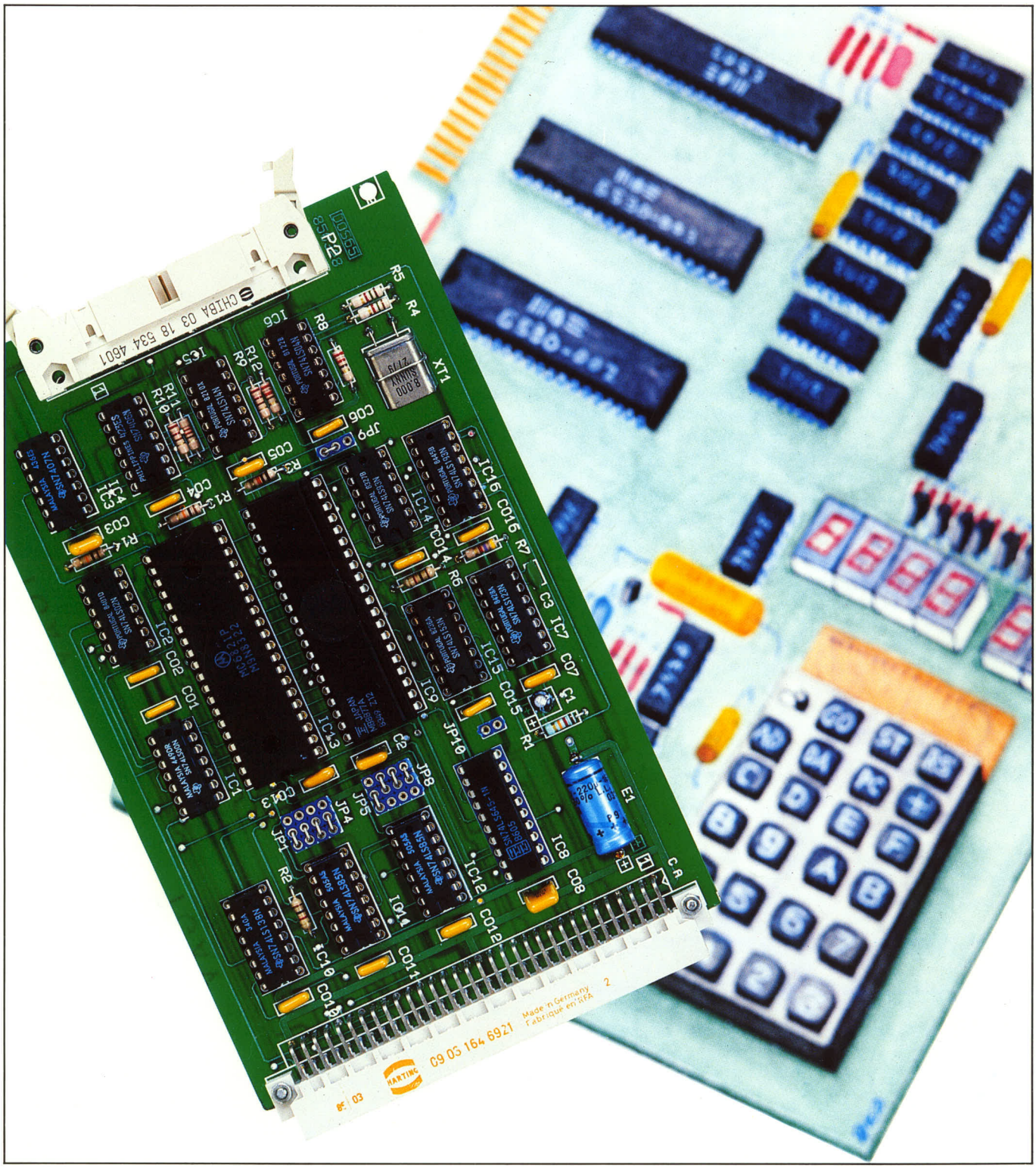
DE µP KENNER

60

Dertiende jaargang nr. 1 februari 1989



INFORMATIE.

De uP Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.

- Het stimuleren en ondersteunen van het gebruik van microcomputers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 6 maal per jaar de uP Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een software-bibliotheek en een technisch forum voor de diverse systemen in stand gehouden.

Landelijke bijeenkomsten:

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

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.

Het telefoonnummer is: 053-303902.

Software Bibliotheek en Technisch Forum:

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

Het Bestuur:

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

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:

Jan D.J. Derksen
Ed Verkadestraat 9-1
7558 TH HENGELO

Adri Hankel
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

De uP Kenner:
De uP Kenner is het huisorgaan van de KIM Gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van deze club.

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

Kopij:
Kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze kopij kan op papier of in machine-leesbare vorm opgestuurd worden aan het redactieadres. De redactie beslist, op basis van bruikbaarheid, publicatiewaarde en actualiteit of en zo ja, wanneer een ingezonden artikel geplaatst wordt.
Geplaatste artikelen blijven het geestelijk eigendom van de auteur en mogen niet zonders diens toestemming door derden gepubliceerd worden.
Helaas kan de redactie noch het bestuur enige aansprakelijk aanvaarden voor de toepassing(en) van de geplaatste kopij.

Redactie.
De redactie wordt 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

Druk:
ACI Offsetdrukkerij B.V.
Langsom 10-16
1066 EW Amsterdam

INHOUDSOPGAVE

Vereniging

Informatie 2
Uitnodiging clubbijeenkomst 5
De vergadering op 21-01-1989 21
Jaarstukken van 1988 22

Algemeen

Redactioneel 4
Van de voorzitter 4
Computers (deel 2) 36
Getallen (deel 3) 41

Communicatie

Datacommunicatie met micro's 6
Ervaring met het "Clubmodem", de BCH1200A van BESAMU 30

DOS-65 Corner

Programmeren in assembler (deel 1) 23
DOS-65 Corner (Coen Kleipool) 28
Spanningen meten met DOS-65 33

Talen/Software

Kalender.pas op DOS-65 39

MS-DOS

De IBM-PC en z'n klonen (deel 2) 49

Redactioneel

Zoals u waarschijnlijk op de cover al gezien hebt, heeft het clubblad een nieuwe naam. Degene die deze naam aan mij doorgaf, G.J.M. op der Heijde, motiveerde zijn keuze zo: Het was de KIM Kenner, het is de 6502 Kenner, het wordt de uP Kenner, het blijft dus een Kenner. Hij heeft mij met dit voorstel echter één probleem bezorgd; ik kan geen mu (griekse letter) printen. We zullen ons dus maar met de 'u' behelpen. Het blad wordt dus de uP (spreek uit: microprocessor) Kenner genoemd.

Behalve de naam van het blad, is er ook op de redactie iets gewijzigd. De redactie heeft sinds kort ook de beschikking over een MS-DOS machine. Dit geeft mij nu nog meer mogelijkheden een kwalitatief goed blad samen te stellen. Ik denk hierbij met name aan de verwerking van figuren. Ik hoop dat ik in de nabije toekomst wat tijd vrij kan maken om eens wat uit te proberen. Een tweede voordeel is waarschijnlijk het feit dat ik gebruik kan gaan maken van spellingcheckers zodat de artikelen van mijn hand in correct nederlands in het blad verschijnen. In deze uitgave is dat nog niet gebeurd, u moet het dus nog doen met de normale portie spelfouten.

Verder is het financiële jaar 1988 afgesloten. De afrekening van de penningmeester kun u in dit blad vinden en dat ziet er niet zo best uit. Wat de KIM-club nodig heeft, zijn zo ongeveer 150 nieuwe leden! Meer leden zijn natuurlijk ook welkom maar die 150 hebben we nodig om weer financieel gezond te worden. Op dit moment draaien we namelijk op de reserve die we in de betere tijden opgebouwd hebben. Dit is ook de reden geweest om de doelstellingen van de club te wijzigen.

Ik zou u twee dingen willen vragen:

Ten eerste, kunt u eens in uw omgeving kijken of er misschien potentiële leden voor de KIM-club zijn? Er moeten volgens mij in Nederland echt wel zo'n 500 mensen te vinden zijn die voor fl. 50,-- zes keer per jaar het clubblad willen ontvangen.

Ten tweede, hebt u misschien tips hoe we het ledental kunnen vergroten? Ik denk hierbij niet alleen aan acties om leden te werven, maar ook aan de inhoud van het clubblad.

Verder nog veel hobbyplezier,
Gert van Opbroek.

VAN DE VOORZITTER

De bijeenkomst bij Forbo te Krommenie was zoals vele jaren voordien ook het geval is geweest een succes. Via dit schrijven wil ik onze gastheer Co Filmer en de firma Forbo nogmaals bedanken voor het ter beschikking stellen van hun kantine tbv onze bijeenkomsten. Wij hopen nog meerdere jaren van deze gelegenheid gebruik te kunnen maken. Tijdens de bijeenkomst heeft zich een nieuwe medewerker gemeld voor het bestuur in de persoon van Jan Derksen Hiermee is het bestuursteam op volle sterkte gekomen. Mooi meegenomen is het feit dat het bestuur uitgebreid is met iemand die de gemiddelde leeftijd van ons team omlaag brengt. Jan ik wens je veel succes toe.

Een nieuwe computer

Aangespoord door de trend der vernieuwingen heb ik inmiddels mijn PC-XT compleet. Ruim een jaar ben ik in de weer geweest om alle delen van mijn computer stukje bij beetje te kopen, maar die mag er dan ook wel zijn. Ik heb mijn systeem onder andere uitgerust met een VGA kaart en een multisync monitor plus een 40mb harde schijf en een 3.5 inch floppy drive, hiermee is het systeem voor een groot gedeelte compatible met de IBM PS/2 computer.
Ik ben wel tot de conclusie gekomen dat de grafische software die nu beschikbaar is voor de gebruikers van IBM compatible systemen die uitgerust zijn met grafische kaarten van hoge resolutie of onbetaalbaar zijn of de hoge resolutie niet gebruiken, maar ik heb goede hoop dat dit niet te lang op zich zal laten wachten. Het installeren van een modem is de laatste faze zodat Jacques Banser ook mij als een notoire pottekijker kan begroeten. Verder besteed ik mijn helaas wat te weinig vrije uurtjes aan het bestuderen van het PC-DOS operating systeem. Speciaal ben ik geinteresseerd in de grafische mogelijkheden van mijn systeem, om die reden heb ik wat lektuur gekocht die mij kunnen helpen om verder te kunnen komen. Een erg interessant boek is Programmer's guide to pc & ps/2 van Richard Wilton, dit boek gaat uitvoerig in op de video performance van de EGA, VGA, HGC en MCGA kaarten.
U ziet dus, ik heb nog veel te doen.

UITNODIGING CLUBBIJEENKOMST

Datum: 18 maart 1989
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP GELDROP
tel: 040-857527

Entreeprijs: fl. 10,--

Routebeschrijving

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

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

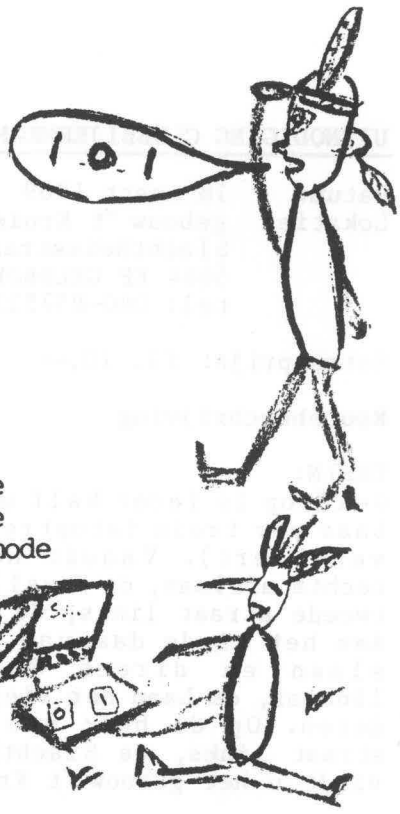
Vanaf Eindhoven door het centrum van Geldrop richting Heeze. Na winkelstraat en daarna het ziekenhuis aan de rechterzijde de eerste straat links bij de stoplichten. Dit is de Laan der vier Heemskinderen. Zie verder boven.

Programma:

9:30	Zaal open met koffie
10:15	Opening
10:30	Voordracht: Titel en spreker nog onbekend. Er wordt naar gestreefd een voordracht over de Commodore Amiga te laten houden. (Meer inlichtingen op het Bulletin Board).
11:30	Forum en markt
12:00	Lunchpauze
	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.



DATACOMMUNICATIE MET MICRO's

1. Inleiding
2. Datavernietiging
   2.1 Computervirussen
   2.2 Beveiliging van informatie
3. Bulletin Boards
4. Populaire communicatieprotocollen
   4.1 Foutcorrigerende codes
   4.2 De checksum
   4.3 CCITT-CRC-16
      4.3.1 CCITT-CRC-16 in hardware
      4.3.2 CCITT-CRC-16 in software
   4.4 Verschil checksum met crc-methode
5. Binaire files
6. Viditel aktueel

1. Inleiding

Dit artikel is te beschouwen als een vervolg op "datacommunicatie met de 6502" wat gepubliceerd is in 6502-kenner nr. 51, blz 14 e.v. In dat artikel worden de elemantaire routines voor terminal-emulatie besproken, en wat de benodigdheden zijn voor datacommunicatie. Dit artikel beschrijft hoe de meest populaire transferprotocollen zijn opgebouwd, hoe foutcorrigerende codes werken, hoe men een virus kan oplopen (!!), hoe men snel berichten op een bulletinboard plaatst, en nog een aantal andere zaken. In het eerste deel word deze materie algemeen behandeld, maar in het tweede gedeelte worden meer specifiek voorbeelden gegeven uit het dos-65 communicatie programma astrid V3.0, waar ik momenteel aan bezig ben.

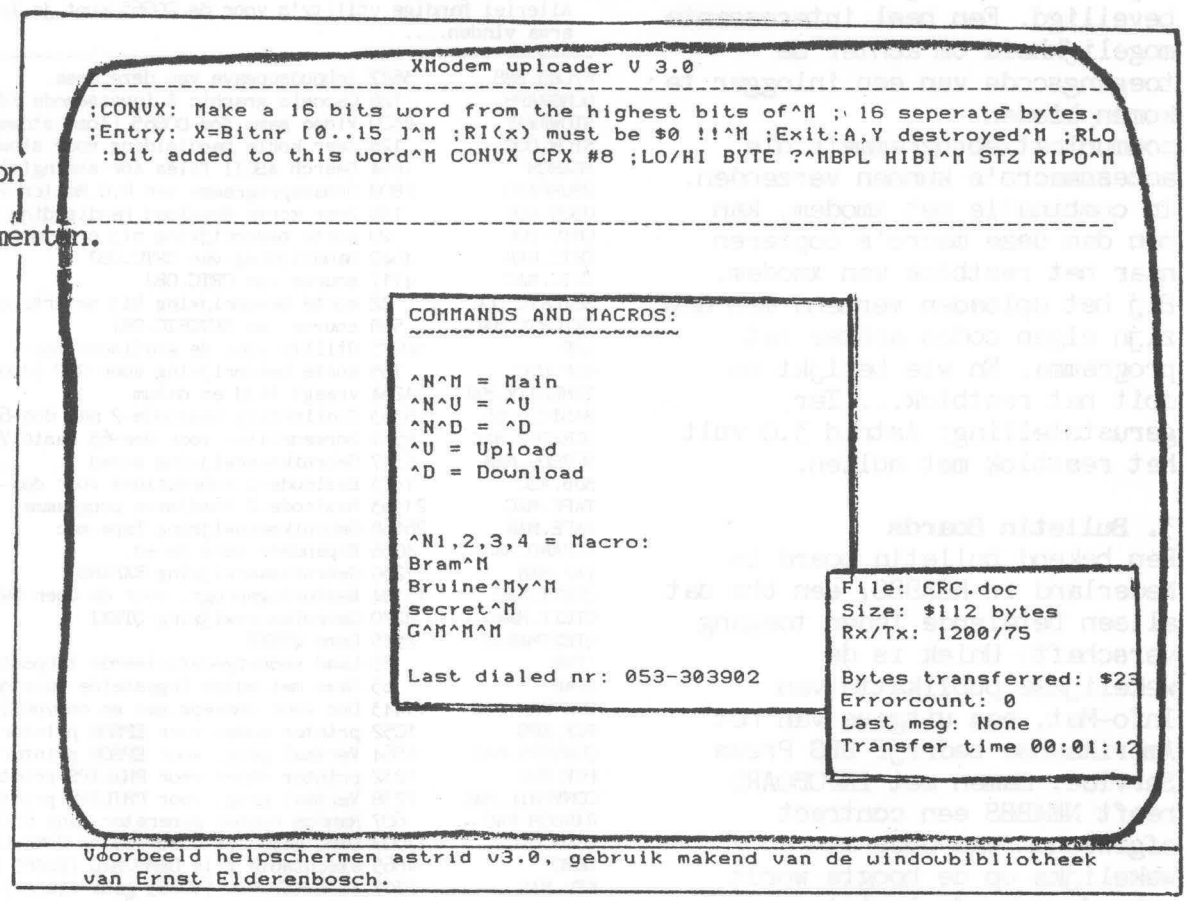
2.1 Computervirussen

MicroAids. Vele datacommunicerende computerenthousiastelingen zijn erg bang om een destructief programma binnen te krijgen van een Bulletin board. Er zijn gevallen bekend van mensen die door telesoftware gedupeerd raakten omdat er een zogenaamd virus in dat programma zat ingebouwd. Een virus haalde de Nationale pers: Als het programma gerund werd begon het de harddisk te formatteren, zodat de gebruiker opeens al zijn software kwijt was.
Wat is zo'n virus eigenlijk ? Een virus is niets meer dan een meestal klein stukje software dat andere software kan vernietigen, of de normale gang van zaken op een systeem ernstig kan verstoren. De meest eenvoudige virussen zijn de direkt aktieve virussen. Direct nadat de computer dit virus oploopt komt dit tot uiting. Men denkt bv een spreadsheet te downloaden, maar... helaas het is een formatteer of delete programma.
Dit kan heel geniepig verwerkt zijn in het programma. (Zie voorbeeld 1)
Iemand die heel goed thuis is in machinecode en zijn systeem, kan door enkele bytes rechtstreeks via de FDC op de systeemschijf te zetten, heel deze schijf onbruikbaar maken. Veel geniepiger zijn echter de virussen die zo af en toe blijk geven van hun kwade wil. Men bereikt dit door het diskoperating systeem aan te vullen met enkele bytes in de

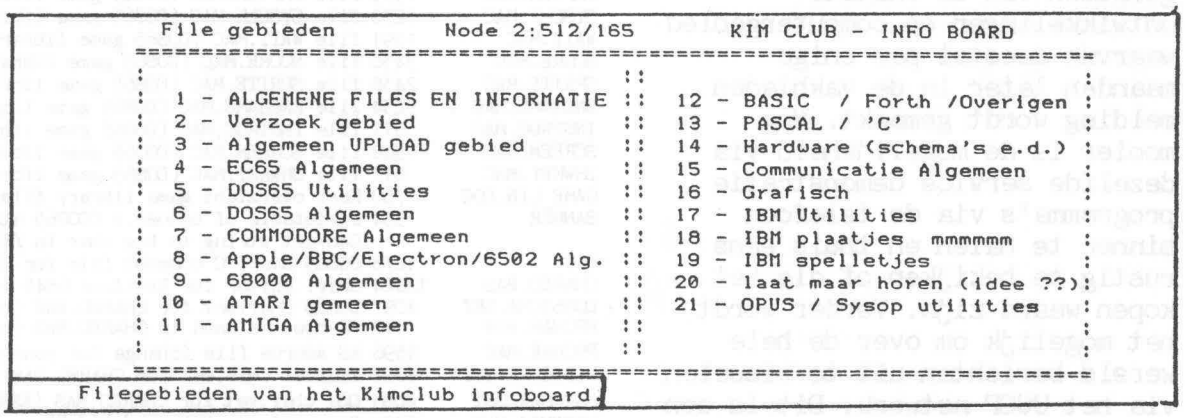
```
1000 PRINT" NUMERIEKE WISKUNDIGE OPLOSMETHODEN"
1010 PRINT" ================================="
1020 PRINT:PRINT" 1. INTEGREREN"
1030 PRINT " 2. DIFFERENTIEREN"
1040 PRINT " 3. NULPUNTEN BEPALEN"
1050 PRINT:PRINT"MAAK UW KEUZE"
--------
2030 IF KEUZE=2 THEN DOS"DEL -Y S:"
```

Voorbeeld van direkt actief virus in dos-65 basic

interruptroutine. Is de datum, maand, dag, tijd, of wat dan ook bv oneven dan word het virus aktief. De incubatietijd is in eerste instantie niet zo gemakkelijk te achterhalen, en soms als men gebruik maakt van een randomgenerator is dit zelfs onmogelijk. Meestal zit er toch een herhalingspatroon in de verschijningsmomenten. Een bekend virus is het cooky-monster. Elke keer als dit virus aktief werd, word het scherm gewist en er een monster op het scherm getekend dat om een koekje vraagt. Indien men dan COOKY intikt verdwijnt het monster weer, en komt het oorspronkelijke scherm weer terug. Maar owee.. als men het monster geen koekjes voert... blijft het monster terugkomen om data op te eten, zodat normaal werken uitgesloten is. Op grote systemen komen ook zelflerende virussen voor, die informatie over de inloggers verzamelen (inlogtijden, toegangswoorden, e.d.) en hem op de meest onmogelijke tijden lastig valt met lastige (soms rake) opmerkingen, waarvan ook de reakties van de gebruiker weer worden opgeslagen, zodat er een volgende keer weer meer te 'zieken' valt, enz. enz.



Voorbeeld helpschermen astrid v3.0, gebruik makend van de windowbibliotheek van Ernst Elderenbosch.



Filegebieden van het Kimclub infoboard.

2. Beveiliging van informatie

Sommige softwarebedrijven beveiligen hun programmatuur tegen illegaal copieren door een soort virus over te dragen zodra er gecopieerd word. Dit virus is meestal niets meer dan het overschrijven van een essentieel programmadeel met de tekst waar men een legale versie kan kopen. Met de opkomst van betaalsoftware dient men goed zijn toegangscodes geheim te houden, omdat anders iemand anders een heleboel software kan downloaden

op kosten van de sloddervos die zijn codes niet goed heeft beveiligd. Een heel interessante mogelijkheid om achter de toegangscode van een inlogger te komen, bieden communicatieprogramma's die accessmacro's kunnen verzenden. In combinatie met xmodem, kan men dan deze macro's copieren naar het restblok van xmodem. Bij het uploaden verzend men dan zijn eigen codes achter het programma. En wie bekijkt nu ooit het restblok... Ter geruststelling: Astrid 3.0 vult het restblok met nullen.

3. Bulletin Boards

Een bekend bulletin board in Nederland is NEABBS, een bbs dat alleen betalende leden toegang verschaft. Uniek is de wekelijkse publikatie van Info-Mat, een uitgave van het Amerikaanse bedrijf BBS Press Service. Samen met INFOBOARD heeft NEABBS een contract afgesloten waardoor men wekelijks op de hoogte wordt gehouden van de laatste ontwikkelingen op computergebied waarvan meestal pas enige maanden later in de vakbladen melding wordt gemaakt. Nog mooier is de mogelijkheid via dezelfde service demonstratie programma's via de telefoon binnen te halen en thuis eens rustig te bekijken of die het kopen waard zijn. Verder wordt het mogelijk om over de hele wereld berichten uit te wisselen via het UUCP netwerk. Dit is een wereldomvattend netwerk van UNIX computers. Bij studenten waarschijnlijk wel bekend van USENET die daar ook bij is aangesloten. Dit netwerk is niet zozeer een koppeling van allerlei bulletin board systemen zoals Fido en Opus, maar een wijd verbreid net van allerlei wetenschappelijke en industriele instellingen. Bepaalde systemen zijn echt bedoeld als conferentie systeem om informatie uit te wisselen over

```
File area £  5 ... DOS65 Utilitie  download area.  23-11-88
--- DOS65 utility area ------------------------------------
  Allerlei handige utility's voor de DOS65 kunt je in deze
  area vinden....
-------------------------------------------------------------
FILES.BBS       3622 Inhoudsopgave van deze area.
NORGRAPH         128 Cancels graphic & inversemode /dos 65 mode: c b
STOWAWAY        8320 Video game for DOS65 (load stowaway G 5000)
STOW.DOC         128 Zeer korte handleiding voor stowaway download
SEARCH          1664 Search ASCII files for string(s) (DOS65)
USURPATO        7808 Schaakprogramma van H.G.Muller voor DOS65
USUR.DOC         128 Zeer korte download handleiding bij USURPATOR
CRTC.DOC          20 korte beschrijving bij crtc.obj
CRTC.HAN        4640 Handlijding van CRTC.OBJ
CRTC.MAC        4717 source van CRTC.OBJ
SETCRTC.DOC       22 korte beschrijving bij setcrtc.obj
SETCRTC.MAC      558 source van SETCRTC.OBJ
GDP            10975 Utility voor de grafische kaart van Elektuur
GDP.DOC          155 korte beschrijving voor GDP (door A. Hankel)
TIMEDATE.MAC   11264 vraagt tijd en datum
BASICO.D65      6363 Toelichting basicode-2 met dos-65
SCRED52.MAC     8531 Screeneditor voor dos-65 basic V2.00
SCRED5.MAN      5347 Gebruiksaanwijzing scred
SUB.ASC         1423 Basicode-2 subroutines voor dos-65 basic V2.00
TAPE.MAC       21363 Basicode-2 load/save programma
TAPE.MAN       26168 Gebruiksaanwijzing Tape.mac
EXPAND.MAC      4066 Expandeer tabs in ed.
TAB.MAN         1230 Gebruiksaanwijzing EXPAND
QTEXT.MAC      13184 Besturingsprogr. voor de Quen DWP 1120 printer
QTEXT.MAN       8270 Gebruiksaanwijzing QTEXT
QTEXTMAN.Q      2545 Demo QTEXT
LTAB              55 Laad voorafgedefinieerde tabposities met ED
STAB              65 Save met edtab ingestelde tabs op disk
CONVPRN.DOC    11443 Doc voor conveps.mac en convphil.mac, 6502 kenner nr 55
PCF.EPS         1052 printer codes voor EPSON printer
CONVEPS.MAC     6564 Vertaal prog. voor EPSON printer
PCF.PHI         1232 printer codes voor PHILIPS printer
CONVPHIL.MAC    7238 Vertaal prog. voor PHILIPS printer
RANDOM.MAC       607 Random number generator (any 65(C)02 system)
DEMO.MAC       17007 Demo game for DOS65 using GAME LIBRARY
DEMO            4865 executable file DEMO.MAC (DOS65 Load 5000, G 5000)
KEY.MAC         7507 file KEY.MAC (DOS65 game library)
JMPVAR.MAC      1579 file JMPVAR.MAC (DOS65 game library)
SYSCLK.MAC      1270 file SYSCLK.MAC (DOS65 game library)
WAIT.MAC        1591 file WAIT.MAC (DOS65 game library)
SCORE.MAC       3436 file SCORE.MAC (DOS65 game library)
SPRITE.MAC      2438 file SPRITE.MAC (DOS65 game library)
ENDGAME.MAC     3234 file ENDGAME.MAC (DOS65 game library)
INSTRUC.MAC      611 file INSTRUC.MAC (DOS65 game library)
SCREEN.MAC      1508 file SCREEN.MAC (DOS65 game library)
SHWKEY.MAC       617 file SHWKEY.MAC (DOS65 game library)
GAME_LIB.DOC     750 Kort overzicht game library files
BANNER           594 executable of banner.c (DOS65 start at $0200)
                     Convert string to big char in file
CCC             1278 DOS65 small C command file for Compiler/Assembler/Run
CHARED.BAS     12084 Char. editor for Elektuur 6845 video EPROM
DOS65CHA.SET    4097 DOS65 Char.set for CHARED.BAS (ready for EPROM)
PRCHAR.BIN       340 MT-routine used in CHARED.BAS (printer dependent)
PRCHAR.MAC      5396 AS source file (change for your printer)
VIDICHAR.SET    4097 Viditel char.set for CHARED.BAS (EPROM readv)
OSICHAR.SET     4081 OSI char.set for CHARED.BAS (EPROM ready)
LEESMIJ.MAN     6484 Documentatie behorend bij RR
RR              2346 Renumber basic programma's, Kim Kenner nr 65
RR.DMP         10725 Hex dump van RR
RR.LST         39119 Gecompileerde versie van RR.MAC
RR.MAC         16601 Source file van RR
README          3041 Handleiding voor onderstaande files
EDDCMD.MAC      3585 Keyboard tabel (hoe is een toets gedefinieerd)
EDDCMD.BIN       223 Gecompileerde versie
EDITOR.DFT      9116 Oorspronkelijke Editor
EDITOR          9116 Nieuwe versie
MAKEED           462 Macro om de benodigde commando's uit te voeren

File area £  5 ... DOS65 Utilitie  download area.
Type ½?' by itself for help
A)rea change  L)ocate       F)ile titles  T)ype (show)  G)oodbye      U)pload
D)ownload     S)tatistics   M)ain menu    C)ontents
Select: ?
```

programmeren, computertalen en diverse soorten van hardware en software zaken zoals data communicatie en digitale beeld verwerking. Fido is echt een netwerk van Personal Computers met systeem beheerders die als hobbyist mee werken onder beheer van de HCC (Hobby Computer Club) Opus is een gelijksoortig netwerk, maar dan onder beheer van sysops die niet verbonden zijn aan de HCC. Effectief zijn de advertenties die gebruikers zo door het hele netwerk verspreiden. Een bericht wordt via 'echo-mail' naar alle aangesloten bulletin boards en computersystemen verstuurd, zodat er een erg grote kans is dat je ook verkoopt wat je aanbiedt of vindt wat je zoekt. Ook bij het oplossen van hard en softwareproblemen bereikt men op deze manier een groot aantal fanatiekelingen en deskundigen. Voor 6502'ers is het kim-info-board het meest aantrekkelijke board, vooral omdat er een berichtengebied voor de leden onderling aanwezig is, waar men informatie kan uitwisselen, iets kan vragen, oplossen, verkopen, te koop vragen, of gewoon voor de gezelligheid een conversatie beginnen kan. In het kader van dit artikel staat een voorbeeld van Dit berichtengebied. Ook is een overzicht van het filegebied opgenomen. Het maandblad 'Byte' drukt iedere maand de meest interessante 'brief-wisseling' van haar bbs af in de rubriek 'best of bix'. (U.S.A.) Ook radio/tv omroepen gaan het medium bulletin-board benutten. Zo verstrekt de NOS beeldkranten en basicodeprogrammatuur via hun fido. Het electronicaweekblad Radio Bulletin heeft een vragengebied op fido NOS (035-45395, msg area £1) De NCRV (079-413921) heeft een bulletin-board waarvan men de teksten kan downloaden van een uitgezonden programma.

```
Inlogscherm!
* Network Address 2:512/165.0 Using BinkleyTerm Version 2.00

        Welkom bij Kim Club INFO BOARD op PCC node 2:512/165
                Nu draaiend onder OPUS V1.03b
            Bekijk de ECHO-MAIL gebieden eens....!!

Please press your Escape key to enter the BBS, or wait a few moments.
Thank you.  Loading KIM CLUB INFO BOARD now.  Please wait...

OPUS-CBCS v1.03b
```

```
Uitlogmsg, na G-commando!

Disconnect [Y,n,?=help]?
Leave a note to Jacques Banser [y,N,?=help]?

Tot ziens Bram

Het is nu  1:03:23 , 05 Nov 88
In deze sessie was het 2 minuten dat je met ons contact hebt gehad !
Dit was jouw 49ste verbinding met KIM CLUB INFO BOARD

Totaal   Upload : 14
Totaal Download : 2

Tot de volgende keer bij : KIM CLUB INFO BOARD  node 2:512/165
```

4. Populaire communicatieprotocollen

De protocollen die momenteel het meest in de belangstelling staan zijn viditel*, kermit*, xmodem, ymodem en zmodem. Vooral XModem (eXtended Modem protocol van Ward Christensen) word erg veel gebruikt. Sommige bbs'en draaien nog een oude versie van dit protocol, (modem, modem7) maar het is allemaal een pot nat. Met xmodem kan men willekeurige data overbrengen in blokken met een vaste bloklengte van 128 bytes. Ieder blok wordt voorafgegaan door een header en een set bloknummers. Het blok wordt

```
It was always thus; even if we're not, it would inevitably have
been always thus.
DEAN LATTIMER

"One Galileo in two thousand years is enough"
POPE PIUS XII

Don't let your mouth write no check that your tail can't cash.
BO DIDDLEY
```

Enkele spreuken waar het kim-info-board inloggers op tracteert.

*Deze protocollen zijn reeds uitvoerig behandeld in vorige uitgaven van dit blad.

afgesloten met een checksum, die bestaat uit de som van de databytes.

bytseq:	1	2	3	4.....131	132
	<SOH>	<BLKNR>	<CMPBLKNR>	<DATABLOK>	<CHECKSUM>
nr. bytes	1	1	1	128	1

Fig: Xmodem blokformaat. (SOH is altijd $01) Indien Xmodemc gebruikt word, bestaat de checksum uit twee bytes, te weten CRCHI en CRCLO.

De ontvanger fungeert als foutdetector en corrector. De foutdetectie vind plaats door vergelijking van de verwachte header en bloknummers met de ontvangen waarden, en door vergelijking van de berekende checksum met de ontvangen checksum. Foutcorrectie gebeurt met de ACK- en NAKsignalen. Is er een fout ontdekt dan zal de ontvanger een NAK naar de zender zenden om aan te geven dat het blok helaas met fouten is ontvangen. De zender zal dan dat blok nog een keer verzenden. Is het blok foutloos ontvangen dan wordt een ACK verzonden naar de zender. Als er tien fouten optreden wordt het transferproces afgebroken. Als optie kent dit protokol de crc-foutcontrole. Indien men met CRC wil downloaden, moet men als startteken i.p.v. een NAK een hoofdletter 'C' naar de host zenden. Indien de host positieve response geeft, dan kan met CRC gewerkt worden. Indien er echter een NAK terugkomt, word automatisch terug gesprongen naar de standaard checksum methode. De bloknummers starten bij $01, en worden daarna modulo 256 steeds met 1 verhoogt, dit betekend dat na blok $FF gecontinueerd word met blok $00. Xmodem kent ook een time-out mechanisme om te voorkomen dat twee computers eindeloos op elkaar blijven wachten.
De nadelen van xmodem zijn:
-Transport van binaire data over 7-bits lijnen is niet mogelijk,
-Controlkarakters worden niet omgezet naar printbare karakters, (geen bezwaar voor Dos-65 systemen)
-Er kunnen geen meerdere files met een commando overgedragen worden,

****
* N.O.S.- HOBBYSCOOP *
* RADIOTEKST *
* BEELDKRANT MET DAARIN *
* OPGENOMEN DE *
* LANDELIJKE COMPUTER AGENDA *
****
nr.177, maa 24 okt '88

GEBRUIK FRANSE MINITEL NEEMT AF
De Franse videotex dienst Minitel wordt geconfronteerd met een afnemend gebruik. Werd er in het tweede kwartaal van 1987 nog 98 minuten per maand per Minitel van de diensten gebruik gemaakt, in het tweede kwartaal van dit jaar is dat gedaald tot 78 minuten. In het eerste kwartaal is in vergelijking met '87 al een forse daling opgetreden van 106 naar 84 minuten per maand. Ook het tweede, derde en vierde kwartaal van '87 zagen met resp. 98, 84 en 85 minuten het gebruik dalen, zo concludeert de studie 'Teletel, lotgevallen van een grootschalig videotex project'. Sinds begin 1987 is vooral het gebruik op de particuliere markt gedaald. Er is daarentegen een forse toename te constateren in het zakelijke gebruik van Minitel. France Telecom maakt na 7 jaar Minitel nog steeds verlies. Per maand worden er 100.000 gratis terminals bij particulieren geplaatst. De studie 'Teletel, lotgevallen van een grootschalig videotex project' is verkrijgbaar bij onderzoeks bureau Telematique Internationale Professionelle en kost f375. Informatie: 09-33-1-47885048. (Uit: Telecom)

THUISGEBRUIK PC'S IN VS
Het aantal PC bezitters in de Verenigde Staten is gelijk gebleven met vorig jaar: 15 procent van de bevolking werkt thuis met een personal computer. Dat meldt het marktonderzoeks bureau Dataquest na een onderzoek onder 2700 Amerikanen naar het gebruik van personal computers. Wel is er een wijziging in de soort systemen, die thuis staan. Vorig jaar kostte 37,3 procent van de systemen meer dan $1000. Dit jaar is dit percentage gestegen naar de 43,7 procent. Veel van de gebruikers hadden vorig jaar nog een systeem van onder de $500, maar kochten dit jaar een nieuw systeem van boven de $1000. De onderzoekers menen dat veel van de machines nu thuis worden gebruikt voor zakelijke toepassingen. 23,8 procent van de ondervraagden had een Commodore computer staan, gevolgd door een Apple (17,4 procent) en IBM (14,3 procent). Andere veelgenoemde leveranciers zijn Tandy, Texas Instruments, Compaq, Leading Edge, Atari, Kaypro en AT&T. Tekstverwerking blijft de meest genoemde toepassing, ook al is er vergeleken met vorig jaar een lichte daling merkbaar. (Uit: Telecom)

De hobbyscoop beeldkrant is zowel via de radio als met de telefoon te ontvangen. (Fido NOS)

-Er is geen restblok aan het einde, zodat er overtollige data verstuurd word.

Ymodem is een uitbreiding van Xmodem. Nu worden er geen datablokken van 128 bytes gebruikt, maar blokken van 1 kbyte. Het voordeel hiervan is dat men minder ACK/NAK/SOH/BLOCKNUMBERS en CHECKSUMS hoeft over te dragen. Dit gaat uiteraard alleen goed bij een goede telefoonverbinding. Als er iets fout gaat, is de snelheidswinst snel verloren. Na een fout schakelt Ymodem over op een kleinere blokgrootte. Blijven er fouten optreden dan verandert Ymodem gewoon weer in Xmodem. Ymodem maakt altijd gebruik van de CCITT-16-CRC checksum methode. Ymodem vangt een groot deel van de nadelen van xmodem op. Door voor SOH geen $01 maar $02 te verzenden, kunnen blokken van 1kbyte overgedragen worden. Bovendien kan men blokken van 128 bytes (SOH=$01) mixen met blokken van 1kbyte (SOH= $02), dit kan nuttig zijn als er regelmatig fouten optreden bij de overdracht van grote blokken, en is uitermate efficient om het laatste deel van een file te verzenden. Indien het laatste blok maar een enkel significant byte bevat is het natuurlijk niet verstandig om dit als een blok van 1kbyte te verzenden. Ymodem begint met bloknr. $00. In dit blok bevind zich informatie die te vergelijken is met het send-init blok van kermit, zoals:
-Filena(a)m(en) van de te overdragen files. Meerdere files kunnen getransporteerd worden ! (batch file transmission)
-De File-lengte kan opgegeven worden, zodat de ontvangende Ymodem de data om een restblok vol te maken, kan weggooien.
-Een string met de tijd/datum in GMT.

bytseq:	1	2	3	4.....1027	1028	1029
	<$02>	<BLKNR>	<CMPBLKNR>	<DATABLOK>	<CRCHI	CRCLO>
nr. bytes	1	1	1	1024	2	

Fig: Ymodem blok met header SOH=$02.

Hoewel Xmodem tekortkomingen heeft, is het toch nog steeds het populairste protocol, mede door zijn eenvoudigheid, en vrij eenvoudige manier van implementatie op ieder computersysteem. Doordat zowat ieder Bulletin-board xmodem ondersteunt word deze populairiteit in stand gehouden. Ymodem vangt een groot deel van de tekortkomingen van xmodem op, en wel op een zodanige manier dat Ymodem veel op Kermit gaat lijken, maar kan niet op alle computers geimplementeerd worden, bijvoorbeeld omdat die niet in staat zijn om met hoge baudrates 1kB bursts op te slaan in hun buffer. Een nadèel blijft dat binaire data niet over 7-bits lijnen verstuurd kan worden. Voor micro [--] micro communicatie is dit echter geen enkel probleem. Kermit kent trouwens al deze problemen niet, en is dan ook het meest ideale protocol wat dit betreft. Helaas is Kermit erg traag vanwege het omzetten van controlkarakters naar printbare karakters, de kleine pakket(=blok)grootte, en i.p.v. een enkele stuurkarakter (bv ACK,NAK) een compleet blok met die informatie. Kermit kent wel variabele bloklengtes, zodat geen data verzonden word die niet tot de file behoort.

Tegenwoordig komt men ook nogal eens protocollen als Zmodem en Jmodem tegen. Helaas heb ik hier geen enkele documentatie over.

4.1 Foutcorrigerende codes

Hoe is dat mogelijk. Een code die zijn eigen fouten corrigeert. Uit het voorgaande blijkt dat dit mogelijk is door ontvangen redundante informatie (bv het checkbyte bij xmodem en kermit) te

vergelijken met de berekende waarde, en indien dit niet overeenstemt word de zender gevraagd opnieuw de gewenste data over te zenden. Zo krijgt men toch de goede data binnen.

4.2 De checksum

De meest toegepaste methode is de checksummethode. Xmodem verdeeld een file

SELECTIE UIT HET KIM=INFO=BOARD BERICHTENGEBIED:

Msg.area 1 ... KIM CLUB (6XXXX) Prikbord, Voor leden onderling.

From: Roelof Heuvel Rec'd
To: Jacques Banser Msg £17, 30-Jul-88 20:21
Subject: 65xxxx

Hebben jullie ook nag een area die over die raatselachtige 65618 (of hoe hete die ook alweer, die 16-biter) gaat. Je hoort, leest en ziet er toch niet meer zoveel van. Of kijk ik niet meer op de goede plaatsen? Groetjes, Roelof,..

From: Fred-Jan Kraan
To: All Msg £7, 03-Aug-88 13:27
Subject: Atom rom

Gevraagd: een Acorn Atom floating point rom (4 KByte). dit voor de completering van mijn Atom. Ik Heb de mogelijkheid om Eproms te copieren.

From: Antoine Megens Rec'd
To: Gert Van.opbroek Msg £5, 02-Oct-88 00:05
Subject: Bouncing Babies & QIX

De bouncing babies die op de PC draait is niet van mij, al lijkt het er wel erg veel op. Uit het C programma blijkt al dat het spel op zich helemaal niet zo'n moeilijk programma is. Wat QIX betreft ligt dat wat moeilijker.... Ik heb al wel een QIX die vrolijk over het scherm dartelt, wat al niet meeviel want in eerdere versies liep hij zichzelf steeds vast in de hoek of in z'n eigen staart. De eerste versie was ook in C geschreven maar toen bleef er te weinig tijd over voor de rest van de bewegingen zoals de 'fuses' en de speler. Dus toen maar opnieuw begonnen in AS. Op het moment ligt het echter even stil door gebrek aan inspiratie.....

Groeten, Antoine

From: Bram Bruine
To: All Msg £6, 10-Oct-88 11:09
Subject: crc xmodem

IN een C-listing van CRC-CCITT staan o.a. de volgende regels:

```
crc [[= 1;
crc += (((c [[=1) & 0400) != 0);
```

Wat betekend deze expressie ? Graag reacties.
Of wie heeft een prgrm om de crc te berekenen ?
vr gr.

From: Jac Kersing
To: Bram Bruine Msg £8, 14-Oct-88 16:06
Subject: Re: crc xmodem

Ha! Dat is C op z'n best...

Het betekend het volgende:

```
  crc [[= 1;
```

crc wordt eenmaal naar links geshift. Zelfde als crc = crc [[1 (en in 6502 iets als ASL als ik me niet vergis, in ieder geval dus alle bits eenmaal naar links en een nul in het laagste bit).

```
crc += ((( c [[= 1 ) & 0400 ) != 0 );
```

Hier worden een aantal dingen gedaan, uiteen gerafeld is het: c [[= 1; (Dus schuif alles eenmaal naar links) (c & 0400) != 0; (Dit levert een 1 als de AND van c (de nieuwe) en 0400 (octaal) ongelijk is aan 0)

```
crc += ... ; (En hier wordt uiteindelijk de crc
```

verhoogd met het resultaat van bovenstaande).

Hopelijk is het nu iets duidelijker.
Succes,

bibi,
Jac

From: Maarten Haakman
To: All Msg £19, 04-Nov-88 07:00
Subject: HAAKMAN-HOST

Hallo allemaal,
dit weekend staat HAAKMAN-HOST tot jullie beschikking er word gewerkt met een verzoek aria waar de meest gevraagde programma's staan voor de rest werkt werkt het als een fido alleen er zijn geen priorities dus inloggen maar. alles kan maar officiele programma's worden niet doorgegeven aan het down gebied je mag ze natuurlijk wel uploaden als ze tenminste compleet en virusvrij zijn, ze worden onder een strenge kontrole goedgekeurd groetjes van HAAKMAN-HOST 033-722175

From: Frank Fiselier Rec'd
To: Jacques Banser Msg £17, 06-Nov-88 13:52
Subject: Kosten

Die inleiding over de Kim club is wel aardig maar ben je niet wat vergeten?
Wat zijn de kosten die verbonden zijn aan het lidmaatschap? Of zijn die zo hoog dat je ze niet durft te vermelden?

Msg.area 3 ... Berichten van en voor de Sysop

From: Jacques Banser
To: Adri Hankel Msg £24, 17-Sep-88 00:38
Subject: Re: LEZEN BERICHTEN

] stel,dat er post voor me is, bijv in 3:10. ik ga naar
] message area 3. hoe kan ik nu direct bericht 10 lezen
] m.b.v. de beschikbare commando's ?
] (dus niet met NEXT, LIST enz.

Gewoon, 10 intikken en return, je krijgt dan automatisch bericht 10 op je scherm ....

Doeiiiiii Jacques

From: Herman Hek
To: Jacques Banser Msg £95, 04-Nov-88 22:29
Subject: Viditel-modem

Jacques,
Je vertelde mij dat jij nog een Viditel-modem had liggen. Is deze misschien te koop?
Mij kameraad heeft wel belangstelling
Dus als het te koop is laat het mij even weten.
En tevens wat de prijs is.

in mootjes van 128 bytes. Na iedere 128 bytes volgt een checksum over de voorgaande 128 databytes. De checksum bestaat uit de 8-bit optelling van deze databytes, een carry word verwaarloosd, is de uitkomst groter dan 1 byte dan word ook de overflow weggegooid. Een voorbeeld hiervan vind men in $5. (Test.bin blok) De checksum over dat blok is 3D. Met het onderstaande programma kan men deze checksum berekenen.

Ook de basic-kermit kent een soortgelijke checksum methode. Voordat deze checksum achter het datablok word geplakt

```
                ;Entry: A = byte to calculate
                ;Exit: updated checksum
CHECKS          CLC                             ;Calculate checksum
                ADC     CHSUM
                STA     CHSUM
                RTS
```

moet het eerst nog bewerkt worden door de subroutine KERCAL. Deze routine telt de hoogste 2 bits op bij de laagste bits. Daarna worden deze twee hoogste bits nul gemaakt en de zo ontstane checksum word dan -na printbaar te zijn gemaakt met TOCHAR- achter de datastroom gezet. Kermit kan gewoon gebruik maken van subroutine CHECKS, en pas aan het einde van een packet converteert men de checksum met KERCAL naar kermit-formaat.

```
        ;Calculate final checksum (chk 1)
        ;entry: A=last updated checksum
        ;exit:  A=tochar(endsum)
KERCAL  STA     SCHK                    ;Save checksum
        AND     #$C0                    ;Isolate b6,7
        LSRA                            ;Shift b6,7 to b1,0
        LSRA
        LSRA
        LSRA
        LSRA
        LSRA
        CLC
        ADC     SCHK
        AND     #$3F                    ;Strip b6,7
        STA     SCHK
        ;JMP    TOCHAR                  ;Tochar +rts

        ;Convert controlchar to printable char by adding $20.
        ;entry: A=controlchar.
        ;exit: A=printable char.
TOCHAR  CMP     #$5F
        BPL     1.F                     ;No controlchar
        CLC
        ADC     #$20
1       RTS
```

Kermit kent ook nog de 2-bytes checksum. Deze word bijna nooit gebruikt, maar als men het wil gebruiken moet men dit in het 'send-initiate'packet aangeven*. Indien de host deze methode ondersteund word de checksum nu in twee bytes berekend volgens onderstaand programmadeel:

```
/*  C H K 2  --  Compute the numeric sum of all the bytes in the packet.  */
unsigned
chk2(pkt) CHAR *pkt; {
    long chk; unsigned int m;
    m = (parity) ? 0177 : 0377;
    for (chk = 0; *pkt != '\0'; pkt++)
      chk += *pkt & m;
    return(chk & 07777);
}
```

4.3 CCITT-CRC-16

De Cyclic Redundancy Check is een bitgeorienteerde methode. Een blok data word gezien als een bitstroom. De CRC-generator voert voor ieder bit een wiskundige bewerking uit op de bitstroom. Het resultaat (een 16-bits check sequence) word aan het einde van de bitstroom (datablok) meegestuurd. Het ccitt-crc-16 polynoom kan men alsvolgt beschrijven:

$$CRC(x) = x^{16} + x^{12} + x^{5} + 1. \quad (x=bitnr)$$

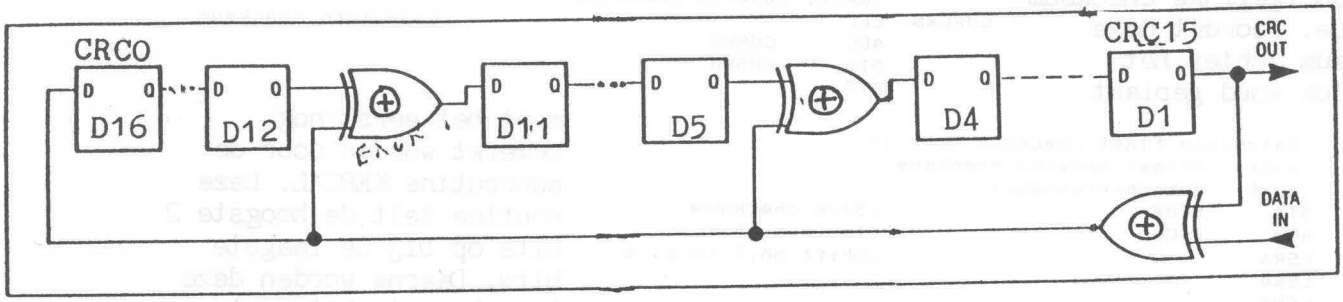
Voor binaire getallen valt hier het 'magische getal' $1021 uit te halen, dat van belang is voor de berekening van de CRC met software.

```
15141312  1110 9 8   7 6 5 4   3 2 1 0=16
 0 0 0 1   0 0 0 0   0 0 1 0   0 0 0 1        $1021
```

Het principe van CRC-berekeningen komt neer op het delen van de bitstroom (the huge number) door een constante. (De polynoom die aangeeft welke bits geset zijn in de deler) Het quotient van de deling word genegeerd, maar de rest (remainder) word gebruikt als block-check-character.

4.3.1 CCITT-CRC-16 in hardware

Het makkelijkst realiseert men een crc-checker met hardware, bestaande uit een cyclisch schuifregister en enkele exorpoorten. Iedere keer als er een nieuw bit binnenkomt schuiven alle voorgaande bits een positie op en komt het nieuwe bit op de laagste positie. (b0)



Uit dit schema blijkt dat de CRC van $00 ook $0000 is. Schuiven we dus uitsluitend een enkel '1'-bit in het schuifregister, dan worden alleen de uitgangen waarvoor een exorpoort staat hoog gemaakt. Op deze manier vind men het getal $1021.

```
$01   ↓          ↓ crc-register: ↓
IN    0 1 2 3  4 5 6 7  8 9 10 11  12 13 14 15
--    0 0 0 0  0 0 0 0  0 0 0  0   0  0  0  0
0     0 0 0 0  0 0 0 0  0 0 0  0   0  0  0  0
0     0 0 0 0  0 0 0 0  0 0 0  0   0  0  0  0
      ...
1     1 0 0 0  0 1 0 0  0 0 0  0   1  0  0  0
      LSB                                  MSB
De Crc word dan (MSB voorop): 0001 0000 0010 0001 ($1021)
```

```
$80   ↓          ↓ crc-register: ↓
IN    0 1 2 3  4 5 6 7  8 9 11 11  12 13 14 15
--    0 0 0 0  0 0 0 0  0 0 0  0   0  0  0  0
1     1 0 0 0  0 1 0 0  0 0 0  0   0  1  0  0
0     0 1 0 0  0 0 1 0  0 0 0  0   0  0  1  0
0     0 0 1 0  0 0 0 1  0 0 0  0   0  0  0  1
0     0 0 0 1  0 0 0 0  1 0 0  0   1  0  0  0
0     1 0 0 0  1 1 0 0  0 1 0  0   0  1  0  0
0     0 1 0 0  0 1 1 0  0 0 1  0   0  0  1  0
0     0 0 1 0  0 0 1 1  0 0 0  1   0  0  0  1
0     0 0 0 1  0 0 0 1  1 0 0  0   1  0  0  1
De Crc word dan: 1001 0001 1000 1000 ($9188)
```

Moderne seriële bouwstenen, zoals de MC68652, hebben crc-logica die d.m.v. een register gelezen kan worden, die door een code in het commandoregister te zetten de Crc-methode selecteert, de crc op nul zet, of het crc-register voorziet van allemaal enen. De meeste asynchrone ACIA's zoals de MC68681 en de 6551 hebben geen ingebouwd crc-register. Men moet dan de crc met software berekenen, en daarna per byte transporteren met de acia. (zenden)

MULTI-PROTOCOL
COMMUNICATIONS CONTROLLER (MPCC)

The MC2652/MC68652 MPCC formats, transmits, and receives synchronous serial data while supporting Bit-Oriented (BOP) or Byte-Control (BCP) protocols. The parallel bus of the MPCC readily interfaces with M6800 and M68000 Microprocessor Families as well as many other 8- or 16-bit processors. Typical applications include intelligent terminals, front-end communications, remote-data concentrators, communication test equipment, and computer-to-computer links.

- DC to 2 Mbps Data Rate
- Bit-Oriented Protocols (BOP): SCLC, ADCCP, HDLC, X.25
  - Character Length—1-to-8 Bits
  - Address Comparison
  - Automatic Detection and Generation of Special Control Characters, i.e., FLAG, ABORT, GA
  - Automatic Zero Insertion and Deletion
  - Short Last Character
  - Idle Transmission of FLAG or ABORT Characters
  - Automatic Generation and Checking of CRC-CCITT FCS
- Byte-Control Protocols (BCP): DDCMP, BISYNC (external CRC)
  - Character Length—5-to-8 Bits
  - SYNC Generation Detection and Stripping
  - Idle Transmission of SYNC or MARK Characters
  - Automatic Generation and Checking of CRC-16 or VRC
- Maintenance Mode for Self-Checking
- Bidirectional, Three-State, 8- or 16-Bit Data Bus
- TTL Compatible
- Compatible with MC2653/MC68653 Polynomial Generator Checker

4.3.2 CCITT-CRC-16 in software

Het is mogelijk om de schuifregisterfunctie om te zetten in software. Dit betekend dat men voor ieder bit een vrij ingewikkelde berekening moet uitvoeren. Gelukkig zijn er algoritmes om de crc per byte te berekenen. Uit 'crc in hardware' is al duidelijk geworden dat men door steeds een bit op te schuiven en door op de posities waar een exorpoort staat het voorgaande bit mee te exoren met de ingangsdata en b15, men na 8 keer schuiven de nieuwe crc heeft. Dit is een vrij bewerkelijke operatie in software, omdat voor ieder bit 16 shifts en 3 exorbewerking moeten worden uitgevoerd.

Beter is het om de crc in een keer over een byte te berekenen. Dit heeft als voordeel dat men tegelijk 8 shifts kan doen (dwz: lda crclo, sta crchi) maar men moet wel meerdere exorbewerkingen uitvoeren. Om de crc per byte te berekenen gaan we acht keer schuiven en nemen alle informatie vanaf de allereerste shift steeds mee naar de volgende shift. Alle variabelen in de vertikale kolommen in onderstaande tabel dienen met elkaar ge-exorred worden. Hier gaan we uit van het getal $1021, waarvan een '1'-bit aangeeft dat er een exorpoort staat.

shift	in	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	--	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
1	M0	M0 C0	C15	C14	C13	C12	C11 M0 C0	C10	C9	C8	C7	C6	C5	C4 M0 C0	C3	C2	C1
2	M1	M1 C1	M0 C0	C15	C14	C13	C12 M1 C1	C11 M0 C0	C10	C9	C8	C7	C6	C5 M1 C1	C4	C3	C2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
8	M7	M7 C7 M3 C3	M6 C6 M2 C2	M5 C5 M1 C1	M4 C4 M0 C0	M3 C3	M2 C2 M7 C7 M3 C3	M1 C1 M6 C6 M2 C2	M0 C0 M5 C5 M1 C1	C15 M4 C4 M0 C0	C14 M3 C3	C13 M2 C2	C12 M1 C1	C11 M0 C0 M7 C7 M3 C3	C10 M6 C6 M2 C2	C9 M5 C5 M1 C1	C8 M4 C4 M0 C0
8	M7	0	0	0	0	0	0	0	0	C15	C14	C13	C12	C11	C10	C9	C8
		X7 X3	X6 X2	X5 X1	X4 X0	X3	X2 X7 X3	X1 X6 X2	X0 X5 X1	X4 X0	X3	X2	X1	X0 X7 X3	X6 X2	X5 X1	X4 X0
CRC.REG =		R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15

Xi= Mi ⊕ Ci, Met Mi het i-de bit of het inputbyte, en C is de initiële crc.

Na 8 verschuivingen kunnen we zien hoe het nieuwe crc-register eruit ziet. Om snelheid te winnen is het mogelijk om de Xi's op te slaan in een opzoek-tabel, want als X bekend is en uiteraard het inbutbyte M en de voorgaande waarde v/h crc-register, kan men het nieuwe crcregister bepalen. Onderstaand programma geschreven in PdV ISO-pascal genereert de opzoektabel voor alle mogelijke combinaties van X. (Noot: De LSB is hier als b1 genummerd)

b0 ≙ b1 in pascalprgr, enz.

```
PROGRAM CRC (INPUT,OUTPUT);
{ ### Dit programma berekend de waarden van het CRC-CCITT polynoom,
      afhankelijk van alle 8 bits ingangscombinaties.
      CRC-CCITT wordt o.a. door xmodem gebruikt voor foutdetectie
      CRC-CCITT=x^16+x^12+x^5+1 (polynoom)
      Voltooid op 24 sept 1988, B. de Bruine
      Bron: IEEE magazine, jaar 1983, afl. juni
            Aram Perez: Bytewise crc-calculations     ### }
TYPE word = ARRAY[1..16] OF BOOLEAN;
VAR x8, x7, x6, x5, x4, x3, x2, x1, x: BOOLEAN;
    v: word;
    i,loop:INTEGER;

PROCEDURE binhex(VAR v: word; nibble: INTEGER);
{Zet een 16 bits boolean array om naar een hexadecimale (print)waarde.
 Input: v=boolean array die maximaal 16 bits mag bevatten
        nibble=Het aantal nibbles dat geprint moet worden,
               bv: nibble=4 betekend 16 bits. (max. 4)
 Output: Print hex-equivalent op het actieve device. }
 VAR i,j,h,k,a,b,c :INTEGER; vi: ARRAY[1..16] OF INTEGER;
     hex: ARRAY[0..15] OF CHAR;
```

```
  BEGIN
   FOR i:=0 TO 15 DO   (lees hex-array waarden in)
    BEGIN
     IF i<10 THEN hex[i]:=CHR(48+i) ELSE hex[i]:=CHR(55+i);
    END;
   FOR i:= 1 TO 16 DO
    BEGIN
     IF v[i]=false THEN vi[i]:=0 ELSE vi[i]:=1;
    END;
          (algoritme groeperen per 4 bits)
    k:=0;
    j:=(nibble*4)-3;
    REPEAT
        a:=j+1;
        b:=j+2;
        c:=j+3;
     h:=vi[j]+(vi[a]*2)+(vi[b]*4)+(vi[c]*8);
     WRITE(hex[h]);
     j:=j-4;
     k:=k+1;
    UNTIL k=nibble;
  END;

FUNCTION xor(x, y: BOOLEAN): BOOLEAN;
 (Xor exclusived orred de variabelen x en y)
 BEGIN
 xor:= (x AND (NOT(y))) OR (y AND (NOT(x)));
 END;

BEGIN   (of program)

FOR i := 1 TO 16 DO v[i]:=FALSE;
loop:=0;

FOR x1 := FALSE TO TRUE DO
 FOR x2 := FALSE TO TRUE DO
  FOR x3 := FALSE TO TRUE DO
   FOR x4 := FALSE TO TRUE DO
    FOR x5 := FALSE TO TRUE DO
     FOR x6 := FALSE TO TRUE DO
      FOR x7 := FALSE TO TRUE DO
       FOR x8 := FALSE TO TRUE DO
       BEGIN
        v[1]:= xor(x8,x4);  (* R0=XOR(X7,X8) in shifttabel)
        v[2]:= xor(x7,x3);
        v[3]:= xor(x6,x2);
        v[4]:= xor(x5,x1);
        v[5]:= x4;
        v[6]:= xor(v[1],x3);
        v[7]:= xor(v[2],x2);
        v[8]:= xor(v[3],x1);
        v[9]:= v[4];
        v[10]:= x4;
        v[11]:= x3;
        v[12]:= x2;
        v[13]:= xor(v[1],x1);
        v[14]:= v[2];
        v[15]:= v[3];
        v[16]:= v[4];
        WRITE(loop, ':  ');   (test variabele)
        loop:=loop+1;
        binhex(v,4); WRITELN;
       END;

WRITELN( 'Ready!' );
END.
```

Opzoektabel voor Crc-ccitt-16 berekening

0:	0000	50:	1611	100:	2C22	150:	E37F	200:	5844
1:	1021	51:	0630	101:	3C03	151:	F35E	201:	4865
2:	2042	52:	76D7	102:	0C60	152:	02B1	202:	7806
3:	3063	53:	66F6	103:	1C41	153:	1290	203:	6827
4:	4084	54:	5695	104:	EDAE	154:	22F3	204:	18C0
5:	50A5	55:	46B4	105:	FD8F	155:	32D2	205:	08E1
6:	60C6	56:	B75B	106:	CDEC	156:	4235	206:	3882
7:	70E7	57:	A77A	107:	DDCD	157:	5214	207:	28A3
8:	8108	58:	9719	108:	AD2A	158:	6277	208:	CB7D
9:	9129	59:	8738	109:	BD0B	159:	7256	209:	DB5C
10:	A14A	60:	F7DF	110:	8D68	160:	B5EA	210:	EB3F
11:	B16B	61:	E7FE	111:	9D49	161:	A5CB	211:	FB1E
12:	C18C	62:	D79D	112:	7E97	162:	95A8	212:	8BF9
13:	D1AD	63:	C7BC	113:	6EB6	163:	8589	213:	9BD8
14:	E1CE	64:	48C4	114:	5ED5	164:	F56E	214:	ABBB
15:	F1EF	65:	58E5	115:	4EF4	165:	E54F	215:	BB9A
16:	1231	66:	6886	116:	3E13	166:	D52C	216:	4A75
17:	0210	67:	78A7	117:	2E32	167:	C50D	217:	5A54
18:	3273	68:	0840	118:	1E51	168:	34E2	218:	6A37
19:	2252	69:	1861	119:	0E70	169:	24C3	219:	7A16
20:	52B5	70:	2802	120:	FF9F	170:	14A0	220:	0AF1
21:	4294	71:	3823	121:	EFBE	171:	0481	221:	1AD0
22:	72F7	72:	C9CC	122:	DFDD	172:	7466	222:	2AB3
23:	62D6	73:	D9ED	123:	CFFC	173:	6447	223:	3A92
24:	9339	74:	E98E	124:	BF1B	174:	5424	224:	FD2E
25:	8318	75:	F9AF	125:	AF3A	175:	4405	225:	ED0F
26:	B37B	76:	8948	126:	9F59	176:	A7DB	226:	DD6C
27:	A35A	77:	9969	127:	8F78	177:	B7FA	227:	CD4D
28:	D3BD	78:	A90A	128:	9188	178:	8799	228:	BDAA
29:	C39C	79:	B92B	129:	81A9	179:	97B8	229:	AD8B
30:	F3FF	80:	5AF5	130:	B1CA	180:	E75F	230:	9DE8
31:	E3DE	81:	4AD4	131:	A1EB	181:	F77E	231:	8DC9
32:	2462	82:	7AB7	132:	D10C	182:	C71D	232:	7C26
33:	3443	83:	6A96	133:	C12D	183:	D73C	233:	6C07
34:	0420	84:	1A71	134:	F14E	184:	26D3	234:	5C64
35:	1401	85:	0A50	135:	E16F	185:	36F2	235:	4C45
36:	64E6	86:	3A33	136:	1080	186:	0691	236:	3CA2
37:	74C7	87:	2A12	137:	00A1	187:	16B0	237:	2C83
38:	44A4	88:	DBFD	138:	30C2	188:	6657	238:	1CE0
39:	5485	89:	CBDC	139:	20E3	189:	7676	239:	0CC1
40:	A56A	90:	FBBF	140:	5004	190:	4615	240:	EF1F
41:	B54B	91:	EB9E	141:	4025	191:	5634	241:	FF3E
42:	8528	92:	9B79	142:	7046	192:	D94C	242:	CF5D
43:	9509	93:	8B58	143:	6067	193:	C96D	243:	DF7C
44:	E5EE	94:	BB3B	144:	83B9	194:	F90E	244:	AF9B
45:	F5CF	95:	AB1A	145:	9398	195:	E92F	245:	BFBA
46:	C5AC	96:	6CA6	146:	A3FB	196:	99C8	246:	8FD9
47:	D58D	97:	7C87	147:	B3DA	197:	89E9	247:	9FF8
48:	3653	98:	4CE4	148:	C33D	198:	B98A	248:	6E17
49:	2672	99:	5CC5	149:	D31C	199:	A9AB	249:	7E36
								250:	4E55
								251:	5E74
								253:	3EB2
								255:	1EF0

Het algoritme verloopt nu alsvolgt:

1e. Exor het inputbyte met het hi-byte van de voorgaande crc.
    Dit levert de index voor de opzoektabel. X = (M ⊕ (R15..R8))
    Nu kan de nieuwe crc gevormd worden:

2e. Crclo word Xlo,
    Crchi word Xhi exor voorgaande crclo.

EXOR: resultaat is 0 als ⊕ beide bits gelijk zijn

Voorbeeld:	In	CRC(oud)		X	CRC(nieuw)	
	--	0000 0000 0000 0000	nvt	0000 0000 0000 0000		$0000
	01	0000 0000 0000 0000	$01	0001 0000 0010 0001		$1021
	01	0001 0000 0010 0001	$11	0010 0011 0001 0000		$2310

Als de tweede $01 ontvangen word, bevat het crc-register reeds $1021. Om X te bepalen word het ingangsbyte ($01) ge-exorred met crc-hi($10). Dit levert de waarde $11 (17 decimaal). In de opzoektabel vinden we bij 17 het getal $0210, waarvan het lage byte het nieuwe lage byte van het CRC-register word. (vgl schuif 8 x rechts). Het hoge byte word verkregen door $02 te exorren met $21. De nevenstaande subroutine berekend op deze manier de crc.

```
        ;--VARIABELEN CRC-CALCULATION
CRC     RES     2,0                     ;Crc register

        ;SUBROUTINE FOR CRC-CALCULATION
        ;CALCRC calculate crc about a byte
        ;Entry: A=char
        ;       CRC=previous value
        ;Exit:  CRC updated.
        ;Destroys: A,X
CALCRC  EOR     CRC+1                   ;CRChi(OLD) xor input
        TAX                             ;X=index in lookup table
        LDA     CRC                     ;CRChi:=TABhi XOR CRClo(OLD)
        EOR     TAB+$0100,X
        STA     CRC+1
        LDA     TAB,X                   :CRClo:=TABlo
        STA     CRC
        RTS

        ;CRC Look-up table (512 bytes)
        org     $a600   ;lo byte of x-value

TAB     fcb     $00
        fcb     $21
        fcb     $42
        ........

        org     $a700

        fcb     $00     ;hi byte of R-value
        fcb     $10
        ......
```

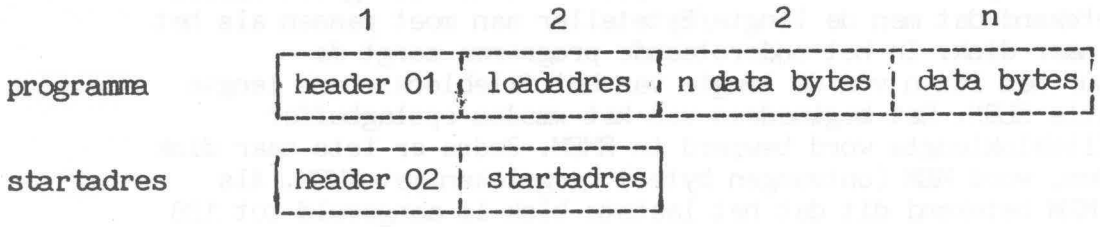
4.4 Verschil checksum met crc-methode

De checksum vangt niet alle fouten, bv $80+$80 levert hetzelfde op als $00 + $00. Daar een overflow afgekapt word, is de controle minder betrouwbaar. Crc detecteerd alle een- en twee-bit fouten, alle oneven bit-errors, alle 'burst'errors kleiner of gelijk aan de orde van de polynoom (16), en de meeste fouten van een hogere orde. Crc kan zelfs een- en tweebitfouten corrigeren. Hier word bij xmodem/kermit geen gebruik van gemaakt. Grofweg valt te berekenen dat de checksummethode een kans heeft van 92% om de fout(en) te detecteren. Voor crc is dit ongeveer 99%.

5. Binaire files

Een apart probleem vormen de binaire files op een dos-65 systeem. Bij binaire files staat locatie- en lengte informatie gewoon tussen de databytes, voorafgegaan door een herkenningsteken. Hiermee is het mogelijk een programma op meerdere plaatsen in het geheugen te laden.

Het formaat van binaire files wordt voor data bytes gedefinieerd door header byte 01, een twee bytes loadadres, een twee bytes lengte specificatie en data bytes. Een startadres specificatie heeft header 02 gevolgd door een twee bytes tellend startadres.



In het twee byte adres of in de twee byte lengte specificatie komt eerst het low byte gevolgd door het high byte.

Voorbeeld: De file Test.mac bevat de volgende instructies:

```
        ORG     $A000
        LDA     #0
        STA     $0

        ORG     $0200
        LDA     #0
        STA     $0

        ORG     $0101
        LDA     #0
        STA     $0

        END
```

Na assembleren ontstaat de file Test.bin die er als volgt uit ziet:

```
      SOH  load addr.  block length  DATA      SOH  load addr  enz...
0000  01  00 A0  04 00  A9 00 85 00 01  00 02  04 00 A9 00   ................
0010  85 00 01 01 01 04 00 A9 00 85 00                       ...........
```

Xmodem verzend Test.bin als het volgende Block:

```
        Xmodem header
7000  01 01 FE 01 00 A0 04 00 A9 00 85 00 01 00 02 04
7010  00 A9 00 85 00 01 01 01 04 00 A9 00 85 00 00 00
7020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7080  00 00 00 3D 04
            checksum   EOT
```

Wat is nu het probleem bij datacommunicatie ? Om te beginnen moet men het fileformaat binair maken. (b in de spec van het CAT commando) Dit is helaas niet voldoende. Daar protocollen als Xmodem met vaste bloklengtes werken, en geen restblok op het einde van een file kennen, worden in het laatste blok dus bytes getransporteerd die niet meer bij de file horen. Voor niet-binaire files is dit geen enkel probleem. De gebruiker negeert deze bytes gewoon. Maar Dos-65 weet geen raad met deze restbytes als het om een bin file gaat. Xmodem introduceert dus een fout. Stel dat na de 01 header in een bin file een lengte van 4 bytes gespecificeerd is. Xmodem maakt hier 128 bytes van om het blok te vullen. Zodra men deze file met dos-65 wil laden/runnen hangt het systeem. Als men een .bin file correct op disk wil zetten, moet men:
1. Het fileformaat op BIN zetten, ($E2)
2. Overtollige bytes verwijderen.
Om de overtollige bytes te verwijderen moet men controleren of een 01-header passeert. Is dat zo, dan moet men de navolgende lengtebytes copieren. (ELEN) Door een byteteller alle databytes te laten tellen, en deze te vergelijken met ELEN kan men het einde van een datablock vaststellen. Hierna kan men opnieuw gaan zoeken naar een 01-header. Om te voorkomen dat men op het verkeerde moment of in de datastroom naar 01 gaat zoeken, moet men met een aantal vlaggen aangeven wanneer naar een 01 gezocht mag worden. Het word nog iets gecompliceerder, omdat er ook nog een 02-header kan voorkomen. Deze kan men negeren, maar men moet wel de navolgende twee bytes (startadres) overslaan, want daar kan ook een 01 byte in staan. Astrid V3.0 werkt met een groot xmodem opslagbuffer. Dit betekend dat men de lengte/Byteteller aan moet passen als het buffer geleegd word naar disk. In het onderstaande programma zorgt de subroutine BISTOR voor het lezen van de lengte van het fileblock. Deze lengte word bewaard in locatie ELEN. Het beginadres van het xmodem opslagbuffer vermeerderd met de filebloklengte word bewaard in EMEM. Zodra er iets naar disk geschreven moet worden, word MEM (ontvangen bytes) vergeleken met EMEM. Als EMEM kleiner is dan MEM betekend dit dat het laatste blok is aangevuld tot 128 met overtollige bytes. Dit programma copieert EMEM dan naar MEM, omdat de write-disk routine alles naar disk schrijft tot en met het adres dat in MEM staat. Op deze manier worden

aanvulkarakters ge-elimineerd. Omdat er meerdere fileblokken voorafgegaan door een 01-header verzonden kunnen worden, vergelijkt BYTELEN de lengte die gelezen is na de 01-header (ELEN) met de getelde bytes (BINBYT). Als deze gelijk zijn worden de tellers gereset, en word de vlag BIN op 00 gezet, wat betekend dat er weer naar een 01/02 gespeurd mag worden. Zodoende word de EMEM naar MEM copie alleen uitgevoerd bij het laatste blok, wat de bedoeling is. Indien men onderstaand programma wil opnemen in een downloadroutine, moet in de lus dat het karakter binnenhaald,

Routines om bij download aanvulbytes te verwijderen:

```
                        ;Values for correct save of a .bin file
33E4 00         BCOUNT  RES     1,0     ;Counts received byte after soh
33E5 00         BIN     RES     1,0     ;Flag=0 if fifo=bin and no $01 found
                                        ;Bin=0 enable search 01/02
33E6 FF         BINEN   RES     1,$FF   ;0=enable bcount
33E7 00         RCHAR   RES     1,0     ;Temp. saveloc of last received char.
33E8 0000       ELEN    RES     2,0     ;Length of a file block
33EA 0000       EMEM    RES     2,0     ;Real endaddr (absolute from beglo)
33EC 0000       BINBYT  RES     2,0     ;Bytecount in one block preceeded
                                        ;with $01

                        ;Write file(block) naar disk
3592 AD CD07    BWRIDI  LDA     FIFO
3595 C9 E2              CMP     #$E2            ;Binary ?
3597 D0 2D              BNE     OK3
3599 A5 91              LDA     MEM+1           ;Calculate real endadrress
359B CD EB33            CMP     EMEM+1
359E 10 02              BPL     1.F
35A0 80 07              BRA     2.F
35A2 A5 90      1       LDA     MEM             ;Is mem >= emem ??
35A4 CD EA33            CMP     EMEM
35A7 10 13              BPL     OK1
35A9 38         2       SEC                     ;Calculate emem for next block
35AA AD EA33            LDA     EMEM
35AD E5 90              SBC     MEM
35AF 8D EA33            STA     EMEM
35B2 AD EB33            LDA     EMEM+1
35B5 E5 91              SBC     MEM+1
35B7 8D EB33            STA     EMEM+1
35BA 80 0F              BRA     OK2
                OK1     MOVEM   EMEM,MEM        ;Dont store restblock rubbish
35C6 20 5424    OK3     JSR     WRIDI           ;Write storagememory to disk
35C9 80 06              BRA     1.F
35CB 20 5424    OK2     JSR     WRIDI
35CE 20 7638            JSR     BIEND           ;Absolute endadrr in emem
35D1 60         1       RTS

                        ;Bistor: isolates fileblocklength from datastream in emem/+1
                        ;Note: bin must be $0 if fifo is e2 at entry xrx/xkx
380E AD E533    BISTOR  LDA     BIN
3811 F0 44              BEQ     45.F
3813 C9 02              CMP     #2              ;Skip 2 bytes start address
                                                ;(02 header)
3815 F0 5B              BEQ     47.F
3817 C9 01              CMP     #1
3819 F0 57              BEQ     47.F
381B AD E633            LDA     BINEN           ;Enable bincount only after
381E D0 49              BNE     44.F            ;01 is found
3820 EE E433            INC     BCOUNT          ;Update bincount
3823 AD E433            LDA     BCOUNT
3826 C9 04              CMP     #4
3828 D0 12              BNE     42.F
                        MOVE    RCHAR,EMEM      ;Save filelength in header
3830 8D E833            STA     ELEN            ;In elen (check next soh)
                ;And in emem (real file end)
                        MOVEQ   ENDMEM,BINBYT+1 ;Safety end
3838 8D EB33            STA     EMEM+1          ;For sure
383B 60                 RTS
383C C9 05      42      CMP     #5
383E D0 29              BNE     44.F
                        MOVE    RCHAR,EMEM+1
3846 8D E933            STA     ELEN+1
                        MVI     $FF,BINEN       ;Stop counting
384E 8D EC33            STA     BINBYT          ;Reset blocklength counter
3851 8D ED33            STA     BINBYT+1
3854 4C 7638            JMP     BIEND           ;Calc abs. file-end
3857 AD E733    45      LDA     RCHAR
385A C9 01              CMP     #$01            ;01/02header ??
385C D0 0C              BNE     46.F
385E 9C E633            STZ     BINEN           ;Enable counting for f1
3861 8D E433            STA     BCOUNT          ;Initialise counter
                        MVI     $FF,BIN         ;Show length readed
3869 60         44      RTS
386A C9 02      46      CMP     #$02            ;02 header ?
386C D0 07              BNE     48.F
386E 8D E533            STA     BIN             ;Bin :=2
3871 60                 RTS
3872 CE E533    47      DEC     BIN
3875 60         48      RTS                     ;48 is eigenlijk een error
```

Commonly used generator polynomials.

CRC-16	$X^{16} + X^{15} + X^2 + 1$
SDLC (IBM, CCITT)	$X^{16} + X^{12} + X^5 + 1$
CRC-12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$
CRC-16 REVERSE	$X^{16} + X^{14} + X^2 + 1$
SDLC REVERSE	$X^{16} + X^{11} + X^4 + X^3 + X^2 + 1$
LRCC-16	$X^{16} + X^2 + 1$
LRCC-8	$X^8 + 1$

```
                ;Help subr. of bistor. also used bij full storage memory
                ;Function: emem:=mem+emem.
3876 18         BIEND   CLC
3877 A5 90              LDA     MEM             ;Calculate fileblock-end
3879 6D EA33            ADC     EMEM
387C 8D EA33            STA     EMEM
387F A5 91              LDA     MEM+1
3881 6D EB33            ADC     EMEM+1
3884 8D EB33            STA     EMEM+1          ;By endblock verify endmem
3887 60                 RTS

                ;Werking: emem is het werkelijke eindadres v/d file
                ;Na een eof dient emem gecopieert te worden naar mem
                ;De restchars om een blok vol te maken voor xmodem
                ;Worden niet meegesaved op disk
                ;Indien de file langer is dan het werkgeheugen van
                ;Xmodem, word emem aangepast door mem(oud) af te trekken
                ;Van emem, en daar de geresette mem(nieuw) bij op te tellen.

                ;Bytelen compares bytecount with bytelength in header
                ;If equal, check on next soh is enabled
                ;Call this routine only if fifo=$e2 (bin)
3888 EE EC33    BYTELEN INC     BINBYT
388B D0 03              BNE     2.F
388D EE ED33            INC     BINBYT+1
3890 AD ED33    2       LDA     BINBYT+1
3893 CD E933            CMP     ELEN+1
3896 D0 0B              BNE     1.F
3898 AD EC33            LDA     BINBYT
389B CD E833            CMP     ELEN
389E D0 03              BNE     1.F
38A0 9C E533    INIBY   STZ     BIN             ;Enable check on next soh
38A3 60         1       RTS
```

een JSR BISTOR, en JSR BYTELEN staan. Eveneens moet het ontvangen karakter in RCHAR worden gezet, en dient men bij de start de tellers te initialiseren. In astrid 3.0 zit deze routine in de xmodem en kermitdownloader. Noot 1: Met kermit kan men binaire files probleemloos downloaden, tenminste als ze ook met kermit zijn geupload. (geen restkarakters) Omdat men nooit weet hoe een programma op een BBS komt, moet men ook voor kermit de 01-check subroutines toepassen. Indien men een programma upload naar de dos-65 area van het kim-bbs zet er dan duidelijk bij dat het om een BIN file gaat ! Noot 2: Het kan ondanks deze routines toch nog fout gaan als in de overtollige databytes een $01-karakter staat. Gelukkig vullen bijna alle xmodemprogramma's het blok aan met spaties, EOF's ($04) of nullen.

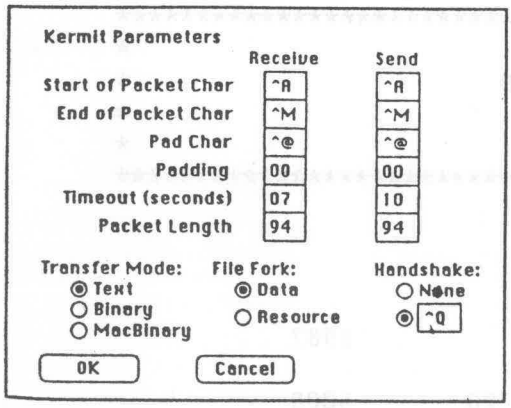
6. Viditel aktueel

Comnet videotex heeft zijn videotexhost uitgebreid met enkele tekstcommando's. Videotex is ontwikkeld om met een aangepast telefoontoestel of een

Teleshopping
Splinternieuw is de Teleshopping mogelijkheid via ComNet. Als éérste is gestart met de mogelijkheid computerboeken, computercursussen en computertoebehoren via Teleshopping te bestellen. Maar ook het aanvragen van catalogussen, proefabonnementen op computertijdschriften etc. gaat via ComNet geheel automatisch. In de informatiebestanden van Kluwer, Data Becker, Weka en Verwijs & Stam treft u de laatste nieuwe uitgaven aan op computergebied. Indien u iets besteld, dan wordt dit rechtstreeks door de betrokken leverancier aan u verzonden.

Trefwoorden
Ook is er in ComNet nu een nieuwe zoekmethode: Trefwoorden. Naast de vertrouwde methode van het zoeken met *'paginacijfer#* of de direkte indexkeuzen is er nu de mogelijkheid bijgekomen om direkt een trefwoord in te typen. Bijvoorbeeld het trefwoord *'kluwer#* brengt u direkt in het bestand van Kluwer. Evenzo brengt *'telesoftware#* u direkt naar de telesoftware index.

Eigen reklamepagina
Nieuw is ook de mogelijkheid een eigen pagina in ComNet te huren. Deze pagina kunt u dan op ieder gewenst moment wijzigen en gebruiken voor allerlei reklame- of verkoopdoeleinden. Als u privé uw auto wilt verkopen of zakelijk uw diensten of produkten wilt aanprijzen dan is de reklamepagina een fantastisch middel om nú al 12.500 mensen te bereiken. Dit aantal groeit met zo'n 1000 per maand! Zo'n pagina kost slechts f.95,-- per jaar. Vraag hem aan via trefwoord *'reklame#* in ComNet.



afstandbediening alle mogelijke functies te realiseren. Dit betekend dat men met 0,1,2,3,4,5,6,7,8,9,# en * de hele databank moet besturen. Nu iedereen toch een computer heeft is het veel handiger om direkt een commando in te tikken. De oude methode blijft ook toepasbaar. Slimme knutselaars die teletekst beelden van de tv in hun computer willen opslaan, om ze daarna te bewerken of te printen, komen aan hun trekken in de HCC-nieuwsbrief 108, blz 98 e.v. Waarschijnlijk kan viditel-65 deze beelden correct weergeven en printen. (??)

De vergadering op 21-01-1989

Een klein deel van de bijeenkomst in Krommenie op 21 januari j.l. had de status van ledenvergadering. Op deze ledenvergadering zijn de volgende zaken aan de orde gekomen:

- Jaarverslag 1988:

  Het jaarverslag 1988 is nog opgesteld door de vorige penningmeester, John van Sprang. Dit jaarverslag is als bijlage bij dit verhaaltje gevoegd. De ledenvergadering ging accoord met het jaarverslag.

  Daar het niet mogelijk gebleken is de kas door de zittende kascontrôle-commissie te laten controleren, is er staande de vergadering een nieuwe commissie benoemd die in Krommenie de kas gecontroleerd en in orde bevonden heeft.

  De financiële administratie en het beheer van de financiën van de club is met ingang van 1 januari overgedragen aan de nieuwe penningmeester Jacques Banser.

- Bestuursverkiezing

  Jan Derksen, de DOS-65 software-coördinator heeft zich beschikbaar gesteld voor een functie in het bestuur. Op de ledenvergadering was hiertegen geen bezwaar. De overige bestuursleden heten Jan van harte welkom in het bestuur.

- Huishoudelijk Reglement

  Op de ledenvergadering in Almelo is het huishoudelijk reglement, zoals afgedrukt in de 6502 Kenner nr. 58 goedgekeurd. Wel werd aan de commissie verzocht een artikel over het bestaan en functioneren van de kascontrôle-commissie op te nemen met als uitgangspunt een zittingsduur van twee jaren. Ieder jaar treedt er dan één lid af.

  Het bestuur en de commissie waren van mening dat de voorgestelde regeling niet zinvol is omdat dan leden van de kascontrôle-commissie gedurende twee jaar verplicht lid van de vereniging moeten blijven. Zou dit niet het geval zijn, dan kan het namelijk voorkomen dat niet-leden inzage krijgen in de (vertrouwelijke) financiële stukken van de vereniging. Aan de ledenvergadering werd voorgesteld de kascontrôle-commissie niet nu bij huishoudelijk regelement te regelen en artikel 11 te laten vervallen. De ledenvergadering ging hiermee accoord. In het gepubliceerde regelement wordt het volgende gewijzigd:

  Artikel 11 vervalt.
  Artikel 12 wordt artikel 11

Verder werd bij stemming besloten dat het huisorgaan met ingang van de dertiende jaargang "De uP Kenner" (de microprocessor kenner) genoemd wordt. De bedenker, G.J.M. op der Heijde, wordt hartelijk bedankt en kan binnenkort de boekenbon verwachten.

Namens het bestuur:
Gert van Opbroek

==========

GEVRAAGD: VIC-20 SOFTWARE

Bert van Tiel
J. van Beamontstraat 21
2805 RN Gouda

vraagt software voor de VIC-20. Hij is geïnteresseerd in alle soorten software en in alle vormen (papier, cassette, kunt (wilt) missen, dan kunt contact opnemen met Bert van Tiel of met de redactie.

****
*
* KIM GEBRUIKERS CLUB NEDERLAND
* JAARSTUKKEN VAN 1988
*
****

Balans per 31 december 1988

AKTIVA	1988	1987
Inventaria	4.304,20	5008
Voorraden	pm	pm
Te ontvangen posten	0	0
Geldmiddelen	22.199,88	29059
	------,--	------
TOTAAL	26.504,08	34.067,+

PASSIVA		
Vrije reserve	19.798,20	26140
Vooruitontvangen contributie	4.200,	7650
Te betalen posten	2.505,88	277
	-------,--	-------
TOTAAL	26.504,08	34.067,

STAAT VAN BATEN EN LASTEN OVER 1988

LASTEN	1988	1987
Druk kosten 6502 kenner	16.288,22	16230
Verzendkosten	0	1398
Redactie kosten	414,03	1327
Bestuur kosten	1.030,52	985
HCC dagen	0	0
Afschrijving inventaris	2.504	1410
	------,--	------
TOTAAL	20.236,77	21.350,

BATEN		
Contributie	11.708,	18290
Opbrengst advertentie	900,	900
Verkoop losse 6502 kenner	40,	628
Verkoop cas. en manuals	0,	725
Bijeenkomsten	486,	798
Rente bank en giro	454,62	573
Dos 65	1.519,30	5493
Div (o.a. ohio disk)	0	250
	------,--	------
TOTAAL	15.107,92	27.657,

Opgemaakt door J.F.v.Sprang
tulp 71
Krimpen a/d IJSSEL

Programeren in Assembler (deel 1)

Een stukje jeugdsentiment

Tijdens de bijeenkomst in Almelo hoorde ik van een aantal bezoekers dat het zo moeilijk is om met programeren in assembler te beginnen als je er nog nul komma nul vanaf weet. Nou ben ik zelf ook eens (in een vaag verleden) begonnen op deze, toendertijd nog moeizame, weg. Damals (het was de tijd van Echte Programmeurs) had ik nog geen assembler tot mijn beschikking maar ik wilde toch wel eens wat anders dan dat eeuwige BASIC, dus wat deed de gek? Hij begon met zijn blote bolletje de assembler codes in hexadecimaal in te typen met behulp van een opcode tabel. Al snel ben je dan een kei in het uitrekenen van relatieve spronginstructies en de meeste opcodes kun je uit je hoofd opdreunen. Toch is programeren op die manier niet eenvoudig en grote programma´s kun je eigenlijk wel vergeten. Een ander zeer groot nadeel is dat je je programma´s op papier zeer goed moet documenteren omdat je anders zo de draad kwijt bent. Ook het tussenvoegen van verbeteringen was erg moeilijk omdat alle addressen dan moesten worden veranderd. Het enige alternatief was een spronginstructie tussenvoegen en de gewenste code op een vrij adres neerzetten. De zo gemaakte programma´s doen denken aan een bekend Italiaans gerecht....

Even het geheugen opfrissen (dynamische RAM´s ??)

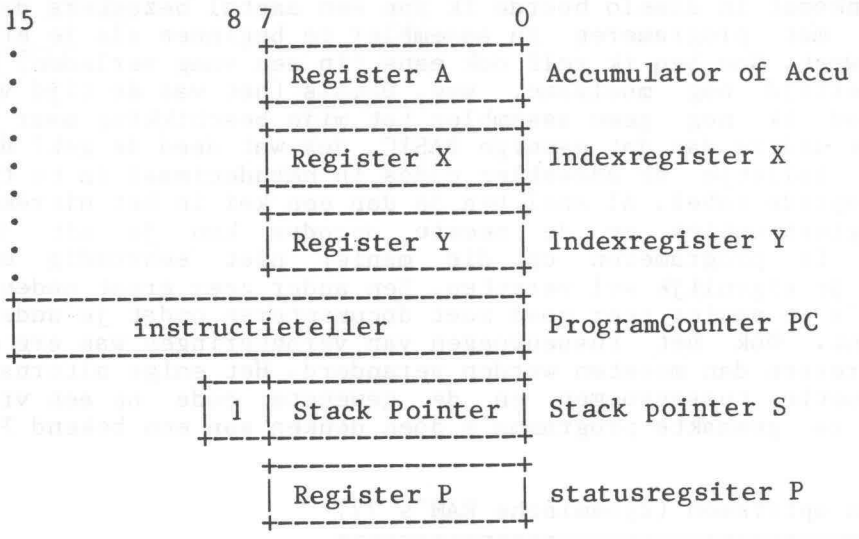
Maar genoeg gemijmerd over die goede oude tijd. Om te beginnen eerst even wat gegevens die je gewoon moet weten over die ouwe trouwe 6502 microprocessor. Het beestje is een 8-bits microprocessor, wat wil zeggen dat hij zowel extern als intern met een 8 lijn brede databus werkt. Al die lijntjes kunnen maar twee toestanden aannemen, er staat of nul of vijf volt op, uit of aan, oftewel logisch 0 of logisch 1 zijn, kortweg 0 of 1. Dit nu is het befaamde ´bit´ binnen de processor, de kleinste hoeveelheid informatie. Er zijn 8 van die bits, dat betekent dat er 2 tot de macht 8 (even wat wiskunde er tussendoor, dat doet het altijd goed dacht ik zo) dus 256 mogelijke situaties zijn te beschrijven. Een groep van 8 bits heet een byte, een groep van vier een nibble. Op dit lage nivea werkt de computer dus met nullen en enen. De computer kan dus niet eens tot twee tellen!! Nu is dat ´nul´ en ´een´ gedoe voor den Mensch wat moeilijk te lezen en om dat dus iets gemakkelijker te maken is het hexadecimale stelsel op de proppen gekomen. Tellen we in het decimale (deci = 10) stelsel van 0 tot 9, in het hexadecimale (hexa = 16) stelsel wordt dat van 0 tot (consequent blijven) 15. Er moeten dus zes andere symbolen bedacht worden voor die getallen boven de 9, mag ik ze even voorstellen:

A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

Nemen we nu een nibble (=4 bits) dan kunnen we daar 2^4 = 16 verschillende getallen mee maken. Dus precies van 0 tot 15, oftewel $0 tot $F. Een byte is dus met precies twee hexadecimale karakters te coderen. Komen die bytes per kilo dan heet dat kilobyte of Kb, maar anders dan bij de slager is die kilo hier geen 1000 maar 1024 (alweer een macht van 2). (Dit wist je waarschijnlijk allemaal al en ik noem het hier alleen maar ter lering ende vermaeck)

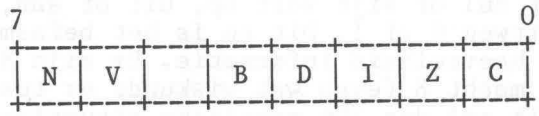
De 6502 registers

De 6502 heeft intern een aantal registers waar bepaalde zaken in worden opgeslagen, hieronder staan ze:



Het statusregister

Het statusregister bevat de volgende bitjes:



N = Negative flag, ´1´ als het resultaat v.e. bewerking negatief is
V = oVerflow flag, ´1´ als het resultaat v.e. bewerking niet meer ´past´ binnen een byte.
B = Break flag, ´1´ als een BRK instructie werd uitgevoerd.
D = Decimal flag, ´1´ als de 6502 decimaal rekent.
I = Interrupt flag, ´1´ disabled de IRQ ingang
Z = Zero flag, ´1´ als als het resultaat v.e. bewerking nul is
C = Carry flag

Ieder van deze bitjes zegt dus iets over de toestand (status) van de processor. In latere delen zal ik verder ingaan op de betekenis van deze bitjes.

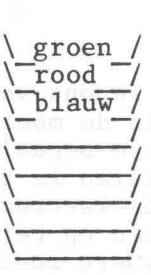
De Program Counter

Zoals je ziet bestaat er slecht een 16 bits register binnen de 6502 en dat is de PC (Program Counter). Dit register wijst steeds naar die geheugenlocatie waar de volgende instructie vandaan komt. Omdat het een 16 bits register is kun je er dus 2^16 = 65536 locaties mee bereiken. Het adres bereik van de 6502 bedraagt dus (slechts) 64 Kbyte. Omdat het PC register 16 bits breed is en onze databus slechts 8 bits breed is spreken we ook wel van een PCH en een PCL. Hierbij is PCH de bovenste (Hoog) en PCL (Laag) de onderste 8 bits van de PC. Op die manier kun je het adresbereik van 64K opdelen in z.g.n. pagina´s van 256 bytes (het gebied dat de PCL kan beslaan). Een snelle rekensom leert ons dat er 256 van die pagina´s in de adresmap van onze 6502 voorkomen. Een aantal belangrijke pagina´s zijn de allereerste, dus pagina 0 ook wel bekend als de ´zero page´ (Nee, heeft niets te maken met dat japanse vliegtuig), de tweede pagina, dus pagina 1 wordt gebruikt voor de stack of stapel (zie ook verder) en de allerlaatste pagina (pagina $FF) wordt gebruikt voor een aantal

adressen die de 6502 moet weten als hij een RESET, IRQ, NMI of BRK uitvoert (de bekende vectoren). Ook deze vectoren zal ik later nog eens bespreken.

De Stack Pointer

Een ander belangrijk register is de SP (Stack Pointer). Dit register begint zoals je ziet altijd met het achtste adres bit '1', dit betekent dat de stack of stapel van de 6502 (slechts) 256 bytes groot is en altijd gesitueerd is op geheugen bladzijde 1. De stack is een zogenaamd LIFO (Last In First Out) geheugen, dus wat er het laatste heen is gestuurd komt er als eerste weer uit. Stack is het engelse woord voor stapel en daarmee kun je de werking inderdaad het gemakkelijkste mee verklaren. Men neme een stapel borden, men plaatse er een blauw bord, vervolgens een rood en als laatste een groen bord op (nee, je wilt NIET weten wat voor servies ik gebruik!). De stapel ziet er dus als volgt uit:



<-- Stack Pointer wijst steeds naar de eerste vrije locatie op de stapel. Nemen we nu, zonder geweld te gebruiken, een bord van de stapel dan ziet een blind paard dat dat het groene bord moet zijn. De Stackpointer zakt een bord want die plaats is weer vrij. Vervolgens kunnen we zonder brokken te maken het rode bord van de stapel pakken en als laatste het blauwe. We zien dus dat de volgorde van afpakken precies andersom is als bij het erop zetten.

Bij dit voorbeeld blijkt nog een andere eigenschap van een stack, er kan steeds maar een bord (of byte) tegelijk op de stapel worden gezet, en ook maar een tegelijk er weer afgehaald. De stackpointer in de vijfenzestignultwee heeft als (rare) eigenschap dat ie niet van laag naar hoog maar van hoog naar laag doorschuift bij het plaatsen van borden (bytes) op de stack. De stapel staat a.h.w. op zijn kop.

De Accumulator

Register A of Accumulator, kortweg Accu, is een van de belangrijkste registers binnen de processor. Accumuleren betekent opslaan en dat is precies wat dit register doet, het slaat het resultaat op van een bewerking. Die bewerking kan bijvoorbeeld bestaan uit optellen, aftrekken, logische AND etc, etc.
Al deze bewerkingen worden uitgevoerd door een beestje binnen de 6502 met de cryptische naam ALU (Arithmetic Logic Unit), vrij vertaald kan dit beestje dus zowel reken- als logische bewerkingen uitvoeren. Deze ALU heeft twee ingangen en een uitgang. Op beide ingangen komen twee getallen te staan en op de uitgang komt het resultaat van de (met instructies) gekozen bewerking. In bijna alle gevallen komt het ene getal uit de Accu en het andere uit het geheugen, het resultaat van de bewerking komt dan meestal in de Accu terecht. Verder wordt de Accu gebruikt om dingen van en op de stack te zetten, een soort doorgeef-luikje dus.

De index registers X en Y

Deze beide registers zijn (helaas) 8 bits. Ze kunnen worden gebruikt voor opslag van gegevens (extra Accu's) maar ze zijn bedoeld om als index gebruikt te worden. Anders dan bij de Accu kun je deze registers niet gebruiken voor bewerkingen als optellen en aftrekken. Via een aantal index adreseer methodes, die ik later zal behandelen, kun je met X of Y bijvoorbeeld een tabel doorlopen. Het nadeel van het feit dat ze 8 bits zijn is evident, met 8 bits kun je namelijk maar 256 bytes of een pagina bestrijken. Om dus het hele 64K gebied of tabellen die langer zijn dan 256 bytes te kunnen behandelen met X of Y, moeten we een truuk special bedenken. Hoe die truuk in z'n werk gaat zien we later bij de adresseermogelijkheden.

Het eerste wankele stapje

We kennen nu alle registers die de 6502 heeft maar nu willen we er ook wel eens wat mee doen. Om ook eens op het allerlaagste nivo gewerkt te hebben gaan we het (programma) (tussen haakjes dus), invoeren met den hand. We gaan dus nog niet gebruik maken van luxe hulpmiddelen zoals de EDITOR of de ASSEMBLER. Op het DOS65 systeem is standaard een monitor aanwezig. Zo'n monitor kan o.a. gebruikt worden om een kijkje te nemen in het geheugen, maar ook om zoals in dit geval, een kort programma in te voeren c.q. wijzigen.

Het programma dat we nu gaan invoeren gebruikt maar 5 instructies, dat zijn:

LDA	LoaD Accumulator
ADC	ADd with Carry
CLC	CLear Carry
STA	STore Accumulator
RTS	ReTurn from Subroutine

De laatste instructie zal ik later bespreken, hij dient in ons voorbeeld alleen om op een nette manier terug (=return) te komen in de monitor. De eerste instructie kan gebruikt worden om de Accumulator met een bepaald getal te 'laden' m.a.w. met een getal vullen. Zoals we al hadden gezien is register A slechts 8 bits, dus dat getal ligt tussen 0 en 255. De tweede instructie kan gebruikt worden om een tweede getal bij de waarde van de Accu op te tellen. Helaas heeft de 6502 geen instructie die die operatie uitvoert zonder daar ook de Carry bij te tellen. Het Carry bitje zal ik later nog bespreken, maar het is zo wel duidelijk dat die Carry 0 moet zijn om geen invloed te hebben op de optelling. Vandaar de derde instructie die niets anders doet dan het Carry bitje uit het statusregister P met de waarde 0 te laden. De vierde instructie zet het getal wat in A zit ergens neer (store=opslaan). Dat kunnen we hier dus mooi gebruiken om het resultaat van de optelling te bewaren. In het voorbeeld gebruiken we alleen de ABSOLUTE ADDRESS adresseermethode, d.w.z. alle addressen die voorkomen zijn 16 bits zonder offset of wat dan ook. Maar genoeg gezwetst, het programma ziet er zo uit:

```
LDA     GETAL1          ; Laad de Accu
CLC                     ; zet de Carry op 0
ADC     GETAL2          ; tel getal bij A
STA     RESULT          ; bewaar het resultaat
RTS                     ; ga terug naar de MONITOR
```

Nu moeten we gaan bepalen op welk adres we het programma gaan neerzetten en welke adressen gebruikt gaan worden voor GETAL1, GETAL2 en RESULT. We zien hier ook mooi dat het RAM geheugen van de computer voor twee dingen gebruikt kan worden; 1) Opslaan van data (GETAL1, GETAL2, RESULT)
2) Opslaan van programma (ons eerste stapje)

Wat we ook nog op moeten zoeken zijn de opcodes die horen bij de gekozen instructies. Ik neem aan dat iedereen wel ergens zo'n lijstje heeft liggen (zo niet, dan kan er in overleg met de Redactie wel het een en ander geregeld worden, dacht ik zo?). Als we dan kijken in de tabel voor absoluut adresseren dan vinden we de volgende opcodes:

LDA=$AD, STA=$8D, ADC=$6D, CLC=$18 en RTS=$60

In dit voorbeeld beginnen we het programma op adres $1000 en we zetten data op de adressen $0200 voor GETAL1, $0201 voor GETAL2 en $0202 voor RESULT. Deze keuzes kunnen we in ons voorbeeld invullen (de getallen in deze lijst zijn allemaal hexadecimaal):

```
1000 AD 00 02          LDA     GETAL1          ; Laad de Accu
1003 18                CLC                     ; zet de Carry op 0
1004 6D 01 02          ADC     GETAL2          ; tel getal bij A
1007 8D 02 02          STA     RESULT          ; bewaar het resultaat
100A 60                RTS                     ; ga terug naar de MONITOR
```

We zien gelijk iets raars, GETAL1 staat op $0200 maar in dit voorbeeld wordt het adres van GETAL1 op 00 02 gezet lijkt het wel. Dat het 16 bits adres in tweeen geknipt moest worden was wel duidelijk, we hebben immers alleen RAM in onze computer die 8 bits breed is. De 6502 leest altijd eerst het LSB (Least Significant Byte) en dan pas het MSB (Most Significant Byte). Vandaar dus dat rare omdraaien. Trouwens dit is niet specifiek iets van de 6502, een heleboel processoren werken op dezelfde manier. Lastig voor ons arme stervelingen, maar het is nu eenmaal niet anders. We zien nog iets in deze listing, het adres aan het begin van de regel loopt op. Dat adres kun je bijna vergelijken met het regelnummer in een BASIC programma. Het geeft aan waar een instructie begint. De eerste instructie staat op het gekozen start adres $1000, die instructie is 3 bytes lang (=Opcode + adres(low) + adres(high)) dus de volgende opcode begint op $1003 etc.,etc.

Nu kunnen we aan de slag met de MONITOR. Op DOS65 nivo typen we eerst:
MEMFILL 1000,1100,0
Dat leest straks wat gemakkelijker, vervolgens
MON
en als alles goed is krijg je dan de MON prompt:

```
MON65 2.01
HaViSoft
MON>
```

Het MON commando om direct in het geheugen te werken is @, dus:

```
MON> @ 1000
1000 00 AD      ; gebruik <LINEFEED> om naar de volgende regel te gaan
1001 00 00
1002 00 02
1003 00 18
1004 00 6D
1005 00 01
1006 00 02
1007 00 8D
1008 00 02
1009 00 02
100A 00 60      ; klaar, dan stoppen met <RETURN>
```

Om te controleren of je alles goed hebt ingetypt kun je de in MON65 ingebakken disassembler gebruiken:

```
MON> D 1000,100A
1000  AD 00 02           LDA  $0200
1003  18                 CLC
1004  6D 01 02           ADC  $0201
1007  8D 02 02           STA  $0202
100A  60                 RTS
```

Nu kunnen we dit programma eens uitvoeren, maar eerst moeten we de data nog invullen op de adressen GETAL1, GETAL2 en RESULT. De laatste HOEFT niet, maar anders geloof je straks niet dat het echt gewerkt heeft!

```
MON> @ 0200
0200 09 34              ; GETAL1 = $34 = 52 dec.
0201 11 12              ; GETAL2 = $12 = 18 dec.
0202 19 00              ; RESULT = 00
```

Zo dat was dat, ik hoop niet dat je nu al hebt afgehaakt want wat we hier nu

hebben gewrocht kun je in BASIC met een simpele regel maken:

```
1000 RESULT = GETAL1 + GETAL2
```

De Echte Volhouder gaat echter Door! Met het 'E'xecute commando kunnen we het programma NU uitvoeren:

```
MON> E 1000
MON>
```

Als de MON65 prompt nu niet terugkomt zoals hierboven dan zit je in diep in de problemen. Je hebt dan waarschijnlijk ergens een typefout gemaakt. Als het wel goed gaat kunnen we nu het wereldschokkende resultaat gaan aanschouwen:

```
MON> @ 0200
0200 34                 ; GETAL1 nog steeds 34
0201 12                 ; GETAL2 nog steeds 12
0202 46                 ; RESULT = $46 = 70 dec. en dat is GOED!!!!
                        ; (Gaat U door voor de hoofdprijs?)
```

Als je weer van de doorstane emoties bent bijgekomen kun je eens wat andere getallen invullen op de plaatsen GETAL1 en GETAL2. Denk er wel om dat het resultaat van de optelling moet passen in 8 bits want anders klopt het resultaat niet meer!! Hoe we dit probleem gaan oplossen zien we in de volgende les van deze cursus (voorop gesteld dat er een tweede les komt als iemand dit gelezen heeft.....).

Antoine Megens

P.S. Ik ben bereikbaar via ons aller BBS (Tel: 053-303902), dus laat eens een berichtje achter wat je van dit artikel vindt en/of wat je in de loop van deze cursus behandeld gehad zou willen hebben. De juiste schrijfwijze van mijn naam zie je hierboven.

DOS-65 CORNER 08.10.88

door: Coen Kleipool,
Val de Perier
F-83310 Cogolin.

Toen ik in de 6502 Kenner van september las dat de redactie moeite had om het blad vol te krijgen, schaamde ik mij een beetje. Ik liep al een tijdje rond met een paar ideeen, maar het op papier zetten had ik maar steeds uitgesteld. Nu werken de zomerse temperaturen hier (tussen de 30 en 40 graden) niet bevorderlijk op intellectuele of kreatieve prestaties.

Dit wetenschappelijke gegeven kan met gemakkelijk verifieren op het strand, waar men zelden iemand ziet lezen en men zich beperkt tot het slaperig staren naar andermans vrouwen. De nadenkertjes blijven daarentegen in de schaduw.

Ter bevordering van mijn intelect had ik een airconditioner besteld om mijn hete computervertrek te koelen, maar mijn vrouw liet het apparaat in de keuken installeren onder met motto: "eten is belangrijker dan programmeren". Als iemand hiervoor een afdoende repliek heeft houd ik mij aanbevolen.

EEN TREURIG VOORVAL

Op de vergadering van mei was ik getuige van een treurig voorval dat ik niettemin wil vermelden, omdat het zo leerzaam is. Ik herinner mij trouwens uit mijn jeugd dat mijn ouders vervelende zaken altijd als leerzaam afspiegelden, terwijl je van de leuke dingen zelden iets kon leren.

Onze sysop kwam wat verlaat binnen en haastte zich om zijn systeem op te stellen. Hij sloot per ongeluk zijn keyboard aan op de RS232 connector met het bekende gevolg dat de 12 volt hiervan het keyboard snel deed overlijden. Een dure grap; het had trouwens ook een printer kunnen zijn.

Dit deed mij wel de schellen van de ogen vallen en ik begrijp nu waarom IBM een male connector voor de RS232 monteert en een female voor de rest. Toen ik indertijd de Dos65 manuals vertaalde, informeerde ik naar het geslacht van de connectors en vernam dat het allemaal females moesten zijn. Noch mijn informant, noch ikzelf dachten ooit na over de consequenties hiervan.

Nu het kalf verdronken is roep ik alle Dos65-ers op snel hun RS232 connector te vermannen. Bovendien kan je dan een gewone PC kabel gebruiken voor je modem. De straps die nodig zijn omdat Dos65 geen RS232 handshake ondersteunt moet je dan wel aan de computer kant aanbrengen inplaats van in de kabel connector. Dit lijkt een domme en overbodige opmerking, maar ik heb mij hierdoor toch eens in de boot laten nemen.

COMMAND FILE PERIKELEN

Als we in dos65 een commando intypen met een file redirect zoals:" > filenaam command ", dan wordt deze outputfile geopend, de output van command er naartoe geschreven en vervolgens wordt deze outputfile keurig gesloten. Als een dergelijke commandoregel echter in een indirect commandfile voorkomt wordt de outputfile pas gesloten als de commandfile geheel is uitgevoerd en zelf wordt gesloten.

Vraag me niet waarom dit zo is, want ik weet het antwoord niet. Het lijkt mij onlogies dat het verschil maakt of een commando wordt ingetikt of uit de inputbuffer komt, maar het is echt zo en men moet dit verschijnsel maar als een onaantastbaar dogma beschouwen. Aangezien nederlanders veel van dogma's houden zal dit wel geaccep..... etc. Ik glijd af.

Intussen kan het heel lastig zijn, want dit betekent dat de nieuw gecreerde outputfile in de commandfile niet opnieuw kan worden aangeroepen omdat hij nog niet gesloten is. In de praktijk deed dit probleem zich voor toen ik een commandfile wilde schrijven voor het vertalen van viditel files (.vid suffix).

Met het commando Viprint kan met vid-files transformeren in ASCII-files; alle control characters worden verwijderd en verder wordt elke 40 characters een CR gezet, want die staan er dikwijls niet. Daarna is het gemakkelijk om de ASCII-file in de editor nog wat op te schonen en de resterende rommel te verwijderen. De commandfile die viprint aanriep, de output in de B-directory zette en vervolgens de nieuwe ASCII file in de editor zette,luidde als volgt:

```
> B/&1 viprint A/&1.vid
Ed B/&1 <cr>
```

Het bleek dat de Editor wel werd ingelezen maar de ASCII-file niet, zijnde nog niet gesloten. Het ligt nu voor de hand tussen beide regels het commando "Sync" in te voegen want hiermee worden alle files gesloten. Helaas echter ook de oorspronkelijke commandfile, zodat we hiermee nog verder van huis zijn.

Een betere oplossing is het gebruik van het commando ">T", omdat hiermee een output redirect kan worden gereset. Ik wijzigde derhalve mijn commandfile als volgt:

```
> B/&1 viprint A/&1.vid
>T Ed B/&1 <cr>
```

Diegenen die nu denken dat het probleem is opgelost, komen bedrogen uit. Weliswaar komt de filenaam keurig terug als editfile als je in de editor IOS opvraagt, maar hij wordt niet ingelezen. Waarschijnlijk omdat de outputfile pas wordt gesloten na een cr. In arren moede veranderde ik mijn file toen maar als volgt:

```
> B/&1 viprint A/&1.vid
>T ASN B
Ed &1 <cr>
```

Nu bleek de zaak wel te werken, terwijl men als bijkomend voordeel in de goede directory terecht komt. Maar ik zou toch graag eens van een van de Dos65 goeroes willen vernemen waarom ">T Ed filenaam" niet werkte. Zou een van deze heren eens een kort antwoord op het 6502 BBS kunnen zetten ?

Tenslotte probeerde ik ook nog:

```
> B/&1 viprint A/&1.vid
>T Ed
Lo B/&1 <cr>
```

Indirect commandfiles kunnen echter alleen Dos commando's bevatten en dus geen editor commando's uitvoeren. Zoiets zou echter wel makkelijk zijn en ik roep bovengenoemde goeroes op om zich eens hierover te buigen.

Ervaringen met het "CLUBMODEM", de BCH1200A van BESAMU.

E.J.van Kan
Krayenhofflaan 11
1965 AD Heemskerk.
Tel.:02510-44373.

Heemskerk 10-12-1988.

Enige tijd geleden werd het bovengenoemde modem, via Adri Hankel, aan de leden van de club aangeboden tegen een gereduceerde prijs. Naar aanleiding van opmerkingen en vragen van verschillende clubleden, heb ik mijn ervaringen samengevat in onderstaand epistel; wellicht heeft iemand er iets aan.

Aansluiten van het modem aan de DOS65-computer:

1. De aansluitingen op PL1 t/m PL8 behoeven geen enkele wijziging t.o.v. de vermelde aansluitingen in het DOS65-hardware-manual. *3
   LET OP!
   De doorverbindingen 4-5 en 6-8-11 op de RS232-connector, zoals vermeld in het DOS65-hardwaremanual op pag. 9, moeten verwijderd worden, anders werkt het spul niet !!

2. Pen 9, van de RS232-connector aan de computer, is nog nergens mee verbonden; deze pen gaan we gebruiken voor het on-line signaal bij auto-dialing maar dan moeten eerst de schakeling uit Elektuur juli/aug. 1987, pag.78/79 en de invertor-schakeling worden gebouwd en aangesloten; beide schakelingen worden verderop besproken. *2

4. RxC is het benodigde receiver-clocksignaal wat niet door het modem wordt geleverd; het signaal is echter wel aanwezig op PB7 van VIA 2.
   Dit signaal moet aangeboden worden op pen 5 van de ACIA. Maak daartoe een verbinding van pen 5 van de ACIA naar pen 17 van de RS232-connector en van PB7 van VIA 2 naar pen 18 van de RS232-connector op de computer. In de modemkabel, aan de computerzijde, moeten pen 17 en 18 in de stekker worden doorverbonden zodat, bij het insteken van de modemkabel in de RS232-connector het kloksignaal aanwezig is.
   Aan de modemzijde van de kabel GEEN DOORVERBINDINGEN maken. *1

5. De verbindingskabel tussen het modem en de RS232-connector aan de computer heeft bij mij de volgende pen aansluitingen:

Computerzijde:	<=====>	Modemzijde:
Pen 1: Dial uit	aan	Pen 1: Dial in (ad 1)
2: TxD (PL7-2)	aan	2: TxD
3: RxD (PL7-3)	aan	3: RxD
4: RTS (PL7-4)	aan	4: RTS
5: CTS (PL7-5)	aan	5: CTS
6: DSR (PL7-6)	aan	6: DSR
7: GND (PL7-7)	aan	7: GND
8: DCD (PL7-8)	aan	8: CD
9: Line relais signaal	aan	21: Line relais (ad 2)
10 t/m 16 n.c.		9 t/m 19 n.c.
17:- |		n.c.
| doorverbinden		
18:- |		n.c.
19: n.c.		n.c.
20: DTR (PL7-10)	aan	20: DTR
21 t/m 25 n.c.		22 t/m 25 n.c

6. De auto-dialer van Elektuur. *2

De auto-dialer heb ik op een stukje vero-board gebouwd. Wanneer dit met enig overleg gebeurt en mini-relais worden gebruikt kan alles op een printplaatje van circa 10 x 6 cm. zodat de auto-dialer in de bestaande modemkast kan worden gemonteerd. Bij mij hangt de auto-dialer, met behulp van dubbelzijdige plaktape, op z´n kop aan het modem deksel.
De voor de auto dialer noodzakelijke 5 volt en massa kunnen van het modem worden betrokken; op het eerste C-tje direkt achter de spanningsregelaar zijn ze beiden beschikbaar.

7. Ad 1:
Het dial-signaal moet afgenomen worden van PB5 van IC3 en aangesloten worden op pen 1 van de RS232-connector.
In het modem moet pen 1 van de connector (wordt toch niet gebruikt) worden verbonden met de "dial in"-aansluiting van de auto-dialer van Elektuur.

8. Ad 2:
Het signaal dat nodig is om het modem automatisch on-line te schakelen bij het automatisch bellen is niet zonder meer beschikbaar. Het benodigde signaal moet afgenomen worden van PB6 van IC3 en geinverteerd worden (zie schema hieronder). De zenerdiode van 4.3 volt is nodig om het line relais, bij een "on hook" commando, te resetten. Zonder de zener komt het line relais niet terug door de nog aanwezige restspanning.
Het on line-signaal wordt, via de zener, aangesloten op pen 9 van de RS232-connector van de computer.
De aansluitingen van PB5, PB6 en PB7 kunnen alle drie afgenomen worden van PL6 op de CPU-kaart op de respectievelijke pennen 5, 3 en 1; deze pennen worden verder toch niet gebruikt. *4

9. Zie figuur 1 en 2

DE CONFIGURATIE:
==========

Aannemende dat u de door Bram de Bruine ontwikkelde communicatie-programmatuur gebruikt staan hieronder de belangrijkste configuratie-gegevens vermeld.

Viditel 3.0:

- Pulse Dialing
- Full duplex
- Originate
- Formaat: 7E1
- Baudrate: 1200/75
- Split speed
- Transparant modem
- Extra initialisation
- VIA T1 oscil.
- Auto reveal
- Banser VIA PB5, 6 control

In de file viditel moet, na de configuratie, met behulp van de editor, "load 0:Banser.dial" worden gezet.

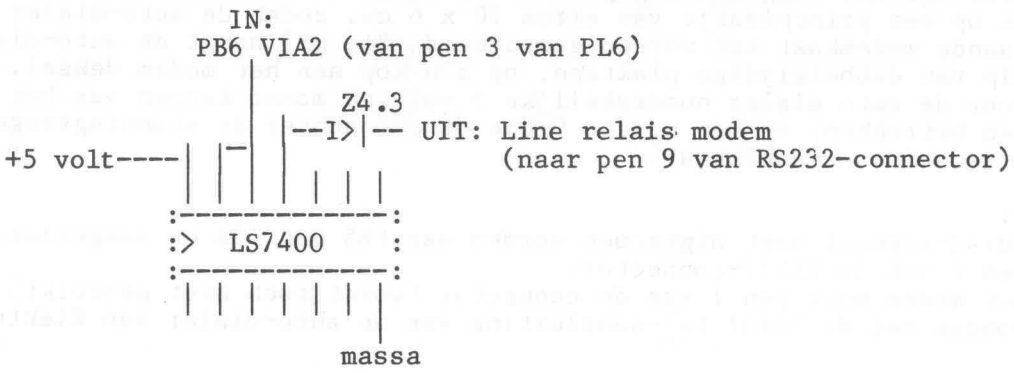
Astrid 3.0 (is nog in testfase):

- Pulse Dialing
- Full duplex
- Originate
- Formaat: 8N1
- Baudrate 300/300
- Split speed
- Transparant modem
- Extra initialisation
- VIA T1 oscil.
- Banser VIA PB5, 6 control

Zoals vermeld is Astrid 3.0 nog in de testfase. De vermelde configuratie is echter ook geldig voor de eerdere versies.

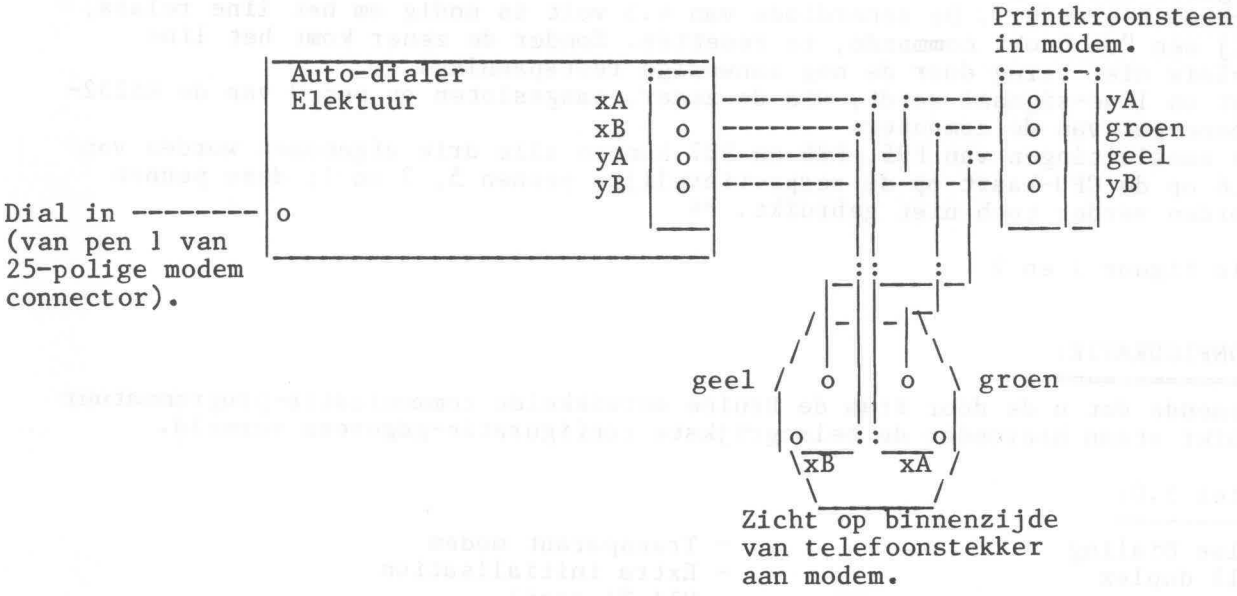
Figuur 1:

Schema inverter voor het automatisch "on line" zetten van het modem.



Figuur 2

Aansluitschema van de auto-dialer.



De aansluitingen aangegeven met geel en groen zijn reeds bestaande aansluitingen.

Tot slot:

Bij mij functioneert het modem, aangesloten op de omschreven wijze, probleemloos op een DOS65-computer welke op 2 MHz draait.
Alle vereiste handelingen, welke zonder auto-dialer met de hand moeten plaatsvinden, worden nu bestuurd met het toetsenbord hetgeen de zaak aanmerkelijk vereenvoudigt.

Geraadpleegde literatuur:

*1: COM65.utilities van Bram de Bruine.
*2: Elektuur juli/augustus 1987, pag.78 en 79.
*3: DOS-65 Hardware-manual van Erwin Visschendijk.
*4: Elektuur Computing nr.1

```
**************************************
***                                ***
***   Spanningen meten met Dos65   ***
***                                ***
**************************************
```

Frank Vandekerkhove,
Sint Michielsstraat 4,
B - 2789 Verrebroek,
03-773.27.94

Toen ik nog meer met het pottenbakken bezig was dan nu, maak je wel eens een grafiek van het bakproces. Regelmatig lees je dan de temperatuur en noteer je die op een blaadje of netjes op mm-papier. En waarom zou Dos65 dat niet kunnen doen?

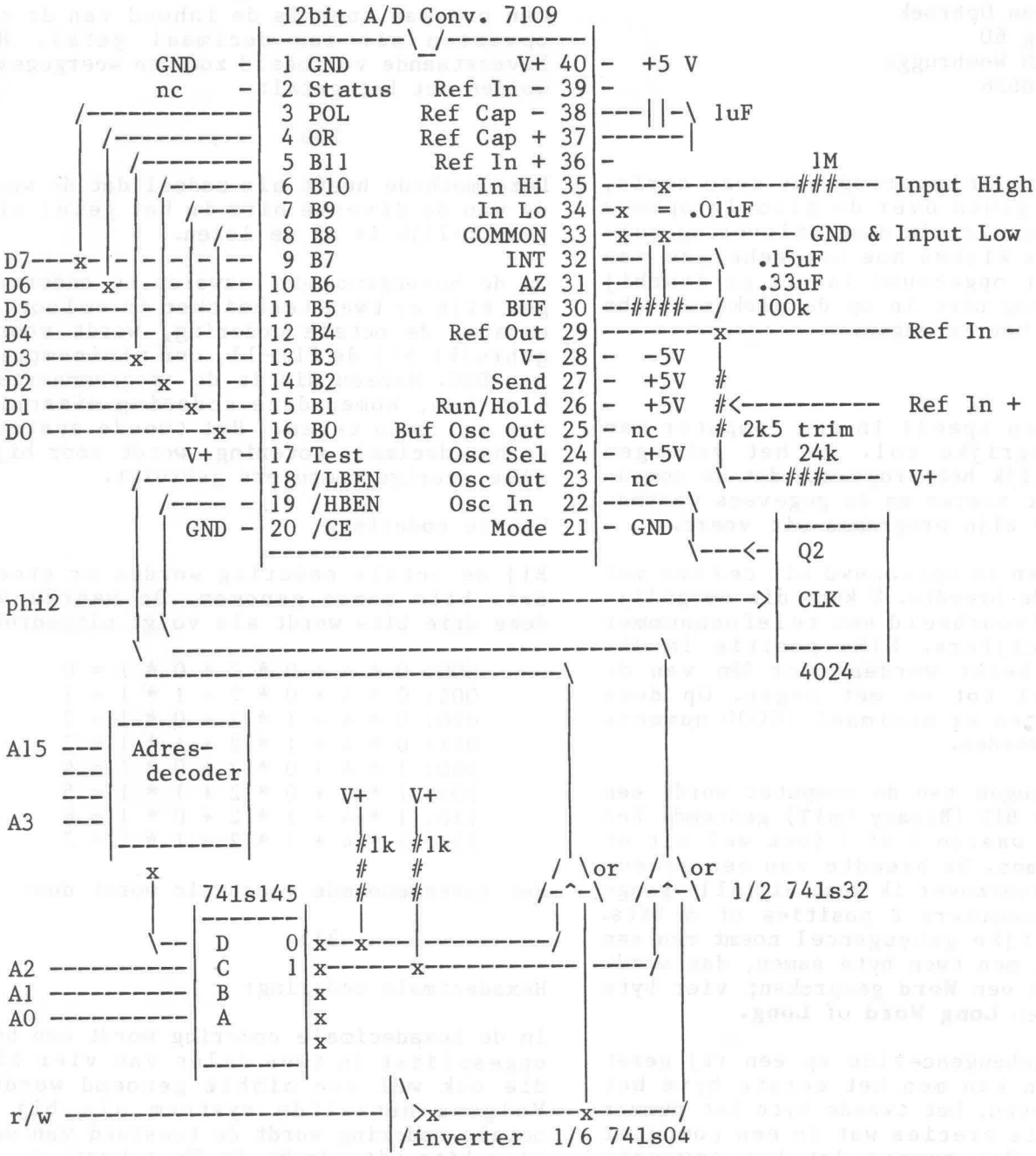
Eerst Dos65 spannigen laten meten: vanwege de eenvoud koos ik voor de A/D converter 7109 van Intersil. Deze A/D converter 7109, met zijn tri-state binaire uitgangen, is de tegenhanger van de 7106 die enorm veel in digitale multimeters wordt gebruikt, mede omdat hij direkt een LCD-display kan aansturen. De 7109 is niet nieuw, noch erg snel, maar wel kompleet met een eigen referentie spanning en toch wat mogelijkheden welke gemakkelijk in de databoeken te vinden zijn. (Ik hoop later de handshake mode met een UART ook voor te stellen.)

In dit artikeltje wil ik nu enkel een aanzet geven. Het schema en het programma zijn heel eenvoudig gehouden: thermokoppelspanning versterking en aanpassing voor NiCr-Ni of PtRh-Pt, het uitgebreid programma en de plotter sturing zijn weggelaten. Wie meer wil weten, of eigen ervaringen wil uitwisselen, kan me altijd schrijven.

De voeding is symmetrisch: + & - 5 Volt. (De computervoeding en een 7905 op de - 12 Volt.) Met de "mode" input laag zijn de uitgangen in tri-state en worden actief met "/CE" en "/LBEN" en/of "/HBEN" (low- en high byte enable). De hoogste vier bits met "POL" en "OR" (polarity en overrange) zijn doorverbonden met de low-byte. De maximale ingangspanning is de dubbele van de aangeboden referentiespanning. (In dit schema: max.= ca. 2 V)

```
; File:     7109.mac
; Program:  7109
; Function: demonstration of the A/D conv. 7109
; By:       Frank Vandekerkhove
;
; *** I/O65 V2.02 subroutines ***
;
posit   equ     $F024           position the cursur to X,Y
prtim   equ     $F583           print the time in status
conver  equ     $FF02           convert 2 bytes hex into 3 bytes dec.
;
; *** io_var ***
;
deci1   equ     $E73A           variables for calculate decimal
deci2   equ     $E73B
deci3   equ     $E73C
```

```
;
; *** Dos65 V2.01 subroutines ***
;
out     equ     $C023           put character
spa     equ     $C032           print space
hexout  equ     $C038           a hexout
print   equ     $C03B           print string after call
;
; *** A/D conv. 7109 ***
;
data    equ     $E200           data form A/D 7109
;
;    ***   START MAIN PROGRAM   ***
;
        org     $1000
;
        jsr     print
        fcc     $0C             clear screen
        fcc     20,10,5         posit cursor
        fcc     '\Ei demo 7109 \En'
        fcc     0
;
read    sei                     disable interrupts
        ldx     data            read data of 7109
        lda     data+1
        stx     input           sava data in input
        sta     input+1
        cli                     enable interrupts
;
        jsr     prtim           update time in status
        ldx     #13             adjust cursor
        ldy     #7
        jsr     posit
testPOL bit     input+1         test b7 (=polarity)
        bmi     POS             if pos=1 then positive
        lda     #'-             get minus-sign
        jsr     out             and print it
        jmp     testOR          jump to next test
POS     jsr     spa             print space
testOR  bit     input+1         test b6 (=overrange)
        bvc     pridec          if or=1 then overrange
        jsr     print
        fcc     ' OR ',0
        jmp     read            read again
pridec  lda     #$0F            clear POL & OR from data
        and     input+1         and keep high byte in accu
        ldx     input           low hex byte in x-reg.
        jsr     conver          covert 2 hex in 3 dec
        lda     deci2           skip high dec
        jsr     hexout          but print med dec
        lda     deci1           get low dec
        jsr     hexout          and print it
;
        jmp     read            read again
;
input   res     2
;
        end
```



Literatuur: * Maxim: "Data Acquisition Catalog 1985"
* Elektuur: 1981 juni: "JC als voltmeter" (G. Sullivan)
1988 mei : "Universele PC-I/O-kaart"

Computers..... (deel 2).

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

Inleiding.

In de vorige aflevering van deze serie, heb ik het gehad over de globale opbouw van een computer. In deze aflevering zullen we eens kijken hoe het geheugen van een compter opgebouwd is. Ik ga daarbij voorlopig nog niet in op de elektronische opbouw van het geheugen.

Geheugen.

Het geheugen speelt in een computer een zeer belangrijke rol. In het geheugen staat namelijk het programma dat de computer uit moet voeren en de gegevens waarmee de computer zijn programma uit voert.

Het geheugen is opgebouwd uit cellen met een bepaalde breedte. U kunt dit vergelijken met bijvoorbeeld een telefoonnummer van vier cijfers. Elke positie in dit getal kan bezet worden door één van de cijfers nul tot en met negen. Op deze manier kunnen er maximaal 10000 nummers gecodeerd worden.

In het geheugen van de computer wordt een positie een BIT (**B**inary Un**IT**) genoemd. Een bit kan de waarde 0 of 1 (ook wel uit of aan) aannemen. De breedte van een geheugencel is voorzover ik weet bij alle gangbare microcomputers 8 posities of 8 bits. Een dergelijke geheugencel noemt men een **Byte**. Neemt men twee byte samen, dan wordt meestal van een **Word** gesproken; vier byte noemt men een **Long Word** of **Long**.

Als alle geheugencellen op een rij gezet worden, dan kan men het eerste byte het nummer 0 geven, het tweede byte het nummer 1 etc. Dit is precies wat in een computer gedaan is. Het nummer dat het gewenste byte aangeeft, is het **Adres** van de geheugencel. Elke geheugencel kan dus aangesproken worden met zijn unieke adres.

Codering van de inhoud van een byte.

De inhoud van een geheugencel kan men aangeven door alle bits in deze cel achter elkaar te zetten dus bijvoorbeeld:

11001101

Dit is de binaire codering. In de praktijk is dit echter nauwelijks hanteerbaar. Wat men ook kan doen is de inhoud van de cel opvatten als een decimaal getal. Het bovenstaande voorbeeld zou dan weergegeven worden met het getal:

205

Deze methode heeft als nadeel dat de waarde van de diverse bits in het getal niet gemakkelijk is af te lezen.

Om de bovengenoemde bezwaren te ondervangen zijn er twee technieken in omloop. De eerste, de octale codering, wordt vooral gebruikt bij de PDP-11, een minicomputer van DEC. Mensen die in de rogrammeertaal C werken, komen deze codering misschien nog wel eens tegen. Het tweede systeem, de hexadecimale notering, wordt voor bijna alle overige computers gebruikt.

Octale codering:

Bij de octale codering worden er steeds drie bits samen genomen. De waarde van deze drie bits wordt als volgt uitgedrukt:

```
000: 0 * 4 + 0 * 2 + 0 * 1 = 0
001: 0 * 4 + 0 * 2 + 1 * 1 = 1
010: 0 * 4 + 1 * 2 + 0 * 1 = 2
011: 0 * 4 + 1 * 2 + 1 * 1 = 3
100: 1 * 4 + 0 * 2 + 0 * 1 = 4
101: 1 * 4 + 0 * 2 + 1 * 1 = 5
110: 1 * 4 + 1 * 2 + 0 * 1 = 6
111: 1 * 4 + 1 * 2 + 1 * 1 = 7
```

Het bovengenoemde voorbeeld wordt dus:

315

Hexadecimale codering:

In de hexadecimale codering wordt een byte opgesplitst in twee delen van vier bits die ook wel een nibble genoemd worden. Volgens hetzelfde systeem als bij de octale codering wordt de toestand van deze vier bits uitgedrukt in één teken:

```
0000: 0 * 8 + 0 * 4 * 0 * 2 + 0 * 1 = 0
0001: 0 * 8 + 0 * 4 * 0 * 2 + 1 * 1 = 1
0010: 0 * 8 + 0 * 4 * 1 * 2 + 0 * 1 = 2
0011: 0 * 8 + 0 * 4 * 1 * 2 + 1 * 1 = 3
0100: 0 * 8 + 1 * 4 * 0 * 2 + 0 * 1 = 4
0101: 0 * 8 + 1 * 4 * 0 * 2 + 1 * 1 = 5
0110: 0 * 8 + 1 * 4 * 1 * 2 + 0 * 1 = 6
0111: 0 * 8 + 1 * 4 * 1 * 2 + 1 * 1 = 7
1000: 1 * 8 + 0 * 4 * 0 * 2 + 0 * 1 = 8
1001: 1 * 8 + 0 * 4 * 0 * 2 + 1 * 1 = 9
1010: 1 * 8 + 0 * 4 * 1 * 2 + 0 * 1 = A
```

```
1011: 1 * 8 + 0 * 4 * 1 * 2 + 1 * 1 = B
1100: 1 * 8 + 1 * 4 * 0 * 2 + 0 * 1 = C
1101: 1 * 8 + 1 * 4 * 0 * 2 + 1 * 1 = D
1110: 1 * 8 + 1 * 4 * 1 * 2 + 0 * 1 = E
1111: 1 * 8 + 1 * 4 * 1 * 2 + 1 * 1 = F
```

Het bovenstaande voorbeeld wordt dus:

CD

Om duidelijk te maken welke wijze van codering gebruikt wordt, moet men in het algemeen door middel van een voorteken aangeven welke codering men gebruikt. Veel voorkomende voortekens zijn:

Geen	Decimaal
%	Binair
$ of 0x	Hexadecimaal

Voor de volledigheid geef ik ook nog even de BCD-codering. Deze codering wordt onder meer gebruikt op de IBM-mainframes. Hierbij wordt een byte ook opgesplitst in twee delen van vier bits echter van de zestien mogelijke combinaties worden alleen de eerste tien gebruikt. De overige zes zijn illegale codes. Het bovengenoemde voorbeeld kan ik dus niet in BCD uitdrukken.

Teken-codering.

Daar ook de teksten die naar printers en andere uitvoerapparaten gestuurd worden in het geheugen staan, zijn de tekens van deze teksten ook gecodeerd. Om een teken te coderen wordt nagenoeg altijd één byte per teken gebruikt. Voor de manier waarop bijvoorbeeld de letter 'A' opgeslagen wordt zijn er twee standaards. EBCDIC en ASCII. EBCDIC van Extended Binary Coded Decimal Interchange Code) wordt gebruikt op de IBM-mainframes. De overige computers gebruiken allemaal ASCII van American Standard Code for Information Interchange). Binnen de ASCII-standaard worden 128 van de mogelijke 256 bitcombinaties gedefinieerd. Dit zijn de codes $00 t/m $7F. De overige 128 codes worden meestal gebruikt voor grafische tekens of iets dergelijks maar hierin bestaat helaas nog geen standaard.

Busbreedte.

Eén van de verschillen tussen de diverse soorten microcomputers is de busbreedte.

1. De adresbus.

Om een bepaalde geheugencel aan te spreken, wordt door de processor het adres van de cel op de adresbus gezet. Dit betekent dat bepaalde lijnen op de adresbus op een spanning van 5 Volt gezet worden en de overige lijnen worden op 0 Volt gehouden. Het adres wordt zodoende binair gecodeerd op de adresbus gezet. De geheugencel met het bijbehorende adres wordt dan aangesproken en afhankelijk van een signaal in de control-bus zal deze geheugencel zijn inhoud op de databus zetten (read- of lees-cyclus) of zal de toestand van de databus overgenomen worden in de geheugencel (write- of schrijfcyclus).

Het aantal lijnen in de adresbus bepaalt dus hoeveel geheugen op de processor aangesloten kan worden.

In de tabel staan enkele voorbeelden (de afkorting kB betekent kilobyte of te wel eenheden van 1024 byte, evenzo de afkorting MB voor Megabyte = eenheden van 1024 * 1024 en GB voor Giga byte = 1024 * 1024 * 1024 byte):

Processor	Lijnen	Grootte
6502	16	64 kB
Z80	16	64 kB
8088	20	1 MB
80286	24	16 MB
68008	24	16 MB
68000	24	16 MB
68020	32	4 GB

2. De Databus.

Stel, een processor wil bij de inhoud van een geheugencel de waarde 4 optellen. Hij moet hiervoor eerst de inhoud van de geheugencel lezen, intern in de processor er vier bij optellen en daarna de nieuwe waarde terugschrijven. Nu kan het voorkomen dat bij deze optelling het resultaat zo groot wordt, dat deze niet meer in een geheugencel past. Vaak neemt men dan de inhoud van de volgende (of vorige) geheugencel en telt hierbij het deel dat niet in de andere cel past op. (Zie getallen deel 1, Kenner nr. 58). Dit betekent dus weer een byte lezen, bewerken en terugschrijven.

De nieuwere processoren (8088, 68000) kunnen voor deze gevallen vaak intern meerdere bytes gelijktijdig bewerken. In dat geval zou het handig zijn, als de processor in één keer ook twee of meer geheugencellen kan lezen of schrijven. Dit betekent dus dat er in totaal bijvoorbeeld 16 bits gelijktijdig benaderd worden. Om dit te kunnen moet de databus dan dus uit

16 lijnen bestaan en spreekt men van een 16 bits processor. De processor spreekt dan de cel, aangegeven door het adres op de adresbus en de cel met het naast hoger gelegen adres aan. In de tabel staan voor een aantal processoren de breedte van de databus opgegeven.

Processor	Lijnen
6502	8
Z80	8
8088	8
80286	16
68000	16
68020	32

In aflevering 1 is al aangegeven dat er bij de 68000 beperkingen aan het aangeboden adres zijn. Bij deze processor bestaat de adresbus namelijk in werkelijkheid niet uit 24 doch uit 23 bits. Het laagste bit is namelijk niet op de adresbus beschikbaar. De processor adresseert namelijk altijd twee bytes. Door middel van signalen in de control-bus geeft hij bovendien aan of hij het byte op het lage adres, het hoge adres of beide wil benaderen. De inhoud van het byte met het lage adres wordt via de hoogste bits van de databus benaderd, het byte met het hoge adres via de lage bits van de databus. Dit betekent dat, als men twee bytes gelijktijdig wil benaderen men altijd een byte-paar of word moet benaderen dat begint op een even adres.

Geheugengebruik

Wat doet een processor met zijn geheugen? Welnu, die vraag is eigenlijk zeer eenvoudig te beantwoorden. In het geheugen vindt de processor het programma dat hij uit moet voeren en de data die hij met dat programma moet verwerken. De processor kan dus niet met iets werken dat niet in het geheugen beschikbaar is. Als de programmeur van een computer wil dat de processor op een bepaald moment de getallen 2 en 3 bij elkaar optelt, dan staan de opdracht "Optellen" en de getallen 2 en 3 altijd ergens in het geheugen.

Wat nu als de informatie (programma of gegevens) niet in het geheugen staat maar bijvoorbeeld ergens op een floppy disk? Dan draait de processor een programmaatje waarmee hij de floppy-disk controler (een stuk programmeerbare I/O) vraagt of die zo vriendelijk wil zijn de betreffende gegevens op te halen. De controler zoekt de gegevens op en daarna zijn er in principe twee mogelijkheden:

1) De controler geeft de gegevens byte voor byte via de databus aan de processor die ze daarna in het geheugen zet.

2) De processor geeft aan waar hij de gegevens in het geheugen wil hebben waarna de controler ze daar zelfstandig neerzet. Nadat dit gebeurd is, krijgt de processor een seintje dat de controler de opdracht uitgevoerd heeft.

De eerste techniek vinden we o.a. bij DOS-65, de tweede bij de PC's en wordt DMA naar **D**irect **M**emory **A**ccess genoemd.

Nadat de informatie opgehaald is, kan deze door de processor verwerkt worden. Moeten er daarna weer gegevens op schijf teruggeschreven worden of uitgeprint worden op bijvoorbeeld een printer, dan worden de gegevens meestal weer ergens in het geheugen achtergelaten waarna ze door een controler verder behandeld worden. Ook hier kunnen weer de twee bovengenoemde technieken gebruikt worden. Hoe één en ander in zijn werk gaat wordt in een volgende aflevering behandeld.

Afsluiting

In deze aflevering zijn een aantal basisbegrippen over het geheugen van een computer geïntroduceerd. Ik hoop dat de beripen Bit, Byte en Hexadecimale notatie u vanaf nu bekend in de oren zullen klinken.

Ik heb mij in deze aflevering niet bezig gehouden met de manier waarop het allemaal elektonisch in zijn werk gaat. Ook heb ik me niet uitgelaten over de verschillende typen geheugen. Ik wil, als ik met de logische beschrijving van de computer klaar ben, daar in een volgende aflevering nog graag op terugkomen.

In de volgende aflevering van deze serie zullen we eens een kijkje nemen binnen de processor van een microcomputer.

Tenslotte wil ik ook de assemblercursus voor DOS-65 van Antoine Megens van harte bij u aanbevelen. Deze cursus is weliswaar 6502-georienteerd maar bevat zoveel algemene informatie dat die ook voor anderen zeer interessant kan zijn.

==========

KALENDER.PAS op DOS-65.

From: Antoine Megens
To: All
Subject: KALENDER.PAS

Msg #22, 29-Dec-88 22:11

Het KALENDER.PAS programma moet op de volgende punten worden aangepast om te draaien op het DOS65 systeem:

- Verwijder de regels met ASSIGN en CLOSE (dit was systeem afhankelijk). Voor DOS65 moet de filenaam achter de executable worden gezet, bijv. u:KAL KALENDER.TXT voor file KALENDER.TXT

- Verkort de programma naam in KAL o.i.d. KALENDER.PAS geeft een error (naam te lang?)

- Controleer of de laatste END wel is afgesloten met een . , dus END.

- Verander de check voor de ondergrens bij PROCEDURE Jaartal in
  IF (P_Jaar <= C_Ondergrens) OR ..... (etc)
  anders krijg je die foutmelding
  >>>>>>>>>> FOUT IN JAARTAL <<<<<<<<<<<
  kennelijk wordt de expressie niet goed bepaald, hij is namelijk TRUE (probeer maar eens WRITELN(P_JAAR < C_Ondergrens); op die plaats) (Ik heb een korter programma geprobeerd met een zelfde expressie en daar ging het wel goed, dus wellicht nog een bug?) Ook in PROCEDURE Kopregel staat zo'n expressie, verander dit in:
  IF (P_Kwartaal <= 0) OR ..... (etc)
  Dat blijft hetzelfde want er wordt met INTEGER waardes gewerkt. Kennelijk werkt de < functie niet altijd goed, de <= functie wel.

Als je deze wijzigingen hebt aangebracht moet je de file compileren met de -E optie want er wordt gebruik gemaakt van '_' karakters in de variabele namen. Dus:
PASCAL -E KAL
Als dat werkt kun je de geproduceerde BIN file RENAMEn naar KAL en testen:
U:KAL KALENDER.TXT
Na invoer van het jaartal verschijnt de kalender op het scherm (met een paar spaties teveel achter de regel, waardoor er een AUTO Linefeed optreedt) en tevens wordt deze data in de file KALENDER.TXT gezet.
Succes ermee!!

Electronic Greetings, Antoine

Een waarschuwing tegen het misbruiken van variant records in Pascal

Pieter de Visser

Met dit artikel reageer ik op de twee Pascal programma's van Gert van Opbroek in De 6502 kenner 59. In beide programma's worden variant records gebruikt op een manier die niet in overeenstemming is met de Pascal definitie (beschreven in de derde editie van de "Pascal user manual and report" van Jensen en Wirth).
Volgens de Pascal definitie kan er hoogstens één variant van een bepaalde variant part actief zijn. Indien er een tag field is, is dat de variant die overeenkomt met de waarde van dat tag field; omdat er wordt geëist dat bij alle mogelijke waarden van het tag type er een variant bestaat, is er dus altijd een variant actief (als de waarde van het tag field gedefinieerd is tenminste). Indien er geen tag field is, is de actieve variant die welke het laatst gebruikt is. Bovendien is de waarde van een inactieve variant niet gedefinieerd.
Als voorbeeld zal ik een deel van een typedefinitie uit het programma "Kalender" van Gert gebruiken:

```
type T_C8  = PACKED ARRAY [1.. 8] OF CHAR;
     T_C32 = PACKED ARRAY [1..32] OF CHAR;
     T_C72 = PACKED ARRAY [1..72] OF CHAR;
     T_Mogelijkheid = 1..4;
     T_Regel = PACKED RECORD
                 Weekdag            : T_C8;
                 CASE T_Mogelijkheid OF
                 1: ( Regel_Totaal  : T_C72);
                 2: ( Voorloop      : T_C32;
                      Jaartal       : T_C8;
                      Naloop        : T_C32);
                 3: ...
                 4: ...
               END;
```

Het type T_Regel wordt gebruikt voor het formatteren van de uitvoer. Dit wordt bijv. gedaan door eerst het Regel_Totaal field geheel met spaties te vullen, vervolgens de eigenlijke uitvoer in het Jaartal field te zetten, en tenslotte het Regel_Totaal field af te drukken. Uit het voorgaande volgt dat dit volgens de Pascal definitie niet is toegestaan.
Hoewel het in principe mogelijk is dat een Pascal implementatie in een dergelijk geval een foutmelding geeft, gaat het in veel gevallen wel goed. Veel implementaties zullen de eerste en tweede variant van T_Regel op dezelfde plaats in het geheugen zetten, en de fields van de tweede variant bovendien in de volgorde van declaratie. Andere implementaties kunnen hiertoe gedwongen worden door een PACKED RECORD te declareren.
Bij DOS-65 Pascal gaat het echter een beetje anders.
De DOS-65 Pascal compiler deelt, bij het bepalen van de offsets van de fields in een fieldlist, de fields op grond van hun type in twee groepen in: enerzijds de fields met zgn. pordinal types (integer, char, boolean, enumerated types, subranges en pointers), anderzijds de overige fields. De compiler deelt offsets voor de fields in een fieldlist van laag naar hoog uit. Eerst worden offsets gegeven aan de fields met een pordinal type in de fixed part; de volgorde in het geheugen is hierbij dezelfde als die van de declaraties, maar omgekeerd t.o.v. de volgorde binnen een declaratie. Indien er een variant part gedefinieerd is, worden daarna, per variant, recursief offsets gegeven aan de fields in die variant. De begin-offset is hierbij voor alle varianten gelijk . Tenslotte worden offsets uitgedeeld aan de fields met een non-pordinal type in de fixed part, in omgekeerde volgorde t.o.v. hun declaratie.
Deze werkwijze heeft tot gevolg dat, als er een pointer naar een record in page 0 staat, in vrijwel alle gevallen de fields met een pordinal type met de [...],y addressing mode bereikt kunnen worden. Hierdoor kan er efficiëntere code gegenereerd worden. Een nadeel is wel dat programma's die uitgaan van een andere implementatie, niet goed zullen werken.
In het gegeven voorbeeld gaat het toevallig wel goed. De eerste en tweede variant van T_Regel komen op dezelfde plaats in het geheugen te staan. Weliswaar is hierbij de volgorde van de fields in de tweede variant eerst Naloop, dan Jaartal en tenslotte Voorloop, maar omdat Voorloop en Naloop even lang zijn en niet gebruikt worden, maakt dat geen verschil.
De moraal van dit verhaal is inmiddels wel duidelijk: het is gevaarlijk te proberen het type mechanisme van Pascal d.m.v. variant records te omzeilen. In de meeste gevallen is dit ook niet nodig; dit geldt ook voor Kalender. Wie gewoon netjes de Pascal definitie volgt bij zijn programma's, kan dit alles verder dus gewoon vergeten.

Getallen (deel 3)

Door Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

Inleiding.

In de eerste aflevering van deze serie heb ik wat algemene getaltheorie behandeld en de manier waarop een computer met gehele getallen (Integers) omgaat. In de tweede aflevering zijn de floating-point (drijvende komma) getallen geïntroduceerd. Bovendien is toen uitgelegd hoe een dergelijk getal in het geheugen van een computer opgeslagen wordt.

In deze en de volgende aflevering gaan we ons bezig houden met de manier waarop in een computer berekeningen met floating-point getallen uitgevoerd worden. Bij deze artikelen is software voor een floating-point pakket voor de 6502 micro-processor gevoegd. Deze software is afgeleid van een reeks artikelen in het maandblad mc (ref. 1 t/m 4). Mensen die geen 6502-processor in hun systeem hebben en we graag een fp-pakket zouden willen hebben, kunnen deze misschien in dit blad vinden. Mc publiceerde een dergelijk pakket voor 68000, 8088 en Z80.

Eén van de reacties op de eerste aflevering was de opmerking dat er bij berekeningen zoveel geschoven wordt. Dat sloeg dan met name op het vermenigvuldigen en delen van gehele getallen. Degene die de opmerking maakte zal nu tot de conclusie komen dat bij floating point getallen het allemaal nog enkele factoren erger is. In het bijgevoegde programma wordt haast niets anders gedaan dan roteren en schuiven.

Optellen.

Hoe twee gehele getallen bij elkaar opgeteld worden is in deel 1 behandeld. In deze aflevering gaan we eens kijken hoe twee drijvende komma getallen bij elkaar opgeteld worden. Kijken we eens naar het volgende voorbeeld:

```
1.234E+2 + 3.000E+1 =
1.234E+2 + 0.300E+2 = 1.534E+2
```

In dit voorbeeld zien we een manier om dergelijke berekeningen uit te voeren. We nemen het getal met de kleinste exponent en veranderen dit getal zodanig dat de exponent van beide getallen gelijk is. Daarna tellen we de mantissa's van de getallen bij elkaar op. De exponent wordt daar dan aan toegevoegd.

In een tweede voorbeeld zien we dat we het getal soms opnieuw moeten normeren:

```
9.934E-2 + 5.000E-3 =
9.934E-2 + 0.500E-2 = 10.434E-2 =
                       1.043E-1
```

Verder komt het natuurlijk ook wel voor dat de exponenten van de getallen zoveel verschillen dat het resultaat niet meer merkbaar door het kleinste getal beïnvloed wordt:

```
1.000E+12 + 1.000E-12 = 1.000E+12
```

Tenslotte nog even over het afronden. Iedereen kent wel de manier waarop normaal afgerond wordt, namelijk 5 en hoger naar boven en 4 en lager naar beneden dus:

```
1.234E+2 + 9.002E+2 = 10.236E+2 =
                       1.024E+1
```

vooropgesteld dat we het resultaat uit willen drukken in 4 decimalen.

Negatieve getallen.

Hebben we een optelling van twee negatieve getallen, dan zal het resultaat altijd negatief zijn:

```
-1.23E+1 + -2.23E+1 = -3.46E+1
```

Anders wordt het bij optellingen van een positief en een negatief getal:

```
-2.31E-4 + +1.10E-4 = -1.21E-4
-1.10E-4 + +2.31E-4 = +1.21E-4
```

In dit geval heeft het resultaat het teken van het getal met de grootste absolute waarde. De absolute waarde van het resultaat is nu het verschil in absolute waarde tussen het grootste getal en het kleinste getal.

Algoritme.

Om twee floating point getallen bij elkaar op te tellen kan men gebruik maken van het volgende algoritme (rekenvoorschrift):

1) Vergelijk de absolute waarde van beide getallen en verwissel ze eventueel zodat het kleinste getal bij het grootste getal opgeteld wordt.

2) Verschuif de komma in het kleinste getal zover naar voren dat de exponent van beide getallen gelijk wordt; voor elke positie verschuiving wordt de exponent met één opgehoogd. Verschillen de exponenten meer dan het aantal gewenste decimalen voor het resultaat + 1, dan is het resultaat het grootste getal.

3) Zijn de tekens van de getallen gelijk, dan de mantissa's bij elkaar optellen, zijn ze ongelijk, dan worden ze van elkaar afgetrokken.

4) Bij het optellen kan het voorkomen dat het resultaat groter is dan het grootste getal dat we weer kunnen geven. Dit noemen we overflow en in dit geval wordt het kenmerk "oneindig" als resultaat gegeven. Evenzo kan het voorkomen dat het resultaat kleiner wordt dan het kleinste getal, in absolute zin, dat we weer kunnen geven. In dat geval wordt het getal 0 (nul) als resultaat gegeven.

5) Na optellen kan het zijn dat we de komma één positie naar links moeten schuiven, na aftrekken kan het voorkomen dat de komma meerdere posities naar rechts geschoven moet worden. De exponent wordt hierbij steeds bijgewerkt. Deze bewerking heet normeren.

6) Het resultaat krijgt tenslotte het teken van het grootste getal (in absolute zin), tenzij het resultaat 0 is. Nul heeft altijd teken + en exponent 0.

Aftrekken.

Als je kunt optellen, dan kun je eigenlijk ook aftekken. Inverteer het teken van de aftrekker en tel de zo onstane getallen met het bovenstaande algoritme bij elkaar op:

```
-1.2E+1 -  2.0E+00 =
-1.2E+1 + -2.0E+0 = -1.0E+1
```

Binaire getallen.

De voorbeelden in de vorige paragrafen waren allemaal voorbeelden met decimale getallen. Uiteraard is de bewerking voor binaire getallen niet anders. Ook hier wordt met de mantissa geschoven totdat de exponenten gelijk zijn, waarna de mantissa's, afhankelijk van de tekens, van elkaar afgetrokken of bij elkaar opgeteld worden. Ook in dit geval vindt er daarna een normering plaats.

Om te demonstreren dat het allemaal echt werkt, heb ik een tweetal subroutines geschreven die getallen in het IEEE-single precision formaat bij elkaar op kunnen tellen of van elkaar af kunnen trekken. Deze routines zijn ontwikkeld op mijn JUNIOR met PROTON-SENIOR DOS en zouden vrij eenvoudig omgezet moeten kunnen worden naar elk ander 6502-systeem.

De routines werken precies volgens de bovenstaande algoritmen. Punt van aandacht is misschien de manier waarop de parameters aan de routines doorgegeven worden. Dit gebeurt namelijk via de stack, een techniek die ik vooral ken van de andere systemen waarop ik werk.

De aanroep van de routines is als volgt:

1) Zet de eerste parameter op de stack, het laagste byte als laatste (dus op het laagste adres).

2) Zet hierna de tweede parameter op de stack ook met het laagste byte op het laagste adres.

3) Roep de routine aan. in de routine wordt eerst het return-adres weggehaald, waarna de parameters één voor één ingelezen worden. Mensen met een macro-assembler kunnen hiervoor het beste een macro schrijven want deze bewerking zal in de toekomst nog wel vaker voorkomen.

4) Voor de return uit de subroutine wordt eerst het resultaat op de stack geschreven waarna het return-adres weer weggeschreven wordt. De aanroepende routine vindt dus het resultaat boven op de stack (met het laagste byte als eerste).

Tenslotte.

In de volgende afleveringen hoop ik routines voor vermenigvuldigen en delen te kunnen presenteren. Bovendien wil ik ook de conversie-routines van en naar ASCII-strings in assembler schrijven.

```
6502 Floating Point Package    PROTON 650X ASSEMBLER  V4.4   PAGE: 0001

0001  0000                    .TIT ´6502 Floating Point Package´
0002  0000                    .OPT GEN
0003  0000                    .OPT PR1
0004  0000           ;
0005  0000           ; ********************************************************
0006  0000           ;
0007  0000           ; Floating point package for the 6502 microprocessor
0008  0000           ;
0009  0000           ; Written by G. van Opbroek
0010  0000           ; on JUNIOR with Proton Senior DOS
0011  0000           ; (c) Copyright 1989 Kim Gebruikersclub Nederland.
0012  0000           ;
0013  0000           ; ********************************************************
0014  0000           ;
0015  0000                    *=$0000           ; Page zero definitions
0016  0000                    .EX1
0017  0000           ;
0018  0000                    *=$0200           ; Load address
0019  0000                    .EX1
0020  0000           ;
0021  0000           ; Define page zero locations
0022  0000           ;
0023  0000           ; Return address
0024  0000           ;
0025  0000  0000     RETADD    .WORD 0
0026  0002           ;
0027  0002  00       MAN1      .BYTE 0,0,0       ; Mantissa of parameter 1
0027  0003  00
0027  0004  01
0028  0005  00       EXP1      .BYTE 0           ; Exponent of parameter 1
0029  0006  06       SIGN1     .BYTE 0           ; Sign of parameter 1
0030  0007  00       MAN2      .BYTE 0,0,0       ; Mantissa of parameter 2
0030  0008  00
0030  0009  09
0031  000A  00       EXP2      .BYTE 0           ; Exponent of parameter 2
0032  000B  00       SIGN2     .BYTE 0           ; Sign of parameter 2
0033  000C           ;
0034  0200                    .EX1
0035  0200           ;
0036  0200           ; ********************************************************
0037  0200           ;
0038  0200           ; Tools for simple operations
0039  0200           ;
0040  0200           ; 1: ROTLEF:  Rotate n bytes left by 1 bit.
0041  0200           ;             X : start address on page zero
0042  0200           ;             Y : n = number of bytes to rotate
0043  0200           ;             C : Input-bit/Output-bit
0044  0200           ;
0045  0200           ; 2: ROTRIG:  Rotate n bytes right by 1 bit.
0046  0200           ;             X : start address on page zero
0047  0200           ;             Y : n = number of bytes to rotate
0048  0200           ;             C : Input-bit/Output-bit
0049  0200           ;
0050  0200           ; ********************************************************
0051  0200           ;
0052  0200  3600     ROTLEF    ROL $00,X         ; Rotate byte
0053  0202  E8                 INX               ; Point to next byte
0054  0203  88                 DEY               ; Decrement counter
0055  0204  D0FA               BNE ROTLEF
```

```
6502 Floating Point Package     PROTON 650X ASSEMBLER  V4.4    PAGE: 0002

0056  0206  60              RTS
0057  0207            ;
0058  0207  7600    ROTRIG  ROR $00,X          ; Rotate byte
0059  0209  CA              DEX               ; Point to previous byte
0060  020A  88              DEY               ; Decrement counter
0061  020B  D0FA            BNE ROTRIG
0062  020D  60              RTS
0063  020E            ;
0064  020E            ;
0065  020E            ; Floating point addition:
0066  020E            ; based on: Hagen Volzke Fliesskomma - Arithmetik und
0067  020E            ;           IEEE-Spezifikation
0068  020E            ;
0069  020E            ;           mc 11/88 page 78
0070  020E            ;
0071  020E            FADD                      ; Get parameters from stack
0072  020E  68              PLA                 ; Get return address and save it
0073  020F  8500            STA RETADD
0074  0211  68              PLA
0075  0212  8501            STA RETADD+1
0076  0214            ;
0077  0214  A900            LDA #0              ; Clear signs
0078  0216  850B            STA SIGN2
0079  0218  8506            STA SIGN1
0080  021A            ;
0081  021A            ; Get second parameter from stack (4 byte)
0082  021A            ;
0083  021A  A207            LDX #<MAN2          ; Zero-page relative address
0084  021C  A004            LDY #4              ; 4 byte
0085  021E  68      FADDP1  PLA                 ; Get byte from stack
0086  021F  9500            STA $0000,X         ; Store byte
0087  0221  E8              INX
0088  0222  88              DEY
0089  0223  D0F9            BNE FADDP1
0090  0225            ;
0091  0225            ; Get first parameter from stack (4 byte)
0092  0225            ;
0093  0225  A202    SUBENT  LDX #<MAN1          ; Zero-page relative address
0094  0227  A004            LDY #4              ; 4 byte
0095  0229  68      FADDP2  PLA                 ; Get byte from stack
0096  022A  9500            STA $0000,X         ; Store byte
0097  022C  E8              INX
0098  022D  88              DEY
0099  022E  D0F9            BNE FADDP2
0100  0230            ;
0101  0230            ; Rotate operands to get exponents in one byte
0102  0230            ;
0103  0230  18              CLC
0104  0231  A202            LDX #<MAN1          ; Get zero-page relative address of MAN1
0105  0233  A005            LDY #5              ; Rotate 5 byte, result is:
0106  0235  200002          JSR ROTLEF          ; mmmmmmm0 mmmmmmmm mmmmmmmm
                                                ; eeeeeeee 0000000s
0107  0238            ;
0108  0238  18              CLC
0109  0239  A505            LDA EXP1            ; Exponent zero ?
0110  023B  F001            BEQ FADD1           ; Yes, denormalized, Ror in 0
0111  023D  38              SEC                 ; No, Ror in  1
0112  023E            ;
0113  023E  A204    FADD1   LDX #<MAN1+2        ; Get zero-page relative address
```

```
6502 Floating Point Package    PROTON 650X ASSEMBLER V4.4   PAGE: 0003

0114  0240  A003            LDY #3              ; Rotate mantissa
0115  0242  200702          JSR ROTRIG          ; mmmmmmmm mmmmmmmm mmmmmmm
                                                ; eeeeeeee 0000000s
0116  0245          ;
0117  0245  18            CLC
0118  0246  A207          LDX #<MAN2          ; Get zero-page releative address of MAN2
0119  0248  A005          LDY #5              ; Rotate 5 byte, result is:
0120  024A  200002        JSR ROTLEF          ; mmmmmmm0 mmmmmmmm mmmmmmmm
                                                ; eeeeeeee 000000s
0121  024D          ;
0122  024D  18            CLC
0123  024E  A50A          LDA EXP2            ; Exponent zero ?
0124  0250  F001          BEQ FADD2           ; Yes, denormalized, Ror in 0
0125  0252  38            SEC                 ; No, Ror in  1
0126  0253          ;
0127  0253  A209    FADD2 LDX #<MAN2+2        ; Get zero-page relative address
0128  0255  A003          LDY #3              ; Rotate mantissa
0129  0257  200702        JSR ROTRIG          ; mmmmmmmm mmmmmmmm mmmmmmm
                                                ; eeeeeeee 0000000s
0130  025A          ;
0131  025A          ; The greatest parameter has to be parameter 1,
0132  025A          ; so compare the parameters and exchange when necessary
0133  025A          ;
0134  025A  A004            LDY #4              ; Test 4 bytes
0135  025C  B90100  FADDL3 LDA MAN1-1,Y        ; Get byte of parameter 1
0136  025F  D90600          CMP MAN2-1,Y        ; Compare with byte of parameter 2
0137  0262  9007            BCC FADDEX          ; Exchange when parameter 2 is greater
0138  0264  D014            BNE FADDNE          ; Do not exchange, parameter 1 is greater
0139  0266  88              DEY
0140  0267  D0F3            BNE FADDL3
0141  0269  F00F            BEQ FADDNE          ; 4 bytes tested --> parameters are equal
0142  026B          ;
0143  026B  A005    FADDEX LDY #5              ; Exchange parameters
0144  026D  B601    FADDL4 LDX MAN1-1,Y
0145  026F  B90600          LDA MAN2-1,Y
0146  0272  990100          STA MAN1-1,Y
0147  0275  9606            STX MAN2-1,Y
0148  0277  88              DEY
0149  0278  D0F3            BNE FADDL4
0150  027A          ;
0151  027A  38      FADDNE SEC                 ; Calculate difference between
                                                ; exponents
0152  027B  A505            LDA EXP1
0153  027D  E50A            SBC EXP2            ; We have to shift mantissa 2
                                                ; by this number
0154  027F  18              CLC                 ; Clear the carry for the future
0155  0280  F012            BEQ FADD3           ; If zero >> no shift
0156  0282  C919            CMP #25             ; If > 25 --> result is parameter 1
0157  0284  18              CLC                 ; Clear carry for rounding
0158  0285  102F            BPL FADDR
0159  0287          ;
0160  0287          ; Align mantissa´s
0161  0287          ;
0162  0287  A003    FADDL5 LDY #3              ; Rotate 3 bytes
0163  0289  A209            LDX #<MAN2+2        ; Rotate parameter 2
0164  028B  18              CLC                 ; Shift in zero
0165  028C  200702          JSR ROTRIG          ; Rotate right
0166  028F  AA              TAX                 ; Decrement difference
0167  0290  CA              DEX
```

6502 Floating Point Package PROTON 650X ASSEMBLER V4.4 PAGE: 0004

```
0168  0291  8A              TXA             ; We want to preserve carry
0169  0292  D0F3            BNE FADDL5      ; Difference <> 0 --> Continue
0170  0294          ;
0171  0294  A200    FADD3     LDX #0          ; Start with low byte
0172  0296  A003              LDY #3          ; 3 bytes
0173  0298  A506              LDA SIGN1       ; Compare sign bits
0174  029A  450B              EOR SIGN2
0175  029C  D00C              BNE FADDSU      ; If unequal subtract, else
                                              ; add mantissa's
0176  029E          ;
0177  029E          ; Signs are equal: we have to add the mantissa's
0178  029E          ;
0179  029E  B502    FADDL6    LDA MAN1,X      ; Get mantissa 1
0180  02A0  7507              ADC MAN2,X      ; Add mantissa 2
0181  02A2  9502              STA MAN1,X      ; Store result in parameter 1
0182  02A4  E8                INX
0183  02A5  88                DEY
0184  02A6  D0F6              BNE FADDL6
0185  02A8  F00C              BEQ FADDR
0186  02AA          ;
0187  02AA          ; Signs are unequal, subtract the mantissa of the smaller
0188  02AA          ; number from the mantissa of the greater one.
0189  02AA          ;
0190  02AA  38      FADDSU    SEC             ; Subtract mantissa's
0191  02AB  B502    FADDL7    LDA MAN1,X      ; Get mantissa 1
0192  02AD  F507              SBC MAN2,X      ; Subtract mantissa 2
0193  02AF  9502              STA MAN1,X      ; Save result in parameter 1
0194  02B1  E8                INX
0195  02B2  88                DEY
0196  02B3  D0F6              BNE FADDL7
0197  02B5  18                CLC             ; Carry is always set; clear it !!
0198  02B6          ;
0199  02B6          ; Normalize the result
0200  02B6          ;
0201  02B6  B010    FADDR     BCS FADD4       ; Normalize and round the result
0202  02B8  A003              LDY #3          ; Test mantissa for zero
0203  02BA  B90100  FADDL8    LDA MAN1-1,Y
0204  02BD  D027              BNE FADD5       ; Result <> 0
0205  02BF  88                DEY
0206  02C0  D0F8              BNE FADDL8
0207  02C2  8505              STA EXP1        ; Store zero in the exponent too
0208  02C4  8506              STA SIGN1       ; And the sign
0209  02C6  F050              BEQ FADDRE      ; Restore stack and exit
0210  02C8          ;
0211  02C8  901C    FADD4     BCC FADD5       ; Carry set ? Yes, shift result
0212  02CA  A204              LDX #<MAN1+2    ; Shift mantissa right
0213  02CC  A003              LDY #3          ; 3 byte
0214  02CE  200702            JSR ROTRIG
0215  02D1  E605              INC EXP1        ; Increment exponent
0216  02D3  F034              BEQ FADD6       ; If zero after INC ---> overflow !
0217  02D5          ;
0218  02D5          ; When carry is set, add 1 for rounding
0219  02D5          ;
0220  02D5  900F              BCC FADD5       ; Carry clear --> guard bit was zero
0221  02D7  A003              LDY #3          ; 3 Byte
0222  02D9  A200              LDX #0
0223  02DB  F602    FADDLA    INC MAN1,X
0224  02DD  D007              BNE FADD5       ; No carry in addition
0225  02DF  E8                INX
```

6502 Floating Point Package PROTON 650X ASSEMBLER V4.4 PAGE: 0005

```
0226  02E0  88              DEY
0227  02E1  D0F8            BNE FADDLA
0228  02E3  38              SEC                 ; Carry on this addition
0229  02E4  F0E2            BEQ FADD4           ; Re-do normalize
0230  02E6          ;
0231  02E6  A505    FADD5   LDA EXP1            ; Test exponent
0232  02E8  F014            BEQ FADD7           ; Zero --> denormalized
0233  02EA  C9FF            CMP #$FF            ; Overflow ?
0234  02EC  F01B            BEQ FADD6           ; Return overflow
0235  02EE          ;
0236  02EE          ; This can be a normalized number, shift until the
0237  02EE          ; MS-bit is one and shift this bit out of the number
0238  02EE          ;
0239  02EE  A003            LDY #3              ; 3 byte mantissa
0240  02F0  A202            LDX #<MAN1          ; Rotate left
0241  02F2  18              CLC                 ; Shift in zero bit
0242  02F3  200002          JSR ROTLEF
0243  02F6  C605            DEC EXP1
0244  02F8  90EC            BCC FADD5           ; Continue until carry is set
                                                ; (MS-bit)
0245  02FA  E605            INC EXP1            ; Exponent is decremented 1 to much; correct
0246  02FC  B01A            BCS FADDRE          ; Number is normalized: restore
                                                ; stack and exit
0247  02FE          ;
0248  02FE          ; Denormalized numbers: EXP = 0 MS-bit is 0
0249  02FE          ; If MS-bit = 1 --> number becomes normalized
0250  02FE          ;
0251  02FE  A004    FADD7   LDY #4              ; Rol in one bit
0252  0300  18              CLC
0253  0301  A202            LDX #<MAN1          ; Rotate 1 bit and put this in exp
0254  0303  200002          JSR ROTLEF
0255  0306  4C1803          JMP FADDRE          ; Restore stack and exit
0256  0309          ;
0257  0309          ; Overflow: preserve sign, clear mantissa, exponent = $FF
0258  0309          ;
0259  0309  A9FF    FADD6   LDA #$FF            ; Exponent = maximum
0260  030B  8505            STA EXP1
0261  030D  A900            LDA #$00            ; Mantissa = 0
0262  030F  A203            LDX #3              ; 3 byte
0263  0311  9501    FADDL9  STA MAN1-1,X
0264  0313  CA              DEX
0265  0314  D0FB            BNE FADDL9
0266  0316  F000            BEQ FADDRE          ; Restore stack and exit
0267  0318          ;
0268  0318          ; Shift the result to get the correct format,
0269  0318          ; put the result on the stack, the return address
0270  0318          ; and return
0271  0318          ;
0272  0318  4606    FADDRE  LSR SIGN1           ; Get sign bit
0273  031A  A205            LDX #<MAN1+3        ; Shift in sign-bit
0274  031C  A004            LDY #4              ; Rotate 4 byte
0275  031E  200702          JSR ROTRIG
0276  0321          ;
0277  0321          ; Save result on the stack (4 byte)
0278  0321          ;
0279  0321  A004            LDY #4              ; 4 byte
0280  0323  B90100  FADDLB  LDA MAN1-1,Y        ; Get byte
0281  0326  48              PHA
0282  0327  88              DEY
```

```
6502 Floating Point Package     PROTON 650X ASSEMBLER  V4.4    PAGE: 0006

0283  0328  D0F9              BNE FADDLB
0284  032A            ;
0285  032A  A501              LDA RETADD+1
0286  032C  48                PHA
0287  032D  A500              LDA RETADD
0288  032F  48                PHA
0289  0330  60                RTS
0290  0331            ;
0291  0331            ; Floating point subtraction:
0292  0331            ; based on: Hagen Volzke Fliesskomma - Arithmetik und
0293  0331            ;           IEEE-Spezifikation
0294  0331            ;
0295  0331            ;           mc 11/88 page 78
0296  0331            ;
0297  0331            FSUB                    ; Get 1 parameter from stack
0298  0331  68                PLA             ; Get return address and save it
0299  0332  8500              STA RETADD
0300  0334  68                PLA
0301  0335  8501              STA RETADD+1
0302  0337            ;
0303  0337  A900              LDA #0          ; Clear signs
0304  0339  850B              STA SIGN2
0305  033B  8506              STA SIGN1
0306  033D            ;
0307  033D            ; Get second parameter from stack (4 byte)
0308  033D            ;
0309  033D  A207              LDX #<MAN2      ; Zero-page relative address
0310  033F  A004              LDY #4          ; 4 byte
0311  0341  68      FSUBP1    PLA             ; Get byte from stack
0312  0342  9500              STA $0000,X     ; Store byte
0313  0344  E8                INX
0314  0345  88                DEY
0315  0346  D0F9              BNE FSUBP1
0316  0348  A50A              LDA EXP2        ; Get sign bit
0317  034A  4980              EOR #$80        ; Invert it
0318  034C  850A              STA EXP2
0319  034E            ;
0320  034E            ; Subtraction is addition with an inverted subtractor
0321  034E            ;
0322  034E  4C2502            JMP SUBENT      ; Continue with FADD
0323  0351                    .END

 ERRORS: 0000                           <0000>
```

Referenties.

1: Hagen Völzke: Fliesskomma-Aritmetik und IEEE-Specifkationen
mc 10/88 blz. 123
2: Hagen Völzke: Fliesskomma-Aritmetik und IEEE-Specifkationen
mc 11/88 blz. 78
3: Hagen Völzke: Fliesskomma-Aritmetik und IEEE-Specifkationen
mc 12/88 blz. 91
4: Hagen Völzke: Fliesskomma-Aritmetik und IEEE-Specifkationen
mc 1/89 blz. 66
5: Hagen Völzke: Fliesskomma-Aritmetik und IEEE-Specifkationen
mc 2/89 blz. 65

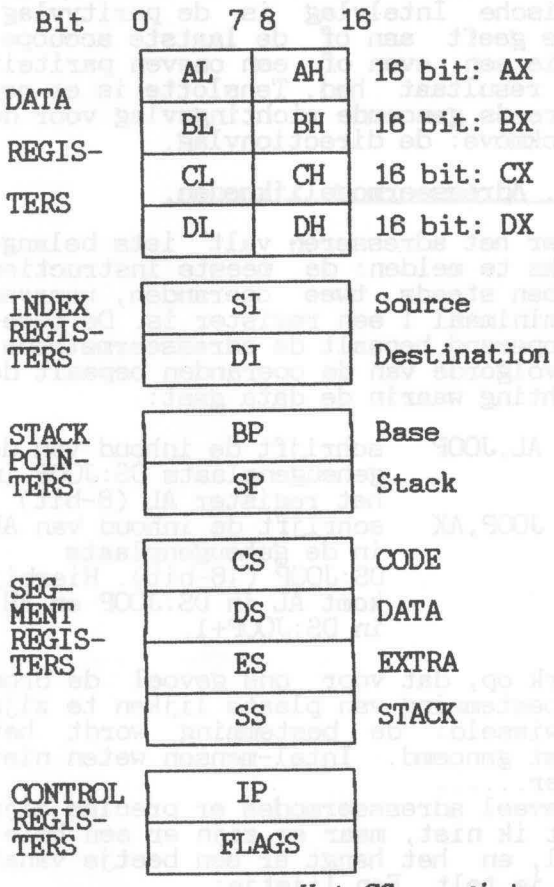
De IBM-PC en z'n klonen (Deel 2).

Door: Nico de Vries.

In deel 1 hebben we kennis gemaakt met de PC-, klonen- en compatibleswereld. Omdat dit een technisch georienteerd verhaal moet worden, zullen we ons in dit deel onledig houden met het hart van ieder computersysteem: de processor. In alle PC's zit een processor die ontwikkeld is door Intel, de nestor op dit gebied. De PC en de PC/XT zijn in de meeste gevallen uitgerust met een 8088 CPU. Deze processor heeft een (gemultiplexte) 8-bit databus, 20 adreslijnen en is gehuisvest in een 40-pins behuizing. Met 20 adreslijnen kun je 1 Mbyte adresseren.
Sommige klonen zijn uitgerust met een variant van de 8088: de 8086. Deze processor is geheel gelijk aan de 8088 maar heeft een gemultiplexte 16-bit databus. De PC/AT en zijn klonen hebben zonder uitzondering een 80286 CPU die eveneens 16 bit is.

2.1. Het registermodel.

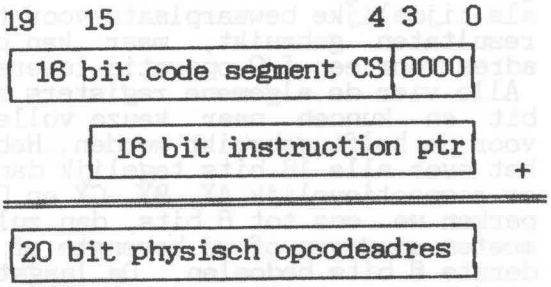
Wil je een idee krijgen hoe een processor is opgebouwd, dan kijk je naar het register- of programmeermodel. Voor de 8088 is dat hieronder gegeven.



HELP! zult u nu wel denken. Zijn we bij de 6502 gewend aan 6 registers (A, X, Y, PC, SP en vlaggen), hier zijn dat er wel wat meer! En wat zijn eigenlijk: segment registers?

2.2. Segmentering.

Het meest opvallende aan het registermodel zijn de segment registers. Dit komt, omdat de 8088 een zogenaamd gesegmenteerde processor is. In het verhaal over RISC en CISC processors van Gert van Opbroek was te lezen, dat het door elkaar gooien van programma code, data en stacks eigenlijk maar een slordige bezigheid is. Welnu: de 8088 is zodanig gemaakt, dat deze drie zaken ieder in een eigen geheugengebied kunnen wonen, en via aparte registers kunnen worden aangesproken. De methode waarop dat gebeurt wordt aangeduid met segmentering. Zo zijn er dus minimaal een programmacodesegment, een data segment en een stacksegment. De registers CS (codesegment), DS (datasegment) en SS (stacksegment) wijzen ieder binnen het 1 Mbyte adresbereik een gebied van 64 kbyte aan voor code, data en stack.
Het registermodel laat zien, dat alle registers 16 bit zijn. Toch is de adresbus 20 bit breed, dus: hoe komen we aan de overige vier bits? Dit gaat als volgt. Als voorbeeld nemen we de programmateller, die bij de 8088 de Instruction Pointer ofwel de IP heet. De instruction pointer is 16 bit en kan dus 64k adresseren. We weten inmiddels, dat de opcodes uit het codesegment komen, dat aangewezen wordt door het codesegment CS. Het 20-bit adres van de huidige opcode wordt nu als volgt verkregen:



Het CS register wordt 4 bit naar links geschoven, en vervolgens wordt hierbij de instruction pointer opgeteld. De laagste vier bits (een bereik van 16 bytes) komen dus altijd uit de instruction pointer. Het is niet nodig dat bit 11 tot 0 van het code segment nul zijn: er vindt altijd een volledige optelling plaats. Dit heeft onder andere tot gevolg, dat de vier segmenten elkaar kunnen overlappen of zelfs kunnen samenvallen.
Ofschoon er een 20-bit adres ontstaat, wordt er bij de 8088 altijd met 16-bit notaties gewerkt: zo komt de volgende

instructie vanaf het adres CS:IP, ofwel uit het code segment, met offset IP. Met andere woorden: bij de 8088 heeft een volledig adres twee delen: een segment of aanwijzer en een offset. Als CS 8765 bevat en IP bevat 1234, dan wordt dit genoteerd als: 8765:1234. Het werkelijke adres is dan:

```
 87650
  1234 +
--------
 88884
```

Voor de stack geldt hetzelfde verhaal, alleen zit de stack altijd in het stacksegment SS. De huidige stackpointer is SP. De stackpointer is net als bij de 6502 downgoing: als je iets op de stack zet wordt de stackpointer lager.

2.3. De algemene registers.

De dataregisters hebben meerdere functies en worden ook wel general purpose (algemene doeleinden) registers genoemd. Register A is de eigenlijke accumulator (vandaar de A) Het register B wordt ook wel het base register genoemd, terwijl register C te boek staat als count register. Register D heeft geen specifieke functie en zou dus het diversen register genoemd kunnen worden.

Deze namen zijn niet geheel toevallig gekozen. Het register B(ase) kan gebruikt worden voor indirect adresseren en levert dan de basis offset. Register C (count) doet bij loops dienst als teller die automatisch wordt afgelaagd. Bij schuifoperaties geeft het C register het aantal bits aan dat geschoven moet worden.
Register A is inderdaad de accumulator: optellen, aftrekken, delen en vermenigvuldigen is alleen mogelijk met dit register. Register D wordt voornamelijk als tijdelijke bewaarplaats voor tussenresultaten gebruikt, maar kan ook het adres voor een I/O operatie leveren.

Alle vier de algemene registers zijn 16 bit en kunnen naar keuze volledig of voor de helft gebruikt worden. Hebben we het over alle 16 bits tegelijk dan heten ze respectievelijk AX, BX, CX en DX. Beperken we ons tot 8 bits, dan zullen we moeten aangeven of we bovenste of de onderste 8 bits bedoelen. De laagste acht worden aangesproken met AL, BL, CL en DL, de hogere helften met AH, BH, CH en DH.

Bij het ophalen of wegzetten van data uit het geheugen wordt naar keuze het datasegment DS of het extra segment ES gebruikt.

2.4. De indexregisters.

Er zijn twee indexregisters aanwezig, SI en DI. Zoals de namen al aangeven worden ze onder andere gebruikt bij blockmoves, die direct als instructie beschikbaar zijn. Hierbij geeft SI de bron-offset, en DI de destination-offset. De blockmove kan alleen binnen een segment plaats vinden. Voor het segment wordt meestal ES gebruikt. Verder is het mogelijk een blockmove te doen in words of in bytes, en kan met een vlag worden aangegeven in welke richting er verplaatst moet worden. Bij een blockmove doet CX dienst als teller voor de hoeveelheid te verplaatsen data. De tweede functie van de registers SI en DI is het indexeren van adressen, net als de registers X en Y in de 6502.

2.5. De overige registers.

Het register BP kan dezelfde functies vervullen als de stackpointer. Dit wordt voornamelijk gebruikt om datastacks mee te maken.

Het vlaggen register heeft de gebruikelijke set vlaggen: Carry, Sign, Zero, Interrupt enable, Break en Overflow zullen bekend in de oren klinken. Een typische Intelvlag is de parityvlag: deze geeft aan of de laatste accuoperatie een even of een oneven pariteit als resultaat had. Tenslotte is er nog de reeds genoemde richtingsvlag voor de blockmove: de directionvlag.

2.6. Adresseermogelijkheden.

Over het adresseren valt iets belangrijks te melden: de meeste instructies hebben steeds twee operanden, waarvan er minimaal 1 een register is. De tweede operand bepaalt de adresseermethode. De volgorde van de operanden bepaalt de richting waarin de data gaat:

MOV AL,JOOP	schrijft de inhoud van de geheugenplaats DS:JOOP in het register AL (8-bit)
MOV JOOP,AX	schrijft de inhoud van AL in de geheugenplaats DS:JOOP (16-bit). Hierbij komt AL in DS:JOOP en AH in DS:JOOP+1.

Merk op, dat voor ons gevoel de bron en bestemming van plaats lijken te zijn verwisseld: de bestemming wordt het eerst genoemd. Intel-mensen weten niet beter.....

Hoeveel adresseermodes er precies zijn weet ik niet, maar er zijn er een heleboel, en het hangt er een beetje vanaf hoe je telt. Een lijstje:

8088	6502
Immediate	Immediate
Register	'Implied'
Memory	Absolute
Base	Indirect
Indexed	Indirect Ind.
Base Indexed	Indirect Ind.
Short	Zero page
Near	Absolute
Far	-

Dit lijkt kort, maar bijna alles mag in bytes en in words (8- respectievelijk

16-bit) terwijl indexering met BX, SI, DI of een constante mag of met BX+SI, BX+DI, BP+BX, BP+SI, BP+DI, of met BP+BX+SI+constante, enzovoort.
Base is met 1 register, al dan niet met een constante als index. Indexed is base met een tweede register als index, terwijl index based dus twee indexen heeft: een register en een constante.
Short, Near en Far addressing komen voor bij JMP instructies (en de instructies die we bij de 6502 branches zouden noemen). De onvoorwaardelijke JMP kan geschieden met 1 byte offset (Short), met twee bytes (Near) of naar een ander segment (Far). Dit laatste kost niet drie maar vier bytes, voor segment en offset gescheiden. De conditionele sprongen kunnen alleen Short worden gedaan. De CALL tenslotte kan Near of Far zijn.

2.7. Reset en interrupts.

De 8088 heeft in principe geen vectoren in ROM. Het startadres na een reset is altijd FFFF:0000. Op deze plek (geheel bovenin de 1Mbyte adresruimte) zit meestal (EP)ROM en er staat meestal een JMP FAR naar een absoluut adres, waar als eerste de segment registers worden geinitialiseerd.
De interrupts zijn doorlopend genummerd, beginnend met 0. Behalve uit de hardware kun je een interrupt ook softwarematig opwekken met een INT-instructie. De interrupts zijn als volgt genummerd:

0: Divide error
1: Break (vergelijk 6502 BRK)
2: NMI
3: BRK3 instructie
4: BRKV instructie
5: gereserveerd
6: gereserveerd, enzovoort

Naarmate het nummer van de interrupt hoger wordt, wordt zijn prioriteit lager. Merk op dat de NMI niet de hoogste prioriteit heeft.
De CPU is naast een NMI aansluiting ook van een INT (6502 IRQ) pin voorzien. Zoals we al gezien hebben is er een aparte vector voor de INT-interrupt. De bedoeling is namelijk, dat het interrumperende device bij de interrupt-acknowledge een byte op de databus zet, dat het interrupt-nummer aangeeft. De 8259 interruptcontroller is zo'n IC. In de PC en de PC/XT zit 1 zo'n chip, die zodanig wordt ingesteld, dat de 8 interrupts die het kan verwerken verwijzen naar de vectoren 8 t/m F.
Naast de normale manier van interruptgeneratie kan het in de 8088 ook direct via software. Hiervoor is de INT instructie, die als operand het vectornummer heeft. In 1 van de volgende delen zullen zien dat van deze mogelijkheid uitgebreid gebruik gemaakt wordt in het BIOS en in MS-DOS. Ook is deze constructie handig voor het testen van een interruptbron in software.

2.8. De V20 en de V30.

De 8088 (8-bit) en de 8086 (16-bit) werden ontwikkeld door Intel. Deze beide processoren zijn op de databusbreedte na, exact aan elkaar gelijk. Beide typen werken volgens het microcodeprincipe. Bij microcode is iedere opcode die de CPU moet uitvoeren opgeslagen in de CPU als een klein programmaatje, dat wordt uitgevoerd door een in de CPU aangebrachte mini-CPU. Het voordeel van deze methode is de korte ontwikkeltijd voor de fabrikant en het feit dat de instructieset niet 'logisch' in elkaar hoeft te zitten (vergelijk dit met de 6502: er is duidelijk sprake van een matrix in de instructieset). Het belangrijkste nadeel wordt gevormd door het feit dat deze methode relatief veel clockcycli en dus veel tijd vergt.
De Japanse fabrikant NEC heeft de moeite genomen om de 8088 om te werken van microcode naar directe decodering. Tevens werden een aantal extra's toegevoegd, waaronder een tweede interne databus, waardoor bijvoorbeeld een registerswap in 1 clockcyclus kan worden gedaan tegen in de 8088 drie. Een verder extra is een instructieprefetch mechanisme, dat vast de volgende opcode ophaalt uit het geheugen. Al deze zaken zorgden ervoor, dat de nieuwe CPU sneller is dan de 8088 bij dezelfde clocksnelheid. NEC noemde de omgewerkte 8088 uPD70108, of V20, en de omgewerkte 8086 heet uPD70116 of V30. Beide V-processors kunnen zonder meer in een PC worden toegepast: ze zijn pin-to-pin compatibel. In een PC(/XT) verkrijgt men op die manier circa 10% snelheidswinst. Nog een hersenkrakertje: de V-CPU's kunnen sneller delen dan vermenigvuldigen, terwijl dit bij alle andere processors (ook de 8088/86) precies andersom is!
En passant werden nog een paar extraatjes aan de V20/V30 toegevoegd: een aantal instructies van de 80188/186 en een emulatiemode. De V20 kan de Intel 8080 CPU emuleren, de V30 gaat nog een stapje verder en emuleert zelfs de Z80.
Beide processoren zijn in CMOStechniek gemaakt en gebruiken dus zeer weinig stroom.

2.9. In deel 3....

Gaan we het hardware-blokschema van de PC(/XT) bekijken. Tot de volgende keer.

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