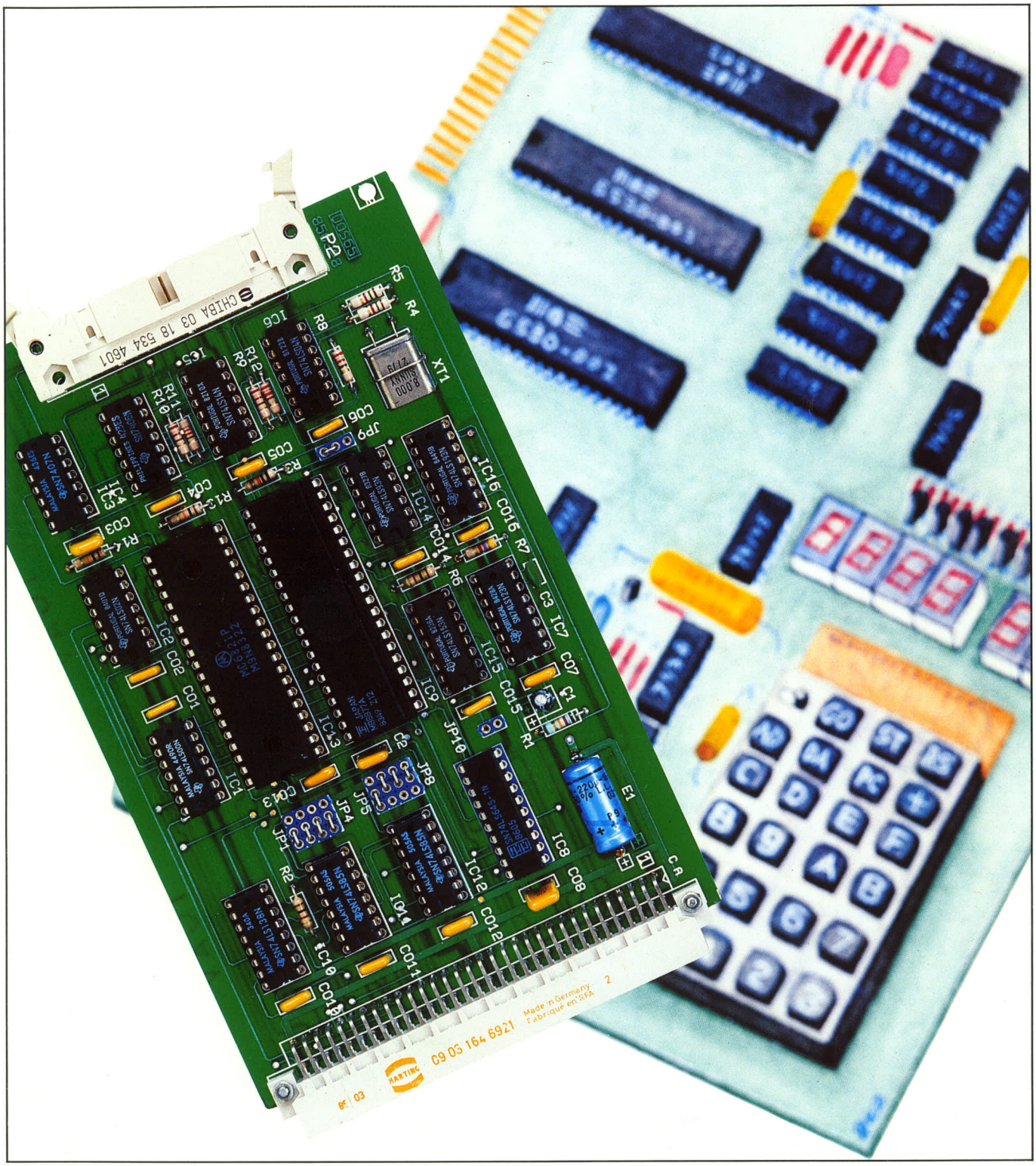
DE 6502 KENNER 59

Twaalfde jaargang nr. 6 december 1988



INFORMATIE.

De 6502 Kenner is een uitgave van de KIM Gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Het doel van de vereniging is het bevorderen van de kennisuitwisseling tussen gebruikers van computers die zijn opgebouwd rond een microprocessor uit de 6500-familie. Voorbeelden hiervan zijn onder andere: Elektuur EC-65, Commodore 64, Apple][, Elektuur Junior, Atari 600 en 800.

De eerder genoemde kennisuitwisseling komt onder andere tot stand door 6 maal per jaar de 6502 Kenner te publiceren, door de organisatie van landelijke bijeenkomsten voor de leden, het instandhouden van een softwarebibliotheek op cassette, floppy disk en papier en het beschikbaar stellen van een Bulletin Board.

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 6502 Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld. Dit Bulletin Board valt onder de verantwoordelijkheid van één van de bestuursleden en wordt bediend door een zgn. Sysop.

Software Bibliotheek:

Voor het beheer van de Software Bibliotheek streeft het bestuur er naar zgn. Software Coördinators te benoemen. Hierbij wordt gedacht aan een drietal coördinators; één voor EC-65(K) en Junior met OHIO DOS-65D, één voor DOS-65 en één voor diverse andere systemen zoals onder andere Elektuur Junior.

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.

Voorzitter:
Rinus Vleesch Dubois
Emiliano Zapataplein 2
2033 CB HAARLEM
Telefoon 023-330993

Secretaris:
Gert Klein
Diedenweg 119
6706 CM WAGENINGEN
Telefoon 08370-23646

Penningmeester:
Jacques H. Banser
Haaksbergerstraat 199
7513 EM Enschede
Telefoon 053-324137

Leden:

Adri Hankel (Bulletin Board)
Willem Kloosstraat 32
7606 BB ALMELO
Telefoon 05490-51151

Gert van Opbroek (Redactie 6502 Kenner)
Bateweg 60
2481 AN WOUBRUGGE
Telefoon 01729-8636

Nico de Vries
Mari Andriessenrade 49
2907 MA CAPELLE A/D IJSSEL
Telefoon 010-4517154

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

==========

Inhoud

De 6502 Kenner:

De 6502 Kenner wordt bij verschijnen gratis toegezonden aan alle leden van de KIM Gebruikersclub Nederland. De kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Alle kopij wordt door de redactie op bruikbaarheid en publicatiewaarde beoordeeld. Deze twee criteria, in samenhang met de actualiteit, bepalen of en zo ja wanneer het stuk gepubliceerd wordt. De redactie streeft er naar de kopij zoveel mogelijk in zijn oorspronkelijke vorm te plaatsen, Nederlandstalige kopij wordt daarom in principe niet naar een andere taal vertaald. De redactie streeft er naar een Nederlandstalig blad te maken doch het staat de auteur vrij een artikel geheel of gedeeltelijk in een andere taal te schrijven.
Helaas kan de redactie, noch het bestuur, enige aansprakelijkheid aanvaarden voor de toepassing(en) van de gepubliceerde kopij.

Verschijningsdata:

De 6502 Kenner verschijnt op de derde zaterdag van de maanden februari, april, juni, augustus, oktober en december.

Redactie.

De redactie wordt momenteel gevormd door:
Gert van Opbroek

Correspondenten:
Bram de Bruine
Antoine Megens
Nico de Vries
Rinus Vleesch Dubois

Redactieadres:
Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

INHOUDSOPGAVE

==========

Redactioneel

Op het moment dat ik dit schrijf, is het sinterklaasavond. Vanavond moet het blad absoluut klaar want als hij morgen niet naar de drukker gaat, dan ben ik bang dat u nummer 59 van De 6502 Kenner niet voor de kerstdagen in huis hebt, en dat zou ik toch jammer vinden. Ik hoop dus, dat u dit voor de kerstdagen kunt lezen.

Allereerst zal ik u dus maar zeer prettige kerstdagen en een dito jaarwisseling toewensen. Ik hoop dat we elkaar weer in 1989 mogen ontmoeten. Overigens, voor ik het vergeet, u hebt allemaal een acceptgirokaart ontvangen voor de contributie voor 1989. Wilt u die contributie a.u.b. zo spoedig mogelijk overmaken? We zijn namelijk een hoop dingen van plan en het zou toch jammer zijn als u dan niet meer lid van de club bent.

Ja, zo aan het einde van een jaar, hoor je altijd even terug te kijken naar het afgelopen jaar. Dit was het eerste jaar dat ik verantwoordelijk was voor de uitgave van De 6502 Kenner (ik denk ook het laatste jaar, maar daarover straks meer). Toen ik bijna een jaar geleden de redactie op mij nam, deed ik dat omdat ik geen alternatief zag. Nu echter kan ik u vertellen, dat ik het zeer leuk werk vind en dat ik het graag nog een poosje blijf doen. Ik ben op de ledenvergadering herkozen, dus ik kan nog twee jaar redacteur blijven. Daarna wil ik de tekstverwerker overdragen aan een ander, vooral omdat ik het niet goed vind dat één persoon dit ambt vele jaren achtereen bekleedt.

Ik zoek wel ondersteuning bij een andere bestuurstaak. In het blad staat ergens een oproep voor een bestuurslid. Dit lid zou ledenwerving en public relations als taak kunnen krijgen. Mijn voorganger deed dit naast zijn redacteursschap er ook allemaal bij, maar dat gaat, gezien mijn dagelijks werk echt niet. Mocht u dus iets voor een dergelijke functie voelen, schroom niet, meldt u aan. Het is overigens op de zes bestuursvergaderingen die we jaarlijks houden uitermate gezellig.....

Nu dan waarom dit waarschijnlijk de laatste 6502 Kenner zal zijn. In de ledenvergadering is goedkeuring verleend aan het plan processor onafhankelijk te worden. Dit betekent dus, dat we niet alleen meer 6502 Kenners zullen zijn, maar ook 8088, 68000 etc. kenners hopen te worden. Daarom wil ik de naam van het blad met ingang van de dertiende jaargang wijzigen. U mag zelf de nieuwe naam kiezen. Ik heb een prijsvraag uitgeschreven voor het bedenken van een nieuwe naam. Als u dus HET idee heeft, laat het me weten en misschien verdient u dan wel de hoofdprijs.

Nu over de inhoud van het blad. In deze aflevering staat onder ander het tweede deel van het PALlen verhaal van Nico de Vries. Verder is hij, in het kader van de nieuwe doelstelling, begonnen aan een serie over PC-compatibles en MS-DOS. U vindt dit in de MS-DOS Corner. Ik denk dat, al hebt u geen compatible, dit toch een interessant artikel kan zijn.

Verder staat er nog een primeur in het blad: een volwaardig Pascal programma. Ik bedacht me opeens, dat het systeem dat ik als tekstverwerker gebruik, ook de beschikking heeft over een Pascal vertaler. Aangezien ik, onder andere door mijn werk, een ruime Pascal-ervaring heb (ik droom wel eens in Pascal), heb ik eens getracht een leerzaam Pascal programma te schrijven. Dit programma gebruikt, afgezien van pointers en set´s (voor niet ingewijden, vergeet deze kreten) zo´n beetje alle mogelijkheden die Pascal in huis heeft. Verder heb ik geprobeerd er uitgebreid commentaar bij te schrijven.

Van Antoine Megens heb ik CHARED geplaatst. Hiermee kunt u, als u een EPROM-programmer hebt, uw eigen character generator maken. Wat het pakket mogelijk ook interessant maakt, is het feit dat er een printerroutine bij is waarmee u de tekens in bit-image mode (dus punt voor punt) naar een printer stuurt.

Bram de Bruine zond mij een klein vervolg op mijn verhaal over CISC en RISC, hij houdt hierin een pleidooi voor een nieuwe 6502, gebaseerd op RISC-technieken (interessante gedachte pagina 0 in registers.....).

Dan als laatste de serie Computers. Deze serie ontstond na een discussie op de laatste ledenvergadering. Het ging in die discussie over het feit dat het niveau van het blad voor sommige mensen misschien wat aan de hoge kant is. De auteurs zijn vaak professionals en ook de nieuwelingen moeten de kans krijgen uit de groeien naar de Kenner die we graag willen zijn. Ik probeer in de serie computers vanaf de basis te beschrijven. Om mensen die de KIM en Junior niet meegemaakt hebben ook te leren hoe een computer werkt.

Tenslotte veel hobbyplezier en tot 1989.

Uw redacteur: Gert van Opbroek.

DE 6502 KENNER

Algemeen

RISC

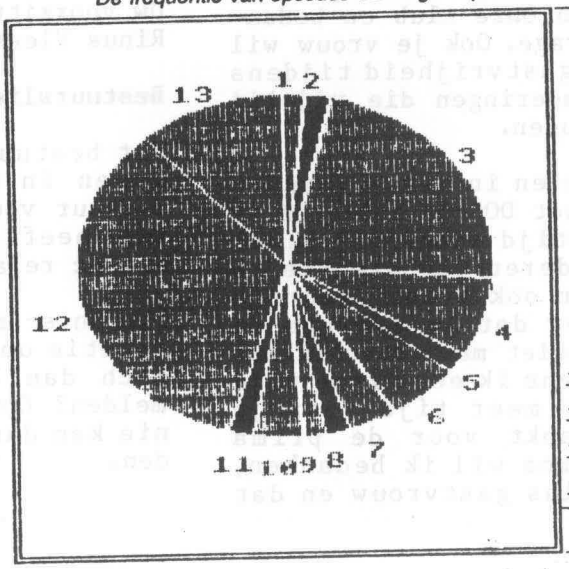
Bij het ontwerp van een microprocessor is men er jarenlang van uitgegaan dat de processor vele malen sneller is dan het geheugen. Minimaliseren van het aantal geheugentransporten werd daarom HET ontwerpprincipe. In de jaren zestig nam de behoefte toe aan hogere programmeertalen. Dit had als gevolg dat de microprocessor ontwikkelaars probeerden hele complexe instructies te maken, met zo min mogelijk geheugentransport. Men wilde als het ware een pascal-instructie uitdrukken in een enkele machinecode mnemonic. Dit leidde tot operatiecodes die zeer lange en ingewikkelde bewerkingen uitvoerden. De instructieset van deze microprocessoren (bv 80x86 en 680x0) nam explosief toe, met als gevolg dat de microcode (een soort hardware operating systeem in de processor) steeds ingewikkelder werd. Men vond dat toen geen enkel probleem, daar sterke minituarisering het mogelijk maakte om vele transistoren te integreren in een enkele chip. Pas sinds kort worden er vraagtekens geplaatst bij dergelijke ingewikkelde microcodes. Bij metingen is namelijk gebleken dat de meeste compilers geen of nauwelijks gebruik maken van deze ingewikkelde instructies. De meest gebruikte instructies zijn: Branch 30%
Load 35%
Store 15%
Overigen 20%

Dit betekend dat 80% van die vernuftig uitgedachte paden in de microcode nauwelijks geactiveerd worden. Jammer dan, zou je denken. Helaas is het nog erger: Ook de eenvoudige Load/store en branchinstructies worden door deze microprogrammering op een zeer omslachtige en tijdrovende manier uitgevoerd. Ergo, als men al die ingewikkelde instructies vergeet, de hele complexe microcode vervangt door eenvoudige snelle transistoren, dan heeft men een razendsnelle processor. Door deze eenvoudige opzet is het ook nog eens mogelijk om uitgebreid pipelining toe te passen. Dit is een methode om alvast data gereed te zetten voor de volgende bewerking, zodat er niet gewacht hoeft te worden. Eventueel kan er dan ook nog parallel gewerkt worden. (vergelijk met transputer idee) Om de snelheid op te voeren kan men zgn. cachegeheugen toevoegen. Dit is zeer snel geheugen bv in de chip geintegreerd, zodat geen databustransport noodzakelijk is. Inmiddels zijn dergelijke microprocessoren bekend onder de naam RISC-processors, (Reduced Instruction Set) Alle andere processoren noemen we gemakshalve maar CISC (Complex Instruction Set Computer) Acorn heeft de ARchiMedes Risc computer ontwikkeld, en dit is werkelijk een razendsnelle computer, waarvoor veel software leverbaar is. De vraag is: "Word het een standaard", of blijkt de IBM/Intelconfiguratie in de toekomst aantrekkelijker ?

Semi-Risc met de 6502 ?

Net als men in taalstudies kan concluderen dat een spatie en een 'e' de meest gebruikte karakters zijn, kan men in een computerprogramma meten wat de meest gebruikte opcode is. Men kan nog verder gaan: kijken welke instructieparen het meest voorkomen. Zo heeft men eens bij een BBC-computer gemeten welke instructie het vaakst volgt na een DEX-instructie. Dit bleek de BNE instructie te

De frequentie van opcodes die volgen op de DEX-instructie.



1:RTS	1%
2:ASL	1%
3:BPL	22%
4:STX a	7%
5:DEY	1%
6:TXA	5%
7:STA a	3%
8:TYA	1%
9:STA a.X	1%
10:LDY z.X	3%
11:LDA a.X	1%
12:BNE	33%
13:BEQ	11%

a = absolute
z = Zero-page

zijn. Waarschijnlijk zal er sprake zijn van een loop, waar men in blijft zolang niet aan de vergelijkingsvoorwaarde van de BNE instructie is voldaan. Reduced instructieset betekend niet dat er geen complexe instructies mogen voorkomen, het gaat er om dat de veelgebruikte instructies SNEL verwerkt worden. Zo zal een blockmove-instructie veelal aanwezig zijn, maar door vele niet gebruikte opcodes uit de microcodering te gooien, word deze blockmove wel veel sneller afgehandeld. ZO zou men het instructiepaar DEX/BNE kunnen vervangen door DEXN label. De instructies DEX en BNE vervallen dan !!! Die weggelaten functies moeten overgenomen worden door combinaties van andere instructies. Een geschikte microcode kan ervoor zorgen dat de processor enkelvoudige instructies niet meer, of niet meer optimaal zal uitvoeren, maar in plaats daarvan efficient combinaties van twee of meer instructies kan verwerken. Zo ontstaat een minimaal aantal instructies, waarmee het nog net mogelijk is de oorspronkelijke instructies te reproduceren, maar dan als een aantal losse instructies met de nieuwe microcode. Dit lijkt in eerste instantie op CISC, maar word toch RISC door het schrappen van ca. 80% van de oude weinig gebruikte code.

Terug naar af ?
==========

De Cmos 6502 kent extra instructies. (CISC-idee) Zou het niet mogelijk zijn om met de moderne technologie de standaard 6502 opnieuw te ontwerpen, zonder ingewikkelde microcode ? Toen de 6502 werd ontwikkeld was dat het maximaal haalbare concept. Het moet toch mogelijk zijn om de 6502 volgens de RISC-filosofie te maken. Is al dat gehobby met die 6502 toch niet voor niet geweest...

Literatuur: De 6502 kenner nr. 57
Databus, juni 1986.

Van de voorzitter

POSITIEF.

Zo mag ik de instelling van mijn collega's in het bestuur wel noemen. In het bijzonder John van Sprang en Erwin Visschedijk. Beiden treden uit het bestuur na vele jaren van fijne samenwerking. John de man die de financiele zaken bijhield en een positieve bijdrage leverde aan de club, zelfs in zeer moeilijke perioden (en dat mag ook wel eens gezegd worden) bleef hij het bestuur trouw. John wij wensen jou nog vele jaren plezier in onze club en bedanken jou voor je bijdrage. Ook je vrouw wil ik bedanken voor de gastvrijheid tijdens de vele bestuursvergaderingen die wij bij jou thuis mochten houden.

Erwin, jouw bijdrage en in het bijzonder de ontwikkeling van het DOS-65 gebeuren is een ieder bekend. Altijd stond jij klaar om je kennis aan anderen ten dienste te stellen, waarvan dan ook gretig gebruik werd gemaakt. Jammer dat je vanwege de drukke werkzaamheden niet meer beschikbaar bent, maar ook jou wens ik een fijne tijd toe als clublid met meer tijd voor je hobby's. Erwin bedankt voor de prima samenwerking, ook Erna wil ik bedanken, zij is een eerste klas gastvrouw en dat hebben wij ondervonden tijdens de vele bestuursvergaderingen bij jullie thuis.

Ik wens de opvolgers van beide bovengenoemde heren zeer veel sterkte toe, maar ik ben er van overtuigd dat ook zij zullen aarden in het bestuur en naar ik wens, nog vele jaren met ons de club zullen runnen.

Alle leden wens ik een plezierige feestdagen toe, en ik hoop u een hand te kunnen schudden op de bijeenkomst op 21 januari te Krommenie.

Uw voorzitter,
Rinus Vleesch Dubois.

Bestuurslid gezocht.

Het bestuur zoekt iemand die zitting wil nemen in het bestuur. De taak die het bestuur voor deze functionaris in gedachten heeft is actieve ledenwerving en public relations.

Indien er mensen zijn die deze belangrijke functie op zich willen nemen, willen zij zich dan bij een van de bestuursleden melden? Op de ledenvergadering in Krommenie kan dan een verkiezing gehouden worden.

De ledenvergadering op 19-11-1988

Door Gert van Opbroek

Dit stukje moet niet gezien worden als het officiële verslag van de vergadering, dat wordt namelijk door de secretaris gemaakt, maar meer als een overzicht van de belangrijkste besluiten.

Ik vond het aantal mensen dat aanwezig was op de vergadering en de bijeenkomst redelijk tot goed. Over het algemeen zijn er op bijeenkomsten ruim twintig man en dat was nu ook het geval. De stemming op de vergadering was goed te noemen en ik ben dan ook van mening dat we een goede vergadering gehad hebben. Nu in het kort even de belangrijkste punten:

Doelstelling van de club en beleidsplan 1989+.

De vergadering ging accoord met de wijziging van de doelstellingen zoals in nummer 57 geformuleerd is. Dit betekent dat de volgende zaken ingevuld gaan worden:

- Het blad krijgt met ingang van de dertiende jaargang een nieuwe naam en de inhoud van het blad zal aansluiten bij de nieuwe doelstelling.
- De nieuwe doelstelling zal op de informatiepagina vermeld worden.
- Het bestuur bereidt een eventuele naamsverandering en wijziging van de statuten voor. Deze wijzigingen zullen conform de statuten t.z.t. aan de leden voorgelegd worden.
- Er zal een uitgebreide ledenwerfactie gestart worden om de club, met zijn nieuwe doelstelling in brede kring bekend te maken.

Conceptbegroting 1989.

De begroting voor 1989, zoals opgesteld door de penningmeester werd aangenomen. Deze begroting is in deze 6502 Kenner opgenomen.

Verkiezing van kascontrolecommissies.

De kascontrolecommissies voor 1988 en 1989 bestaan weer uit twee leden. Tot mijn grote schaamte, weet ik niet de namen van alle commissieleden.

Huishoudelijk reglement.

Het huishoudelijk reglement, zoals voorgesteld door de commissie huishoudelijk reglement is aangenomen en dus in werking getreden. Dit reglement is afgedrukt in nummer 58 van de 6502 Kenner. De vergadering kwam met het voorstel artikel 11 uit het voorstel inderdaad te gebruiken voor het beschrijven van het bestaan en het functioneren van de kascontrolecommissie. De vergadering wilde bovendien vastleggen dat deze commissie bestaat uit twee leden waarvan er ieder jaar één lid aftreedt. De commissie zal dit punt verder uitwerken en proberen deze wijziging op de bijeenkomst van januari aan de leden voor te leggen.

Komende en gaande bestuursleden.

De door het bestuur voorgedragen kandidaten zijn zonder tegenkandidaten verkozen. De vacature Visschedijk is niet ingevuld. Het bestuur streeft ernaar deze vacature op de bijeenkomst van januari toch in te vullen. Binnen het bestuur bestaat de behoefte een bestuurslid te belasten met het doen van Public Relations en het werven van leden. Degene(n) onder de leden die deze taak als een uitdaging zien wordt (worden) verzocht contact op te nemen met één van de bestuursleden.

Rondvraag.

Naar aanleiding van een discussie over de mensen die we met onze club aan willen spreken en de doelstelling van de club kwam uit de vergadering een vraag over het niveau van de clubactiviteiten en de inhoud van het blad naar voren. De vrager maakte zich zorgen over het feit dat een deel van de leden een hoog kennisniveau op computergebied hebben en een ander deel als beginners beschouwd moet worden. Hij vroeg zich, terecht, af de beginners in staat gesteld zouden worden ook door te groeien naar een hoger niveau.

Daar deze vraag voor een deel betrekking heeft op de inhoud van het blad, denk ik dat het goed is als ik daar, als redacteur, een antwoord op tracht te geven.

Ik ben mij er inderdaad van bewust dat het niveau van de artikelen in het blad vaak vrij hoog is. Dit heeft er alles mee te maken dat deze artikelen meestal geschreven worden door professionals op het gebied van de automatisering. Om ook tegemoed te komen aan de leden die dit niveau nog niet hebben zal ik ervoor zorgen dat er in de 6502 Kenner een aantal pagina's met artikelen voor juist deze mensen opgenomen wordt. Deze rubriek start in deze uitgave.

****
*
*
* KIM GEBRUIKERS CLUB NEDERLAND
*
*
****

Krimpen a/d ijssel 14 november 1988

BEGROTING 1989

BATEN :

Contributie 300 * f. 50,=	f.	15000,00
Losse kim kenners	f.	100,00
Bijeenkomsten	f.	750,00
Reclame	f.	900,00
Dos 65	f.	950,00
totaal		--------- +
	f.	17.700,00

LASTEN :

6502 KENNER 6* f. 2500,=	f.	15000,00
Afschrijving inventaris	f.	2500,00
Sysop	f.	1200,00
Bestuur kosten	f.	1500,00
Reclame - ledenwervingsactie	f.	1500,00
totaal		--------- +
	f.	21.700,00
verschil	f.	-4000,00
		--------- +
Tekort voor het jaar 1989	f.	(4.000,00)
		--------- +
totaal	f.	17.700,00

De begroting voor het boekjaar 1989 zal een te kort geven van F 4000,=
Dit komt o.a. :
- Doordat het aantal leden dit jaar met 125 is verminderd.
- verkoop van losse kimkenners zal af nemen.
- dos-65 is verzadigd.

De verliezen van het boekjaar 1989 kunnen door de algemene reserve worden opgebracht.
Voorstel om de club na 1989 te behouden is o.a. :
- Een goede leden werfactie.
- Meer reclame in de Kim Kenner.
- Nieuwe projecten opstarten.
- Veel medewerking van de LEDEN.
- Een nieuw beleid.

Uitnodiging voor de clubbijeenkomst

Datum: Zaterdag 21 januari 1988
Locatie: Nieuwe kantine FORBO-Krommenie
Industrieweg 12
1566 JP Assendelft
Telefoon: 075-291911
Entree: fl.10,-- voor het gedeelte na 11:00 uur

Routebeschrijving

Per auto:

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

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

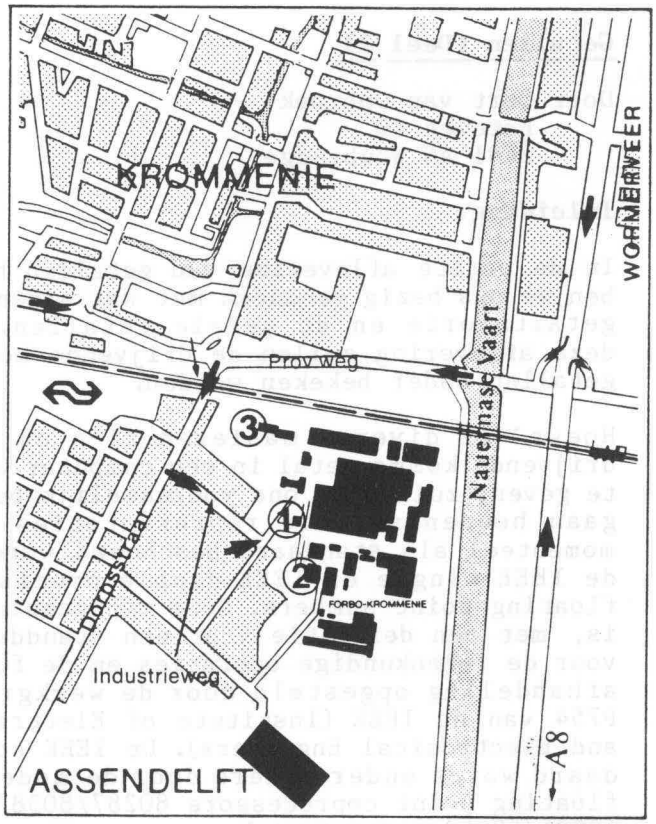
3. Uit de richting Alkmaar: Snelweg Alkmaar-Haarlem. Afslag Uitgeest-Zaandam. Bij kruising linksaf. Bij de 3e stoplichten rechtsaf, de spoorbaan over. Verder volgens punt 2.

Per trein:

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

Programma:

9:30	Zaal open. Ontvangst met koffie.
10:00	Opening door de voorzitter en verwelkoming door de gastheer:Co Filmer.
10:10	Diashow over FORBO-Krommenie.
10:30	Ledenvergadering Agenda: - Behandeling jaarverslag 1988 - Bestuursverkiezing - Wijziging Huish. Reglement: regelen kascontrole commisssie.
10:45	Koffiepauze.
11:00	Spreekbeurt van Nico de Vries over PC-compatibles en/of MS-DOS.
12:00	Forum en markt, naam van het clubblad
12:30	Lunch, aangeboden door FORBO-Krommenie.



① **Portier**
Portier
Portier
Gateman
Portero

② **Ontvangstcentrum**
Salle de reception
Empfangsraum
Reception building
Sala de recepcion

③ **Kantoor**
Bureaux
Büros
Offices
Oficinas

④ **Centraal magazijn**
Magasin central
Zentrallager
Central warehouse
Almacen central

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

Aansluitend het informele gedeelte bedoeld om kennis, ervaring en **Public Domain** software uit te wisselen. 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 enkele aansprakelijkheid voor de gevolgen van het in bezit hebben van illegale software.

Getallen (Deel 2)

Door Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

Inleiding.

In de eerste aflevering van getallen hebben we ons bezig gehouden met wat algemene getaltheorie en de gehele getallen. In deze aflevering zullen de drijvende komma getallen nader bekeken worden.

Hoewel er diverse manieren zijn om een drijvende komma getal in een computer weer te geven, zullen we ons voornamelijk bezig gaan houden met die representaties die momenteel als standaard beschouwd worden, de IEEE single en IEEE double precision floating point numbers. Deze representatie is, met een definitie voor een standdaard voor de rekenkundige operaties en de foutafhandeling opgesteld door de werkgroep P754 van de IEEE (Institute of Electrical and Electronical Engeneers). De IEEE standaard wordt onder andere door de moderne floating point coprocessors 80287/80387 en 68881/68882 ondersteund.

Getallen kleiner dan 1.

In de vorige aflevering is verteld dat het gehele getal 123 als volgt opgevat moet worden:

```
123 =
     1 * 10 tot de macht 2 +
     2 * 10 tot de macht 1 +
     3 * 10 tot de macht 0
```

Evenzo geldt voor 1011:

```
1011 =
     1 *  2 tot de macht 3 +
     0 *  2 tot de macht 2 +
     1 *  2 tot de macht 1 +
     1 *  2 tot de macht 0
```

Deze notatie kan eventueel uitgebreid worden met negatieve exponenten voor de decimale resp. binaire fractie:

```
1.23 =
     1 * 10 tot de macht 0    +
     2 * 10 tot de macht -1   +
     3 * 10 tot de macht -2

1.011 =
     1 *  2 tot de macht 0    +
     0 *  2 tot de macht -1   +
     1 *  2 tot de macht -2   +
     1 *  2 tot de macht -3
```

Drijvende komma getallen

Een volgende stap is het invoeren van een exponent in de notatie. In dat geval wordt het getal 123.45 geschreven als:

```
123.45 = 1.2345E+2
```

Hierbij wil de toevoeging E+2 zeggen: Vermenigvuldig het getal 1.2345 met tien tot de macht 2, dus met honderd.

Bekijken we het getal 1.2345E+2 nogmaals, dan kunnen het getal verdelen in twee delen. In de eerste plaats het deel 1.2345. Dit deel noemen we de mantissa van het getal. In deze mantissa staat de decimale punt meteen achter het eerste cijfer. Dit wil zeggen dat het getal genormeerd is. Het tweede deel dat we kunnen onderscheiden is het deel E+2. Dit noemen we de exponent van het getal. Omdat in het voorbeeld in de exponent de machten van tien aangegeven worden, heet het grondtal van de exponent 10.

Voor getallen kleiner dan 1, krijgen we een negatieve exponent. Bijvoorbeeld:

```
1.2345E-2 = 0.012345
```

dit wil zeggen: vermenigvuldig 1.2345 met tien tot de macht min 2. Dit is hetzelfde als delen door tien tot de macht plus 2. Als laatste kan er voor een getal nog een voorteken staan. Dit voorteken is + of - en geeft aan of het getal groter of kleiner dan nul is. Het weglaten van het voorteken betekent dat het getal 0 of groter is.

Binaire getallen.

Bij een binaire representatie van drijvende komma getallen is de bovengenoemde opdeling ook te maken. Als we naar de IEEE notatie kijken, dan zien we het volgende:

- Een voorteken van 1 bit. Is dit bit geset, dan is het getal negatief
- Een mantissa van 23 bits (single) of 52 bits (double)
- Een exponent van 8 bits (single) of 11 bits (double)

Uiteraard zijn zowel de mantissa als de exponent in een binaire vorm weergegeven. Bovendien geeft de exponent de macht van twee aan waarmee het getal vermenigvuldigd moet worden, het grondtal van de exponent is dus twee.

Op mijn 68000 wordt een IEEE drijvende komma getal als volgt in het computergeheugen opgeslagen:

Bit: 1 09876543 21098765432109876543210

S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM

S: Tekenbit (bit 31)
E: Exponent (bit 23 t/m 30)
M: Mantissa (bit 0 t/m 22)

NB. Op een 68000 is het gebruikelijk dat het meest significante byte op het laagste adres staat. In andere voorbeelden zullen we zien dat dit niet altijd het geval is.

Weergegeven is het single precision floating point number, het double precision floating point number is analoog gecodeerd.

Het tekenbit staat dus in het meest significante bit, daarna volgt de exponent en daarna de mantissa.

Van de mantissa worden 23 resp. 52 bits weergegeven, toch is de nauwkeurigheid 24 resp. 53 bits. Hoe zit dat? Welnu, dat heeft met de manier waarop een getal genormeerd wordt te maken. Dat gaat als volgt: De exponent is zodanig dat het voorste bit van een getal altijd een 1 is. Bovendien staat de decimale punt meteen achter deze 1. Omdat het voorste bit altijd een 1 is, hoeven we deze in een genormeerd getal eigenlijk ook niet aan te geven, hij is toch altijd 1. Welnu, dat is precies wat er gedaan wordt, het voorste bit van de mantissa is in een genormeerd getal altijd een 1 en wordt daarom maar weggelaten.

Enkele voorbeelden.

Het is vaak een leuke bezigheid zelf eens te onderzoeken hoe iets op je eigen computer gedaan wordt. In navolging van het maandblad mc (ref. 2), worden een paar programma's gegeven waarmee de afzonderlijke bits van een floating point getal afgedrukt kunnen worden. Het zijn twee verschillende programma's:

Programma 1.

Programma 1 is een Pascal-programma. Dit programma is geschreven in standaard (ISO) Pascal (ref 3) en zou dus ook op bijvoorbeeld een DOS-65 systeem gebruikt kunnen worden. Het afgedrukte programma is getest op een Apple II compatible met Z80, CP/M en Turbo Pascal.

De genoemde combinatie van hard- en software slaat drijvende komma getallen niet volgens de IEEE standaard op. Een getal wordt in dit geval opgeslagen in zes bytes waarvan het meest significante byte op het hoogste adres ligt. Bovendien is de volgorde van mantissa en exponent omgedraaid ten opzichte van de IEEE standaard. Een verdere uitleg over het programma staat in een aparte paragraaf.

Programma 2.

Programma 2 is geschreven in de programmeertaal C volgens het boek van Kernigan and Ritchie (Ref. 4). Dit programma is gebruikt op een mc68000 onder het operating systeem OS9/68k met de bijbehorende Microware C-compiler. Deze combinatie van hard- en software gebruikt drijvende komma getallen volgens de IEEE standaard. Bovendien wordt het meest significante byte op het laagste adres opgeslagen. Ook van dit programma staat de uitleg in een aparte paragraaf.

Met de genoemde programma's zijn enkele voorbeelden bekeken. Een overzicht van deze voorbeelden staat in tabel 1.

Kijken we naar deze voorbeelden, dan vallen ons enkele dingen op:

- Het getal nul wordt in beide gevallen weergegeven met allemaal nullen. Dit is altijd zo (per definitie) en is gedaan om op eenvoudige wijze op dit resultaat te kunnen testen.
- Onder CP/M heeft een floating point getal een lengte van 6 byte, onder OS9 een lengte van vier byte. Onder OS9 wordt echter ook een double precision versie ondersteund met een lengte van 8 byte, onder CP/M is dit niet het geval.
- Beide systemen hebben 8 bits ingeruimd voor de exponent van het getal, bij CP/M staat de exponent echter achter de mantissa, terwijl hij onder OS9 er voor staat. Het tekenbit van het hele getal staat vooraan in het getal. De overige 23 resp. 39 bits bevatten de mantissa.
- De mantissa is voor gelijke getallen op beide systemen gelijk, alleen heeft CP/M meer bits, dus een hogere nauwkeurigheid (meer cijfers achter de komma).
- Hoewel het drijvende komma getal en de mantissa gelijk is, ziet de expo-

nent er op beide systemen verschillend uit. Dit heeft met de definitie van een genormeerd getal te maken en met de binaire constante die bij de exponent opgeteld is. Hoe dit precies in elkaar zit, wordt zo uitgelegd.

- In beide gevallen verschillen positieve en negatieve getallen alleen in het tekenbit. Een drijvende komma getal is dus weergegeven als **Signed Magnitude** (zie deel 1).

Verdere beschrijving.

Zoals in de vorige paragraaf al is opgemerkt, geeft het tekenbit aan of het getal positief of negatief is. Het veranderen van het tekenbit maakt van een negatief getal een positief getal of andersom. Dit gaat dus net zo als bij de normale decimale getallen. Een getal bestaat uit een teken (sign) en iets dat de grootte (magnitude of absolute waarde) aangeeft, vandaar de term Signed Magnitude.

Bekijken we nu het getal 1:

Als we het getal 1 in machten van twee gaan ontbinden dan krijgen we het volgende:

1 = 2 tot de macht 0

Dit getal zou er dus als volgt uit kunnen zien:

Sign = 0
Mantissa = 10000000 00000000 00000000
Exponent = 0000000

Dus als 0 00000000 1.00000000000000000000

Bij de IEEE-notatie staat de decimale punt altijd meteen achter het eerste bit. Omdat het eerste bit altijd een 1 is, wordt dit bit ook maar weggelaten.

Omdat het voor het rekenen met floating point getallen (zie deel 3) lastig is als er grote verschillen tussen positieve en negatieve exponenten zijn, wordt de exponent weergeven in de Ecxess notatie (zie deel 1). In de IEEE notatie is dit de excess 127 notatie waarbij het getal 127 bij de exponent opgeteld wordt. Onder CP/M is dit het getal 128 en spreken we van de excess 128 notatie.

OS9/68k met Microware C

FP	HEX	BIN
0.0000e+0	0	0 00000000 00000000000000000000000
5.0000e-1	3f000000	0 01111110 00000000000000000000000
-5.0000e-1	bf000000	1 01111110 00000000000000000000000
1.0000e+0	3f800000	0 01111111 00000000000000000000000
-1.0000e+0	bf800000	1 01111111 00000000000000000000000
2.0000e+0	40000000	0 10000000 00000000000000000000000
-2.0000e+0	c0000000	1 10000000 00000000000000000000000
3.0000e+0	40400000	0 10000000 10000000000000000000000
-3.0000e+0	c0400000	1 10000000 10000000000000000000000
1.0000e+6	49742400	0 10010010 11101000010010000000000
1.0000e-6	358637bd	0 01101011 00001100011011111011101
8.3000e-1	3f547ae1	0 01111110 10101000111101011100001
1.2345e+0	3f9e0419	0 01111111 00111100000010000011001

Apple II (CP/M) met Turbo Pascal

0.0000e+0	000000000000	0 00000000000000000000000000000000000000000000 00000000
5.0000e-1	000000000080	0 00000000000000000000000000000000000000000000 10000000
-5.0000e-1	800000000080	1 00000000000000000000000000000000000000000000 10000000
1.0000e+0	000000000081	0 00000000000000000000000000000000000000000000 10000001
-1.0000e+0	800000000081	1 00000000000000000000000000000000000000000000 10000001
2.0000e+0	000000000082	0 00000000000000000000000000000000000000000000 10000010
-2.0000e+0	800000000082	1 00000000000000000000000000000000000000000000 10000010
3.0000e+0	400000000082	0 10000000000000000000000000000000000000000000 10000010
-3.0000e+0	c00000000082	1 10000000000000000000000000000000000000000000 10000010
1.0000e+6	742400000094	0 11101000010010000000000000000000000000000000 10010100
1.0000e-6	0637bd05af6d	0 00001100011011110111101000001011010111 01101101
8.3000e-1	547ae147ae80	0 10101000111101011100001010001111010111 0 10000000
1.2345e+0	1e0418937481	0 00111100000010000011000100100110111010 0 10000001

Behalve het verschil van 1 t.g.v. het verschil tussen excess 127 en excess 128 blijkt er in de exponent nog een verschil van 1 te zijn. Dit heeft te maken met de definitie van een genormeerd getal. In een IEEE-getal staat de komma meteen achter het eerste bit, onder CP/M staat de komma voor het eerste bit. Dus:

```
5.0000E+0 =

IEEE:
sign       = 0
mantissa   = 1.01000000000000000000000
exponent   = 10000001          (2)

CP/M:
sign       = 0
mantissa= .10100000000000000000000..
exponent   = 10000011          (3)
```

Daar het voorste bit van de mantissa weggelaten wordt, zien de getallen er als volgt uit:

```
IEEE:
0 10000001 01000000000000000000000

CP/M:
0 01000000000000000000000.. 10000011
```

Overflow/Underflow

Als een getal, bijvoorbeeld in een berekening, in absolute waarde te groot wordt, dan heet dit overflow. Wordt het getal te klein, dan is dit underflow.

In de IEEE-specificatie voor single precision is aangegeven dat de exponent in een genormaliseerd getal ongelijk aan nul altijd tussen -127 en +128 (dus 0 < exponent < ff) ligt. Is de exponent 0, dan wil dit zeggen dat het getal te klein is geworden om nog in genormeerde vorm weer te geven. Het getal kan dan vaak nog wel weergegeven worden, alleen moet er aan de mantissa geen bit 1 toegevoegd worden.

Voorbeeld:

```
0 00000001 00000000000000000000000 =

sign       = 0
mantissa   = 1.00000000000000000000000
exponent   = -126

kortom: 2 tot macht 0 *
        twee tot de macht -126.
```

Wordt dit getal door twee gedeeld, dan krijgen we:

```
0 00000000 10000000000000000000000
sign       = 0
mantissa   = 1.00000000000000000000000
exponent   = -127

kortom: 2 tot de macht 0 *
        2 tot de macht -127
```

Delen we dit getal nogmaals door twee, dan krijgen we:

```
0 00000000 01000000000000000000000
sign       = 0
mantissa   = 0.10000000000000000000000
exponent   = -127

kortom: 2 tot de macht -1 *
        2 tot de macht -127
```

Wordt het getal nog kleiner, dan staan er steeds meer nullen vooraan in de mantissa en komen we tenslotte bij de representatie van het getal nul. Hoewel IEEE voorziet in een manier om getallen weer te geven die kleiner zijn dan het kleinste getal dat nog in genormeerde vorm weergegeven kan worden, zijn er toch getallen die zo klein zijn dat ze niet weergegeven kunnen worden. Dit heet **Underflow.**

Een exponent +128 (allemaal enen) heeft ook een bijzondere betekenis. Dit zijn de zogenaamde NANs = Not A Number (geen getal). Verder is een getal waarvan in de mantissa en de exponent alle bits geset zijn de representatie van oneindig. Het tekenbit geeft hierbij aan of er +oneindig of -oneindig bedoeld wordt. In de IEEE specificatie is vastgelegd wanneer een berekening een NAN of oneindig oplevert.

Als van het resultaat van een berekening de absolute waarde groter wordt dan het grootste getal dat in een floating point formaat opgeslagen kan worden, dan noemen we dit **Overflow.** In de praktijk zal dit altijd met oneindig aangegeven worden.

Samenvatting.

In dit deel is de manier waarop in een computer een drijvende komma getal weergegeven wordt behandeld. Aan de hand van enkele voorbeelden is dit nog wat verder uitgewerkt.

Als bijlage bij dit artikel zijn een tweetal programma's gegeven waarmee men zelf zijn of haar systeem kan onderzoeken. Mocht u nog een opslagmethode vinden die afwijkt van de twee voorbeelden die ik gegeven heb, dan wil ik daar graag over ingelicht worden.

Programma 1.

```
1   PROGRAM Float(input,output);
2
3   (*
4    * Convert a floating point number to an hexadecimal and a binary bit array
5    *
6    * This program is tested on an Apple II with CP/M and Turbo Pascal
7    *
8    * Author Gert van Opbroek  20/11/88
9    *
10   * This program was adapted from mc november 1988 page 79
11   *)
12
13  CONST C_fp_bytes = 6;                (* Implementation dependent *)
14
15  TYPE  T_byte       = CHAR ;
16        T_byte_array = PACKED ARRAY [1..C_fp_bytes] OF T_byte;
17
18        (*
19         * T_conversion is used for the conversion of a real
20         * to an byte array.
21         *)
22
23        T_conversion = RECORD
24                       CASE BOOLEAN OF
25                         TRUE:   (fpnum : REAL);
26                         FALSE:  (l_int : T_byte_array)
27                       END;
28
29  VAR  number : T_conversion;
30       i      : INTEGER;
31  (*
32   * Hex writes one byte in hexadecimal notation
33   *)
34
35  PROCEDURE Hex(h : T_byte);
36
37  VAR x : INTEGER;
38
39  BEGIN
40    x := ord(h) DIV 16 ;
41    IF x > 9
42    THEN write(chr (x+55))
43    ELSE write(chr (x+48));
44
45    x := ord(h) MOD 16;
46    IF x > 9
47    THEN write(chr (x+55))
48    ELSE write(chr (x+48))
49  END;
50
51  (*
52   * Bin writes one byte in binary notation
53   *)
54
55  PROCEDURE Bin(h : T_byte);
56
57  VAR x,i : INTEGER;
58
59  BEGIN
60    x := ord(h);
61    FOR i := 1 to 8 DO
```

```
62      BEGIN
63        x := x * 2;
64        IF x > 255
65        THEN write('1')
66        ELSE write('0');
67        x := x MOD 256
68      END;
69    END;
70
71    BEGIN
72      WHILE TRUE DO
73      BEGIN
74        write('Please give a floating point number: ');
75        readln(number.fpnum);
76
77        writeln('FP-format:  ',number.fpnum);
78
79        (*
80         * MS byte on lowest address (implementation dependent
81         *)
82
83        write('HEX-format: ');
84        FOR i := C_fp_bytes DOWNTO 1 DO
85        BEGIN
86          Hex(number.l_int[i]); write(' ')
87        END;
88        writeln;
89
90        write('BIN-format: ');
91        FOR i := C_fp_bytes DOWNTO 1 DO
92        BEGIN
93          Bin(number.l_int[i]); write(' ')
94        END;
95        writeln
96
97      END;
98    END.
```

Programma 2.

```
1   #include <stdio.h>
2   #define void int
3
4   /*
5    * Convert a floating point number to an hexadecimal and a binary bit array
6    *
7    * This program is tested on a mc68000 with OS9/68k and Microware C.
8    *
9    * Author Gert van Opbroek              21/11/88
10   *
11   * This program is adapted from mc november 1988 page 79
12   */
13
14  /*
15   * Bin prints a floating point number in binary notation
16   */
17
18  void Bin(f)
19  long *f;
20
21  {
22          int i,k;
23          printf("BIN-format: ");
24          for (i = 0;i < sizeof(float) * 8;i++)   {
```

```
25                      k = ((*f & 0x80000000) != 0);
26                      printf("%1d",k);
27                      *f <<= 1 ;
28              }
29      }
30
31      main()
32      {
33              float number;
34              long  *pointer;
35              pointer = &number;
36              while(1)
37              {
38                      printf("Please enter floating point number: ");
39                      scanf("%f",&number);
40                      printf("\nFP-format:  %g, \nHex-format: %8x\n",number,*pointer);
41                      Bin(pointer);
42                      printf("\n");
43              }
44      }
```

Beschrijving van de programma´s.

1: **Programma 1: Het Pascalprogramma.**

Dit programma is geschreven in ISO-Pascal en getest onder CP/M met Turbo Pascal. Voor DOS-65 is het noodzakelijk d.mv. een compilerswitch aan te geven dat in namen underscores (´_´) toegestaan zijn.

Het hoofdprogramma start op regel 71. In een WHILE lus worden getallen ingegeven en in drie formaten afgedrukt. Aangezien de conditie in de WHILE altijd true (waar) is, wordt de lus dus oneindig vaak doorlopen. Het einde van de WHILE lus wordt gevormd door de END op regel 97.

Nadat het floating point getal op regel 74 ingelezen is, wordt hij in het floating point formaat afgedrukt op regel 77.

In regel 84 t/m 87 wordt voor elk byte in het getal de routine Hex aangeroepen die een byte in HEX-formaat afdrukt. In de constante C_fp_bytes staat het aantal bytes dat bekeken moet worden terwijl de FOR lus met het sleutelwoord DOWNTO aangeeft dat de index van de lus iedere keer verlaagd moet worden.

Regel 90 t/m 94 bevatten een FOR lus die ervoor zorgt dat de bytes in een binaire vorm afgedrukt worden.

Pascal is heel streng in zijn types. Dit wil zeggen dat we zonder speciale maatregelen niet zondermeer de bytes waaruit een floating point getal kunnen benaderen. Eén van de methodes om dit netjes te doen is door middel van een datastructuur. Deze datastructuur is in de TYPE declaratie in regel 12 t/m 27 gedefinieerd (in Pascal termen: gedeclareerd). De constructie met CASE in het record geeft zogenaamde varianten aan. De inhoud van het record kan opgevat worden als een floating point getal door hem te benaderen met:

```
number.fpnum
```

Ook kan men dit getal als een byte ARRAY opvatten door hem te benaderen met:

```
number.l_int[i]
```

waarbij i de index in de array is.

Aan de procedure Hex uit regel 35 t/m 49 wordt de waarde van het byte dat we hexadecimaal af willen drukken als parameter meegegeven. Verder wordt er binnen de procedure een lokale variabele x gedeclareerd die alleen binnen deze procedure bestaat.

In regel 40 worden de hoge vier bits bekeken die in regel 42 of 43 als hexadecimaal teken afgedrukt worden.

Ook aan de procedure Bin uit regel 59 t/m 69 wordt de waarde van het byte als parameter meegegeven. Bovendien worden de lokale variabelen x en i in regel 57 gedeclareerd.

In de lus in regel 61 t/m 68 wordt steeds het voorste bit van het byte geisoleerd en afgedrukt. Dit wordt gedaan door het getal iedere keer met 2 te vermenigvuldigen. Is dit resultaat groter dan 255, dan is het bit geset, anders is het bit clear. Door van het resultaat de rest van een deling door 256 (MOD) te nemen, houden we de lage bits over voor de volgende doorgang door de lus.

2: **Programma 2: Het C programma**

Ook in programma 2 zien we de while lus. In C geeft het getal 1 true aan dus hierin wijkt het programma niet af van de versie in Pascal.

In de gebruikte C implementatie heeft een long (4 byte integer) dezelfde lengte als een float. Door middel van het definieren van een pointer wordt de float omgezet in een long. In C mag dit, waarbij de gebruikte C compiler wel een waarschuwing op regel 35 geeft. in deze regel wordt het startadres van de float toegekend aan de pointer op een long.

In regel 40 zien we dat C een long af kan drukken in Hex-formaat zonder dat daar verdere bewerkingen voor nodig zijn.

De functie Bin uit regel 18 t/m 29 is van dezelfde opzet als in het Pascal programma, alleen is de for lus nu in de functie opgenomen. Bovendien wordt er gebruik gemaakt van het feit dat true (waar) aangegeven wordt met een 1 en false (onwaar) met een nul.

Referenties.

1: Hagen Völzke: Fliesskomma-Arithmetik und IEEE-Spezifikation Teil 1; mc 10/1988 pag. 123.
2: Hagen Völzke: Fliesskomma-Arithmetik und IEEE-Spezifikation Teil 2; mc 11/1988 pag. 78.
3: Jensen and Wirth: Pascal User Manual and Report, Third Edition.
4: Kernigan and Ritchie: The C Programming Language

PRIJSVRAAG: Wie verzint een nieuwe naam voor het blad?

De huidige naam van het blad is De 6502 Kenner. Deze naam is nu al weer zo'n jaar of zes in gebruik. Daarvoor heette het blad de KIM Kenner, naar de naam van de club.

De naam van het blad heeft alles te maken met de doelstelling van de club. In de tijd dat het blad de KIM Kenner genoemd werd, hielden de leden zich voornamelijk bezig met de KIM. Daarna kwamen onder andere de Junior, de Apple II, de Octopus en DOS-65. Omdat dit geen KIM's waren en de naam KIM Kenner niet bij deze systemen past, werd het De 6502 Kenner.

Nu, aan het begin van 1989, is besloten dat de club zich breder op gaat stellen ten aanzien van het type processor. Dit betekent dat de naam De 6502 Kenner niet meer past bij het blad. We hebben dus een nieuwe naam nodig!! Ik zou het heel prettig vinden als we met ingang van de dertiende jaargang een nieuwe naam voor het blad zouden kunnen voeren.

Ik roep daarom uw hulp in!

Wie verzint er een leuke naam die aansluit bij de nieuwe richting die de club inslaat?

U kunt uw bijdrage tot 15 januari bij mij inleveren.

De winnaar wordt beloond met een blad dat de door hem bedachte naam draagt en met een boekenbon van F.50,--. Op de bijeenkomst in Krommenie op 21 januari 1988 krijgen de daar aanwezige leden de kans uit de ingezonden namen De Naam te kiezen.

Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

CHARED: Editor for the DOS65 character generator.

Met behulp van CHARED kan men de Charactergenerator die in een DOS65 systeem zit zelf wijzigen.

Het pakket bestaat uit:

1: Een basic programma met de eigenlijke editor (CHARED.BAS)
2: Een assemblerprogramma met een printroutine (PRCHAR.MAC)
3: Diverse charactersets waarvan die van DOS65 als voorbeeld afgedrukt is (DOS65.SET)

Deze software staat op het Bulletin Board en kan eventueel via de DOS65 coördinator besteld worden.

```
1000 REM +-------------------------+
1001 REM |        CHARED.BAS       |
1002 REM |-------------------------|
1003 REM |        A.G.MEGENS       |
1004 REM |   febr.1983 V.1  OSI    |
1005 REM |   sept.1984 V.2  OS65D3 |
1006 REM |    jan.1988 V.3  DOS65  |
1007 REM +-------------------------+
1008 POKE &003D,&005F:REM top of RAM at $5FFF to
1009 POKE &003C,&00FF:REM protect buffer at $6000
1009 X=FRE(X):REM adjust rest of pointers
1010 WI=7:REM WIDTH OF MATRIX - 1
1011 HI=9:REM HEIGHT OF MATRIX - 1
1015 DIM CHAR(WI,HI):TE=0:BASE=&6000:MAX=255
1018 FOR I=0 TO WI:FOR J=0 TO HI:CHAR(I,J)=0:NEXT J,I
1020 CX=41:CY=2:X0=CX:Y0=CY:OC=0:ER$=""
1022 CHANGED=0:NTSAVD=0:STNSAVD=0
1025 CL$=CHR$(26):REM clear to end_of_line
1026 CS$=CHR$(25):REM clear to end_of_screen
1027 AX=PEEK(&AAC0):REM save mode byte
1030 HOME:CURSOR TO 2,1
1033 PRINT " Character editor V.3"
1034 PRINT "----------------------"
1035 CR$=CHR$(13)
1040 GOSUB 10000:REM show matrix
1060 CURSOR TO 4,1
1075 PRINT"G = Goto character x"
1076 PRINT"L = Load char.set from disk"
1077 PRINT"S = Save char.set on disk"
1078 PRINT"I = Invert character"
1079 PRINT"Q = Quit"
1080 PRINT"E = Erase matrix"
1081 PRINT"P = Put matrixchar. in set"
1082 PRINT"C = Copy matrixchar. from x"
1083 PRINT"< = Previous character"
1084 PRINT"> = Next character"
1085 PRINT"H = Hardcopy of char.set"
1086 PRINT"1 = Set bit"
1087 PRINT"0 = Clr bit              ^E"
1088 PRINT"Cursor control with    ^S  ^D"
1089 PRINT"standard EDITOR keys:    ^X"
1092 GOTO 5000:REM show first char.
1097 REM +--------------------+
1098 REM |     MAIN LOOP      |
1099 REM +--------------------+
1100 CURSOR TO CY,CX:IF OC=1 THEN INVERSE
1101 PRINT "[]":NORMAL
1102 CURSOR TO 20,27
1105 IF CHANGED THEN INVERSE:PRINT "Modified":NORMAL:STNSAVD=1:GOTO 1107
```

```
1106 PRINT "        "
1107 CURSOR TO 21,20
1108 IF NTSAVD THEN INVERSE:PRINT "(not saved in set)":NORMAL:GOTO 1110
1109 PRINT "                    "
1110 GOSUB 22000           :REM CHECK KEYS
1115 IF DX+DY<>0 THEN 1100 :REM SHOW NEW CURSOR
1120 IF DA=0 THEN GOTO 1110:REM NO KEYS AT ALL
1130 REM +--------------------+
1131 REM | CHECK COMMAND-KEYS |
1132 REM +--------------------+
1139 REM convert to uppercase
1140 IF DA$<="z" AND DA$>="a" THEN DA$=CHR$(DA-(ASC("a")-ASC("A")))
1150 FOUND=0
1155 CM$="GLSIQEPC<>H10":REM allowed commands
1160 FOR I=1 TO LEN(CM$)
1165 :   IF DA$ = MID$(CM$,I,1) THEN FOUND=I:I=LEN(CM$)
1170 NEXT I
1175 IF FOUND THEN 1180
1177 ER$="Unknown command":GOSUB 11000:GOTO 1100
1180 GOSUB 11100:REM erase message line
1181 IF FOUND>10 THEN 1185
1182 ON FOUND    GOTO 6000,8000,7000,1560,1990,1530,3000,9000,1510,1500
1185 ON FOUND-10 GOTO 4000,1300,1310
1190 STOP
1297 REM +--------------------+
1298 REM | '1' & '0' command  |
1299 REM +--------------------+
1300 OC=1:GOSUB 1400:CHANGED=1:NTSAVD=1:GOTO 1100
1310 OC=0:GOSUB 1450:CHANGED=1:NTSAVD=1:GOTO 1100
1400 REM *** SET DOT ON ***
1405 CURSOR TO CY,CX
1410 INVERSE:PRINT "  ":NORMAL:RETURN
1450 REM *** SET DOT OFF ***
1455 CURSOR TO  CY,CX
1460 NORMAL:PRINT "  ":RETURN
1497 REM +-----------------------------------+
1498 REM | '>' Next and '<' Prev. command    |
1499 REM +-----------------------------------+
1500 TE=TE+1:CHANGED=0:NTSAVD=0:GOTO 5000
1510 TE=TE-1:CHANGED=0:NTSAVD=0:GOTO 5000
1511 REM +--------------------+
1512 REM |  'E'rase command   |
1513 REM +--------------------+
1530 FOR I=0 TO HI
1535 : FOR J=0 TO WI
1540 :  IF CHAR(J,I)=0 THEN 1550
1545 :  CURSOR TO Y0+2*I,X0+3*(WI-J)
1547 :  GOSUB 1460:CHAR(J,I)=0
1550 NEXT J,I
1552 OC=0:CHANGED=1:NTSAVD=1
1553 GOTO 1100
1554 REM +---------------------+
1555 REM |  'I'nverse command  |
1556 REM +---------------------+
1560 FOR I=0 TO HI
1565 : FOR J=0 TO WI
1566 :  CURSOR TO (Y0+2*I),(X0+3*(WI-J))
1570 :  IF CHAR(J,I)=1 THEN CHAR(J,I)=0:GOSUB 1460:GOTO 1590
1580 :  CHAR(J,I)=1:GOSUB 1410
1590 NEXT J,I:CHANGED=1:NTSAVD=1
1600 IF OC=0 THEN OC=1:GOTO 1100
1610 OC=0
1620 GOTO 1100
1630 :
```

```
1986 REM +------------------+
1987 REM | 'Q'uit command   |
1988 REM +------------------+
1989 REM --- 'Q'uit command ---
1990 IF STNSAVD THEN 2010
2000 HOME:PRINT:PRINT:POKE &AAC0,AX:END
2010 CURSOR TO 22,1
2020 PRINT "Are you sure ? (modified set not stored on disk)"
2030 GET C$:IF C$="" THEN 2030
2040 IF C$="Y" OR C$="y" THEN 2000
2050 CURSOR TO 22,1:PRINT CL$:POKE 22,0
2060 GOTO 1100
2996 REM +--------------------+
2997 REM | 'P'ut char. in set |
2998 REM +--------------------+
3000 CURSOR TO 22,1:POKE 22,0
3003 PRINT"Saving character nr.";TE;" "
3005 CHAR(WI-(CX-X0)/3,(CY-Y0)/2)=OC
3010 FOR I=0 TO HI
3015 : B=0
3020 : FOR J=0 TO WI
3030 :  IF CHAR(J,I) THEN B=B+INT(2^J+.5)
3050 : NEXT J
3060 : POKE (BASE+16*TE+I),B
3065 NEXT I:NTSAVD=0
3068 CURSOR TO 22,1:POKE 22,0:PRINT CL$
3080 GOTO 1100
3996 REM +----------------------+
3997 REM | 'H'ardcopy of buffer |
3998 REM +----------------------+
4000 CURSOR TO 22,1:PRINT "Please wait --> printing";CL$;:POKE 22,0
4005 CURSOR TO 23,1
4010 DOS"LOAD PRCHAR.BIN":REM ** GET MT PROGRAM **
4030 CALL &A000
4060 CURSOR TO 22,1:PRINT CL$
4070 GOTO 1110
4995 REM +-----------------+
4996 REM | Show BUFFER(TE) |
4998 REM +-----------------+
5000 ER$="Reached end of buffer"
5001 IF TE<0   THEN GOSUB 11000:TE=0  :GOTO 1100
5002 IF TE>MAX THEN GOSUB 11000:TE=MAX:GOTO 1100
5003 CURSOR TO 20,1:POKE 22,0:PRINT"Showing character nr.";TE;"  "
5004 CURSOR TO 20,27:PRINT "        "
5005 CURSOR TO 21,20:PRINT "                  "
5010 FOR I=0 TO HI
5015 : B=PEEK(BASE+TE*16+I)
5020 : FOR J=0 TO WI
5030 : CHAR(J,I)=SGN(B AND 2^J)
5035 : CURSOR TO Y0+2*I,X0+3*(WI-J)
5040 : IF CHAR(J,I) THEN GOSUB 1410:GOTO 5060
5050 : GOSUB 1460
5060 NEXT J,I
5070 OC=CHAR(WI-(CX-X0)/3,(CY-Y0)/2)
5080 GOTO 1100
5996 REM +-----------------+
5997 REM | 'G'oto character |
5998 REM +-----------------+
6000 CURSOR TO 22,1:POKE 22,0
6010 PRINT "Which character (0-255) ";CL$;:POKE 22,0
6030 INPUT C$
6035 GOSUB 12000:REM remove spaces
6040 IF C$="" THEN CURSOR TO 22,1:PRINT CL$;:POKE 22,0:GOTO 1110
6045 ER$="Enter max. 3 digits"
```

```
6050 IF LEN(C$)>3 THEN GOSUB 11000:GOTO 6000
6055 ER$="Illegal number"
6060 FOR I=1 TO LEN(C$)
6070 : Q$=MID$(C$,I,1)
6080 : IF Q$<"0" OR Q$>"9" THEN GOSUB 11000:I=LEN(C$):NEXT:GOTO 6000
6090 NEXT I
6091 ER$="Number out of range, must be between 0 and"+STR$(MAX)
6094 IF VAL(C$)>MAX OR VAL(C$)<0 THEN GOSUB 11000:GOTO 6000
6095 TE=VAL(C$):CHANGED=0:NTSAVD=0
6096 CURSOR TO 22,1:PRINT CL$;:POKE 22,0
6097 GOSUB 11100
6100 GOTO 5000
6996 REM +--------------------+
6997 REM | ´S´ave set on disk |
6998 REM +--------------------+
7000 QU$="Give filename to save set "
7030 GOSUB 14000
7035 IF C$="" THEN 7096
7080 D$="SAVE "+C$+" "
7082 DC=BASE:GOSUB 13000:D$=D$+HX$:REM convert BASE to HEX
7085 DC=BASE+16*MAX:GOSUB 13000:D$=D$+","+HX$
7090 DOS D$:REM SAVE <filename> <start>,<end>
7095 CURSOR TO 23,1:PRINT "Character set saved in file ";C$
7096 CURSOR TO 22,1:PRINT CL$:POKE 22,0:STNSAVD=0
7100 GOTO 1110
7996 REM +----------------------+
7997 REM | ´L´oad set from disk |
7998 REM +----------------------+
8000 IF STNSAVD=0 THEN 8009
8001 CURSOR TO 22,1
8002 PRINT "Are you sure ? (current set is modified but not stored on disk)"
8003 GET C$:IF C$="" THEN 8003
8004 IF C$="Y" OR C$="y" THEN 8009
8005 CURSOR TO 22,1:PRINT CL$:POKE 22,0:GOTO 1100
8009 HOME:DOS"DIR":CURSOR TO 22,1:POKE 22,0
8010 QU$="Give filename of set "
8030 GOSUB 14000
8035 IF C$="" THEN CURSOR TO 22,1:PRINT CL$:POKE 22,0:GOTO 1030
8080 D$="LOAD "+C$+" "
8082 DC=BASE:GOSUB 13000:D$=D$+HX$:REM convert BASE to HEX
8090 DOS D$:REM LOAD <filename> <start>
8091 CURSOR TO 22,1
8092 PRINT "Press <RETURN> to continue....";CL$;:POKE22,0
8093 GET C$:IF C$="" THEN 8093
8094 IF ASC(C$)<>13 THEN 8093
8095 CURSOR TO 22,1:PRINT CS$
8099 TE=0:CHANGED=0:NTSAVD=0:STNSAVD=0
8100 GOTO 1030
8996 REM +------------------+
8997 REM | ´C´opy character |
8998 REM +------------------+
9000 CURSOR TO 22,1:POKE 22,0
9010 PRINT "Copy from which character (0-255) ";CL$;:POKE 22,0
9030 INPUT C$
9035 GOSUB 12000:REM remove spaces
9040 IF C$="" THEN CURSOR TO 22,1:PRINT CL$;:POKE 22,0:GOTO 1110
9045 ER$="Enter max. 3 digits"
9050 IF LEN(C$)>3 THEN GOSUB 11000:GOTO 9000
9055 ER$="Illegal number"
9060 FOR I=1 TO LEN(C$)
9070 : Q$=MID$(C$,I,1)
9080 : IF Q$<"0" OR Q$>"9" THEN GOSUB 11000:I=LEN(C$):NEXT:GOTO 9000
9090 NEXT I
9091 ER$="Number out of range, must be between 0 and"+STR$(MAX)
```

```
 9094 IF VAL(C$)>MAX OR VAL(C$)<0 THEN GOSUB 11000:GOTO 9000
 9095 ER$="Number is equal to current char. number"
 9096 IF VAL(C$)=TE THEN GOSUB 11000:GOTO 9000
 9100 FOR I=0 TO HI
 9110 : B=PEEK(BASE+VAL(C$)*16+I)
 9120 : POKE (BASE+TE*16+I),B
 9130 NEXT I:CHANGED=1:NTSAVD=1
 9140 CURSOR TO 22,1:PRINT CL$;:POKE 22,0
 9150 GOSUB 11100
 9160 GOTO 5000
 9997 REM +------------------------------+
 9998 REM | Show (empty) character matrix |
 9999 REM +------------------------------+
10000 CURSOR TO 1,40
10001 PRINT CHR$(27);"F";:REM graphic mode on
10002 PRINT "P";
10005 FOR J=0 TO WI-1
10010 : PRINT "XXW";
10015 NEXT J
10017 PRINT "XXV"
10020 FOR I=0 TO HI
10025 CURSOR TO 2+2*I,40
10030 : FOR J=0 TO WI
10035 :   PRINT "Y  ";
10040 : NEXT J
10042 PRINT "Y"
10044 CURSOR TO 3+2*I,40
10046 IF I=HI THEN 10080
10048 : PRINT "QXX";
10050 : FOR J=1 TO WI
10055 :   PRINT "ZXX";
10060 : NEXT J
10070 : PRINT "U"
10080 NEXT I
10085 PRINT "R";
10090 FOR J=0 TO WI-1
10095 : PRINT "XXS";
10100 NEXT J
10105 PRINT "XXT"
10115 PRINT CHR$(27);"G":REM graphic mode off
10160 RETURN
10999 REM -------> ERROR MESSAGES <---------
11000 CURSOR TO 23,1:PRINT ER$;CL$:RETURN
11099 REM -----> ERASE ERROR MESSAGES <-----
11100 CURSOR TO 23,1:PRINT CL$:RETURN
12000 REM remove leading and trailing spaces in filename
12010 IF LEFT$(C$,1)=" "  AND LEN(C$)>1 THEN C$=MID$(C$,2):GOTO 12010
12020 IF RIGHT$(C$,1)=" " AND LEN(C$)>1 THEN C$=LEFT$(C$,LEN(C$)-1):GOTO 12020
12030 RETURN
13000 REM ---> convert decimal DC to hexadecimal HX$ <----
13010 HX$="0123456789ABCDEF"
13200 H1=INT(DC/4096):D=DC-4096*H1:L1=INT(D/256):D=D-256*L1
13220 H0=INT(D/16):D=D-16*H0:L0=D+1
13230 H1=H1+1:H0=H0+1:L1=L1+1
13240 HX$=MID$(HX$,H1,1)+MID$(HX$,L1,1)+MID$(HX$,H0,1)+MID$(HX$,L0,1)
13250 RETURN
13999 REM ---> get filename <----
14000 CURSOR TO 22,1:PRINT QU$;CL$;:POKE 22,0
14005 INPUT C$
14010 GOSUB 12000:REM remove spaces
14020 IF C$="" THEN 14090
14041 ER$="Filename too long, max. 14 char."
14042 IF LEN(C$)>14 THEN GOSUB 11000:GOTO 14000
14045 ER$="Illegal character in filename"
```

```
14050 FOR I=1 TO LEN(C$)
14060 : Q$=MID$(C$,I,1)
14070 : IF Q$=" " OR Q$="," THEN GOSUB 11000:I=LEN(C$):NEXT:GOTO 14000
14080 NEXT I
14090 RETURN
21996 REM +------------------+
21997 REM |  CURSOR ROUTINE  |
21998 REM +------------------+
22000 DX=0:DY=0:DA=0
22005 POKE &AAC0,(AX AND 127):REM disable ^S^Q
22010 GET DA$:IF DA$="" THEN RETURN
22015 DA=ASC(DA$)         :REM ** DOS65 ED codes **
22020 IF DA=5  THEN DY=-2:REM **   ^E is up     **
22040 IF DA=24 THEN DY= 2:REM **   ^X is down   **
22050 IF DA=19 THEN DX=-3:REM **   ^S is left   **
22070 IF DA=4  THEN DX= 3:REM **   ^D is right  **
22080 IF DX=0 AND DY=0 THEN RETURN
22085 GOSUB 11100:REM erase message line
22090 REM move cursor (with wrap-around)
22120 CURSOR TO CY,CX
22121 CHAR(WI-(CX-X0)/3,(CY-Y0)/2)=OC
22125 IF OC=1 THEN GOSUB 1400: REM leave a dot here
22127 IF OC=0 THEN GOSUB 1450: REM leave a space here
22130 CX=CX+DX:CY=CY+DY
22140 IF (CX-X0)/3 > WI THEN CX=X0:CY=CY+2
22145 IF (CX-X0)/3 < 0  THEN CX=X0+WI*3:CY=CY-2
22150 IF (CY-Y0)/2 > HI THEN CY=Y0
22155 IF (CY-Y0)/2 < 0  THEN CY=Y0+2*HI
22160 OC=CHAR(WI-(CX-X0)/3,(CY-Y0)/2)
22180 RETURN
```

```
    1    ;=========================================================================
    2    ; PRCHAR
    3    ; Print character set in $6000-$6FFF in bit-image mode on matrix-printer,
    4    ; used as subroutine in CHARED.BAS program. Can also be used standalone.
    5    ;
    6    ; char.data in the ELEKTOR 6845/6545 Video-card EPROM is stored as follows:
    7    ;
    8    ;   B B B B B B B B
    9    ;   7 6 5 4|3 2 1 0
   10    ;   . . . .|. . . . = 00  1
   11    ;   . X X X|X X . . = 7C  2
   12    ;   . . X .|. . X . = 22  3
   13    ;   . . X .|. . X . = 22  4
   14    ;   . . X X|X X . . = 3C  5
   15    ;   . . X .|. . X . = 22  6
   16    ;   . . X .|. . X . = 22  7
   17    ;   . X X X|X X . . = 7C  8
   18    ;   . . . .|. . . . = 00  9
   19    ;   . . . .|. . . . = 00 10
   20    ;
   21    ; 10 subsequent bytes form one char. Next char. is 16 bytes further.
   22    ;
   23    ; For bit image printing on my printer (COSMOS-80) the data is send in
   24    ; a string:
   25    ;
   26    ; <ESC> K <n1> <n2> <data>
   27    ;
   28    ; <ESC> K to enter bit-image mode in normal density
   29    ; <n1>    is low order byte of data counter
   30    ; <n2>    is high order byte
   31    ; <data>  is bit image data (max. 640 bytes)
   32    ;
```

```
33    ;                B7  B6  B5  B4  B3  B2  B1  B0
34    ; DOT             |   |   |   |   |   |   |   |
35    ;  . <-----------+   |   |   |   |   |   |   |
36    ;  . <---------------+   |   |   |   |   |   |
37    ;  . <-------------------+   |   |   |   |   |
38    ;  . <-----------------------+   |   |   |   |
39    ;  . <---------------------------+   |   |   |
40    ;  . <-------------------------------+   |   |      1st line
41    ;  . <-----------------------------------+   |
42    ;  . <---------------------------------------+
43    ;  . <------------------------------------- B7   2nd line
44    ;  . <------------------------------------- B6
45    ;
46    ; So the bit-image data is composed of all bits of one char. with the same
47    ; weight, e.g. all B7 combined form the first column of the bit-image
48    ; representation of the char. Because the char. matrix is 10 x 8, two extra
49    ; bits are packed on the next line. They come from the 9th and 10th byte.
50    ; Prior to printing the line-spacing is set to 21/216 inch (dots connect with
51    ; next line) All printer strings are in tables, so easy to change for other
52    ; printers. The characters are seperated on the line with 2 empty columns.
53    ;=============================================================================
54    PRINIT equ     $F0F4            ;printer initialisation routine
55    PROUT  equ     $F88C            ;printer output routine
56    PRTEXT equ     $F79D            ;print string on screen
57    ESC    equ     $1B              ;ESC code
58    LF     EQU     $0A              ;LINEFEED code
59    CR     EQU     $0D
60    FF     EQU     $0C              ;FORMFEED
61    PTR    EQU     $E0
62    MAX    EQU     640              ;max. number of dots/line
63    CHRPLIN        EQU     MAX/10   ;number of char./line (8 + 2 columns)
64
65           org     $A000
66
67    START  jmp     PRCHAR
68    STRINGS        equ     $
69    LINESP fcc     ESC,'3',21,0     ;line spacing string
70    BITIMG fcc     ESC,'K'          ;bit-image string
71           fdb     MAX
72           fcc     0
73    RESPR  fcc     ESC,'0',LF,0     ;reset line-spacing to 1/8 inch
74    ;
75    PRCHAR LDA     #0               ;set pointer to start ($6000)
76           LDY     #$60             ;of char.set data
77           STA     PTR
78           STY     PTR+1
79           STA     LINECNT
80           JSR     PRINIT           ;init printer output
81           BCC     CONT             ;printer ok
82    NOPRINT        JSR     PRTEXT   ;else send error message
83           fcc     'Printer not present or off-line\n\r',0
84           RTS                      ;and return to caller
85    CONT   LDX     #LINESP-STRINGS  ;define line spacing
86           JSR     SNDPRIN          ;send string to printer
87    NXTLIN LDX     #BITIMG-STRINGS  ;enter bit-image mode
88           JSR     SNDPRIN
89           LDA     PTR
90           LDY     PTR+1
91           STA     PSAV
92           STY     PSAV+1
93           LDA     #0
94           STA     COUNT
95    FIRSTLN        LDX     #0       ;compose first line
96           LDA     #%10000000       ;start with B7
```

```
 97          STA     MASK
 98   NXTBlT LDY     #0              ;clear bit-image data
 99          STY     DATA
100   GETDATA        LDA    [PTR],Y ;get char. data
101          AND     MASK            ;get bit
102          SEC
103          BNE     SHFT            ;if set shift in '1'
104          CLC                     ;else '0'
105   SHFT   ROL     DATA
106          lNY
107          CPY     #8              ;for all 8 bytes?
108          BNE     GETDATA         ;no, keep going
109          LDA     DATA            ;get bit-image data
110          JSR     PROUT           ;send to printer
111          LSR     MASK            ;point to next column
112          lNX
113          CPX     #8              ;do this 8 times
114          BNE     NXTBlT
115          LDA     #0
116          JSR     PROUT           ;two empty columns to
117          JSR     PROUT           ;seperate characters
118          LDA     PTR
119          CLC
120          ADC     #16             ;next char. is 16 bytes away
121          STA     PTR
122          BCC     NlNC
123          lNC     PTR+1
124   NlNC   lNC     COUNT
125          LDA     COUNT
126          CMP     #CHRPLIN        ;number of char./line
127          BNE     FlRSTLN
128          LDA     #LF             ;send LF to empty buffer
129          JSR     PROUT
130          LDA     PSAV            ;restore pointer
131          LDY     PSAV+1
132          STA     PTR
133          STY     PTR+1
134          LDX     #BlTlMG-STRlNGS ;enter bit-image mode
135          JSR     SNDPRlN
136          LDA     #0
137          STA     COUNT
138   SCNDLN LDX     #0              ;compose second line
139          LDA     #%10000000      ;start with B7
140          STA     MASK
141          LDA     PTR
142          LDY     PTR+1
143          STA     PSAV
144          STY     PSAV+1
145   NXTBlTT        LDY     #8      ;clear bit-image data
146          LDA     #0
147          STA     DATA
148   GETDAT LDA     [PTR],Y         ;get char. data
149          AND     MASK            ;get bit
150          SEC
151          BNE     SHFTT           ;if set shift in '1'
152          CLC                     ;else '0'
153   SHFTT  ROL     DATA
154          lNY
155          CPY     #10             ;for next 2 bytes (8+2 = 10)?
156          BNE     GETDAT          ;no, keep going
157          LDY     #6
158   MVlN   CLC
159          ROL     DATA
160          DEY
```

```
161             BNE     MVIN
162             LDA     DATA            ;get bit-image data
163             JSR     PROUT           ;send to printer
164             LSR     MASK            ;point to next column
165             INX
166             CPX     #8              ;do this 8 times
167             BNE     NXTBITT
168             LDA     #0
169             JSR     PROUT           ;two empty columns to
170             JSR     PROUT           ;seperate characters
171             LDA     PTR
172             CLC
173             ADC     #16             ;next char. is 16 bytes away
174             STA     PTR
175             BCC     NOINC
176             INC     PTR+1
177     NOINC   INC     COUNT
178             LDA     COUNT
179             CMP     #CHRPLIN        ;number of char./line
180             BNE     SCNDLN
181             LDA     #LF             ;send LF to empty buffer
182             JSR     PROUT
183             INC     LINECNT
184             LDA     LINECNT
185             CMP     #(256/CHRPLIN)  ;number of lines to print all 256 char.
186             BEQ     EXIT
187             JMP     NXTLIN
188             LDA     #CR             ;send CR
189             JSR     PROUT
190     EXIT    LDX     #RESPR-STRINGS  ;reset printer
191             JSR     SNDPRIN         ;send string to printer
192             RTS
193     ;
194     SNDPRIN         LDA     STRINGS,X       ;send string until NULL
195             BEQ     SNDEX
196             JSR     PROUT
197             INX
198             BNE     SNDPRIN         ;branch always
199     SNDEX   RTS
200     ;
201     PSAV    fcc     0,0
202     DATA    fcc     0
203     MASK    fcc     0
204     COUNT   fcc     0
205     LINECNT         fcc     0
206             END     START
```

Addr	01	23	45	67	89	AB	CD	EF
0000	0000	0000	000f	0f0f	0f0f	0000	0000	0000
0010	0000	0000	00f0	f0f0	f0f0	0000	0000	0000
0020	0000	0000	00ff	ffff	ffff	0000	0000	0000
0030	0012	2449	1224	4912	2400	0000	0000	0000
0040	0010	0804	3e10	0804	0000	0000	0000	0000
0050	0000	081c	3e1c	0800	0000	0000	0000	0000
0060	0000	0000	1c1c	0000	0000	0000	0000	0000
0070	0070	4870	4874	0404	0700	0000	0000	0000
0080	0070	4870	4f78	0601	0e00	0000	0000	0000
0090	0048	4878	485f	0404	0400	0000	0000	0000
00a0	0040	4040	4f78	0c08	0800	0000	0000	0000
00b0	0044	4428	101f	0404	0400	0000	0000	0000
00c0	0078	4060	4f48	0c08	0800	0000	0000	0000
00d0	0038	4040	3e09	0e0a	0900	0000	0000	0000
00e0	0038	4030	0876	0909	0600	0000	0000	0000
00f0	0038	4030	0f72	0202	0700	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0100	0000	0000	0f08	0808	0808	0808	0808	0808
0110	0808	0808	0f08	0808	0808	0808	0808	0808
0120	0808	0808	0f00	0000	0000	0000	0000	0000
0130	0808	0808	ff00	0000	0000	0000	0000	0000
0140	0808	0808	f800	0000	0000	0000	0000	0000
0150	0808	0808	f808	0808	0808	0808	0808	0808
0160	0000	0000	f808	0808	0808	0808	0808	0808
0170	0000	0000	ff08	0808	0808	0808	0808	0808
0180	0000	0000	ff00	0000	0000	0000	0000	0000
0190	0808	0808	0808	0808	0808	0808	0808	0808
01a0	0808	0808	ff08	0808	0808	0808	0808	0808
01b0	0078	4070	4778	0601	0e00	0000	0000	0000
01c0	0000	0000	00ff	ffff	ffff	ff00	0000	0000
01d0	0030	4058	4f38	0601	0e00	0000	0000	0000
01e0	0000	00ff	ffff	0000	0000	0000	0000	0000
01f0	ffff	ff00	0000	0000	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0200	0000	0000	0000	0000	0000	0000	0000	0000
0210	0008	0808	0800	0808	0000	0000	0000	0000
0220	0024	2424	0000	0000	0000	0000	0000	0000
0230	0024	247e	247e	2424	0000	0000	0000	0000
0240	0008	3f48	3e09	7e08	0000	0000	0000	0000
0250	0061	6204	0810	2343	0000	0000	0000	0000
0260	0038	4428	1029	4639	0000	0000	0000	0000
0270	000c	0c08	1000	0000	0000	0000	0000	0000
0280	0004	0810	1010	0804	0000	0000	0000	0000
0290	0010	0804	0404	0810	0000	0000	0000	0000
02a0	0000	082a	1c2a	0800	0000	0000	0000	0000
02b0	0000	0808	7f08	0800	0000	0000	0000	0000
02c0	0000	0000	0000	1818	1020	0000	0000	0000
02d0	0000	0000	7f00	0000	0000	0000	0000	0000
02e0	0000	0000	0000	1818	0000	0000	0000	0000
02f0	0000	0204	0810	2040	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0300	003c	464e	5a72	623c	0000	0000	0000	0000
0310	0008	1828	0808	083e	0000	0000	0000	0000
0320	003c	4202	1c20	407e	0000	0000	0000	0000
0330	003c	4202	1c02	423c	0000	0000	0000	0000
0340	0004	0c14	247e	0404	0000	0000	0000	0000
0350	007e	407c	0202	423c	0000	0000	0000	0000
0360	001c	2040	7c42	423c	0000	0000	0000	0000
0370	007e	4204	0810	1010	0000	0000	0000	0000
0380	003c	4242	3c42	423c	0000	0000	0000	0000
0390	003c	4242	3e02	023c	0000	0000	0000	0000
03a0	0000	0018	1800	1818	0000	0000	0000	0000
03b0	0000	0018	1800	1818	1020	0000	0000	0000
03c0	0008	1020	4020	1008	0000	0000	0000	0000
03d0	0000	003e	003e	0000	0000	0000	0000	0000
03e0	0008	0402	0102	0408	0000	0000	0000	0000
03f0	001e	2101	0608	0008	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0400	001e	214d	555e	403e	0000	0000	0000	0000
0410	0018	2442	7e42	4242	0000	0000	0000	0000
0420	007c	2222	3c22	227c	0000	0000	0000	0000
0430	001c	2240	4040	221c	0000	0000	0000	0000
0440	007c	2222	2222	227c	0000	0000	0000	0000
0450	007e	4040	7840	407e	0000	0000	0000	0000
0460	007e	4040	7840	4040	0000	0000	0000	0000
0470	003c	4240	404e	423c	0000	0000	0000	0000
0480	0042	4242	7e42	4242	0000	0000	0000	0000
0490	001c	0808	0808	081c	0000	0000	0000	0000
04a0	000e	0404	0404	4438	0000	0000	0000	0000
04b0	0042	4448	5068	4442	0000	0000	0000	0000
04c0	0040	4040	4040	407e	0000	0000	0000	0000
04d0	0041	6355	4941	4141	0000	0000	0000	0000
04e0	0042	6252	4a46	4242	0000	0000	0000	0000
04f0	001c	2241	4141	221c	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0500	007c	4242	7c40	4040	0000	0000	0000	0000
0510	001c	2241	4945	221d	0000	0000	0000	0000
0520	007c	4242	7c48	4442	0000	0000	0000	0000
0530	003c	4240	3c02	423c	0000	0000	0000	0000
0540	007f	0808	0808	0808	0000	0000	0000	0000
0550	0042	4242	4242	423c	0000	0000	0000	0000
0560	0041	4122	2214	1408	0000	0000	0000	0000
0570	0041	4141	4149	5522	0000	0000	0000	0000
0580	0041	2214	0814	2241	0000	0000	0000	0000
0590	0041	2214	0808	0808	0000	0000	0000	0000
05a0	007f	0204	0810	207f	0000	0000	0000	0000
05b0	003c	2020	2020	203c	0000	0000	0000	0000
05c0	0000	4020	1008	0402	0000	0000	0000	0000
05d0	003c	0404	0404	043c	0000	0000	0000	0000
05e0	0008	1422	0000	0000	0000	0000	0000	0000
05f0	0000	0000	0000	007f	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0600	0018	1808	0400	0000	0000	0000	0000	0000
0610	0000	003c	023e	423d	0000	0000	0000	0000
0620	0040	405c	6242	625c	0000	0000	0000	0000
0630	0000	003c	4240	423c	0000	0000	0000	0000
0640	0002	023a	4642	463a	0000	0000	0000	0000
0650	0000	003c	427e	403c	0000	0000	0000	0000
0660	000c	1210	7c10	1010	0000	0000	0000	0000
0670	0000	003a	4642	463a	023c	0000	0000	0000
0680	0040	405c	6242	4242	0000	0000	0000	0000
0690	0008	0018	0808	081c	0000	0000	0000	0000
06a0	0002	0006	0202	0202	221c	0000	0000	0000
06b0	0040	4048	5068	4442	0000	0000	0000	0000
06c0	0018	0808	0808	081c	0000	0000	0000	0000
06d0	0000	0076	4949	4949	0000	0000	0000	0000
06e0	0000	005c	6242	4242	0000	0000	0000	0000
06f0	0000	003c	4242	423c	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0700	0000	005c	6242	625c	4040	0000	0000	0000
0710	0000	003a	4642	463a	0202	0000	0000	0000
0720	0000	005c	6240	4040	0000	0000	0000	0000
0730	0000	003e	6018	067c	0000	0000	0000	0000
0740	0000	107c	1010	120c	0000	0000	0000	0000
0750	0000	0042	4242	463a	0000	0000	0000	0000
0760	0000	0041	4122	1408	0000	0000	0000	0000
0770	0000	0041	4949	4936	0000	0000	0000	0000
0780	0000	0042	2418	2442	0000	0000	0000	0000
0790	0000	0042	4242	463a	023c	0000	0000	0000
07a0	0000	007e	0418	207e	0000	0000	0000	0000
07b0	000c	1010	2010	100c	0000	0000	0000	0000
07c0	0008	0808	0008	0808	0000	0000	0000	0000
07d0	0018	0404	0204	0418	0000	0000	0000	0000
07e0	0030	4906	0000	0000	0000	0000	0000	0000
07f0	00ff	ffff	ffff	ffff	0000	0000	0000	0000

Addr	01	23	45	67	89	AB	CD	EF
0800	ffff	ffff	fff0	f0f0	f0f0	ffff	ffff	ffff
0810	ffff	ffff	ff0f	0f0f	0f0f	ffff	ffff	ffff
0820	ffff	ffff	ff00	0000	0000	ffff	ffff	ffff
0830	ffed	dbb6	eddb	b6ed	dbff	ffff	ffff	ffff
0840	ffef	f7fb	c1ef	f7fb	ffff	ffff	ffff	ffff
0850	ffff	f7e3	c1e3	f7ff	ffff	ffff	ffff	ffff
0860	ffff	ffff	e3e3	ffff	ffff	ffff	ffff	ffff
0870	ff8f	b78f	b78b	fbfb	f8ff	ffff	ffff	ffff
0880	ff8f	b78f	b087	f9fe	f1ff	ffff	ffff	ffff
0890	ffb7	b787	b7a0	fbfb	fbff	ffff	ffff	ffff
08a0	ffbf	bfbf	b087	f3f7	f7ff	ffff	ffff	ffff
08b0	ffbb	bbd7	efe0	fbfb	fbff	ffff	ffff	ffff
08c0	ff87	bf9f	b0b7	f3f7	f7ff	ffff	ffff	ffff
08d0	ffc7	bfbf	c1f6	f1f5	f6ff	ffff	ffff	ffff
08e0	ffc7	bfcf	f789	f6f6	f9ff	ffff	ffff	ffff
08f0	ffc7	bfcf	f08d	fdfd	f8ff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0900	ffff	ffff	f0f7	f7f7	f7f7	f7f7	f7f7	f7f7
0910	f7f7	f7f7	f0f7	f7f7	f7f7	f7f7	f7f7	f7f7
0920	f7f7	f7f7	f0ff	ffff	ffff	ffff	ffff	ffff
0930	f7f7	f7f7	00ff	ffff	ffff	ffff	ffff	ffff
0940	f7f7	f7f7	07ff	ffff	ffff	ffff	ffff	ffff
0950	f7f7	f7f7	07f7	f7f7	f7f7	f7f7	f7f7	f7f7
0960	ffff	ffff	07f7	f7f7	f7f7	f7f7	f7f7	f7f7
0970	ffff	ffff	00f7	f7f7	f7f7	f7f7	f7f7	f7f7
0980	ffff	ffff	00ff	ffff	ffff	ffff	ffff	ffff
0990	f7f7	f7f7	f7f7	f7f7	f7f7	f7f7	f7f7	f7f7
09a0	f7f7	f7f7	00f7	f7f7	f7f7	f7f7	f7f7	f7f7
09b0	ff87	bf8f	b887	f9fe	f1ff	ffff	ffff	ffff
09c0	ffff	ffff	ff00	0000	0000	00ff	ffff	ffff
09d0	ffcf	bfa7	b0c7	f9fe	f1ff	ffff	ffff	ffff
09e0	ffff	ff00	0000	ffff	ffff	ffff	ffff	ffff
09f0	0000	00ff	ffff	ffff	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0a00	ffff	ffff	ffff	ffff	ffff	ffff	ffff	ffff
0a10	fff7	f7f7	f7ff	f7f7	ffff	ffff	ffff	ffff
0a20	ffdb	dbdb	ffff	ffff	ffff	ffff	ffff	ffff
0a30	ffdb	db81	db81	dbdb	ffff	ffff	ffff	ffff
0a40	fff7	c0b7	c1f6	81f7	ffff	ffff	ffff	ffff
0a50	ff9e	9dfb	f7ef	dcbc	ffff	ffff	ffff	ffff
0a60	ffc7	bbd7	efd6	b9c6	ffff	ffff	ffff	ffff
0a70	fff3	f3f7	efff	ffff	ffff	ffff	ffff	ffff
0a80	fffb	f7ef	efef	f7fb	ffff	ffff	ffff	ffff
0a90	ffef	f7fb	fbfb	f7ef	ffff	ffff	ffff	ffff
0aa0	ffff	f7d5	e3d5	f7ff	ffff	ffff	ffff	ffff
0ab0	ffff	f7f7	80f7	f7ff	ffff	ffff	ffff	ffff
0ac0	ffff	ffff	ffff	e7e7	efdf	ffff	ffff	ffff
0ad0	ffff	ffff	80ff	ffff	ffff	ffff	ffff	ffff
0ae0	ffff	ffff	ffff	e7e7	ffff	ffff	ffff	ffff
0af0	ffff	fdfb	f7ef	dfbf	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0b00	ffc3	b9b1	a58d	9dc3	ffff	ffff	ffff	ffff
0b10	fff7	e7d7	f7f7	f7c1	ffff	ffff	ffff	ffff
0b20	ffc3	bdfd	e3df	bf81	ffff	ffff	ffff	ffff
0b30	ffc3	bdfd	e3fd	bdc3	ffff	ffff	ffff	ffff
0b40	fffb	f3eb	db81	fbfb	ffff	ffff	ffff	ffff
0b50	ff81	bf83	fdfd	bdc3	ffff	ffff	ffff	ffff
0b60	ffe3	dfbf	83bd	bdc3	ffff	ffff	ffff	ffff
0b70	ff81	bdfb	f7ef	efef	ffff	ffff	ffff	ffff
0b80	ffc3	bdbd	c3bd	bdc3	ffff	ffff	ffff	ffff
0b90	ffc3	bdbd	c1fd	fdc3	ffff	ffff	ffff	ffff
0ba0	ffff	ffe7	e7ff	e7e7	ffff	ffff	ffff	ffff
0bb0	ffff	ffe7	e7ff	e7e7	efdf	ffff	ffff	ffff
0bc0	fff7	efdf	bfdf	eff7	ffff	ffff	ffff	ffff
0bd0	ffff	ffc1	ffc1	ffff	ffff	ffff	ffff	ffff
0be0	fff7	fbfd	fefd	fbf7	ffff	ffff	ffff	ffff
0bf0	ffe1	defe	f9f7	fff7	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0c00	ffe1	deb2	aaa1	bfc1	ffff	ffff	ffff	ffff
0c10	ffe7	dbbd	81bd	bdbd	ffff	ffff	ffff	ffff
0c20	ff83	dddd	c3dd	dd83	ffff	ffff	ffff	ffff
0c30	ffe3	ddbf	bfbf	dde3	ffff	ffff	ffff	ffff
0c40	ff83	dddd	dddd	dd83	ffff	ffff	ffff	ffff
0c50	ff81	bfbf	87bf	bf81	ffff	ffff	ffff	ffff
0c60	ff81	bfbf	87bf	bfbf	ffff	ffff	ffff	ffff
0c70	ffc3	bdbf	bfb1	bdc3	ffff	ffff	ffff	ffff
0c80	ffbd	bdbd	81bd	bdbd	ffff	ffff	ffff	ffff
0c90	ffe3	f7f7	f7f7	f7e3	ffff	ffff	ffff	ffff
0ca0	fff1	fbfb	fbfb	bbc7	ffff	ffff	ffff	ffff
0cb0	ffbd	bbb7	af97	bbbd	ffff	ffff	ffff	ffff
0cc0	ffbf	bfbf	bfbf	bf81	ffff	ffff	ffff	ffff
0cd0	ffbe	9caa	b6be	bebe	ffff	ffff	ffff	ffff
0ce0	ffbd	9dad	b5b9	bdbd	ffff	ffff	ffff	ffff
0cf0	ffe3	ddbe	bebe	dde3	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0d00	ff83	bdbd	83bf	bfbf	ffff	ffff	ffff	ffff
0d10	ffe3	ddbe	b6ba	dde2	ffff	ffff	ffff	ffff
0d20	ff83	bdbd	83b7	bbbd	ffff	ffff	ffff	ffff
0d30	ffc3	bdbf	c3fd	bdc3	ffff	ffff	ffff	ffff
0d40	ff80	f7f7	f7f7	f7f7	ffff	ffff	ffff	ffff
0d50	ffbd	bdbd	bdbd	bdc3	ffff	ffff	ffff	ffff
0d60	ffbe	bedd	ddeb	ebf7	ffff	ffff	ffff	ffff
0d70	ffbe	bebe	beb6	aadd	ffff	ffff	ffff	ffff
0d80	ffbe	ddeb	f7eb	ddbe	ffff	ffff	ffff	ffff
0d90	ffbe	ddeb	f7f7	f7f7	ffff	ffff	ffff	ffff
0da0	ff80	fdfb	f7ef	df80	ffff	ffff	ffff	ffff
0db0	ffc3	dfdf	dfdf	dfc3	ffff	ffff	ffff	ffff
0dc0	ffff	bfdf	eff7	fbfd	ffff	ffff	ffff	ffff
0dd0	ffc3	fbfb	fbfb	fbc3	ffff	ffff	ffff	ffff
0de0	fff7	ebdd	ffff	ffff	ffff	ffff	ffff	ffff
0df0	ffff	ffff	ffff	ff80	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0e00	ffe7	e7f7	fbff	ffff	ffff	ffff	ffff	ffff
0e10	ffff	ffc3	fdc1	bdc2	ffff	ffff	ffff	ffff
0e20	ffbf	bfa3	9dbd	9da3	ffff	ffff	ffff	ffff
0e30	ffff	ffc3	bdbf	bdc3	ffff	ffff	ffff	ffff
0e40	fffd	fdc5	b9bd	b9c5	ffff	ffff	ffff	ffff
0e50	ffff	ffc3	bd81	bfc3	ffff	ffff	ffff	ffff
0e60	fff3	edef	83ef	efef	ffff	ffff	ffff	ffff
0e70	ffff	ffc5	b9bd	b9c5	fdc3	ffff	ffff	ffff
0e80	ffbf	bfa3	9dbd	bdbd	ffff	ffff	ffff	ffff
0e90	fff7	ffe7	f7f7	f7e3	ffff	ffff	ffff	ffff
0ea0	fffd	fff9	fdfd	fdfd	dde3	ffff	ffff	ffff
0eb0	ffbf	bfb7	af97	bbbd	ffff	ffff	ffff	ffff
0ec0	ffe7	f7f7	f7f7	f7e3	ffff	ffff	ffff	ffff
0ed0	ffff	ff89	b6b6	b6b6	ffff	ffff	ffff	ffff
0ee0	ffff	ffa3	9dbd	bdbd	ffff	ffff	ffff	ffff
0ef0	ffff	ffc3	bdbd	bdc3	ffff	ffff	ffff	ffff

Addr	01	23	45	67	89	AB	CD	EF
0f00	ffff	ffa3	9dbd	9da3	bfbf	ffff	ffff	ffff
0f10	ffff	ffc5	b9bd	b9c5	fdfd	ffff	ffff	ffff
0f20	ffff	ffa3	9dbf	bfbf	ffff	ffff	ffff	ffff
0f30	ffff	ffc1	9fe7	f983	ffff	ffff	ffff	ffff
0f40	ffff	ef83	efef	edf3	ffff	ffff	ffff	ffff
0f50	ffff	ffbd	bdbd	b9c5	ffff	ffff	ffff	ffff
0f60	ffff	ffbe	bedd	ebf7	ffff	ffff	ffff	ffff
0f70	ffff	ffbe	b6b6	b6c9	ffff	ffff	ffff	ffff
0f80	ffff	ffbd	dbe7	dbbd	ffff	ffff	ffff	ffff
0f90	ffff	ffbd	bdbd	b9c5	fdc3	ffff	ffff	ffff
0fa0	ffff	ff81	fbe7	df81	ffff	ffff	ffff	ffff
0fb0	fff3	efef	dfef	eff3	ffff	ffff	ffff	ffff
0fc0	fff7	f7f7	fff7	f7f7	ffff	ffff	ffff	ffff
0fd0	ffe7	fbfb	fdfb	fbe7	ffff	ffff	ffff	ffff
0fe0	ffcf	b6f9	ffff	ffff	ffff	ffff	ffff	ffff
0ff0	ff00	0000	0000	0000	ffff	ffff	ffff	ffff

Programmeerbare Logica, wat is dat? (Deel 2).

Door: Nico de Vries

In deel 1 hebben we kennis gemaakt met de beginselen van Booleaanse algebra, en de ontstaansgeschiedenis van de oudste soort programmeerbare logica: PALs. De ontwikkelingen hebben echter niet stil gestaan. In dit deel worden dan ook de laatste ontwikkelingen en de andere soorten programmeerbare logica bekeken.

6. Wat worden die krengen heet!

De in het vorige deel van dit verhaal beschreven PALs hadden niet alleen voordelen, maar ook nadelen. Zo zijn deze PALs gemaakt in zogenaamde bipolaire techniek. Dit heeft tot gevolg dat de PALs redelijk snel zijn: de standaardtypen hebben een doorlooptijd van 35 ns (ongeveer 4 LS-poorten). Bij de allersnelste typen loopt dit terug tot 10 ns. Bipolaire techniek heeft ook een nadeel: stroomverbruik. Standaard PALs nemen 180 tot 210 mA uit de voeding op: 1 Watt dissipatie! Deze PALs worden in bedrijf behoorlijk heet.

Een aantal typen zijn dan ook leverbaar met een lager stroomverbruik. Dit gaat echter ten koste van de snelheid. Dit probleem wordt ten dele opgelost door het combineren van snellere ICs opleverende technieken met het vergroten van de interne pull-up weerstanden. De PALs blijven dan even snel, maar gebruiken minder stroom. Op deze manier is men erin geslaagd, om PALs te maken die minder dan 50 mA verbruiken, maar die toch de standaard snelheidsspecificaties bezitten.

Een andere manier ter verkleining van het stroomverbruik is om PALs te maken in CMOS-techniek. Dit heeft echter wat langer geduurd dan oorspronkelijk voorzien: in CMOS is het moeilijk om 'bipolaire' snelheden te halen, terwijl vooral het fabriceren van een betrouwbare fuse geen sinecure is. Een verder probleem was het grote aantal pull-up weerstanden in de PAL: deze gebruiken ook stroom als de ingangen niet veranderen. Zo'n PAL zou dus niet voldoen aan het CMOS-criterium: stilstand is gelijk aan geen voedingsstroom. Er zijn inmiddels PALs in CMOS verkrijgbaar die redelijke snelheidsspecificaties hebben. Echte zero-power standby PALs zijn echter schaars: deze meeste hebben een standby verbruik van ca. 100 uA.

De meeste CMOS PALs zijn gebaseerd op EPROM-technieken voor zover het de programmeerbare verbindingen betreft. Vaak kunnen deze typen dan ook in twee uitvoeringen verkregen worden: met en zonder ruitje, oftewel wis- en opnieuw programmeerbaar of eenmalig te programmeren.

7. Toeters en bellen.

In het voorgaande hebben we gezien, dat een PAL beperkt wordt door het aantal producttermen per uitgang, en door zijn architectuur, die vastligt in het type. Dit is er in principe de oorzaak van, dat er zoveel verschillende typen zijn: Monolithic Memories maakte op een gegeven moment wel 30 verschillende basistypen. Een stap in een andere richting is dan ook, ook de architectuur programmeerbaar te maken. Een eenvoudig voorbeeld daarvan waren de PALs met programmeerbare uitgangspolariteit. Maar het kan nog verder: zo zou je per uitgang moeten kunnen kiezen tussen registerwerking of niet, terugkoppeling vanaf de pin of vanaf de Q-niet uitgang van de flip-flop en dergelijke. Ook een groter aantal producttermen per uitgang levert een meer flexibele PAL op.

De eerste stap in deze richting werd gedaan door AMD. Zij ontwierpen de PAL22V10, een 24-pins PAL met 10 uitgangen, die 8 tot zelfs 16 producttermen hebben. Ook kan de gebruiker met een paar fuse-jes bepalen welke uitgangen registers bevatten en welke niet, en tevens is de uitgangspolariteit programmeerbaar. Het ontwerp van de PAL22V10 was zodanig uitgekiend, dat deze PAL zo ongeveer de standaard 24-pins PAL is geworden. Van deze PAL bestaat inmiddels ook een CMOS-versie. De EPLDs (Erasable Programmable Logic Device) van Intel en Altera vallen ook onder de familie logica met gedeeltelijk programmeerbare architectuur. EPLDs zijn gemaakt in CMOS, met EPROM eigenschappen: dus wisbaar met UV-licht.

Het ei van Columbus op dit gebied wordt wellicht aangeboden door Lattice. Zij produceren een tweetal ICs, die niet met PAL, maar met de weinig flatteuze naam GAL (Generic Array Logic) worden aangeduid. De GALs hebben net als de PAL22V10 een programmeerbare architectuur, die echter zodanig gemaakt is, dat men met een 20-pins en een 24-pins IC (respectievelijk de GAL16V8 en de GAL20V8) meer dan 25 verschillende typen PALs kan emuleren. Verder heeft Lattice de programmerfabrikanten zover kunnen krijgen, dat de programmer eventueel de omzetting van PAL naar GAL kan verzorgen. Dit houdt voor de gebruiker in, dat hij zonder meer van PALs over kan gaan op GALs. De gebruiker krijgt door het gebruik van GALs namelijk nog twee extra voordelen: GALs zijn gemaakt in CMOS-techniek, en gebruiken dus minder stroom dan PALs, en zij zijn gemaakt in een EEPROM-technologie, dat wil zeggen, GALs zijn electrisch wis- en programmeerbaar. Dit maakt een GAL ideaal voor prototypewerk: bij een ontwerpverandering gewoon

even opnieuw programmeren. Er hoeft zelfs niet op het wissen gewacht te worden. In het prototype van de virtual diskkaart met statische RAMs zit bijvoorbeeld een GAL20V8. De V in de GAL-nummers staat overigens voor 'versatile', een Engels woord voor veelzijdig.

8. Andere soorten programmeerbare logica.

Tot dusver hebben we gezien, dat een PAL een programmeerbaar AND-array heeft: de verbindingen naar de AND-poorten zijn programmeerbaar uitgevoerd. Het uitgangsdeel is vast bedraad en bevat de OR-poorten. Deze AND-OR volgorde volgt automatisch uit het feit dat iedere logische functie altijd kan worden uitgedrukt door een som van producten. Het kan natuurlijk ook anders.

Het omgekeerde, een vast AND-array en een programmeerbaar OR-array bestaat ook. Deze vorm is zelfs ouder en kennen we allemaal: de PROM. Een PROM (en dus ook een EPROM en een EEPROM) bestaat uit een adresdecoder (het vaste AND-array) en programmeerbare data (het programmeerbare OR-array). De adresaansluitingen zijn de ingangen, de data-aansluitingen de uitgangen. De PROM wordt echter relatief weinig voor logica ingezet, vanwege een probleempje. Dat probleem is niet snelheid, want er bestaan PROMs die 20 ns toegangstijd hebben. Het probleem is glitches. Stel er verandert een adreslijn (ingang), en de data op een bepaalde uitgang zou hierbij dezelfde moeten blijven. Dit gebeurt in principe ook, zij het, dat de adresdecoder in de PROM aan het werk gaat en een andere geheugencel gaat adresseren. Tijdens de verandering van de ene naar de andere geheugencel zitten we even in het niemandsland: de uitgang is ongedefinieerd. Hierdoor gaat de uitgang misschien even laag, of gaat even zweven. Zoiets wordt een glitch genoemd, en is inherent aan alle PROMs. De belangrijkste oorzaak van glitches zijn looptijdverschillen op de chip zelf. Een voorbeeld van een PROM en een EPROM toegepast als logische bouwsteen vindt men in de EPROM-programmer: de pindecoder-EPROM en de VPP-decoder-PROM. Door zijn architectuur heeft de PROM geen beperking in het aantal producttermen per uitgang.

De combinatie van zowel een programmeerbaar AND-array en een programmeerbaar OR-array bestaat ook. Ze is ongeveer even oud als de eerste PAL en wordt FPLA (Field Programmable Logic Array) genoemd. De schepper van deze familie is Signetics. FPLAs hebben de eigenschap, dat een bepaalde productterm die in meerdere uitgangen voorkomt, maar 1 keer gemaakt hoeft te worden. Hierdoor kennen deze ICs in de praktijk geen producttermbeperkingen. FPLAs hebben vanaf het prille begin steeds programmeerbare uitgangspolariteit gehad. FPLAs met registers in de uitgangen bestaan ook: ze heten dan FPLS (Field Programmable Logic Sequencer). Deze architectuur lijkt het meest optimaal, maar is het toch niet. Ten eerste kan een ontwerp zodanig worden opgezet, dat het glitch-probleem weer de kop opsteekt, ofschoon men bij het ontwerp van de FPLAs zelf hieraan bijzondere aandacht heeft geschonken. Ten tweede heeft men te maken met twee arrays: AND en OR. Het signaal moet dus door twee arrays heen, waardoor de FPLA inherent trager is dan een PAL of een PROM. Een tweede nadeel van het dubbele array is, dat een relatief eenvoudige FPLA al gauw tweemaal zoveel fuses bevat dan een PAL van dezelfde grootte. Daar fuses veel chipruimte vereisen, en chipgrootte mede de snelheid bepaalt, is ook dit een oorzaak van snelheidsbeperkingen.

9. Hoe programmeer je PALs?

Er treedt bij PALs, in tegenstelling tot PROMs een bijzonder probleem op: je kunt het array niet rechtstreeks via de pinnen bereiken, althans niet op fuseniveau. Alle PALs, EPLDs, GALs en FPLAs moeten dan ook eerst in een speciale programmeerstand worden gebracht. Dit gebeurt zonder uitzondering door het aanleggen van spanningen hoger dan 5 Volt op bepaalde pinnen. Hierdoor veranderen de ingangen meestal in adres- en stuurlijnen, en via de uitgangen zijn dan meestal rechtstreeks de fuses in het array beschikbaar. Toch is hiermee het probleem nog niet helemaal opgelost: de PAL16L8 bijvoorbeeld, heeft 10 echte ingangen, maar 2048 fuses. 10 ingangen kunnen slechts 1024 fuses adresseren. Er moet dus ook nog gemultiplext worden. Bij FPLAs wordt dit probleem nog een factor groter.

Een tweede hinderpaal is de metalen fuse zelf. Er is sprake van een doorsmeltproces en dat kost energie. De typische programmeerstroom bij een PAL ligt dan ook bij ca. 750 mA (piek). Als je dus meerdere fuses tegelijk wilt programmeren (in verband de met snelheid van programmeren) moet de programmer in totaal al gauw Amperes kunnen leveren!

Die metalen fuse levert nog een ander probleem op. Omdat je die dingen maar 1 keer kunt programmeren, kun je in de fabriek niet kijken of de PAL wel goed zal programmeren. Het uur der waarheid slaat dus pas bij de eindgebruiker als hij de PAL programmeert. Vandaar dat de fabrikanten van programmeerbare logica die metalen fuses bevat, meestal niet een opbrengst van 100% maar van 95% tot 98% beloven.

In de programmeerstand kan van een logisch IC niet alleen het array rechtstreeks worden geprogrammeerd, maar ook worden uitgelezen. Men kan dus eenvoudig een duplicaat maken, door een IC eerst in te lezen, en vervolgens een blanco exemplaar met het uitgelezen array te programmeren.

Het niet rechtstreeks bereikbaar zijn van het array kan ook ten voordele gebruikt worden. Alle programmeerbare logische ICs zijn voorzien van een mogelijkheid om de programmeer/uitleesstand permanent te uit te schakelen. Dit wordt gedaan door een speciale fuse, die het functioneren van de fusedecoder lamlegt, waardoor het array niet meer gelezen kan worden. Hierdoor kan een buitenstaander niet meer eenvoudig een kopie van een PAL maken, anders dan door het aanbieden van allerlei ingangscombinaties en het bestuderen van de uitgangen. Deze bijzondere fuse wordt meestal de 'security fuse' genoemd en kan door een goede logicaprogrammer geprogrammeerd worden. Na zo'n actie kan de PAL dus niet meer zonder meer gecopieerd worden.

Het zal uit het bovenstaande duidelijk geworden zijn, dat het maken van een goede PAL-programmer geen sinecure is. Bij CMOS-ICs ligt de zaak meestal wat minder gevoelig omdat deze vaak wisbaar zijn, terwijl de vereiste spanningen en stromen ook bescheidener zijn. Het ingewikkelde adresseren blijft echter, vooral bij typen met programmeerbare architectuur.

10. Hoe ontwerp je nu een PAL?

Tot dusver hebben we het alleen over de PALs zelf gehad. In het eerste deel hebben we zelfs een eenvoudig ontwerpje 'met de hand' gedaan. Het zal echter een ieder duidelijk zijn, dat deze manier van werken met kruisjes zetten in een plattegrond geen gebruikersvriendelijke benadering van de PAL inhoudt. Vanaf het prille begin bestaat er dan ook software, die deze uitzoekerij van de ontwerper overneemt.

Het oudste programma voor ontwerpen van PALs komt van de uitvinder MMI en heet eenvoudig PALASM (assembler voor PALs). PALASM is in staat om van een genormaliseerde sourcefile met Booleaanse vergelijkingen een fuseplattegrond (meestal fuseplot genoemd) te genereren. Het werkt vrij eenvoudig: geef alle pinnen van de PAL die je gaat gebruiken een naam, en schrijf vervolgens in Booleaanse vergelijkingen de gewenste logische verbanden tussen de in- en uitgangen op. Dit levert voor iedere uitgang een vergelijking op. PALASM doet niet aan vereenvoudigingen, en verwerkt ook geen haakjes: de gebruiker moet de vergelijkingen dus helemaal uitwerken. Dit heeft als voordeel dat men van te voren kan zien of een ontwerp qua producttermen in een PAL past, maar als nadeel dat het ontwerp vrij veel voorbereidingstijd kan vergen.

Moderne assemblers voor logica kunnen echter veel meer. Zo accepteren ze vrijwel allemaal drie vormen van invoer: de vergelijking, de waarheidstabel en bollen- of toestandsdiagrammen, alsmede mengvormen hiervan. Deze laatste worden gebruikt bij logica met registers. Ook mag men zonder uitzondering haakjes gebruiken: de assembler rekent de uiteindelijke vergelijkingen zelf uit en controleert of alles in het opgegeven IC past. Alle assemblers, ook PALASM, genereren als uitvoer een documentatiefile en een zogenaamde JEDEC-file. Deze JEDEC-file kan door een logicaprogrammer worden gelezen, en heeft een universeel, genormaliseerd formaat.

Een mogelijkheid die ook door alle pakketten geboden wordt, is simulatie. Dit wil zeggen, dat de gebruiker in de sourcefile kan beschrijven hoe hij verwacht dat het te programmeren IC zal werken. De assembler kan deze informatie dan denkbeeldig toevoeren aan het ontworpen IC, en controleren of dit inderdaad het geval is. Op deze manier kan men een logisch IC ontwerpen achter het toetsenbord: pas als de simulatie naar wens verloopt programmeert men het prototype. Vooral in de snelheid waarmee dit simulatieproces verloopt en in de gedetailleerdheid van de foutmeldingen lopen de diversen pakketten sterk uiteen.

Bekende logische assemblerpakketten zijn naast PALASM, ABEL en CUPL. Deze twee pakketten zijn zeer universeel en snel, en kunnen niet alleen voor PALs maar ook voor PROMs, GALs, EPLDs en FPLAs assembleren en simuleren. In tegenstelling tot PALASM dat van MMI afkomstig is, en dus alleen MMI PALs ondersteunt, zijn ABEL en CUPL niet merkgebonden. Een laatste voorbeeld van een merkgebonden pakket is PLPL van AMD. Al deze pakketten draaien op PC's of klonen hiervan, en varieren in prijs tussen fl. 800.- en fl. 6000.-. Niet iets voor de hobbyist dus.

11. Besluit.

Dit was een beknopt overzicht over programmeerbare logica. Het is niet de bedoeling van dit verhaal dat de lezer na lezing compleet en uitgebreid over dit soort ICs geïnformeerd is, maar meer bedoeld om een idee te geven van de wereld die schuil gaat achter een misschien nu wat meer zeggend typenummer als PAL16L8. In dit artikel lag verder sterk de nadruk op PALs: dit komt omdat deze het meest gebruikt worden, en door de achtergrond van de schrijver.

Zoals reeds gesteld, zijn PALs inmiddels uitgegroeid tot standaardcomponenten, waar zelfs electronica-hobbytijdschriften als Elektuur nauwelijks meer omheen kunnen. Programmeerbare logica is immers niets anders dan je eigen custom-IC maken, zij het op beperkte schaal.

Computers..... (deel 1).

Door Gert van Opbroek
Bateweg 60
2481 AN Woubrugge
01729-8636

Inleiding

Op de laatste ledenvergadering en uit gesprekken met leden is mij gebleken, dat er binnen de club behoefte bestaat aan enkele publicaties die vooral gericht zijn op die mensen die beginnen met de computerhobby. Welnu, dit verhaaltje wil hiervoor een eerste aanzet zijn.

Ik denk dat het goed is, te beginnen bij het begin van onze club. Onze club is nu twaalf jaar geleden opgericht door mensen die op hun werk te maken kregen met de KIM. Waarschijnlijk hadden ze ook zelf zo'n systeem en om de kennis over dit systeem te bewaren en te verspreiden, is de KIM Gebruikersclub Nederland opgericht.

Wat is die KIM eigenlijk?

De KIM is een zogenaamde single board computer. Dit wil dus zeggen dat op één printplaat een complete computer opgebouwd is, dit in tegenstelling tot de grote kasten van de grote computers zoals de IBM 370 en de PDP-11. Dit computertje kon gebruikt worden voor bijvoorbeeld het besturen en regelen van machines. De mensen die de club opgericht hebben gebruikten de KIM ook voor deze zaken.

In figuur 1 is een foto afgedrukt van de KIM. Het is een printplaat met daarop een aantal elektronische onderdelen. Verder zitten er op de KIM een toetsenbordje met de toetsen 0 t/m 9, A t/m F en enkele toetsen met een twee-letterige code en een zestal 7 segment displays. Op de rand van de printplaat zijn twee connectoren aangebracht voor uitbreidingen en het kunnen aansluiten van de KIM aan andere schakelingen.

In figuur 2 is de interne opbouw van de KIM schematisch weergegeven. In een volgende paragraaf zal deze opbouw verder besproken worden.

Hoewel de KIM uit slechts een klein aantal onderdelen bestaat is het toch een complete computer. Deze computer is niet alleen het startpunt van de KIM Gebruikersgroep Nederland geweest maar ook de start van de hobby-computers.

De club is zich in de loop van de tijd gaan ontwikkelen als een club van specialisten. Het blad heet niet voor niets De 6502 Kenner. Wij willen hiermee aangeven dat we de 6502 door en door kennen. Deze kennis is opgebouwd door het feit dat een deel van de leden zelf hun systeem gebouwd en misschien zelfs wel ontworpen hebben. Verder is er in de club altijd veel in assembler geprogrammeerd en is er door een aantal leden systeemsoftware gemaakt. DOS-65 is in deze categorie een van de laatste producten.

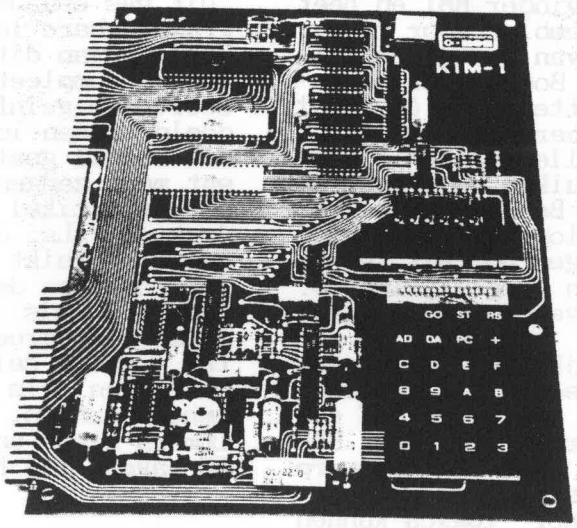


Fig. 1 : Photo of KIM-1

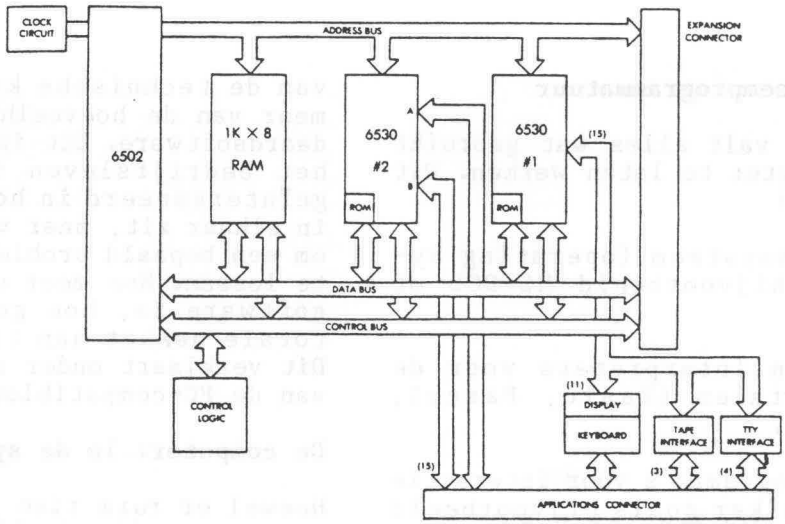


Fig. 2 : KIM-1 Internal Organization

Op dit moment is het zo dat besloten is op dezelfde voet door te gaan. Wel gaan we proberen de basis van de club te vergroten door ook de andere typen processoren te gaan ondersteunen. In plaats van 6502 Kenners worden we dus Microcomputer Kenners.

In dit artikel zullen we eens gaan kijken hoe een microcomputer globaal opgebouwd is. In de volgende delen zullen we steeds een laagje dieper graven. Ik weet niet waar deze excursie zal eindigen maar ik hoop dat degenen die meegaan op excursie na afloop van mening zullen zijn, dat het de moeite waard geweest is. Uiteraard houdt ik me voor op- en aanmerkingen en eventuele bijdragen zoals altijd van harte aanbevolen.

De computer: de buitenste laag

Voor de meeste mensen bestaat een computer momenteel uit:

- Een systeemkast met daarin een hoeveelheid elektronica
- Een toetsenbord
- Een beeldscherm
- Eén of meer diskettestations en misschien zelfs wel een harde schijf (Winchester)
- Eventueel een printer, een muis een plotter .....

Hoewel het allemaal verschillende dingen zijn, kunnen we toch alle apparatuur onderbrengen in drie categorieën:

Categorie 1: De computer

De computer is het hart van het systeem. In deze categorie valt het grootse deel van de inhoud van de systeemkast. Dit is met name de processor die het werk doet en het hoofdgeheugen waar in staat wat voor werk de processor moet doen.

Categorie 2: Achtergrond- of massageheugen

In deze categorie vallen de floppy disks en de harde schijven. Als het systeem een cassette-recorder gebruikt, valt deze ook in deze categorie. De processor kan alleen werken met gegevens die in het hoofdgeheugen staan. Als deze gegevens op achtergrondgeheugen staan, dan moeten ze eerst opgehaald, geladen worden.

Categorie 3: Invoer en uitvoer

Alle overige apparatuur valt in de categorie Invoer en Uitvoer of I/O naar Input en Output. Dit is apparatuur waarmee informatie het systeem binnenkomt of waarmee informatie uit het systeem gaat.

Ook in de software is een opdeling te maken. Hier is de opdeling echter veel minder vast als bij de computer. Zelf hanteer ik de volgende opdeling:

Categorie 1: Systeemprogrammatuur

In deze categorie valt alles wat gebruikt wordt om de computer te laten werken. Dit zijn onder andere:

- Het bedrijfssysteem (operating systeem) dus bijvoorbeeld MS-DOS of DOS-65.
- Vertalers en interpreters voor de programmeertalen (Basic, Pascal, Forth .....).
- Speciale programma's voor interactie met de gebruiker zoals bijvoorbeeld MS-Windows onder MS-DOS of GEM voor de Atari of de Amiga.

Categorie 2: Standaardsoftware

Dit zijn algemene programma's waarmee de computergebruiker werkt. Het specifieke kenmerk van standaardsoftware is dat het algemeen van opzet is. Voorbeelden van standaardsoftware zijn:

- Tekstverwerkers (Wordstar, WordPerfect, ...)
- Pakketten voor gegevensbeheer (database). Voorbeelden hiervan zijn DBASE, ORACLE .....
- Spreadsheets, in het nederlands ook wel electronische kladblokken genoemd zoals bijvoorbeeld Lotus 123, Supercalc etc.
- Module-bibliotheken, ook wel Toolboxes genoemd. Een goed voorbeeld hiervan is de DOS-65 Game Library die in nr. 57 van De 6502 Kenner gestaan heeft.

Categorie 3: Toepassings-software

In de categorie van de toepassings- of applicatie software valt de rest van de software. Dit is meestal software die voor één specifieke taak ontwikkeld is. Dit kunnen dus bijvoorbeeld programma's in Basic zijn maar in mijn visie ook applicaties in bijvoorbeeld DBASE of Lotus 123. Het gebruiken van zoveel mogelijk standaard software bij het maken van applicatie programmatuur zorgt ervoor dat de ontwikkeling van deze applicatie software stukken eenvoudiger gaat. In het bedrijfsleven kan dit de kostprijs van de software aanzienlijk verlagen.

De populariteit van een bepaald computersysteem hangt tegenwoordig meestal niet af van de technische kwaliteiten maar veel meer van de hoeveelheid beschikbare standaardsoftware. Dit is ook logisch, want in het bedrijfsleven is men niet zo zeer geïnteresseerd in hoe mooi het technisch in elkaar zit, maar veel meer wat het kost om een bepaald probleem met de computer op te lossen. Hoe meer en beter de standaardsoftware is, hoe goedkoper meestal het totale pakket aan toepassings-software. Dit verklaart onder andere de populariteit van de PC-compatibles en MS-DOS.

De computer: In de systeemkast

Hoewel er ruim tien jaar tussen de KIM en de moderne 20 MHz 80386 systemen ligt, is er in de opbouw van de systemen maar weinig veranderd. De microcomputers zijn groter en sneller en goedkoper geworden, maar iets revolutionairs is er in die tien jaar niet gebeurd. Kortom, en nu schop ik misschien de bezitters van een modern renpaard tegen het zere been, om een computer te beschrijven kunnen we beste de KIM eens wat beter bekijken. Er zijn natuurlijk best verschillen, en waar ze relevant zijn, zal ik dat ook aangeven.

In figuur 2 is reeds de interne opbouw van de KIM weergegeven, in figuur 3 is schematisch een meer algemene microcomputer getekend.

Eén van de meest belangrijke onderdelen van een computer is de processor. Dit is het hart van het systeem. In een computer wordt de processor meestal aangeduid met CPU van Central Processing Unit. In het nederlands ook wel Centrale Verwerkings Eenheid (CVE) genoemd. Elke computer heeft één over meerdere processors. Het kenmerkende van een microcomputer is het feit dat voor de processor een micro processor (MPU in figuur 3) gebruikt is.

In vroeger tijd bestond de processor van grote computers (IBM 370 bijvoorbeeld) uit een groot aantal onderdelen. In een micro processor zijn deze onderdelen allemaal geïntergreerd in slechts één bouwsteen: de micro processor. Ten opzichte van zijn grote broers had de micro processor minder snelheid en mogelijkheden maar toch was er een markt voor die dingen. Tegenwoordig is het onderscheid niet zo duidelijk meer te maken, het hart van grote computers bestaat tegenwoordig ook vaak uit één of meer micro processors. Ik krijg tegenwoordig sterk de indruk dat de grootte van de kast bepaalt of iets een micro, mini, supermicro, supermini, supercomputer of mainframe is.

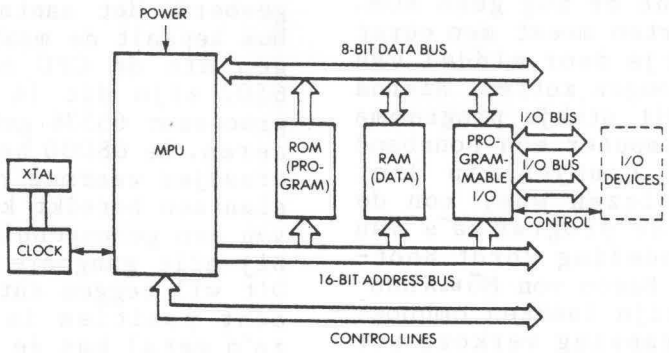


Fig. 3 : Architecture of a Standard Microprocessor System

Goed, in figuur 2 en 3 zien we de CPU geflankeerd door in eerste instantie de klok. Deze levert de hartslag voor de CPU en de rest van de computer. De klok wordt gestuurd door een kristal. Het aantal pulsen dat de klok per seconde geeft (aantal hartslagen) wordt uitgedrukt in Hertz (Hz). Omdat de pulsen met miljoenen per seconde komen spreken we meestal over MegaHertz (MHz) wat wil zeggen miljoen pulsen per seconde. Nu is het zo dat, bij hetzelfde type processor, meer pulsen per seconde ook een snellere computer geeft, het systeem kan dan alles in minder tijd. Vergelijken we computers met verschillende types CPU, dan klopt dit niet meer, een 1 MHz 6502 is ongeveer even snel als een 4 MHz Z80. Bovendien kan het ook nog voorkomen dat het systeem weliswaar een snelle klok heeft maar toch niet zo snel is, omdat de computer gewoonweg niet alle pulsen gebruikt. Er worden dan dus pulsen overgeslagen (wait states). In grote lijnen klopt het echter wel dat bij twee computers met dezelfde CPU de computer met de hoogste kloksnelheid de snelste is.

Rechts naast de CPU vinden we het geheugen of hoofdgeheugen van de computer. Dat is in de eerste plaats geheugen waar de computer alleen uit kan en mag lezen (ROM van Read Only Memory) en verder geheugen waarin zowel gelezen als geschreven kan worden (RAM Random Access Memory). De CPU kan alleen maar iets doen met informatie die in het geheugen aanwezig is. Dit betekent dat alle gegevens die de computer gebruikt in het hoofdgeheugen moeten zitten, maar ook het programma waarin precies staat hoe de computer zijn opdracht uit moet voeren. Een processor doet namelijk niets uit zichzelf. Hij kan een aantal eenvoudige (nou ja) instructies uitvoeren zoals bijvoorbeeld:

Haal de inhoud uit geheugenplaats X
Tel hierbij 7 op
Schrijf het resultaat weg in Y

De opdracht voor de processor staat in het geheugen als een aantal van dergelijke simpele instructies.

Nu heeft RAM één heel belangrijk nadeel: Als de spanning wegvalt, is de inhoud van dit geheugen verloren. Daarom heeft een computer altijd een stuk ROM waarin die programma's staan die altijd voor de com-

puter beschikbaar moeten blijven. Eén van deze programma's is bijvoorbeeld het opstartprogramma van de computer.

In vroeger tijd bestond er nog geen ROM. Om computers op te starten moest men eerst een klein programmaatje door middel van schakelaars in het geheugen zetten. Hierna liet men de computer dit stukje programma uitvoeren waarna de computer een ponsband (een reep papier met gaatjes) in liet lezen. Nadat dit ingelezen was, kon de computer de rest van de programma's van schijf halen. Deze handeling wordt Bootstrapping genoemd naar Baron von Münchhausen die zichzelf aan zijn laarzen omhoogtrok. Later is Bootstrapping verkort tot Booten en zo is deze uitdrukking ontstaan.

Tegenwoordig wordt er meestal gebruik gemaakt van EPROMs (Eraseble Programmable Read Only Memory). Dit wil dus zeggen dat deze ROM's eventueel weer gewist en opnieuw geprogrammeerd kunnen worden.

Om met de buitenwereld te kunnen communiceren, heeft de computer nog een stuk programmeerbare I/O. Hierop kunnen terminals, toetsenborden, muizen, beeldschermen etc. aangesloten worden. In figuur 3 worden dit I/O devices genoemd.

Als laatste heeft een computer meestal een aansluiting voor massageheugen. Strikt genomen is dit vanuit de CPU gezien ook een programmeerbaar stuk I/O; het maakt namelijk voor de CPU niets uit of hij zijn gegevens van een muis, een toetsenbord, of van een schijf krijgt. Ook voor de uitvoer maakt het niets uit of de informatie naar een printer, een beeldscherm of een schijf moet.

In figuur 3 is door middel van een drietal brede pijlen aangegeven hoe de CPU met zijn omgeving communiceert. Deze drie pijlen bestaan in werkelijkheid uit drie bossen draad. Deze bossen draad worden de bussen genoemd. De bussen lopen langs de diverse bouwstenen. Elke bouwsteen krijgt hierbij een aftakking van de bussen. In de praktijk zijn deze bossen draad meestal groepen printsporen, soms (DOS-65) is er zelfs een print waar alleen de bussen op zitten. Op deze zogenaamde busprint zitten dan connectors waarop de printkaarten waaruit het systeem opgebouwd is worden aangesloten.

1: De adresbus.

Elke geheugenplaats in een computer heeft een uniek adres. Door het aanspreken van dit adres kan men de inhoud van deze geheugenplaats lezen of schrijven. Het adres van de geheugenplaats wordt uitgedrukt in een binair getal en de bits van dit getal worden langs de adresbus naar het geheugen gevoerd. Het aantal draadjes in de adresbus bepaalt de maximale hoeveelheid geheugen die de CPU aan kan spreken. Bij de 6502 zijn dit 16 draadjes waarmee deze processor 65536 geheugenplaatsen kan benaderen. De 68000 heeft een adresbus van 24 draadjes waarmee ruim 16 miljoen geheugenplaatsen bereikt kunnen worden. De grootte van een geheugenplaats is, bij mijn weten, bij alle gangbare microprocessors 8 bits. Dit wil zeggen dat er een binair getal van acht posities in past. Elke positie in zo'n getal kan de waarde 0 of 1 aannemen. Een groepje van 8 bits wordt ook wel een byte genoemd.

2: De databus.

Als de processor een adres op de adresbus gezet heeft, kan hij de inhoud van de aangesproken geheugenplaats benaderen. Dit doet hij via de databus. Het aantal draadjes in de databus bepaalt hoeveel informatie er in één keer van of naar het geheugen getransporteerd wordt. Bij de 6502 zijn dit 8 bits, dus de volledige inhoud van een geheugenplaats. Bij de 8086 bijvoorbeeld zijn dit 16 bits, dit is de inhoud van twee geheugencellen. In de praktijk is dit dan de inhoud van de geadresseerde cel plus de inhoud van de cel met het naast hoger gelegen adres. Voor enkele processoren zijn er dan wel beperkingen aan het aangeboden adres (68000) maar dat voert mij op dit moment te ver. De breedte van de databus wordt meestal als kengetal voor de processor aangemerkt. Een processor met een databus van 8 bits heet dan een 8 bitter of een 8 bits CPU en die met een databus van 16 bits een 16 bitter.

3: De Control bus.

Over de control bus wordt stuurinformatie uitgewisseld. Voorbeelden hiervan zijn o.a. een signaal dat aangeeft of een geheugenplaats gelezen of beschreven moet worden, een signaal dat aangeeft dat er een adres op de adresbus staat, signalen die de processor een seintje geven dat een I/O-bouwsteen hulp nodig heeft etc. etc. Ook in de control bus zijn verschillen tussen de diverse typen processoren, deze zullen bij de bespreking van de werkwijze van de processor verder behandeld worden.

Aan het einde van deze paragraaf nog een zaak die de wereld van de micro processors in twee kampen verdeeld heeft. Bij een

deel van de processors (6502, 6800, 6809, 68000 .....) worden de I/O-bouwstenen als onderdeel van het geheugen beschouwd. Dit wil zeggen dat een I/O-bouwsteen zich voor de processor net zo gedraagt als een geheugencel. Er is dus bijvoorbeeld een geheugencel die alle aangeboden informatie meteen op papier afdrukt. Dit wordt Memory Mapped I/O genoemd.

Het tweede kamp bestaat uit bijvoorbeeld de 8080, 8088, Z80, ... Deze processors zien de I/O-bouwstenen niet als geheugencel maar als aparte onderdelen. Om deze onderdelen te benaderen zijn er dus ook aparte bussen en instructies nodig. In principe maakt het natuurlijk niets uit maar in de praktijk had dit wel tot gevolg dat tot voor kort gesteld werd dat de 68000 wel en de 8088 niet in de KIM Gebruikersgroep thuishoort.

Voorbeeld: De KIM.

Het hart van de KIM wordt gevormd door een 1 MHz 6502, geflankeerd door 1024 RAM geheugencellen en 2 6530 bouwstenen. Elke 6530 heeft onder andere de beschikking over 2 I/O poorten met 8 lijnen per stuk, 64 geheugencellen van 8 bits (byte) en 1024 ROM geheugencellen. Verder zijn er mogelijkheden met behulp van de connectors dit systeem uit te breiden.

Op de I/O lijnen zijn het toetsenbord en het display aangesloten. Verder zijn er aansluitingen voorzien voor een terminal (TeleType of TTY, een grote zware kast die een hoop herrie maakt met een toetsenbord en een bolletje die de uitvoer op papier hamert) en een cassette recorder. De I/O-lijnen van de tweede 6530 zijn vrij voor de gebruiker beschikbaar.

In het ROM gedeelte zit het zogenaamde Monitor-programma. Met dit programma kan men informatie vanaf het toetsenbordje in het geheugen zetten en het geheugen met de displays uitlezen. Verder bevat het programma ook routines om informatie met de TTY uit te wisselen en om stukken geheugen op cassette weg te schrijven of van cassette in te lezen.

Al met al was dit voor zijn prijs (+/- fl. 1000) in die tijd een leuk systeem met zeer veel mogelijkheden. Vooral de vrij beschikbare I/O-lijnen vormen voor de echte knutselaars natuurlijk een enorme uitdaging. Bij dit systeem is in de loop der tijd van alles bijgekomen, Basic, Assemblers, diskettestations, terminals, meer geheugen etc. etc. De Junior van Elektuur is een afgeleide van de KIM.

Voorbeeld: een PC-compatible of -kloon

Vergelijken we een willekeurige PC-kloon, met de KIM dan blijken de verschillen behalve in de CPU (6502 versus 8088) voornamelijk in de aantallen te liggen. Een kloon heeft al gauw 650.000 geheugencellen RAM en ook zo'n 128.000 geheugencellen ROM. Verder is de snelheid van het systeem ook wat groter als die van de KIM. Het grote verschil is het feit dat de KIM aan alle kanten aansluitingen voor knutselaars heeft, terwijl dit bij de kloon niet zondermeer het geval is.

Als laatste belangrijke verschil vallen de ingebouwde diskettestations, het toetsenbord en het beeldscherm op. Toch is de interne opbouw niet echt verschillend van figuur 3, alleen het beeldscherm past niet goed in dit plaatje. In een volgende aflevering zal dit verder uit de doeken gedaan worden.

Opvallend is, dat de PC-compatible nu ook zo'n fl 1000,-- kost en ook de standaard hobby computer geworden is.

==========

Te koop:

Wegens overcompleet biedt het bestuur te koop aan:

Een originele Apple II plus met de volgende toebehoren:

- Z80 kaart
- Printerkaart
- Super Serial kaart
- Floppy controler met één originele Apple diskdrive
- Een originele Apple monochroom monitor
- Veel software voor Apple DOS en CP/M

Vraagprijs fl. 500,--

Inlichtingen bij:

Gert Klein
Diedenweg 119
6706 CM Wageningen
08370-23646

Kalender

Hierbij een programma in Pascal. Dit programma is niet getest onder DOS-65 Pascal maar zou wel op een DOS-65 systeem moeten kunnen lopen.
De volgende punten zijn hierbij van belang:
- Omdat in de namen een underscore ´_´ voorkomt, moet hiervoor een compiler-optie meegegeven worden (Zie documentatie).
- De regels met ASSIGN en CLOSE in het hoofdprogramma moeten verwijderd worden. Om de filenaam toe te kennen, moet deze bij het opstarten meegegeven worden.

Mocht u bijzondere dingen tegen komen, laat mij het dan even weten.

Veel succes.

```
PROGRAM KALENDER(INPUT,OUTPUT,KALENDER);

(*
 * Dit programma genereert een kalender voor een in te geven jaar.
 * De kalender wordt op de terminal getoond en weggeschreven in een
 * file die door de interne file-variabele KALENDER wordt aangegeven.
 *
 * Het programma is geschreven in ISO Pascal en uitgetest op een Apple II
 * compatible met Z80-kaart, CP/M en TurboPascal.
 *
 * Het programma is geschreven door:
 *
 * Gert van Opbroek                              Version 1.0/27-11-1988
 * Bateweg 60
 * 2481 AN Woubrugge
 * 01729-8636
 *
 * (c) Copyright: KIM Gebruikergroep Nederland
 *                Het programma mag vrij verspreid worden.
 *                Publicatie van het programma of delen daarvan is slechts
 *                toegestaan na schriftelijke toestemming van het bestuur
 *                van de KGN.
 *)

CONST C_Bovengrens = 2150;                  (* Hoogste toegestane jaartal *)
      C_Ondergrens = 1850;                  (* Laagste toegestane jaartal *)

      (*
       * Constantes voor het schoonmaken van datastructuren en de namen
       * van de dagen van de week en de maanden
       *)

      C_72_Spaties =
'                                                                        ';

      C_8_Spaties  = '        ';      C_Wo  = '  WO  ';
      C_Zo         = '  ZO    ';      C_Do  = '  DO  ';
      C_Ma         = '  MA    ';      C_Vr  = '  VR  ';
      C_Di         = '  DI    ';      C_Za  = '  ZA  ';

      C_Jan        = ' JANUARI ';     C_Jul = '  JULI  ';
      C_Feb        = ' FEBRUARI';     C_Aug = ' AUGUSTUS';
      C_Mrt        = '  MAART  ';     C_Sep = 'SEPTEMBER';
      C_Apr        = '  APRIL  ';     C_Okt = ' OKTOBER ';
```

```
44        C_Mei          = '   MEI    ';    C_Nov = ' NOVEMBER ';
45        C_Jun          = '   JUNI   ';    C_Dec = ' DECEMBER ';
46
47    TYPE  (*
48          * Types om gemakkelijk met arrays van letters (teksten) om te kunnen
49          * gaan.
50          *
51          * NB. De toevoeging 'PACKED' betekent dat er een minimale hoeveelheid
52          *     ruimte gereserveerd moet worden: per letter dus 1 byte
53          *)
54
55          T_C2           = PACKED ARRAY [1.. 2] OF CHAR;
56          T_C3           = PACKED ARRAY [1.. 3] OF CHAR;
57          T_C7           = PACKED ARRAY [1.. 7] OF CHAR;
58          T_C8           = PACKED ARRAY [1.. 8] OF CHAR;
59          T_C10          = PACKED ARRAY [1..10] OF CHAR;
60          T_C32          = PACKED ARRAY [1..32] OF CHAR;
61          T_C72          = PACKED ARRAY [1..72] OF CHAR;
62
63          (*
64          * Type om per regel, per maand de data die op een bepaalde weekdag
65          * vallen in op te slaan. Voor de data zijn drie letters gereserveerd
66          * waarvan er maximaal 2 gebruikt zullen worden, de derde is de
67          * de scheiding tussen de data.
68          *)
69
70          T_MaandInfo    = PACKED RECORD
71                             VoorloopSpaties    : T_C3;
72                             Weekdagen          : PACKED ARRAY [0..5] OF T_C3;
73                             NaLoopSpaties      : T_C3;
74                           END;
75
76          (*
77          * Type om op eenvoudige wijze per kwartaal de kopregel af te
78          * kunnen drukken.
79          *)
80
81          T_MaandKop     = PACKED RECORD
82                             VoorloopSpaties    : T_C7 ;
83                             MaandNaam          : T_C10;
84                             NaloopSpaties      : T_C7 ;
85                           END;
86
87          (*
88          * Type om op eenvoudige wijze de layout van een regel mee op te
89          * maken. Een regel begint altijd met een weekdag, waarin eventueel
90          * alleen spaties staan. Verder zijn er een aantal mogelijkheden.
91          *
92          * 1: Beschouw de rest van de regel als een geheel. Deze mogelijkheid
93          *    wordt gebruikt om:
94          *    - een regel in een keer met spaties te vullen
95          *    - een regel af te drukken
96          * 2: Een datastructuur waarmee het jaartal afgedrukt wordt.
97          * 3: Een datastructuur waarmee per kwartaal de namen van de maand
98          *    afgedrukt kunnen worden.
99          * 4: Een datastructuur waarmee van een kwartaal per weekdag de data
100         *    waarop die weekdag valt afgedrukt kan worden.
```

```
101          *
102          * Een dergelijke opbouw van van record worden VARIANTEN genoemd.
103          * Ook hier is het woord PACKED toegevoegd om er zeker van te zijn
104          * dat de minimaal benodigde ruimte gebruikt wordt en alle velden
105          * in het record aansluitend achter elkaar liggen.
106          *)
107
108         (*
109          * Het record bezit vier mogelijkheden
110          *)
111
112         T_Mogelijkheid = 1..4;
113
114         T_Regel      = PACKED RECORD
115                          Weekdag           : T_C8;
116                          CASE T_Mogelijkheid OF
117                          1: ( Regel_Totaal  : T_C72);
118                          2: ( Voorloop      : T_C32;
119                               Jaartal       : T_C8;
120                               Naloop        : T_C32);
121                          3: ( KopRegel      : PACKED ARRAY [1..3] OF T_MaandKop);
122                          4: ( MaandInfo     : PACKED ARRAY [1..3] OF T_MaandInfo);
123                        END;
124
125         (*
126          * Enummeratie-type met de maanden van het jaar. In dit type worden
127          * de maanden van het jaar opgesomd.
128          *)
129
130         T_E_Maand    = (E_Jan,E_Feb,E_Mrt,E_Apr,E_Mei,E_Jun,
131                         E_Jul,E_Aug,E_Sep,E_Okt,E_Nov,E_Dec);
132
133   (*
134    * Variabelen die in het hele programma bereikt kunnen worden.
135    *)
136
137   VAR   KALENDER : TEXT;
138         Kwartaal : INTEGER;
139         Jaar     : INTEGER;
140
141   (*
142    * Fuctie DagFactor is een oude bekende. Deze functie is onder andere
143    * gebruikt in het programma Datum --> Weekdag conversie
144    * uit De 6502 Kenner nr. 53. Dit betrof toen een versie in C.
145    *
146    * Het algoritme is afkomstig uit de programma ROM van een TI 58
147    * rekenmachine.
148    *
149    * DagFactor berekent bij de ingevoerde dag, maand en jaar een dagnummer.
150    * dit dagnummer is uniek. Door het dagnummer modulo 7 te nemen, krijgen
151    * we een getal dat de dag in de week aangeeft. Hierbij krijgt de zondag
152    * de waarde nul. Door voor de deling 1 van het dagnummer af te trekken,
153    * wordt er voor gezorgd dat de week begint op maandag.
154    *
155    * De datum wordt in waarde- (VALUE) parameters doorgegeven, de dag in
156    * de week wordt als referentie- (REFERENCE) parameter teruggegeven dit wil
157    * zeggen dat bij de aanroep het startadres van de parameter doorgegeven
```

```
158    * wordt. Deze variabele kan dus door DagFactor gewijzigd worden.
159    *
160    * Als functieresultaat wordt doorgegeven of de ingevoerde datum bestaat
161    * en of het jaartal in het vastgestelde gebied ligt.
162    *)
163
164   FUNCTION DagFactor(     P_Dag       : INTEGER;
165                           P_Maand     : T_E_Maand;
166                           P_Jaar      : INTEGER;
167                     VAR P_Weekdag     : INTEGER)       : BOOLEAN;
168
169   (*
170    * Locale variabelen die alleen in DagFactor benandert kunnen worden.
171    *)
172
173   VAR   Factor          : INTEGER;
174         Parameters_OK   : BOOLEAN;
175         Klad            : INTEGER;
176
177   BEGIN
178
179     (*
180      * Controleer eerst of de datum bestaat.
181      *)
182     (*
183      * Het jaar:
184      *)
185
186     Parameters_OK := (P_Jaar >= C_Ondergrens) AND (P_Jaar <= C_Bovengrens);
187
188     IF Parameters_OK THEN
189     CASE P_Maand OF
190
191       (*
192        * De dag voor de maanden met 31 dagen
193        *)
194
195       E_Jan,E_Mrt,E_Mei,E_Jul,E_Aug,E_Okt,E_Dec :
196
197         Parameters_OK := (P_Dag >= 1) AND (P_Dag <= 31);
198
199       (*
200        * De dag voor de maanden met 30 dagen
201        *)
202
203       E_Apr,E_Jun,E_Sep,E_Nov :
204
205         Parameters_OK := (P_Dag >= 1) AND (P_Dag <= 30);
206
207       (*
208        * De dag voor de maand fedruari
209        *)
210
211       E_Feb :
212
213         (*
214          * Schrikkeljaren: Jaren deelbaar door vier en eeuwjaren deelbaar
```

```
215          * door 400
216          *)
217
218         IF (P_Jaar MOD 4 = 0) AND (( P_Jaar MOD 100 <> 0) OR
219                                    ( P_Jaar MOD 400  = 0)) THEN
220              Parameters_OK := (P_Dag >= 1) AND (P_Dag <= 29)
221         ELSE  Parameters_OK := (P_Dag >= 1) AND (P_Dag <= 28);
222
223       END; (* CASE *)
224
225     IF Parameters_OK THEN
226     BEGIN
227
228       (*
229        * Bereken de dagfactor; ORD geeft het volgnummer van de maand.
230        *)
231
232       Klad   := P_Jaar - 1985 ;                        (* Anders Overflow *)
233
234       Factor := 365 * Klad - 4 + P_Dag +
235                 31 * (ORD(P_Maand) - ORD(E_Jan));
236
237       IF P_Maand <= E_Feb THEN
238          Factor := Factor +  (P_jaar - 1) DIV 4  -
239                          3 * ((P_Jaar - 1) DIV 100 + 1) DIV 4
240       ELSE
241          Factor := Factor - (4 * (ORD(P_Maand) - ORD(E_Jan) + 1) + 23) DIV 10 +
242                                  P_Jaar DIV 4 -
243                           3 * (P_Jaar DIV 100 + 1) DIV 4;
244
245       (*
246        * Bepaal de bijbehorende dag in de week, maandag = 0;
247        *)
248
249       P_Weekdag := (Factor - 1) MOD 7 ;
250     END;
251     DagFactor := Parameters_OK;
252   END;
253
254   (*
255    * Procedure Jaartal drukt het jaartal in de kop van de kalender af.
256    * Het betreffende jaartal wordt als waarde (VALUE) parameter meegegeven,
257    * hetgeen betekent dat bij de aanroep de waarde van de parameter gekopieerd
258    * wordt zodat Jaartal het orgineel niet kan veranderen.
259    *)
260
261   PROCEDURE Jaartal(P_Jaar      : INTEGER);
262
263   (*
264    * Lokale variabelen binnen Jaartal, deze kunnen niet buiten Jaartal
265    * benaderd worden.
266    *)
267
268   VAR   Regel : T_Regel;
269         I     : INTEGER;
270
271   BEGIN
```

```
272
273     (*
274      * Test voor de zekerheid het waardenbereik van de parameter af, geef
275      * een foutmelding als dit niet deugd.
276      *)
277
278     IF (P_Jaar < C_Ondergrens) OR (P_Jaar > C_Bovengrens) THEN
279     BEGIN
280       WRITELN('>>>>>>>>>> FOUT IN JAARTAL ',P_Jaar,' <<<<<<<<<<');
281       WRITELN(KALENDER,'>>>>>>>>>> FOUT IN JAARTAL ',P_Jaar,' <<<<<<<<<<');
282     END ELSE
283
284       (*
285        * In het volgende deel moeten we binnen de datastructuur Regel zijn,
286        * door het WITH statement te gebruiken hoeven we niet steeds de
287        * toevoeging Regel. te gebruiken.
288        *)
289
290       WITH Regel DO
291       BEGIN
292
293         (*
294          * Vul de hele datastructuur met spaties.....
295          *)
296
297         Weekdag      := C_8_Spaties;
298         Regel_totaal := C_72_Spaties;
299
300         (*
301          * Neem het jaartal teken voor teken over in de datastructuur, te
302          * beginnen bij het laatste teken.
303          *)
304
305         FOR I := 3 DOWNTO 0 DO
306         BEGIN
307           Jaartal[2 * I + 1] := CHR((P_Jaar MOD 10) + ORD('0'));
308           P_Jaar := P_Jaar DIV 10;
309         END;
310
311         (*
312          * Druk de regel met het jaartal af.
313          *)
314
315         WRITELN(Weekdag,Regel_Totaal);
316         WRITELN(KALENDER,Weekdag,Regel_Totaal);
317       END;
318   END;
319
320   (*
321    * Procedure Kopregel drukt per kwartaal een kop met de maanden in dat
322    * kwartaal af. Het kwartaal waarvoor dit moet gebeuren wordt als waarde-
323    * (VALUE) parameter meegegeven.
324    *)
325
326   PROCEDURE Kopregel(P_Kwartaal : INTEGER);
327
328   (*
```

```
329    * Lokale variabelen binnen Kopregel
330    *)
331
332   VAR   Regel : T_Regel;
333
334   BEGIN
335     WRITELN;  WRITELN(KALENDER);
336     WRITELN;  WRITELN(KALENDER);
337
338     (*
339      * Voor de zekerheid eerst even de parameter aftesten en eventueel
340      * een foutmelding geven.
341      *)
342
343     IF (P_Kwartaal < 1) OR (P_Kwartaal > 4) THEN
344     BEGIN
345       WRITELN('>>>>>>>>>> Fout in kwartaal ',P_Kwartaal,' <<<<<<<<<<');
346       WRITELN(KALENDER,'>>>>>>>>>> Fout in kwartaal ',P_Kwartaal,' <<<<<<<<<<');
347     END ELSE
348
349       (*
350        * een WITH-statement om ons onnodig typewerk te besparen.
351        *)
352
353       WITH Regel DO
354       BEGIN
355
356         (*
357          * Vul de datastructuur met spaties.
358          *)
359
360         Weekdag      := C_8_Spaties;
361         Regel_Totaal := C_72_Spaties;
362
363         (*
364          * Nu voor een kwartaal de kopinformatie invullen. Het CASE-statement
365          * selecteert het juiste kwartaal. Dat blok waarvan het label overeen-
366          * komt met het kwartaal wordt namelijk uitgevoerd.
367          *)
368
369         CASE P_Kwartaal OF
370           1:  BEGIN
371                 Kopregel[1].MaandNaam := C_Jan;
372                 Kopregel[2].MaandNaam := C_Feb;
373                 Kopregel[3].MaandNaam := C_Mrt
374               END;
375           2:  BEGIN
376                 Kopregel[1].MaandNaam := C_Apr;
377                 Kopregel[2].MaandNaam := C_Mei;
378                 Kopregel[3].MaandNaam := C_Jun
379               END;
380           3:  BEGIN
381                 Kopregel[1].MaandNaam := C_Jul;
382                 Kopregel[2].MaandNaam := C_Aug;
383                 Kopregel[3].MaandNaam := C_Sep
384               END;
385           4:  BEGIN
```

```
386                 Kopregel[1].MaandNaam := C_Okt;
387                 Kopregel[2].MaandNaam := C_Nov;
388                 Kopregel[3].MaandNaam := C_Dec
389               END;
390       END;
391
392       (*
393        * Nu de kop nog even afdrukken en klaar is Kees
394        *)
395
396       WRITELN(Weekdag,Regel_Totaal);
397       WRITELN(KALENDER,Weekdag,Regel_Totaal)
398     END;
399
400     WRITELN;  WRITELN(KALENDER)
401   END;
402
403 (*
404  * Procedure DataRegels genereert voor het als parameter doorgegeven kwartaal
405  * de dataregels. Het kwartaal en het jaar worden als waarde- (VALUE)
406  * parameter doorgegeven.
407  *)
408
409 PROCEDURE DataRegels(P_Kwartaal,P_Jaar : INTEGER);
410
411 (*
412  * Locale variabelen
413  *)
414
415 LABEL 999;                              (* Uitsluitend voor foutsituaties *)
416 VAR   Startdagen     : ARRAY [1..3] OF INTEGER;
417       Maanden        : ARRAY [1..3] OF T_E_Maand;
418       I,J,K,I_Weekdag: INTEGER;
419       Dag            : INTEGER;
420       Regel          : T_Regel;
421
422 BEGIN
423
424   IF (P_Kwartaal >= 0) AND (P_Kwartaal <= 4) THEN
425   BEGIN
426
427     (*
428      * Vul de maanden in de array in, het eerste element wordt handmatig
429      * gevuld, de overige elementen zijn de opvolger (SUCCessor) van het
430      * eerste resp. de tweede element.
431      *)
432
433     CASE P_Kwartaal OF
434       1: Maanden[1] := E_Jan;
435       2: Maanden[1] := E_Apr;
436       3: Maanden[1] := E_Jul;
437       4: Maanden[1] := E_Okt;
438     END;
439
440     Maanden[2] := SUCC(Maanden[1]); Maanden[3] := SUCC(Maanden[2]);
441
442     (*
```

```
443         * Bepaal m.b.v. DagFactor de startdagen van de eerste rij voor de
444         * drie maanden.
445         *)
446
447       FOR I:= 1 TO 3 DO
448           IF NOT DagFactor(1,Maanden[I],P_Jaar,I_Weekdag) THEN
449           BEGIN
450             WRITELN('>>>>>>>>>> Ernstige fout opgetreden <<<<<<<<<<');
451             WRITELN(KALENDER,'>>>>>>>>>> Ernstige fout opgetreden <<<<<<<<<<');
452             GOTO 999;
453           END ELSE
454
455             (*
456              * Valt de eerste van de maand op maandag, dan meteen beginnen,
457              * anders eerst enkele dagen overslaan d.m.v. een illegale datum
458              *)
459
460             StartDagen[I] := 1 - I_Weekdag;
461
462       WITH Regel DO
463         FOR K := 0 TO 6 DO
464         BEGIN
465
466           (*
467            * Initialiseer de datastructuur
468            *)
469
470           Regel_Totaal := C_72_Spaties;
471           CASE K OF
472             0: Weekdag := C_Ma;
473             1: Weekdag := C_Di;
474             2: Weekdag := C_Wo;
475             3: Weekdag := C_Do;
476             4: Weekdag := C_Vr;
477             5: Weekdag := C_Za;
478             6: Weekdag := C_ZO
479           END;
480
481           FOR J := 1 TO 3 DO
482             FOR I := 0 TO 5 DO
483             BEGIN
484               Dag := Startdagen[J] + 7 * I + K;
485               IF DagFactor(Dag,Maanden[J],P_Jaar,I_Weekdag) THEN
486               BEGIN
487                 Maandinfo[J].Weekdagen[I,3]   := CHR(Dag MOD 10 + ORD('0'));
488                 IF Dag >= 10 THEN
489                   Maandinfo[J].Weekdagen[I,2] := CHR(Dag DIV 10 + ORD('0'));
490               END;
491             END;
492
493           WRITELN(Weekdag,Regel_Totaal);
494           WRITELN(KALENDER,Weekdag,Regel_Totaal);
495       END;
496     END;
497   999:
498   END;
499
```

```
500   (*
501    * Het hoofdprogramma.
502    *)
503
504   BEGIN
505    (*
506     * De volgende statement is systeemafhankelijk en moet aan uw
507     * eigen systeem aangepast worden.
508     *)
509
510    ASSIGN(KALENDER,'KALENDER.LIS');
511
512    (*
513     * Geef aan dat de file KALENDER geopend moet worden voor schrijven, de
514     * oude inhoud gaat hiermee verloren.
515     *)
516
517    REWRITE(KALENDER);
518
519    (*
520     * In deze REPEAT-lus wordt de invoer ingelezen. De lus wordt pas verlaten
521     * nadat er correcte invoer ingelezen is.
522     *
523     * NB. Het invoeren van iets anders dan een getal doet het programma
524     *     crashen. In echt gebruikersvriendelijke software zou dit ook
525     *     ondervangen moeten worden.
526     *)
527
528    REPEAT
529      WRITE('Geef het jaartal >= ',C_Ondergrens,' <= ',C_Bovengrens,' : ');
530      READLN(Jaar);
531    UNTIL (Jaar >= C_Ondergrens) AND (Jaar <= C_Bovengrens);
532
533    (*
534     * Geef het jaartal uit.
535     *)
536
537    Jaartal(Jaar);
538
539    (*
540     * Geef per kwartaal de kop en de kalender uit.
541     *)
542
543    FOR Kwartaal := 1 TO 4 DO
544    BEGIN
545      Kopregel(Kwartaal);
546      DataRegels(Kwartaal,Jaar)
547    END;
548
549    (*
550     * Het volgende statement is systeemafhankelijk en moet evt. aangepast
551     * worden aan uw eigen systeem.
552     *)
553
554    CLOSE(KALENDER)
555
556   END.
```

De IBM-PC en z'n klonen (Deel 1).

Door: Nico de Vries.

In de vorige nummers heeft u, misschien zelfs wel tot uw grote schrik, kunnen lezen dat het bestuur met de club een beetje een andere kant op wil. Een van die kanten is de wereld van MS-DOS, IBM, klonen uit de oriënt en dus Intel 80XXX processoren. Nu kan ik mij voorstellen dat lang niet iedereen nu precies weet hoe het technische hoe waarom van de 'PC' en zijn klonen in elkaar zit. Dat wordt nog een beetje bemoeilijkt door het feit dat er voor ons 6502-kenners een buitengewoon maffe processor in die apparaten zit. Ook draagt de meestal ronduit gebrekkige documentatie van de meeste klonen niet bij tot een goed inzicht in de hard- en softwaremogelijkheden van de PC.
Daarom deze artikelreeks. Het is bedoeld om de lezer een inzicht geven wat er aan techniek schuil gaat achter de MS-DOS prompt. Ook is het de bedoeling uw belangstelling te prikkelen voor de PC als doel van uw hobby, en niet als middel en/of gereedschap in het dagelijkse leven. De PC is technisch net zo interessant als een DOS-65 of een Octopus computer. Er is maar 1 verschil: hij is kant en klaar gekocht, en niet zelf gebouwd.
Boven dit artikel staat: Deel 1. De delen 2 en 3 zullen heus wel volgen, ik heb stof genoeg. Welk nummer het laatste deel gaat krijgen weet ik niet. Dat is niet gepland. We zien wel. Ik weet ook niet alles van PC's en MS-DOS. Er kunnen dus best fouten en onvolkomenheden in dit verhaal staan. Maar de grote lijn, die klopt wel.....

1.1. Geschiedenis.

Je moet altijd bij het begin beginnen, wil je tenminste begrijpen waarom een aantal dingen vandaag de dag zijn, zoals ze zijn. Dit geldt ook voor PC's, en in het bijzonder voor de AT.
Wacht eens even.... Daar staan al twee dingen die uitleg behoeven! Wat verstaan we eigenlijk onder een PC in dit verband? Dat is niet zo moeilijk: dat zijn alle computers die enigszins compatibel zijn met of zelfs sterk lijken op de oorspronkelijke IBM-PC. Dergelijke computers hebben een 8088 (of soms een 8086) CPU. Een AT is een computer die opwaarts compatibel is met een PC maar die is uitgerust met een 16-bit processor: de 80286.
Maar terug naar het begin. Na de uitvinding van de microprocessor verschenen er een aantal verschillende op de markt. Je kunt twee oer-processors onderscheiden: de Intel 8080 (afgeleid van de allereerste 4004, en de latere 8008), en de Motorola 6800. Zeer spoedig daarna volgde de ons wel bekende 6502, die strikt genomen van de 6800 is afgeleid. Omdat de 6502 al zo oud is, wordt hij meestal ook bij de oer-CPU's gerekend. De 6800 en de 8080 verschilden op 1 belangrijk punt van elkaar: de 6800 (en ook de 6502) heeft de I/O gewoon in de geheugenmap zitten, dat wil zeggen, er is geen verschil in geheugen en I/O. De Intel 8080 daarentegen heeft aparte instructies en ook aparte stuurlijnen om I/O aan te kunnen sturen.
Het duurde niet lang, of er verschenen kleine printjes met wat RAM, een debugger in ROM, een cassette-interface en een hexdisplay op de markt: de kits. De KIM-1 was een van die kits en zo populair dat onze club erdoor ontstaan is. De kit was eigenlijk bedoeld als studiehulp bij het evalueren van een CPU, maar werd in de praktijk ook gebruikt voor het maken van kleine besturinkjes. Hogere programmeertalen waren er niet, en assemblers nauwelijks. Alles ging in hex en op cassettetape!
Na een tweetal jaren veranderde de wereld van de microprocessors als bij donderslag: Commodore kwam op de markt met een apparaat met de weinig flatteuze naam PET. Dit was een computer met voor die tijd een redelijke hoeveelheid RAM (4 of 8 kbyte), een ingebouwde cassetterecorder met interface, een compleet toetsenbord (met grafische symbolen), een scherm en als klap op de vuurpijl een BASIC interpreter. Plotseling kom men comfortabel werken, mede dankzij een uitstekende schermeditor. De PET werd een enorm succes.
Zeer kort daarna verschenen nog twee computers op ongeveer hetzelfde stramien: de Apple en de Tandy TRS-80. Deze apparaten hadden veel met de PET gemeen: BASIC, cassette-opslag, een scherm (zonder schermeditor), en een compleet, volwaardig toetsenbord.
De Apple had net als de PET een 6502 CPU, en bezat voor die tijd iets bijzonders: uitbreidingsslots. In deze slots kon men bijvoorbeeld een printerinterface steken, of (later) een floppy disk controller. Voor deze slots is een veelheid aan kaarten en kaartjes op de markt verschenen. De populairste was waarschijnlijk het Z-80 kaartje, waardoor de Apple ook CP/M kon draaien.
De TRS-80 viel op door het feit dat de computer modulair was opgezet: door het bijkopen van allerlei extra's kon het apparaat buitengewoon ver worden uitgebreid, tot 48k RAM en schijven toe, die in die tijd zeer kostbaar waren.
Deze drie oer-machines evolueerden ieder op hun eigen manier verder. Er verschenen apparaten die duidelijk op het kantoor waren toegespitst (CBM 8032 bijvoorbeeld) of juist voor de huiskamer (VIC-20). Een paar dingen bleven consequent standaard: BASIC in ROM en een cassetteinterface voor de opslag.

Ondertussen stonden de ontwikkelingen niet stil. De 8080 kreeg een krachtiger opvolger in de vorm van de Z-80 van Zilog. Zilog kreeg meteen een proces aan zijn broek: de Z-80 kon namelijk alle 8080 programma's draaien maar had een aanzienlijk krachtiger instructieset, die in grootte alle tot dan toe bekende processoren overtrof. Bij een groot aantal mensen heeft hierdoor het misverstand postgevat dan de Z-80 de krachtigste 8-bit processor is. Dit is onjuist: dat is nog steeds de 6502.....

De industrie en het bedrijfsleven waren inmiddels ook met microcomputers begonnen, en kregen in het beginstadium last van het feit dat er geen goede standaards waren. Zo kan een Apple geen schijfjes van een Commodore lezen, laat staan dat ze elkaars programma's kunnen draaien. Ook hadden de eerste computers hun beperkingen, ook wat snelheid betreft: 6502's op 1 MHz, de TRS-80 met Z-80 op 4 MHz (ongeveer even snel dus!).

Een bedrijf met de naam Digital Research had inmiddels een operating system ontwikkeld voor de 8080. Het heette CP/M hetgeen betekent: Control Program for Microcomputer. CP/M was zodanig opgezet, dat het nagenoeg onafhankelijk was van de hardware. Deze afhankelijkheid werd opgelost in een interfaceprogramma, dat in CP/M het BIOS, of Basic Input/Output System heet. De rest van CP/M is voor alle computers gelijk. Dit gaf een mogelijkheid om programma's te kunnen uitwisselen tussen apparaten van verschillende merken, die echter wel allemaal CP/M draaiden.

Het gevolg laat zich raden, CP/M machines, vooral die met een Z-80 werden razend populair, vooral in de USA. De ontwikkeling van de 64kbit DRAM maakte de droom helemaal waar. Een Z-80 kan DRAMs refreshen zonder hulpschakelingen: de machines kregen eenvoudig 64k RAM. Het bootROM werd na booten uit de memorymap geschakeld zodat er echt 64k RAM beschikbaar was: in de meeste implementaties blijft er dan na het laden van CP/M zelf zo'n 48k over.

De populariteit van CP/M werd nog versterkt door het feit dat er tekstverwerkers en databases verschenen. Sommige waren zo goed en/of werden zo populair, dat ze vandaag de dag nog gebruikt worden. Voorbeelden zijn de WordStar tekstverwerker (eigenlijk een uit de hand gelopen ASCII-editor) en de Dbase II database manager.

Vanaf het begin vroegen de kenners zich af wanneer de mainframegigant IBM zich met de micro's ging bemoeien. Het bleef (leek het) lang stil in het IBM-kamp. Maar in het begin van de jaren tachtig gebeurde het toch.....

1.2. De IBM-PC komt!

IBM noemde zijn eerste microcomputer eenvoudigweg Personal Computer, of afgekort PC. Er was niets revolutionairs aan het apparaat te onderscheiden, op 1 ding na: er zat een relatief nieuwe CPU van Intel in, de 8088.

De oer-PC was voorzien van BASIC in ROM, een volwaardig toetsenbord en een cassetteinterface. De koper kon kiezen uit een groen/zwartscherm voor tekstweergave, of een kleurenscherm dat ook grafische afbeeldingen kon produceren. Er zat 64k RAM in de PC.

IBM had goed nagedacht. De PC was vanaf het begin zodanig opgezet, dat het apparaat kon worden uitgebreid. Men leende het slot-idee van Apple bijvoorbeeld. Ook kon er meer RAM in, indien nodig: de 8088 kan tenslotte 1 Mbyte adresseren. Schijfjes waren ook mogelijk: in het begin ging daar wel 160 kbyte op! Ook de CPU, toen nieuw, was slim gekozen: het is de eenvoudigste van een hele reeks opwaarts compatibele processoren die Intel ontwikkeld heeft. IBM kon dus alle kanten op.

Ook in software. Een PC met diskdrives (nog geen harddisk) kon je met drie verschillende operating systems bestellen. Het eerste was het in universiteitskringen redelijk ingeburgerde USCD. Niemand kocht het. Van het tweede operating system had men de meeste verwachtingen: CP/M-86K, een variant op het op Z-80 machines immens populaire CP/M, toegespitst op de 8088/8086 processoren. Niemand kocht het. IBM bood nog een derde operating system aan waar nog niemand van gehoord had: PC-DOS. Het copyright statement vermeldde de naam Microsoft. Microsoft, zeer bekend van haar BASIC-interpreters, de Macro-80 assembler en een rijtje compilers voor C, BASIC en PASCAL (alle onder CP/M) had het operating system gekocht van een klein bedrijfje in Seattle. IBM implementeerde het op haar PC en noemde het PC-DOS, Microsoft behield zich het recht voor het ook aan anderen te verkopen onder de naam MS-DOS.

PC-DOS, of MS-DOS zo u wilt, werd eenvoudig de standaard op de PC. Niemand weet precies waarom. Het vermoeden bestaat dat de naam Microsoft vertrouwen inboezemde, gekoppeld aan IBM kon dat vertrouwen niet meer stuk. De PC werd gigantisch populair: er stond IBM op en je kon er alle kanten mee op. Al spoedig werd de roep om meer gehoord. IBM bracht een versie uit met 256k RAM standaard en een 10 of 20 Mbyte winchester drive. Deze machine droeg de naam Personal Computer Extended, of afgekort PC/XT. Het werd de standaard voor de jaren tachtig. De vraag was groter dan het aanbod. En dus kwamen er:

1.3. Klonen en compatibles.

Als iemand met een product succes heeft, dan komt er al gauw iemand bij, die een product gaat maken dat er wel op lijkt, maar dat toch voldoende an-

ders is dan het oorspronkelijke. Zo geschiedde ook met de PC en de PC/XT.

Als je product van IBM koopt moet je eerst wennen aan de documentatie. Die is namelijk erg compleet. De Technical Reference Manual bijvoorbeeld bevat zoveel detailinformatie, dat je een PC zo kunt nabouwen. En dat gebeurde dan ook. Eerst in de Verenigde Staten. Daar hebben ze op copyrightgebied een beruchte wetgeving, dus zoiets moet je netjes aanpakken. Er verschenen na verloop van tijd een groot aantal machines die allemaal MS-DOS konden draaien en die allemaal op de IBM-PC leken. En in dat lijken zit nu de clou: ze zijn een ietsje anders. Zolang de software maar netjes van het DOS gebruik maakt, is er niets aan de hand. Wordt het echter machinespecifiek dan strandt het schip: de machine is niet helemaal compatibel. Deze klasse van machines worden dan ook de (bijna) compatibles genoemd. Ze zijn vaak redelijk goed gelijk aan de IBM-PC maar voldoende anders om IBM niet de kans op een succesvol proces te geven. Er zelfs een tijd geweest waarin de mate van compatibiliteit werd uitgedrukt in procenten (hoe doe je zoiets??). De meestgebruikte tests waren de Microsoft Flight Simulator en Lotus 1-2-3, die beide rechtstreeks de machine aanspraken, langs het DOS en soms zelfs langs het BIOS heen. Draaide de compatible deze programma's dan was hij 'compatible', anders waardeloos.

Er zijn echter in de wereld ook landen waar men het niet zo nauw neemt met het copyright. Je zou zeggen in Nederland, maar dat valt mee. Landen als Taiwan en Korea erkennen namelijk het Amerikaanse recht niet, dus ook niet het copyright. En dankzij de uitstekende IBM documentatie kom het gebeuren dat er 1 op 1 kopieën van de PC en de PC/XT uit die landen verschenen: alle onderdelen, tot en met laatste schroefje zijn uitwisselbaar. De printen en de IC-nummering daarop zijn bij sommige machines exact gelijk aan de IBM originelen! Dergelijke apparaten konden dankzij het lage loonpeil in het verre oosten voor een belangrijk lagere prijs worden verkocht dan IBM's officiële produkten, met als verschil: geen ondersteuning bij moeilijkheden. Deze klasse van machines worden klonen genoemd. Klonen zijn automatisch 100% compatibel: het zijn gewoon kopieën van het origineel. Door in een kloon een IBM-ROMset te plaatsen krijg je de super-kloon: alles dat op een PC of PC/XT draait, draait ook op de superkloon.

In de klonen- en compatibleswereld is een hele ontwikkeling geweest. Het resultaat is, dat vrijwel alle klonen en compatibles op hogere kloksnelheden lopen dan hun voorbeelden. Ook hebben de fabrikanten inmiddels geleerd een 100% compatible BIOS te schrijven dat voor IBM geen aanleiding is tot het starten van processen. Het netto resultaat is, dat de kloon en de compatible inmiddels een volwassen product zijn geworden, waar ook grote bedrijven zoals Philips mee op de markt verschijnen. Compatibiliteitsproblemen komen vrijwel niet meer voor, terwijl de prijzen tot op het bot zijn gedaald: ook u lezer, heeft waarschijnlijk wel via het zoveelste PC-prive-project een kloon in huis staan, en anders wel op uw werk. De PC is alom: iedereen doet het erop...

1.4. Verdere ontwikkelingen.

IBM zat intussen niet stil en ontwikkelde een machine met de krachtiger 80286 processor. In tegenstelling tot de 8088 die een 8-bit databus heeft, is de 80286 een 16-bit CPU (de mensen die een 8088 een 16-bitter noemen hebben het bij het verkeerde eind: een 16-bit 8088 heet namelijk 8086). De kloksnelheid van de PC en de PC/XT lag niet bijster hoog: 4.77 MHz. Dit werd in de 80286 machine hoger: 6 MHz. Verder werden technisch gesproken alle zeilen bijgezet om zo hoog mogelijke prestaties te halen binnen de mogelijkheden van toen: zo kreeg het apparaat 2 DMA controllers, een uitgebreide interruptstructuur en slots die 16-bit breed, en toch PC(/XT) compatibel waren. IBM vond dat de machine geavanceerd was en doopte hem Personal Computer, Advanced Technology, of afgekort PC/AT. Ook deze PC is inmiddels zeer uitgebreid gekloond. De AT-compatibles en -klonen zijn in het bedrijfsleven wereldwijd zo populair, dat zij de standaardmachine van dit moment vormen. De aantallen zijn zo groot, dat IBM met haar nieuwste reeks machines nauwelijks voet aan de grond krijgt: men koopt liever een AT-kloon, want die is compatibel met de andere machines op het kantoor en nog goedkoper ook. Door de hoge kloksnelheden (16 MHz is al redelijk normaal) geven de prestaties ook geen reden tot klachten.

De PC/AT, of nog korter AT, is in ieder opzicht opwaarts compatibel met de PC(/XT) reeks: het apparaat accepteert de meeste I/O kaarten van de XT, gebruikt hetzelfde DOS, en heeft dezelfde aansluitingen. Slechts de toetsenborden zijn niet uitwisselbaar. De 80286 processor staat dit alles niet in de weg: de instructieset is een superset van die van de 8088 en de CPU kan in twee modes gebruikt worden: 8088 mode (of real address mode) of in virtual mode. In de laatste mode kan de 80286 16 Mbyte adresseren.

1.5. In deel 2.....

Inmiddels hebben we het over CPU's. In deel 2 daarom een beschrijving van de 8088, het hart van de meeste XT klonen en compatibles.

Inhoud van de 12e jaargang

TECHNITRON TLP-12 LASER PRINTER
— U HEEFT EIGENLIJK GEEN ANDERE KEUZE!

- 12 pagina's per minuut (max.)
- tot 10.000 afdrukken per maand
- 8 ingebouwde lettertypes; 32 afdruk-combinaties
- unieke "FontMaker" service
- unieke "FormsMaker", formulier- en logo service
- 3 ingebouwde hardware-emulaties
- flexibele in- en uitvoer van papier

Technitron DATA

Technitron Data B.V.
Zwarteweg 110, Postbus 14,
1430 AA Aalsmeer
tel. 02977-22456
telefax 02977-40968
telex 13301

Vestigingen in:
BONDSREPUBLIEK DUITSLAND – DENEMARKEN – ENGELAND – FRANKRIJK – ITALIË – NOORWEGEN – VERENIGDE STATEN – ZWEDEN