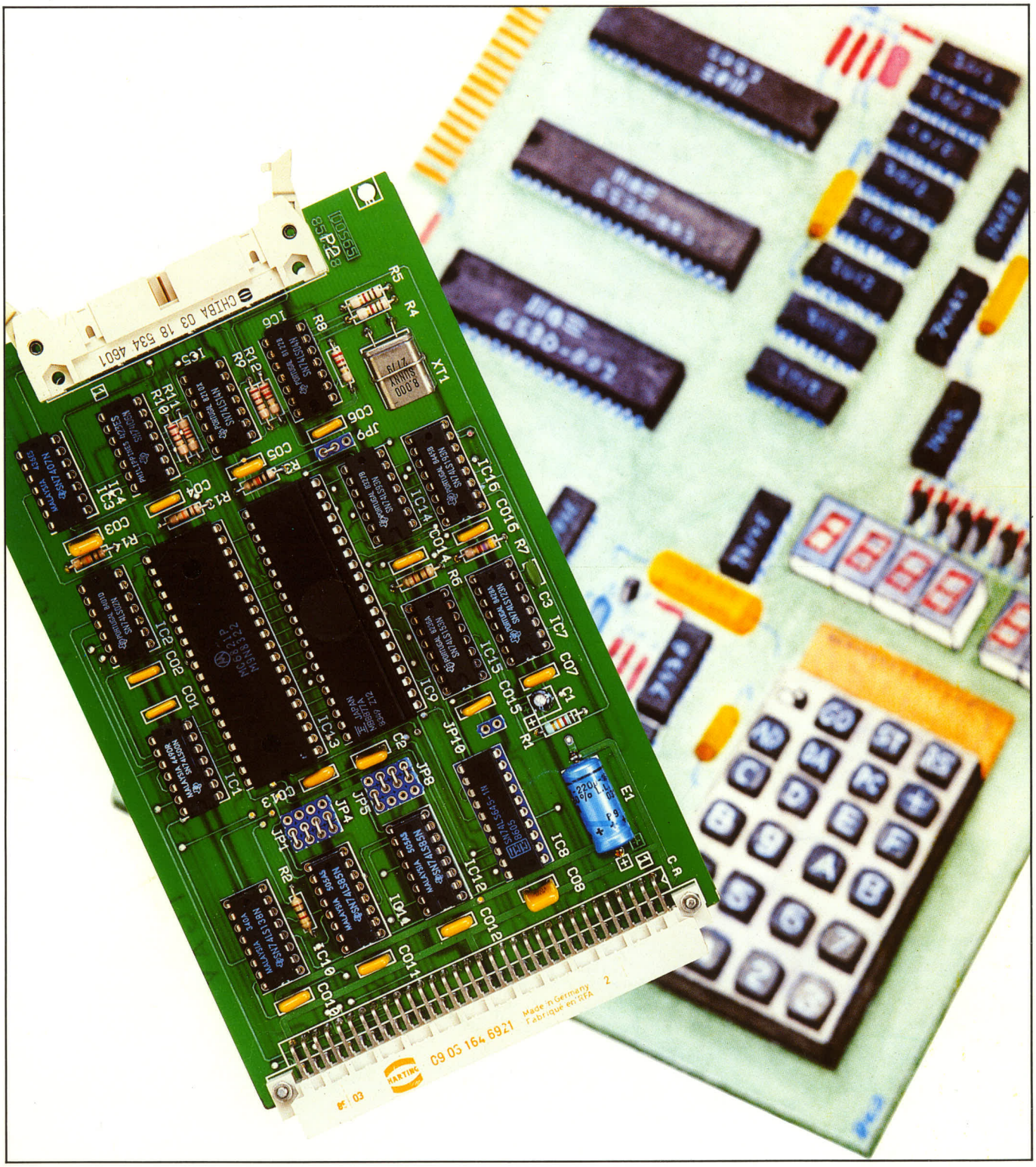
DE 6502 KENNER 52

Elfde jaargang, nr 5, october 1987



DE 6502 KENNER

Het bestuur van de 6502 Kenners zoekt voor haar nog op te richten bulletinboard een

SYSOP

De sysop beheert de soft- en hardware van het bulletin-board. De hardware bestaat uit een IBM(achtige) PC met 20 Mb Winchester. Van de sysop wordt verwacht dat hij zorgt dat het bulletin-board op de aangewezen tijden bereikbaar is, dat hij vragen van leden beantwoord en zorg draagt voor een redelijke mate van discipline van de gebruikers. Vooral in het begin zal dit een behoorlijke tijdsinspanning vragen. Derhalve dient de sysop over een grote dosis enthousiasme te beschikken. Geinteresseerden wordt verzocht kontakt op te nemen met de secretaris.

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Het bestuur van de 6502 Kenners zoekt op korte termijn een

REDACTEUR

De redacteur is verantwoordelijk voor het redigeren en opmaken van het verenigingsblad 'de 6502 Kenner'.Hij heeft namens het bestuur daarin een grote mate van vrijheid. In zijn werkzaamheden wordt de redacteurbijgestaan door een redactie van meerdere personen. De redacteur neemt deel aan de vergaderingen van het bestuur.
Geinteresseerden worden verzocht kontakt op te nemen met de secretaris.

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Het bestuur van de 6502 Kenners zoekt op korte termijn een

PENNINGMEESTER

De penningmeester is als lid van het dagelijks bestuur belast met het beheer van de financien van de vereniging. Hij draagt zorg voor een correcte financiele administratie. Daarnaast dient de penningmeester er zorg voor te dragen dat er tijdig een begroting en een financieel jaarverslag wordt gemaakt. Geinteresseerden wordt verzocht kontakt op te nemen met de secretaris.

COLOFON

De 6502 KENNER is een uitgave van de KIM gebruikers Club Nederland.

Redaktie adres: R.Vleesch Dubois
Florence Nightingalestraat 212
2037 NG Haarlem.

Medewerkers:
Adri Hankel
Gert Klein
Gert van Opbroek
Nico de Vries
Erwin Visschedijk
John van Sprang
Rinus Vleesch Dubois.

De 6502 KENNER verschijnt 6 x per jaar en heeft een oplage van 500 exemplaren.

I.v.m. auteurswetgeving aanvaardt de redaktie geen aansprakelijkheid voor inzendingen. Tenzij anders aangegeven, dient de inzending afkomstig te zijn van de inzender.

De voorpagina is een aquarel van de Kim, een DOS65 controller.

Fotografie: Frank Visschedijk.

Aan alle leden van de "KIM-Gebruikers club Nederland"

Op een bestuursvergadering in september 1987 heeft het bestuur besloten de samenwerking met Willem van Pelt op te zeggen. Naar we hebben vernomen van het bestuur is dit een gevolg van een konflikt wat al een jaar speelt.

Omdat ondergetekenden van mening zijn dat Willem van Pelt tot nu toe de spil is geweest van onze computer club, roept deze beslissing bij ons de volgende vragen op:

-Waarom heeft het bestuur in dit konflikt, hetgeen zijzelf als belangrijk heeft opgevat, geen enkele mededeling aan de leden gedaan door middel van een artikel in de 6502 kenner, maar heeft zij hiermee gewacht tot het besluit omtrent het beeindigen van de samenwerking met Willem van Pelt een feit was?
-Waarom heeft het bestuur geen enkele poging ondernomen het konflikt op een van de bijeenkomsten met de leden te bespreken?
-Waarom heeft het bestuur de overige redactieleden niet benaderd over het konflikt en gepoogd in overleg met hen tot een oplossing te komen?
-Waarom heeft het bestuur niet besloten deze belangrijke beslissing voor te leggen aan de ledenvergadering in november 1987? (Dit zou als bijkomend voor deel hebben gehad dat de distributie van software niet in het gedrang zou zijn gekomen zoals nu het geval is.)
-Op welke gronden is de samenwerking met Willem van Pelt opgezegd?
  -Is dit een gevolg van willekeur van het bestuur jegens de persoon Willem van Pelt?
  -Heeft dit te maken met beleidskwesties met betrekking tot de toekomst strategie van de KIM-Gebruikers club Nederland en verschillende inzichten hier over, waardoor samenwerking tussen de betrokken personen onmogelijk bleek?
  -Is dit gebeurd op grond van schriftelijke klachten van leden van de club?
-Is de indruk juist dat het bestuur de leden van de vereniging moedwillig informatie onthoudt, en op geen enkele manier inspraak van de leden duldt anders dan op een ledenvergadering?

Het bestuur heeft op de bijeenkomst op 19 september in Haarlem toegezegd dat deze kwestie zal worden besproken in de ledenvergadering op 14 november. Gezien de merkwaardige en, voor de leden, duistere gang van zaken eisen wij van het bestuur dat:

-Het bestuur terugkomt op haar besluit de samenwerking met Willem van Pelt op te zeggen.
-Het bestuur uiteen zet hoe zij de communicatie tussen bestuur en de leden denkt te verbeteren.
-Het bestuur inhoudelijke mededelingen doet over het konflikt met Willem van Pelt teneinde hierover een open discussie te hebben.

Omdat het van het grootste belang is dat deze zaak wordt besproken met zoveel mogelijk leden, roepen wij een ieder met klem op de ledenvergadering van 14 november bij te wonen teneinde tot een weloverwogen besluit te komen. Indien het bestuur niet zelf terugkomt op haar besluit de samenwerking met Willem van Pelt op te zeggen, zal een motie waarin het bestuur wordt gevraagd dit alsnog te doen worden ingediend. Indien het niet mogelijk is de vergadering zelf bij te wonen verzoeken wij U een volmacht voor deze stemming, vergezeld van een stemadvies, te zenden aan:

To all members of the "KIM-gebruikers club Nederland"

At a board-meeting in september 1987, the board decided to dismiss Willem van Pelt of his duties as chief editor of "De 6502 Kenner". According to the board this is the result of a conflict that has lasted for a year already.

The undersigners share the opinion that Willem van Pelt is the spirit of our computerclub. Therefore the following questions arise regarding this decision:

-The board stated to consider the conflict as important. Why didn't they inform the members by means of an article in until the decision was made?
-Why has there never been any attempt to discuss this conflict with members who were attending the meetings?
-Why has there never been any attempt to discuss this conflict with the other members of the editorial staff in order to solve the problems?
-Why hasn't the board decided to consult the members about this important decision at the general meeting in november 1987 before make their decision? (In that case there wouldn't be any problems with the distribution of software as there are now.)
-On which grounds the decision to dismiss Willem van Pelt of his duties has been made?
  -Is this the result of feelings against the person of Willem van Pelt?
  -Has this conflict anything to do with the developing policy concerning the future of the KIM-Gebruikers club Nederland and different points of view of the people involved?
  -Has Willem van Pelt perhaps been dismissed as a result of written complaints by members?
-We have the impression that the board is deliberately witholding information from the members of the club and is obstructing members to express their opinion in these matters outside the general meeting. Is this a correct impression?

At the meeting in Haarlem on 19 september 1987 the members of the board promised to discuss this matter with the members at the coming general meeting on 14 november 1987. In view of the strange and, to members, obscure cource of events we demand the following:

-the board to recall the decision to dismiss Willem van Pelt of his duties as chief editor of "De 6502 Kenner".
-the board to come forward with their intentions and ideas how to improve communication between the members and the board.
-the board to present full explanation about the nature of the conflict between Willem van Pelt and the board, in order to have an open discussion.

As we think it is of enormous importance to discuss this matter with as much members as possible, we urge everybody to attent the general meeting on 14 november 1987 so that an extensively deliberated decision can be taken.
If the board does not recall their decision to dismiss Willem van Pelt of his duties, on this general meeting an formal proposal will be introduced to recall it after all.
If you are not able to visit the general meeting personally, please send your authorization to vote, along with your preference, to:

Coen Boltjes
Noordeinde 5
3121 KG Schiedam
The Netherlands

Note:
Deze brief van Coen Boltjes is op zijn verzoek, ongecensureerd en met instemming van het bestuur, gepubliceerd.

Het bestuur wil echter benadrukken dat het zich volledig distantieert van geschreven inhoud, welke geheel ter verantwoording is van Coen Boltjes.

Namens het bestuur, R. Vleesch Dubois.

Fernando Lopez	Portugal
Andrew Gregory	England
Marc Lachaert	Belgium
Coen Boltjes	The Netherlands

Queen Boltes
Moordenaars
3127 KD Oud-Beijerland
The Netherlands

Inhoudsopgave

Van het bestuur.

Tijdens de laatste bestuursvergadering heeft het voltallige bestuur besloten om de Heer W. van Pelt uit zijn funktie van redacteur te ontheffen. Dit besluit is genomen om reden van bestuurlijke aard: de heer van Pelt meende meer en meer een eigen koers te moeten varen. Hij stelde menigmaal tijdens bestuursvergaderingen dat; " indien het bestuur vindt dat ik het niet goed doe, moeten zij het maar zeggen dan stap ik zo op". Uiteraard valt er met iemand die zich zo op stelt niet te werken. Diskussie was vooral het groeiend buitenlands taalgebruik. Zelfs vertalers werden door W. van Pelt aangeschreven om Nederlandse kopy te vertalen. Tevens meende de heer van Pelt de prijzen van enige club-produkten zelfstandig te moeten bepalen. Helaas heeft het bestuur deze moeilijke maar noodzakelijke beslissing moeten nemen. Alle dingen bestaan minstens uit twee, zo ook hier. De heer W. van Pelt heeft zeer veel werk verricht, dit heeft zeker bijgedragen tot een aantal positieve ontwikkelingen binnen onze club. Hiervoor wil het bestuur de heer W.L. van Pelt zijn erkentelijkheid betuigen. Het bestuur wil nogmaals duidelijk naar voren brengen, dat de redacteur door het bestuur wordt benoemd, en de Kim Kenner onder de verantwoordelijkheid valt van het dagelijks bestuur.

Namens het bestuur.
R. Vleesch Dubois. (voorzitter)

REDAKTIONEEL

PLOF ! ! !

Zomaar een nieuw jasje voor de 6502 Kenner. Ja, ook de indeling van dit blad is aangepast. Was de oude 6502 Kenner dan niet goed? Zeker, maar ook redaktioneel moet men innoveren. De hedendaagse "Desktop-publishing" apparatuur maakt het mogelijk om betere kwaliteit te leveren en sneller te werken. Technitron Data B.V. te Aalsmeer stelde ons een TLP-12 laser printer ter beschikking om de kopy voor dit blad te produceren. De indeling is overzichtelijk gemaakt, zodat een ieder de voor hem interessante informatie snel kan lokaliseren. Er zullen vaste rubrieken worden gepresenteerd, waarmee we zoveel als mogelijk de voor ieder interessante informatie willen publiceren. Graag zouden we dan ook zien dat op deze vaste rubrieken wordt gereageerd door het insturen van kopy, die (onder voorbehoud) in een van deze rubrieken geplaatst zullen worden. Tekst of software kunt u insturen op een 5 1/4 inch floppy disk welke gelezen kunnen worden op de volgende disk operating systemen:

Dos - 65
Ms - Dos
Ohio - Octopus
CP/M - (Apple)
Junior tape

Natuurlijk zijn ook andere informatiedragers mogelijk, zoals getypte of afgedrukte kopy. Deze dienen dan wel overgetypt te worden door de redaktie medewerkers. Bespaar ons werk en stuur uw kopy op floppy. Voor iedere inzending op floppy kan de inzender rekenen op een nieuwe floppy en misschien zetten wij er dan iets leuks voor u op.

De redaktie.

UITNODIGING

Uitnodiging voor de landelijke bijeenkomst op ZATERDAG 14 november 1987.

Lokatie:
Wijkcentrum 't Veurbrook
Jan Tooropstraat 27
7606 JS ALMELO.
Tel.: 05490 - 10353

Routebeschrijving:
Voor degenen die al eerder op een bijeenkomst in Almelo waren, is het eenvoudig: u rijdt naar de u bekende lokatie aan de Jan Steenstraat. Daar aangekomen gaat u steeds rechtdoor, tot u niet verder kunt. Hier gaat u links af. Dit is de Jan Tooropstraat. Met de bocht mee naar rechts. Na plm. 120 meter links:
't Veurbrook.

Vanuit het westen en zuiden (A1 / A35):
1. Aan het einde van de snelweg rechtsaf. Bij het eerstvolgende kruispunt MET VERKEERSLICHTEN linksaf, richting Wierden/Zwolle. Bij de eerstvolgende verkeerslichten rechtdoor. Bij de volgende verkeerslichten (links BP tankstation en Opel garage Kamp) gaat u rechtsaf.
2. U rijdt nu op de Windmolenbroeksweg. Doorrijden tot over de brug, dan de eerste straat rechts. Dit is de W. van Konijnenbrugstraat. Na plm. 50 meter rechtsaf. Dit is de Tooropstraat. Met de bocht mee naar links. Na plm. 50 meter aan de rechterkant: 't Veurbrook.

Vanuit het noorden (via de N 36).
1. Bij de stoplichten rechtsaf, richting streekziekenhuis. U bevindt zich nu op de rondweg om Almelo. Deze weg blijven volgen tot u het BP tankstation ziet. Bij dit kruispunt linksaf. Zie verder punt 2.

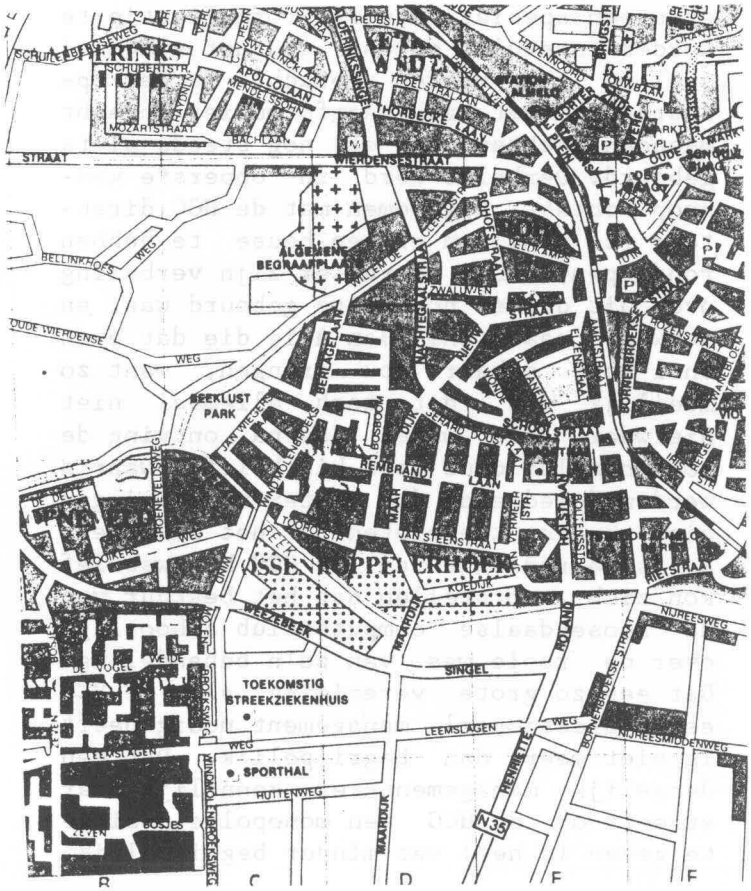
Met het openbaar vervoer.
Vanaf NS station Almelo met de stadsbus naar de wijk Molênbroek. Uitstappen bij de halte Windmolenbroeksweg. Schuin tegenover de bushalte staat een wegwijzer. Daarop staat ook 't Veurbrook vermeld.

PROGRAMMA

Aanvang 10.00 uur.
- opening ledenvergadering
- notulen laatste ledenvergadering
- concept begroting 1988
- verkiezing kaskontrole kommissie
- redaktie 6502 kenner
- komende en gaande bestuursleden
  aftredend en niet herkiesbaar:
  John van Sprang - penningmeester
  aftredend en herkiesbaar:
  Adri Hankel
  Erwin Visschedijk
- kandidaten voor genoemde funkties kunnen zich melden tijdens de vergadering of eerder d.m.v. briefje aan de voorzitter.
- rondvraag
- lunch
- informeel gedeelte.

Toegang gratis.

Konsumpties en/of broodjes tegen betaling.



DE 6502 KENNERS GAAN DE LUCHT IN

De 6502 kenners gaan de lucht in, niet op de rumoerige manier die van Speijk daarvoor koos, maar op een vreedzame manier met een bulletin-board. Het bestuur heeft besloten een eigen bulletin-board voor onze vereniging op te richten. Op het moment dat u dit leest heeft de penningmeester al het groene licht gegeven voor de aanschaf van de noodzakelijke hardware. Het belangrijkste deel van die hardware bestaat uit een IBM(achtige) PC met een 20 Mb Winchester en een auto-answer modem. Welke software we gaan gebruiken is op dit moment nog niet bekend. Het zou het meest voor de hand liggen om een FIDO bulletin board op te zetten. Nog mooier zou het zijn als ons bulletin-board een onderdeel zou kunnen gaan vormen van het landelijke FIDO netwerk. Helaas lijkt daar getuige een ingezonden brief van onze zustervereniging uit Roosendaal in de laatste HCC nieuwsbrief weinig kans op. De Roosendaalse club had van diverse hotemetoten uit het HCC bestuur al mondelinge toezeggingen gekregen dat hun FIDO op het landelijke net kon worden aangesloten. Er hoefde alleen nog maar even een speciaal aanvraagformuliertje te worden ingevuld, en dan zou het allemaal snel in orde worden gebracht. Na het opsturen van het aanvraagformulier en acht maanden wachten was er nog steeds niets gebeurd. Derhalve werd in opperste wanhoop kontakt opgenomen met de HCC direkteur (ze schijnen er een heuse te hebben rondlopen). Die sprak er zijn verbazing over uit dat er nog niets gebeurd was, en verwees naar ene Jan-Arie die dat even snel voor elkaar zou brengen, want zo moeilijk was dat toch allemaal niet nietwaar. Twee weken daarna ontving de Roosendaalse club een kil briefje waarin werd medegedeeld dat zuster-verenigingen niet worden toegelaten tot het FIDO net. Zelfs de redactie van de HCC Nieuwsbrief kon zich voorstellen dat het bestuur van de Roosendaalse computerclub behoorlijk over de rooie was van zo'n behandeling. Dat een zo grote vereniging als de HCC een professioneel management nodig heeft is niet meer dan begrijpelijk. Dat een dergelijk management er kennelijk naar streeft om de HCC een monopolie positie te geven is heel wat minder begrijpelijk. Ik kan me ook werkelijk niet voorstellen dat de HCC leden er behoefte aan hebben om medehobbyisten op zo'n manier te behandelen. Afijn genoeg gekankerd, misschien leren ze bij de HCC nog wel eens dat hobby-verenigingen meer gebaat zijn bij samenwerking dan bij tegenwerking. Intussen zullen wij knus onder elkaar zijn op ons eigen bulletin-board. Wat heeft een dergelijk board u als hobbyist nu te bieden? In principe levert een bulletin-board twee soorten diensten. Allereerst is een bulletin-board een verzamelpunt voor lieden die elkaar iets willen vragen, die iets te vertellen hebben, of die willen kijken wat anderen te melden hebben. Iedereen die kontakt zoekt met een bulletin-board (inlogd) kan er berichten achterlaten of berichten die door andere mensen zijn achtergelaten lezen. Zulke berichten kunnen op allerlei onderwerpen betrekking hebben. Wie met een schijnbaar onoplosbaar probleem zit kan de hulp van medeleden inroepen, of je kunt als je denkt dat je iemand kunt helpen een bericht achterlaten waarin tips staan voor de oplossing van het probleem. Ook is het mogelijk om een al dan niet verhitte discussie aan te gaan over interessante (computer)onderwerpen. Eveneens is het mogelijk om verjaardagsgroeten of schuine moppen achter te laten, maar de praktijk wijst uit dat de lol daar al gauw van af gaat. Op de meeste goede bulletin-boards tref je dat soort zaken tegenwoordig niet meer aan. De tweede dienst die een bulletin-board biedt is die van een file-server. Dat wil zeggen dat het board als plaats dient waar files opgeslagen zijn. Op commando van u als ingelogde gebruiker stuurt het bulletin-board de file naar uw eigen computer toe, waar een bepaald programma er voor zorgt dat het op uw eigen floppy-disk wordt opgeslagen. Dit proces wordt down-loaden genoemd. Ook het omgekeerde (up-loaden) is mogelijk. U kunt een file van uw eigen computer overzenden naar het bulletin-board. Het board zorgt ervoor dat de file zodanig wordt opgeslagen dat anderen hem weer kunnen down-loaden. Uiteraard is dit allemaal reuze handig. Stelt u zich voor dat de listings van de programma's uit de laatste 6502 Kenner altijd op het board aanwezig zijn. Dat bespaart u werkelijk uren type-werk! Al het data-transport tussen u en het

bulletin-board verloopt via het openbare telefoonnet en dat betekent dus dat u over een modem moet beschikken. Gelukkig is er op modem-gebied op het ogenblik een ware prijzenoorlog aan de gang. De 'klonen' fabrikanten in Taiwan hebben een markt voor modems in Europa ontdekt en ervoor gezorgd dat de prijzen in een jaar tijds meer dan gehalveerd zijn. Betaalde je een jaar geleden nog tussen de f 1500,- en f 2000,- voor een 1200 baud full-duplex modem, op dit moment zijn 'Hayes compatible' modems al voor prijzen van rond de f 500,- te koop. Het ligt het meest voor de hand om een modem de kopen dat zowel de V21 norm (300 baud full-duplex) als de V22 norm (1200 baud full-duplex) aan kan. Wilt u ook gebruik maken van data-banken die volgens het PTT 'Viditel' principe werken, dan dient uw modem ook nog de V23 norm aan te kunnen (1200/75 baud). U moet echter wel bedenken dat de door ons gewoonlijk gebruikte 6551 ACIA zonder hardwarematige kunstgrepen niet in staat is met verschillende snelheden te zenden en te ontvangen. Goed, nadat u het nodige ijzerwerk heeft ingékocht en aangesloten op de RS232 interface van uw computer, moet er nog de nodige software komen om met dat modem en het bulletin-board om te kunnen gaan. U moet tenminste over een terminal-emulatie programma beschikken om met het bulletin-board kontakt te kunnen opnemen. Zo'n terminal-emulator is in feite een programma dat niet veel meer doet dan datgene wat u intypt via de RS232 interface naar het modem te sturen, en de characters die het modem via de RS232 interface naar uw computer stuurt op het scherm te zetten. (Zie het artikel van Bram de Bruine in de 6502 Kenner nr 51). Wilt u het bulletin-board alleen maar gebruiken om berichten te lezen en achter te laten dan heeft u hieraan al genoeg. Wilt u echter files kunnen down- en uploaden dan moet u over een programma beschikken dat een of meerdere communicatie 'protocols' begrijpt. Een dergelijk protocol (afspraak) beschrijft hoe een file wordt overgestuurd over de telefoonlijn. De bedoeling van een dergelijk protocol is vooral om verminking van de file tijdens het overzenden te kunnen ontdekken en zonodig te corrigeren. Op bulletin-boards zijn drie protocols populair: ASCII, Xmodem en Kermit. Het laatste protocol is met toestemming van de scheppers naar de gelijknamige kikker uit de 'Muppet show' genoemd. Het ASCII protocol is het primitiefste van de drie. De file, die overigens alleen ASCII characters mag bevatten, wordt als een stroom opeenvolgende bytes over de lijn gezonden. De ontvangende computer moet er dan maar voor zorgen dat hij de ontvangen characters verwerkt. Het Xmodem protocol werkt al een stuk beter. Hier wordt om de zoveel bytes een checksum overgestuurd. De ontvangende computer vergelijkt de checksum met de door hem zelf berekende en geeft aan de zendende computer te kennen of alles ok was of niet. Was er iets niet in orde, dan worden de betreffende bytes nog eens overgezonden. Het Xmodem protocol is redelijk betrouwbaar, maar niet echt bomvast. Met Xmodem kunnen in principe ook binaire files worden overgezonden, maar alleen als zowel de zender als de ontvanger characters van 8 bits kunnen verwerken. Verreweg het beste protocol van dit moment is Kermit. Kermit is aan het begin van de jaren tachtig aan de Columbia University van New York ontwikkeld. Men heeft daarbij sterk gekeken naar professionele protocols als DECnet en OSI (Open Systems Interconnect). Kermit is een 'layered' protocol. Elke computer die over een RS232 interface beschikt en alle printbare ASCII characters ongehinderd verwerkt kan met Kermit werken. Bij Kermit wordt de te verzenden file in stukjes gehakt, de zogeheten packets. De bytes in deze packets worden met bepaalde algorithmes omgezet in printbare characters. Vervolgens worden er een checksum, een volgnummer een typeaanduiding en informatie over de lengte van het packet toegevoegd. Een packet bestaat derhalve uit verschillende lagen van informatie: aan de begin en einde een data link layer (checksum, lengte) om de integriteit van het packet te garanderen, een session layer (volgnummer, type) om de continuiteit van de datastroom te garanderen en een application layer waarin de feitelijke informatie van het packet staat. Er bestaan zeer nauwkeurige afspraken over de wijze hoe de twee Kermits verbinding met elkaar maken aan het begin van de file-transfer. De bedoeling hiervan is dat de Kermits elkaar duidelijk maken wie ze zijn en wat ze kunnen. Er bestaan even

nauwkeurige afspraken over wat er gebeurt als er storing op de lijn is, of wanneer bijvoorbeeld een mainframe responstijd-problemen heeft enz. Dat alles is noodzakelijk om een foutloze overdracht van de file in elke denkbare situatie mogelijk te maken. Op dit moment zijn voor zover mij bekend is zowel voor de Octopus als DOS65 programma's verkrijgbaar die een terminal emuleren en het ASCII en Xmodem protocol ondersteunen. Een communicatieprogramma dat Kermit ondersteunt is er helaas nog niet. Dat heeft waarschijnlijk te maken met het feit dat het schrijven van een goed werkende Kermit geen triviale klus is. Alleen al de manual die het Kermit protocol beschrijft is zo'n 85 pagina's dik! Toch bestaat er de kans dat er over enige tijd een werkende Kermit voor DOS-65 zal bestaan. Voor zover mij bekend zijn er nog geen mensen bezig met het aanpassen van Kermit voor de Octopus. Over de adressen waar de bovengenoemde programma's verkrijgbaar zijn bestaat nu nogal wat onduidelijkheid. Het zal u na het lezen van het bestuurlijke artikel in dit blad niet ontgaan zijn dat er rond het functioneren van de redactie de nodige problemen zijn geweest. De redactie was bij traditie de plaats waar de verspreiding van software plaatsvond. Omdat we momenteel met een interim-redactie werken zijn er een aantal dingen die opnieuw georganiseerd moeten worden, waaronder dus dit. Voor het goed functioneren van een bulletin-board is nog iets anders nodig dan hardware en software: een Sysop. De Sysop is de persoon die de bulletin-board hardware en software onder zijn beheer heeft. Deze beklagenswaardige figuur wordt op de onmogelijkste tijden opgebeld (Hoezo laat?? Ik werk altijd het liefste om halftwee 's nachts...), krijgt de domste vragen te beantwoorden (Oh, moet je een modem dan niet met een nul-modem kabel aansluiten...) en moet bovendien voortdurend voor politieagent spelen om ongedisciplineerde gebruikers in toom te houden (En als je nu nog eens een belgenmop op het file-gebied zet dan ontneem ik je je privileges...). Maar nu even in ernst, voordat we van start kunnen gaan met ons bulletin-board moet er wel iemand gevonden worden die de nodige tijd en energie wil steken in het opzetten ervan. Er moet een vaste plaats komen voor de PC die bijna dag en nacht in bedrijf moet zijn, en er moet een tweede telefoonaansluiting komen. Vooral in het begin, wanneer alles nog niet op rolletjes loopt zal de Sysop er heel wat tijd in moeten steken. Uiteraard worden alle kosten door de club betaald. Elders in dit blad treft u een oproep van het bestuur aan voor een Sysop. Als u belangstelling heeft, neem dan kontakt op met het bestuur. In het volgende nummer van de 6502 kenner hopen wij u een telefoonnummer te kunnen aanbieden waarachter een goed werkend bulletin-board schuil gaat.

Gert Klein 08370-23646

DOS65

LAATSTE NIEUWS

DOS65 versie 2 entries beschrijving.

Alle entries in IO65 staan reeds beschreven in de IO65 manual, maar de DOS65 entries ontbraken nog steeds. Vanaf 1 november is er ook een nederlandstalige DOS65 entries manual beschikbaar. Hierin staat precies beschreven waarvoor de diverse DOS65 entries gebruikt kunnen worden. Het is dan mogelijk door gebruik te maken van bepaalde entries vanuit zelf geschreven machinetaal programma's files te openen en hieruit te lezen of erin te schrijven. Ook het lezen van enkele sectoren wordt dan mogelijk vanuit machinetaal. Het een en ander is voorzien van voorbeelden. Hierbij wordt tevens een diskette geleverd waarop de file DVAR.MAC staat. In deze file staan al deze DOS65 entries en foutmelding definities. Deze file kan dan als library aan een zelf geschreven machinetaal programma gehangen worden. Het pakket is verkrijgbaar bij het officiele DOS65 distributie adres en kost F35,= exclusief F3,= verzendkosten.

5-Sep-87 13:37 taalst.lst Page 1

```
                        ; file          taalst.mac
                        ;
                        ; system        DOS65
                        ;
                        ; author        Ernst Elderenbosch
                        ;
                        ; date          16 aug '87
                        ;
                0200            org     $0200
                        ;
                C003    warm    equ     $c003
                C006    comd    equ     $c006
                C020    getcha  equ     $c020
                C023    prcha   equ     $c023
                C026    getecho equ     $c026
                C02F    prcrlf  equ     $c02f
                C03B    prtext  equ     $c03b
                        ;
                0020    point   equ     $0020           ; pointer into dictionary
                                                        ; (bbeg <= temp <= bend)
                0022    temp    equ     $0022           ; pointer into temporary
                                                        ; buffer (bufsa <= point
                                                        ; <= bufea)
                0024    bufeod  equ     $0024           ; pointer to end of data in
                                                        ; buffer
                0026    ysave   equ     $0026           ; temporary save y register
                0027    flag    equ     $0027
                        ;
                0025    comdky  equ     '%              ; command key
                0008    backsp  equ     8
                003D    sepkey  equ     '=
                002A    wildca  equ     '*
                0023    delkey  equ     '#
                000D    carret  equ     $0d
                000D    endkey  equ     carret
                        ;
0200 4C 7104    init    jmp     start           ; make use of dos65 help facility
0203 C8C5CCD0           fcc     $c8,$c5,$cc,$d0
0207 20 3BC0    help    jsr     prtext          ; normally fcc $4c,$4c,$4c
020A 0D5461616C         fcc     '\rTaalstudie (language study)  version .2
                                  (for dos65 level 2.01)\n'

020F 7374756469
0214 6520286C61
0219 6E67756167
021E 6520737475
0223 6479292020
0228 7665727369
022D 6F6E202E32
0232 2028666F72
0237 20646F7336
023C 35206C6576
0241 656C20322E
0246 3031290A
024A 0D456E6162         fcc     '\rEnables the user to create a small dictionary\n'
024F 6C65732074
0254 6865207573
0259 657220746F
025E 2063726561
0263 7465206120
0268 736D616C6C
026D 2064696374
0272 696F6E6172
0277 790A
0279 0D53796E74         fcc     '\rSyntax  : Taalst'
027E 617820203A
0283 205461616C
0288 7374
028A 0D4F707469         fcc     '\rOptions : none'
028F 6F6E73203A
0294 206E6F6E65
0299 0D6B657973         fcc     '\rkeys    : =  search buffer for word'
029E 202020203A
02A3 203D097365
02A8 6172636820
```

```
02AD 6275666665
02B2 7220666F72
02B7 20776F7264
02BC 0D09202009          fcc     '\r              ? if no match found, then type translation'
02C1 3F20696620
02C6 6E6F206D61
02CB 7463682066
02D0 6F756E642C
02D5 207468656E
02DA 2074797065
02DF 207472616E
02E4 736C617469
02E9 6F6E
02EB 0D0920202A          fcc     '\r       *      search for word ending with wildcard'
02F0 0973656172
02F5 636820666F
02FA 7220776F72
02FF 6420656E64
0304 696E672077
0309 6974682077
030E 696C646361
0313 7264
0315 0D09202023          fcc     '\r       #      search buffer for word and delete'
031A 0973656172
031F 6368206275
0324 6666657220
0329 666F722077
032E 6F72642061
0333 6E64206465
0338 6C657465
033C 0D09202063          fcc     '\r       cr     enters new word into buffer'
0341 7209656E74
0346 657273206E
034B 657720776F
0350 726420696E
0355 746F206275
035A 66666572
035E 0D09202062          fcc     '\r       bs     delete last typed character'
0363 730964656C
0368 657465206C
036D 6173742074
0372 7970656420
0377 6368617261
037C 63746572
0380 0D09202025          fcc     '\r       %      go into command mode'
0385 09676F2069
038A 6E746F2063
038F 6F6D6D616E
0394 64206D6F64
0399 65
039A 0D0A636F6D          fcc     '\r\ncommand mode:'
039F 6D616E6420
03A4 6D6F64653A
03A9 0D09202068          fcc     '\r       h      print help screen'
03AE 097072696E
03B3 742068656C
03B8 7020736372
03BD 65656E
03C0 0D20202020          fcc     '\r          s   save dictionary to disk as name.dct'
03C5 2020202020
03CA 2073097361
03CF 7665206469
03D4 6374696F6E
03D9 6172792074
03DE 6F20646973
03E3 6B20617320
03E8 6E616D652E
03ED 646374
03F0 0D0920206C          fcc     '\r       l      load dictionary from disk (name.dct)'
03F5 096C6F6164
03FA 2064696374
03FF 696F6E6172
0404 792066726F
0409 6D20646973
```

```
040E 6B20286E61
0413 6D652E6463
0418 7429
041A 0D09202063          fcc     '\r      c     clear dictionary'
041F 09636C6561
0424 7220646963
0429 74696F6E61
042E 7279
0430 0D09202077          fcc     '\r      w     wipe screen'
0435 2009776970
043A 6520736372
043F 65656E
0442 0D09202071          fcc     '\r      q     quit program'
0447 2009717569
044C 742070726F
0451 6772616D
0455 0D09202063          fcc     '\r      cr    leave command mode'
045A 72096C6561
045F 766520636F
0464 6D6D616E64
0469 206D6F6465
046E 0D00                fcc     '\r',0
0470 60                  fcc     $60              ; return from help when used
                                                  ; as a subroutine
                         ;
0471 20 1E08    start    jsr     setbuf           ; set pointer to begin temporary buffer
0474 20 2708             jsr     prompt           ;  and print prompt '>>'
0477 20 26C0    loop     jsr     getecho          ; get character and echo
047A C9 25               cmp     #comdky          ; command key?
047C D0 06               bne     1.f
047E 20 FD05             jsr     command          ; go into command mode
0481 4C 7104             jmp     start            ;  and jump to start afterwards
0484 C9 08      1        cmp     #backsp          ; cursor left?
0486 D0 06               bne     2.f
0488 20 E105             jsr     bsrout           ; then decrement pointer
048B 4C 7704             jmp     loop             ;  and get next character
048E C9 3D      2        cmp     #sepkey          ; separator key?
0490 D0 05               bne     3.f
0492 85 27               sta     flag
0494 4C B604             jmp     xlmode           ; then get translation
0497 C9 23      3        cmp     #delkey          ; delete key?
0499 D0 05               bne     4.f
049B 85 27               sta     flag             ; then delete word
049D 4C B604             jmp     xlmode           ; from dictionary
04A0 C9 2A      4        cmp     #wildca          ; wildcard character
04A2 D0 05               bne     6.f
04A4 85 27               sta     flag             ; then print list of words
04A6 4C B604             jmp     xlmode           ;  beginning with given characters
04A9 C9 0D      6        cmp     #carret          ; if carriage return, start again
04AB D0 03               bne     7.f              ;  and forget this word
04AD 4C 7104             jmp     start
04B0 20 E505    7        jsr     storch           ; store into temporary buffer
04B3 4C 7704             jmp     loop             ;  and get new character
04B6 20 E505    xlmode   jsr     storch           ; store the separator key too
04B9 20 3505             jsr     searmat          ; search for matching word
04BC B0 B3               bcs     start            ; word found, ready for next word
04BE A5 27               lda     flag             ; only if separator key is typed
04C0 C9 3D               cmp     #sepkey          ;  translation is asked
04C2 D0 AD               bne     start            ; wildcard or delete flag set
04C4 20 26C0    xlloop   jsr     getecho          ; word not found,
04C7 C9 08               cmp     #backsp          ; get translation from keybd
04C9 D0 06               bne     8.f              ;  and check if backspace typed
04CB 20 E105             jsr     bsrout           ;   then decrement pointer
04CE 4C C404             jmp     xlloop           ;    and get next character
04D1 C9 0D      8        cmp     #endkey          ; end of translation?
04D3 F0 06               beq     newentr          ;  then store word
04D5 20 E505             jsr     storch           ;   in dictionary
04D8 4C C404             jmp     xlloop           ;  else get next character
                         ;
                         ; new entry in library
                         ;
04DB 20 D805    newentr  jsr     chkeod           ; set eod to end of text
04DE 20 1E08             jsr     setbuf           ; reset pointer
04E1 20 ED05             jsr     setfree          ; search for free entry in library
```

```
04E4 A0 00              ldy     #0
04E6 A9 0D              lda     #carret         ; store cr in library
04E8 91 20              sta     [point],y
04EA 20 D307            jsr     incpnt          ; increment pointer
04ED A0 00              ldy     #0
04EF B1 22      1       lda     [temp],y        ; and copy temporary buffer
04F1 91 20              sta     [point],y       ;  to dictionary
04F3 20 E307            jsr     incbuf          ; increment
04F6 20 D307            jsr     incpnt          ;  pointers
04F9 90 10              bcc     8.f             ; dictionary full !!
04FB 38                 sec                     ; end of temporary buffer?
04FC A5 24              lda     bufeod
04FE C5 22              cmp     temp
0500 D0 ED              bne     1.b             ; not yet, so continue
0502 A5 25              lda     bufeod+1
0504 C5 23              cmp     temp+1
0506 D0 E7              bne     1.b             ; not yet, so continue
0508 4C 7104            jmp     start           ; finished
050B 20 3BC0    8       jsr     prtext
050E 0D6C696272         fcc     '\rlibrary full, last entry truncated',0
0513 6172792066
0518 756C6C2C20
051D 6C61737420
0522 656E747279
0527 207472756E
052C 6361746564
0531 00
0532 4C 7104            jmp     start           ; what to do now? ==> operators choice
                ;
                ; search for word match
                ;
0535 20 D805    searmat jsr     chkeod          ; set eod to separator
0538 20 1508            jsr     setbeg          ; start at beginning of dictionary
053B 20 1E08    nextwld jsr     setbuf          ; begin temporary buffer
053E A0 00      1       ldy     #0
0540 B1 20              lda     [point],y       ; get char from dictionary
0542 F0 42              beq     8.f             ; end of dictionary
0544 C9 0D              cmp     #carret         ; next char will be
0546 F0 06              beq     2.f             ;  beginning of next word
0548 20 D307            jsr     incpnt          ; else keep searching
054B 4C 3E05            jmp     1.b
054E 20 D307    2       jsr     incpnt          ; point to first character
0551 A0 00              ldy     #0
0553 B1 22      3       lda     [temp],y        ; and take a look at
0555 C9 3D              cmp     #sepkey         ;  word to be found
0557 F0 12              beq     4.f             ; end of word
0559 C9 2A              cmp     #wildca         ; end of significant
055B F0 33              beq     prtwild         ;  part of word
055D C9 23              cmp     #delkey         ; if delete key encountered
055F D0 03              bne     7.f             ;  then start deleting word
0561 4C AE05            jmp     delete          ;   from dictionary
0564 D1 20      7       cmp     [point],y       ; check character
0566 D0 D6              bne     1.b             ; no match, next word in library
0568 C8                 iny                     ; prepare to check
0569 D0 E8              bne     3.b             ;  next character
056B B1 20      4       lda     [point],y       ; check if word in dictionary
056D C9 3D              cmp     #sepkey         ;  is longer than wanted word
056F D0 CD              bne     1.b             ;   if so, words still don't match
0571 20 D307            jsr     incpnt          ; point to first character
0574 B1 20      5       lda     [point],y       ;  of translation
0576 C9 0D              cmp     #carret         ;   and start printing
0578 F0 0A              beq     6.f
057A 84 26              sty     ysave
057C 20 23C0            jsr     prcha
057F A4 26              ldy     ysave
0581 C8                 iny                     ; increment pointer
0582 D0 F0              bne     5.b             ;  and loop for more
0584 38         6       sec                     ; set carry to indicate match found
0585 60                 rts
0586 A5 24      8       lda     bufeod          ; set pointer to
0588 85 22              sta     temp            ;  correct place
058A A5 25              lda     bufeod+1        ;   in temporary buffer
058C 85 23              sta     temp+1
058E 18                 clc                     ; clc means no match found
```

```
058F 60                 rts
                        ;
                        ; print list of words with matching start characters
                        ;
0590 20 2FC0    prtwild jsr     prcrlf          ; a word is found with
0593 A0 00              ldy     #0              ;  matching starting characters
0595 B1 20      1       lda     [point],y       ;   so we are going to print
0597 F0 12              beq     9.f             ;    word and translation
0599 C9 0D              cmp     #carret         ;     and jump back to
059B F0 0B              beq     2.f             ;      find more matching words
059D 84 26              sty     ysave
059F 20 23C0            jsr     prcha
05A2 A4 26              ldy     ysave
05A4 C8                 iny
05A5 4C 9505            jmp     1.b
05A8 4C 3B05    2       jmp     nextwld         ; go find next matching word
05AB 4C 7104    9       jmp     start           ;  unless we reached end of dictionary
                        ;
                        ; delete the currently found word
                        ;
05AE A5 20      delete  lda     point           ; we found the word
05B0 85 22              sta     temp            ;  that we want to delete
05B2 A5 21              lda     point+1         ;   (don't ask me why
05B4 85 23              sta     temp+1          ;     we want to delete it)
05B6 A0 00              ldy     #0
05B8 B1 20      1       lda     [point],y       ; search for next word
05BA C8                 iny                     ;  because we have to
05BB C9 00              cmp     #0              ;   shift the rest of the
05BD F0 04              beq     2.f             ;    dictionary down to
05BF C9 0D              cmp     #carret         ;     close the gap again
05C1 D0 F5              bne     1.b
05C3 84 26      2       sty     ysave           ; found next word or end of dictionary
05C5 20 0708            jsr     dectmp          ;  anyway we have to copy this character too
05C8 A4 26      3       ldy     ysave           ; get offset
05CA B1 20              lda     [point],y       ; get character
05CC A0 00              ldy     #0              ; and copy it
05CE 91 20              sta     [point],y       ; over deleted word
                        ; cmp #0 ; end reached?
                        ; beq 9.f ; shift rest down
05D0 20 D307            jsr     incpnt          ; point to next character
05D3 B1 20              lda     [point],y
05D5 D0 F1              bne     3.b
05D7 60         9       rts                     ; finished
                        ;
                        ; save pointer
                        ;
05D8 A5 22      chkeod  lda     temp            ; we would like to
05DA 85 24              sta     bufeod          ;  save a pointer to the
05DC A5 23              lda     temp+1          ;   end of text for
05DE 85 25              sta     bufeod+1        ;    later use
05E0 60                 rts
                        ;
                        ; backspace routine
                        ;
05E1 20 F307    bsrout  jsr     decbuf          ; is this all?
05E4 60                 rts                     ;  yes indeed
                        ;
                        ; store character
                        ;
05E5 A0 00      storch  ldy     #0              ; store character
05E7 91 22              sta     [temp],y        ;  in temporary buffer
05E9 20 E307            jsr     incbuf          ;   and update pointer
05EC 60                 rts
                        ;
                        ; search end of data
                        ;
05ED 20 1508    setfree jsr     setbeg          ; now we want
05F0 A0 00      1       ldy     #0              ;  to know where
05F2 B1 20              lda     [point],y       ;   the data in the
05F4 F0 06              beq     9.f             ;    dictionary ends
05F6 20 D307            jsr     incpnt          ;     momentarily
05F9 4C F005            jmp     1.b
05FC 60         9       rts                     ; so now we really know
                        ;
```

```
                        ; command routine
                        ;
05FD 20 3BC0    command jsr     prtext          ; let user know that he's in
0600 203C636F6D         fcc     ' <command> ',0 ;  command mode
0605 6D616E643E
060A 2000
060C 20 26C0            jsr     getecho         ;   and ask him what he wants
060F C9 0D              cmp     #carret         ; nothing at all?
0611 F0 3D              beq     9.f             ;  ok, then return
0613 C9 68              cmp     #'h             ; command h - help screen
0615 D0 03              bne     1.f
0617 4C 0702            jmp     help
061A C9 73      1       cmp     #'s             ; command s - save dictionary
061C D0 03              bne     2.f
061E 4C 5106            jmp     savebuf
0621 C9 6C      2       cmp     #'l             ; command l - load dictionary
0623 D0 03              bne     3.f
0625 4C FD06            jmp     loadbuf
0628 C9 63      3       cmp     #'c             ; command c - clear dictionary
062A D0 03              bne     4.f
062C 4C 6907            jmp     clearbuf
062F C9 77      4       cmp     #'w             ; command w - wipe screen (dos command 'cl')
0631 D0 03              bne     5.f
0633 4C C807            jmp     cls
0636 C9 71      5       cmp     #'q             ; command q - quit program
0638 D0 C3              bne     command         ;  this command was not in the list
063A 68                 pla                     ; balance the stack
063B 68                 pla
063C 20 3BC0            jsr     prtext
063F 0D62796520         fcc     '\rbye bye ...\r',0
0644 627965202E
0649 2E2E0D00
064D 4C 03C0            jmp     warm            ; warm start dos65
0650 60         9       rts                     ; leave command mode
                        ;
                        ; save routine
                        ;
0651 20 1E08    savebuf jsr     setbuf
0654 20 3BC0            jsr     prtext
0657 6176652074         fcc     'ave to file ? ',0
065C 6F2066696C
0661 65203F2000
0666 A0 00              ldy     #0              ; copy command to temp buffer
0668 B9 B506    1       lda     comsave,y       ; set up text
066B F0 06              beq     2.f             ;  'save u:'
066D 91 22              sta     [temp],y
066F C8                 iny
0670 4C 6806            jmp     1.b
0673 84 26      2       sty     ysave           ; then get filename
0675 20 26C0            jsr     getecho
0678 A4 26              ldy     ysave
067A 91 22              sta     [temp],y
067C C8                 iny
067D C9 0D              cmp     #carret
067F D0 F2              bne     2.b
0681 88                 dey                     ; remove the carriage return
0682 98                 tya                     ;  because we want
0683 18                 clc                     ;   to add the file extension
0684 65 22              adc     temp            ;    and addresses later
0686 85 22              sta     temp            ; set pointer to
0688 A5 23              lda     temp+1          ;  old pointer + y register
068A 69 00              adc     #0              ;   because that's where
068C 85 23              sta     temp+1          ;    the carriage return was
068E 20 CC06            jsr     geteof          ; get end address library
0691 A0 00              ldy     #0
0693 B9 BD06    3       lda     comsave2,y      ; start copying
0696 91 22              sta     [temp],y        ;  '.dct 1000'  to the
0698 F0 04              beq     4.f             ;   command line
069A C8                 iny                     ;    and include the
069B 4C 9306            jmp     3.b             ;     end address too
069E 20 3BC0    4       jsr     prtext          ; notify the user
06A1 0D73617669         fcc     '\rsaving ...',0
06A6 6E67202E2E
06AB 2E00
```

```
06AD A0 30              ldy     #bufsa&255      ;  and start the action
06AF A9 08              lda     #bufsa>>8       ;   with a command to the
06B1 20 06C0            jsr     comd            ;    dos65 command interpreter
06B4 60                 rts                     ; that's all folks
                ;
06B5 7361766520 comsave fcc     'save u:',0
06BA 753A00
06BD 2E64637420 comsave2 fcc    '.dct 1000,'
06C2 3130303020
06C7 3866666600 comsave3 fcc    '8fff',0        ; reserved for real end address
                ;
                ; get end address in ascii
                ;
06CC 20 ED05    geteof  jsr     setfree         ; first see where
06CF A5 21              lda     point+1         ;  the dictionary ends
06D1 A2 00              ldx     #0              ;   and then convert the
06D3 48         1       pha                     ;    hex address to text
06D4 6A                 rora                    ;     and insert it
06D5 6A                 rora                    ;      as four ascii characters
06D6 6A                 rora                    ;       in comsave3
06D7 6A                 rora                    ; use the x register
06D8 20 F106            jsr     hexasc          ;  as a flag
06DB 9D C706            sta     comsave3,x      ;   so we can use
06DE 68                 pla                     ;    the same routine twice
06DF 20 F106            jsr     hexasc          ;   once with x=0
06E2 9D C806            sta     comsave3+1,x    ;    and once with x=2
06E5 E0 02              cpx     #2              ; check if finished second round
06E7 F0 07              beq     9.f             ;  then stop it
06E9 A5 20              lda     point           ;   else do some homework
06EB A2 02              ldx     #2              ;    and go at it again
06ED 4C D306            jmp     1.b             ;     with x=2 this time
06F0 60         9       rts
                ;
                ; convert nibble to ascii
                ;
06F1 29 0F      hexasc  and     #$0f            ; convert a nibble
06F3 18                 clc                     ;  to an ascii character
06F4 69 30              adc     #$30            ;   in the oldfashioned way
06F6 C9 3A              cmp     #$3a            ; wasn't that Aznavour
06F8 90 02              bcc     9.f             ;  i just heard?
06FA 69 06              adc     #6              ; carry is set, makes +7
06FC 60         9       rts
                ;
                ; load routine
                ;
06FD 20 1E08    loadbuf jsr     setbuf          ; prepare for loading
0700 20 3BC0            jsr     prtext          ; ask for filename
0703 6F61642066         fcc     'oad file ? ',0
0708 696C65203F
070D 2000
070F A0 00              ldy     #0              ;  but let user think
0711 B9 5C07    1       lda     comload,y       ;   about it for some
0714 F0 06              beq     2.f             ;    microseconds while
0716 91 22              sta     [temp],y        ;     i prepare the first part
0718 C8                 iny                     ;      of the command line
0719 4C 1107            jmp     1.b             ; text: 'load u:'
071C 84 26      2       sty     ysave           ; ok user, do your thing
071E 20 26C0            jsr     getecho         ;  you don't even have
0721 A4 26              ldy     ysave           ;   to type the file extension
0723 91 22              sta     [temp],y        ;    because i will do that
0725 C8                 iny                     ;     after you give me a
0726 C9 0D              cmp     #carret         ;      carriage return
0728 D0 F2              bne     2.b
072A 88                 dey                     ; remove the carriage return
072B 98                 tya                     ;  because i want to
072C 18                 clc                     ;   add the file extension now
072D 65 22              adc     temp            ; unlike the save command
072F 85 22              sta     temp            ;  we don't need addresses
0731 A5 23              lda     temp+1          ;   so the rest is child's play
0733 69 00              adc     #0
0735 85 23              sta     temp+1
0737 A0 00              ldy     #0
0739 B9 6407    3       lda     comload2,y
073C 91 22              sta     [temp],y
```

```
073E F0 04              beq     4.f
0740 C8                 iny
0741 4C 3907            jmp     3.b
0744 20 3BC0    4       jsr     prtext          ; notify the user
0747 0D6C6F6164         fcc     '\rloading ...',0
074C 696E67202E
0751 2E2E00
0754 A0 30              ldy     #bufsa&255      ;  that we are going
0756 A9 08              lda     #bufsa>>8       ;   to use the
0758 20 06C0            jsr     comd            ; command interpreter
075B 60                 rts                     ;     again
                ;
075C 6C6F616420 comload fcc     'load u:',0
0761 753A00
0764 2E64637400 comload2 fcc    '.dct',0
                ;
                ; clear textbuffer
                ;
0769 20 3BC0    clearbuf jsr    prtext          ; human error?
076C 6C65617220         fcc     'lear dictionary - are you sure [y/n]? ',0
0771 6469637469
0776 6F6E617279
077B 202D206172
0780 6520796F75
0785 2073757265
078A 205B792F6E
078F 5D3F2000
0793 20 26C0            jsr     getecho
0796 C9 79              cmp     #'y             ; no human error?
0798 D0 2D              bne     9.f             ;  else no hard feelings
079A 20 3BC0            jsr     prtext          ;   haha, too late now
079D 0D636C6561         fcc     '\rclearing dictionary now ...',0
07A2 72696E6720
07A7 6469637469
07AC 6F6E617279
07B1 206E6F7720
07B6 2E2E2E00
07BA 20 1508            jsr     setbeg          ; all your work gone
07BD A0 00      1       ldy     #0              ;  in a few seconds
07BF 98                 tya                     ; fill with binary null
07C0 91 20              sta     [point],y
07C2 20 D307            jsr     incpnt
07C5 B0 F6              bcs     1.b
07C7 60         9       rts
                ;
                ; wipe screen
                ;
07C8 A0 D0      cls     ldy     #wipebuf&255    ; get command 'cl'
07CA A9 07              lda     #wipebuf>>8     ;  and surrender to the
07CC 20 06C0            jsr     comd            ;   command processor
07CF 60                 rts
                ;
07D0 636C00     wipebuf fcc     'cl',0
                ;
                ; several pointer handling routines
                ;
07D3 E6 20      incpnt  inc     point           ; increment pointer
07D5 D0 02              bne     1.f             ;  to the dictionary
07D7 E6 21              inc     point+1
07D9 38         1       sec                     ;    and check if we are
07DA A9 FF              lda     #bend&255       ;     still between boundaries
07DC E5 20              sbc     point
07DE A9 8F              lda     #bend>>8        ; if we are not
07E0 E5 21              sbc     point+1         ;  we clear the carry
07E2 60                 rts
                ;
07E3 E6 22      incbuf  inc     temp            ; increment pointer
07E5 D0 02              bne     1.f             ;  to the temporay buffer
07E7 E6 23              inc     temp+1
07E9 38         1       sec                     ; and check if at end of buffer
07EA A9 94              lda     #bufea&255      ;  if so, return with
07EC E5 22              sbc     temp            ;   carry clear
07EE A9 08              lda     #bufea>>8       ;  else return with
07F0 E5 23              sbc     temp+1          ;   carry set
```

5-Sep-87 14:06 taalst.lst Page 9

```
07F2 60                 rts
                ;
07F3 A5 22      decbuf  lda     temp            ; check if already at
07F5 C9 30              cmp     #bufsa&255      ;  beginning of temporary buffer
07F7 F0 0D              beq     9.f             ; if so, return
07F9 38                 sec                     ; else decrement pointer
07FA A5 22              lda     temp
07FC E9 01              sbc     #1
07FE 85 22              sta     temp
0800 A5 23              lda     temp+1
0802 E9 00              sbc     #0
0804 85 23              sta     temp+1
0806 60         9       rts
                ;
0807 38         dectmp  sec                     ; decrement pointer
0808 A5 20              lda     point           ;  to dictionary
080A E9 01              sbc     #1              ; no need to check
080C 85 20              sta     point           ;  for boundaries
080E A5 21              lda     point+1         ; only used in
0810 E9 00              sbc     #0              ;  case boundary
0812 85 21              sta     point+1         ;   cannot be reached
0814 60         9       rts
                ;
0815 A9 00      setbeg  lda     #bbeg&255       ; set pointer to
0817 85 20              sta     point           ;  begin of dictionary
0819 A9 10              lda     #bbeg>>8
081B 85 21              sta     point+1
081D 60                 rts
                ;
081E A9 30      setbuf  lda     #bufsa&255      ; set pointer to
0820 85 22              sta     temp            ;  begin of temporary buffer
0822 A9 08              lda     #bufsa>>8
0824 85 23              sta     temp+1
0826 60                 rts
                ;
                ; type prompt
                ;
0827 20 3BC0    prompt  jsr     prtext          ; here you can design
082A 0D3E3E2000         fcc     '\r>> ',0       ;  your personal prompt,
                ; for instance ;
                ; fcc '\r(^_^) ',0 ; would print funny face as prompt
082F 60                 rts
                ;
                ; define 256 byte buffer (please leave size 256 bytes)
                ;  & 32 k dictionary (can be any size you want)
                ;
          0830  bufsa   equ     *               ; start of temporary buffer
          0894  bufea   equ     *+100           ; end of temporary buffer
                ;
          1000  bbeg    equ     $1000           ; dictionary begin
          8FFF  bend    equ     $8fff           ; dictionary end
                ;
          0200          end     init
                        label table
```

backsp	0008	bbeg	1000	bend	8FFF	bsrout	05E1	bufea	0894
bufeod	0024	bufsa	0830	carret	000D	chkeod	05D8	clearbuf	0769
cls	07C8	comd	C006	comdky	0025	comload	075C	comload2	0764
command	05FD	comsave	06B5	comsave2	06BD	comsave3	06C7	decbuf	07F3
dectmp	0807	delete	05AE	delkey	0023	endkey	000D	flag	0027
getcha	C020	getecho	C026	geteof	06CC	help	0207	hexasc	06F1
incbuf	07E3	incpnt	07D3	init	0200	loadbuf	06FD	loop	0477
newentr	04DB	nextwld	053B	point	0020	prcha	C023	prcrlf	C02F
prompt	0827	prtext	C03B	prtwild	0590	savebuf	0651	searmat	0535
sepkey	003D	setbeg	0815	setbuf	081E	setfree	05ED	start	0471
storch	05E5	temp	0022	warm	C003	wildca	002A	wipebuf	07D0
xlloop	04C4	xlmode	04B6	ysave	0026				

Errors detected: 0

Subdirectories.

Misschien ten overvloede, maar voor de mensen die nog niet zo thuis zijn op DOS65 gebied, het volgende voor wat betreft het selecteren van subdirectories. Het systeem kent drie logische namen voor bepaalde geselecteerde directories.

s: - dit is de systeem directory
u: - dit is de user directory
w: - dit is de work directory

Na het commando 'asn <return>' laat het systeem zien waaraan die logische namen zijn toegekend. Als er nog geen andere toekenningen zijn gedaan staan ze alle drie te wijzen naar de hoofddirectory van drive 0. Gebruiken we echter drive 1 als user drive, dan typen we in: 'asn u=1: <return>'. Op dat moment zal DOS65 alle commando's die niet standaard in DOS zitten zoals CAT, COPY, FORMAT enz. van de systeem directory (0) halen en de files die opgehaald moeten worden bij bv. 'ED program.mac' komen van de user: hoofddirectory van drive 1. Typen we echter in: 'asn u=1:a/ <return>' dan is de subdirectory A van drive 1 de user directory geworden. Bij ophalen en wegschrijven van files wordt dan alleen directory A van drive 1 gebruikt. Zo kunnen we ook de work directory naar subdirectories laten wijzen. Typen we in 'asn w=1:b/ <return>' dan is de work directory op drive 1 subdirectory B. We kunnen nu heel simpel iets copieren van subdirectory 1:a/ naar subdirectory 1:b/. Dat gaat dan door het volgende in te typen 'copy u:program.mac w:'. We zien dat de schrijfwijze 'u:' iets gemakkelijker is dan '1:a/'. Als we de user directory laten wijzen naar een subdirectory dan geven de commando's DIR, SDIR en CAT ook alleen maar de informatie over de user directory en niet over de gehele schijf. Alleen het commando 'DIR -/' geeft een directory list van de complete schijf waarop de user directory zit met alle subdirectories. Rest me nog duidelijk te maken waar de work directory voor gebruikt wordt. Bij bv. compileren of assembleren worden er files aangemaakt die men later niet meer nodig heeft. Deze kunnen dan op de work directory gezet worden en kan men later de complete work directory deleten.

Elektuur keyboard.

Er treden problemen op met DOS65 als het Elektuur keyboard zonder aanpassing op het systeem wordt aangesloten. Het probleem zit hem in het feit dat de keyboard decoder van het keyboard niet feilloos werkt. Bij sommige toetsen wordt namelijk het meest significant bit (7) hoog gemaakt. Omdat DOS65 wel graag alle codes van een toetsenbord wil kunnen lezen wordt dit bit er in de software niet uitgefilterd. Alle DOS65 commando's echter zijn in gewoon ASCII en niet met dat bit hoog. De commando interpreter kan de commando's dus niet thuisbrengen en er volgen foutmeldingen. Aangezien IO65 geschikt is gemaakt om karakters in invers video weer te geven en dit is gedaan door het meest significante bit van het af te beelden karakter hoog te maken worden de commando's bij het beschreven probleem geval in invers video op het beeld gezet. Hieraan is dus de fout te herkennen. Het is vrij simpel om het probleem op te lossen. Haal het meest significante bit van het toetsenbord los en verbindt de nu ook losgekomen VIA lijn met de ground. Het gevolg is dat het meest significante bit nu altijd laag is en de commando's goed geinterpreteerd worden. Een nadeel is, dat er nu geen toetswaarden meer met dat bit hoog naar de computer gestuurd kunnen worden. Deze zouden eventueel gebruikt kunnen worden in zelf geschreven programma's of als functietoetsen in de screen editor.

+ +

Verbetering voor de statische RAMkaart.

De statische RAMkaart die werd gepubliceerd in maart '83 blijkt een onvolkomenheidje te bevatten. Sommige SRAM-chips moeten namelijk voor ieder access een nieuwe Chip-Select aangeboden krijgen, omdat deze chip anders het adres op hun adreslijnen niet in-latchen. Als de adreslijnen A13-A15 niet veranderen, gebeurt dit op de genoemde kaart niet. Dit kan eenvoudig verholpen worden door de CS te ANDen met phi2. Buig hiertoe pin 2 van de 74LS156 of 74LS155 naar buiten, en verbindt deze met pin 16 van de 74LS240. De kaart werkt nu goed met alle SRAM chips.

***** D O S - 6 5 V 2.0 F I L E T R A N S F E R *****

Door: H.A.J.Quast
Dekemastate 15
1275 CM Huizen N.H.
tel. 02152-54905

Dit programma verzorgt de dataoverdracht tussen de DOS-65 computer en een andere computer. De interface is op RS-232 basis.

Als tweede computer maak ik gebruik van een P2000T van Philips. Op deze Computer zit o.a. een programma "Familiebestand" in rompack. Met dit programma is het mogelijk om b.v. bestanden aan te maken, ook kan deze gebruikt worden als een eenvoudige tekstverwerker.

Het werken met het programma filetransfer gaat als volgt:

Op de P2000T wordt b.v. een stuk tekst ingetyped. Aan het einde van de tekst wordt een e.o.f. karakter gezet. (ik gebruik op de P2000T meestal het teken @) Vervolgens starten we het programma "FILETRANS" op de DOS-65 computer en geven na de juiste keuze gemaakt te hebben op de P2000T voor de Baudrate, het printkomando. De file wordt nu overgezonden naar de DOS-65 computer en verwerkt met de opgegeven options. Na afloop komt de DOS-65 terug in de commandomode en na enige tijd ~ 10 sec ook de P2000T.

Beschrijving van het filetransfer programma.

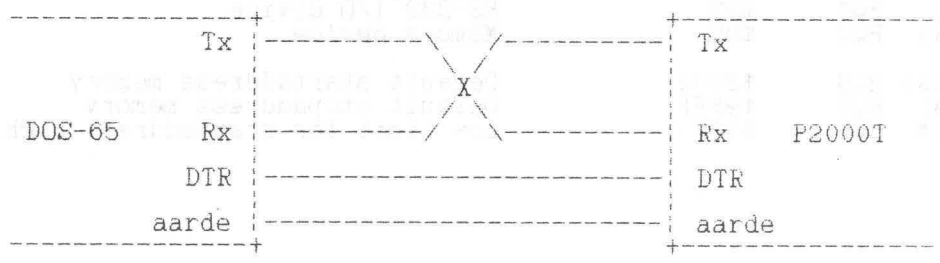
Command: FILETRANS [-HMBE +m,n] FILE
Optiontabel

-H	Help
-M	Data naar geheugen of naar disk (default disk)
-B	Baudrate 1200/2400 b/s (default 2400b/s)
-E	$03 als eof karakter. (default @)
+m	m = Startadres voor laden in het geheugen (default $3000)
n	n = Stopadres voor laden in het geheugen (default $9FFF)
FILE	Filenaam van de datafile

Het filetransfer programma werkt samen met de RS-232 input en met het geheugen of de diskdrive. Na het opstarten van het programma wordt het RS-232 device geinitialiseerd. In het commando kunnen we opgeven voor welke baudrate er gekozen moet worden, 1200/2400 baud (voor andere snelheden kunnen deze waarden worden aangepast in de datatabel BAUD12/BAUD24).
Nadat gekozen is voor de baudrate kan er opgegeven worden welk e.o.f. karakter gebruikt moet worden. Bij het printen via de printerpoort geeft de P2000T geen eof karakter $03. Om nu de DOS-65 computer te kennen te geven dat de file overgezonden is maak ik gebruik van het kar."@" in de tekst op de P2000T. Dit karakter is ook het default karakter voor het programma. Als er gekozen is voor het laden in het geheugen dan kan met de parameter +m,n worden opgegeven waar de data in het geheugen gezet moet worden (default start $3000 stop $9FFF). Het geheugengebied dat opgegeven kan worden loopt van $0100 t/m $9FFF. Er wordt geen test gedaan of het stopadres hoger is dan het startadres.

Als laatste kan opgegeven worden wat de filenaam is waarin de data weggeschreven moet worden. Wanneer de data verstuurd is geeft de DOS-65 computer op het scherm de melding "Data transfer ready", vervolgens kan de data bewerkt worden met de editor.

Het ontvangen kan voortijdig gestopt worden door het indrukken van de toets "@" op de DOS-65 computer.
De RS-232 verbinding tussen de twee computers bestaat uit:



```
ijd  9-May-87 21:37  === file filetrans.mac ===    pagina 1

                ;
                ;
                ; ************** D O S - 6 5   V 2.01 ****************
                ; *
                ; *       F I L E   T R A N S F E R
                ; *
                ; ************************************************
                ;
                ;
                ;       H.A.J.Quast
                ;       Dekemastate 15
                ;       1275 CM Huizen N.H.
                ;       tel. 02152-54905
                ;
                ;
                ; With this program you can send a file from another
                ; computer to your own computer via the RS-232 interface.
                ;
                ; Page zero address
                ;-----------------------------------
          00A0          ORG     $00A0
00A0            OPTMASK RES     1               Option mask
00A1            POINTER RES     2               Commandbuffer pointer
00A3            TEMPX   RES     1               Save address X-reg.
00A4            MEMSTAR RES     2               Startaddress in memory
00A6            MEMEND  RES     2               Endaddress in memory
00A8            FILEOUT RES     1               Outputfilenumber
00A9            EOFCHA  RES     1               EOF character
                ;
                ; System subroutine's
                ;-----------------------------------
          C03B  PRINT1  EQU     $C03B           Print text on screen
          C059  REDROUT EQU     $C059           Redirect output
          C05F  OUTCLO  EQU     $C05F           Close outputredirectfile
          C068  SOPT1   EQU     $C068           Get option
          C06B  SPAR1   EQU     $C06B           Get parameter
          D00C  SWRITE1 EQU     $D00C           Single write in file
          D0B7  ERMES1  EQU     $D0B7           Error messages
          F006  OUTDEV  EQU     $F006           Output devices register
          F009  INPDEV  EQU     $F009           Input devices register
          F00C  INITDEV EQU     $F00C           Init. I/O device
                ;
                ; RS-232 device
                ;-----------------------------------
          E132  ACICMD  EQU     $E132           Command register
          E734  ACCTL   EQU     $E734           ACIA control register
          E735  ACCMD   EQU     $E735           ACIA command register
          E776  WRBEGL  EQU     $E776           Startaddress memory
          E777  WRBEGH  EQU     $E777
          E778  WRENDL  EQU     $E778           Endaddress memory
          E779  WRENDH  EQU     $E779
                ;
                ;
                ; Mask for option-code
                ;-----------------------------------
          0080  HELP    EQU     %10000000       Help
          0040  BAUDR   EQU     %01000000       Baudrate
          0020  MEMDISK EQU     %00100000       Memory or disk
          0010  EOFHEX  EQU     %00010000       EOF char. $03
                ;
                ; Data
                ;-----------------------------------
          0001  DISPLAY EQU     $01             Display output device
          0003  RS232   EQU     $03             RS-232 I/O divice
          0005  MEMORY  EQU     $05             Memory device
                ;
          3000  STARTAD EQU     $3000           Default startaddress memory
          9FFF  STOPAD  EQU     $9FFF           Default stopaddress memory
          0010  LOWLIM  EQU     $10             Low limit for startaddress high byte
```

```
Tijd  9-May-87 21:37  === file filetrans.mac ===    pagina 2

          00A0  HIGLIM  EQU     $A0             High limit for stopaddress high byte
          00B8  BAUD12  EQU     $B8             Data for controlbyte ACIA
          00BA  BAUD24  EQU     $BA             Data for controlbyte ACIA
          0001  COMBYT1 EQU     $01             Data for commandbyte ACIA
          0000  COMBYT2 EQU     $00             Data for commandbyte ACIA
          0040  EOFA    EQU     '@              End of file character
          0003  EOFH    EQU     $03             End of file character
          000A  LFEED   EQU     $0A
                ;
                ;-------------------------------------
                ; Mainprogram
                ;-------------------------------------
          0A00          ORG     $0A00
                ;
                ; Initialisation
                ;
0A00 20 68C0    FILETRANS JSR   SOPT1           Get option
0A03 48                 FCC     'H'             Help
0A04 42                 FCC     'B'             Baudrate 1200/2400 b/s
0A05 4D                 FCC     'M'             Data to memory or disk
0A06 45                 FCC     'E'             Eof char. is $03
0A07 00                 FCB     $0              Option table end
0A08 90 03              BCC     1.f             Branch if option are correct
0A0A 4C 640B            JMP     ERROR
                ;
0A0D 86 A0      1       STX     OPTMASK         Save the option
0A0F A2 40              LDX     #$40
0A11 20 6BC0            JSR     SPAR1           Load parameters
0A14 A4                 FCB     #MEMSTAR
0A15 A6                 FCB     #MEMEND
0A16 00                 FCB     $00             paremeter table end
0A17 90 03              BCC     2.f             Branch if parameter are correct
0A19 4C 640B            JMP     ERROR
                ;
0A1C 84 A1      2       STY     POINTER         Save filepointer address
0A1E 85 A2              STA     POINTER+1       Save the filepointer address
                ;
                ; Test the help command
                ;
0A20 A5 A0      TSHELP  LDA     OPTMASK         Load option masker
0A22 29 80              AND     #HELP           Test help command
0A24 F0 03              BEQ     INIT
0A26 4C 6C0B            JMP     INFO
                ;
                ; Initialisation ACIA
                ;
0A29 A5 A0      INIT    LDA     OPTMASK         Load optionmask
0A2B 29 40              AND     #BAUDR          Test baudrate
0A2D F0 05              BEQ     1.f
0A2F A9 B8              LDA     #BAUD12         Set 1200 baud/s
0A31 4C 360A            JMP     INITACIA
                ;
0A34 A9 BA      1       LDA     #BAUD24         Set 2400 baud/s
0A36 8D 34E7    INITACIA STA    ACCTL
0A39 A2 03              LDX     #RS232
0A3B 20 0CF0            JSR     INITDEV         Init. Input dev. RS-232
0A3E 90 23              BCC     TSEOF           Branch if device ready
0A40 20 3BC0            JSR     PRINT1
0A43 071B692049         FCC     '\a\Ei Inputdevice not ready. \En\r',0
0A62 60                 RTS                     Exit program
                ;
                ; Test which eof char is used.
                ;
0A63 A5 A0      TSEOF   LDA     OPTMASK         Load optionmask
0A65 29 10              AND     #EOFHEX         Test $03 char.
0A67 F0 04              BEQ     1.f
0A69 A9 03              LDA     #EOFH           Load eof char.
0A6B D0 02              BNE     STACHA          Branch allways
0A6D A9 40      1       LDA     #EOFA           Load eof char.
0A6F 85 A9      STACHA  STA     EOFCHA          Save the char.
```

```
ijd  9-May-87 21:37  === file filetrans.mac ===    pagina 3

                     ;
                     ; Test memory or disk
                     ;
A71 A5 A0    TSMEM   LDA     OPTMASK         Load optionmask
A73 29 20            AND     #MEMDISK        Test memory or disk
A75 D0 56            BNE     TOMEM           Branch if memory
                     ;
                     ; Test filename
                     ;
A77 A4 A1    TSFILE  LDY     POINTER         Load the filepointer address
A79 A5 A2            LDA     POINTER+1
A7B A2 E1            LDX     #$E1            file mode
A7D 20 59C0          JSR     REDROUT         Redirect inputfile
A80 90 03            BCC     1.f
A82 4C 640B          JMP     ERROR
                     ;
A85 86 A8    1       STX     FILEOUT         Save inputfilenumber
A87 A9 01    LOOP    LDA     #COMBYT1        Enable RS232 device
A89 8D 32E1          STA     ACICMD          Set command register
A8C A2 03            LDX     #RS232          Load device number
A8E 20 09F0          JSR     INPDEV          Get char.
A91 A0 00            LDY     #COMBYT2        Disable RS232 device
A93 8C 32E1          STY     ACICMD          Set Command register
A96 C5 A9            CMP     EOFCHA          Test eof char.
A98 F0 0E            BEQ     EXIT
A9A C9 0A            CMP     #LFEED          Suppress linefeed
A9C F0 E9            BEQ     LOOP
A9E A6 A8            LDX     FILEOUT         Load outputfile number
AA0 20 0CD0          JSR     SWRITE1         Single write in file
AA3 90 E2            BCC     LOOP
AA5 4C 640B          JMP     ERROR           Branch if error
                     ;
AA8 A6 A8    EXIT    LDX     FILEOUT         Load outputfile number
AAA 20 5FC0          JSR     OUTCLO          Close the outputfile
AAD 20 3BC0  EXIT1   JSR     PRINT1
AB0 1B69204461       FCC     '\Ei Data transfer ready. \En\r',0
ACC 60               RTS                     Exit program
                     ;
                     ;initialisation of memory startaddress
                     ;
ACD A5 A5    TOMEM   LDA     MEMSTAR+1       Test pointer are zero
ACF D0 0D            BNE     STRADR          Branch if not zero
                     ;
                     ; Load default startaddress
                     ;
AD1 A9 00    1       LDA     #STARTAD&255    Startaddress lowbyte
AD3 8D 76E7          STA     WRBEGL
AD6 A9 30            LDA     #STARTAD>>8     Startaddress highbyte
AD8 8D 77E7          STA     WRBEGH
ADB 4C EA0A          JMP     MEMSTOP
                     ;
                     ; Load startaddress
                     ; Test startaddress >=$1000
                     ;
ADE C9 10    STRADR  CMP     #LOWLIM         Test >=$10
AE0 30 56            BMI     RANERR
AE2 8D 77E7          STA     WRBEGH          Startaddress highbyte
AE5 A5 A4            LDA     MEMSTAR         Startaddress lowbyte
AE7 8D 76E7          STA     WRBEGL
                     ;
                     ; initialisation of memory stopaddress
                     ;
AEA A5 A7    MEMSTOP LDA     MEMEND+1        Test pointer are zero
AEC D0 0D            BNE     STOADR          Branch if not zero
                     ;
                     ; Load default end address
                     ;
AEE A9 FF            LDA     #STOPAD&255     Stopaddress lowbyte
AF0 8D 78E7          STA     WRENDL
AF3 A9 9F            LDA     #STOPAD>>8      Stopaddress highbyte
AF5 8D 79E7          STA     WRENDH
```

```
Tijd  9-May-87 21:37  === file filetrans.mac ===     pagina 4

0AF8 4C 070B            JMP     MEMMOVE

                        ;
                        ; Load endaddress
                        ; test endaddress >=$A000
                        ;
0AFB C9 A0      STOADR  CMP     #HIGLIM         test for >= $A0
0AFD 10 39              BPL     RANERR
0AFF 8D 79E7            STA     WRENDH          Stopaddress highbyte
0B02 A5 A6              LDA     MEMEND          Stopaddress lowbyte
0B04 8D 78E7            STA     WRENDL
                        ;
                        ; Initialisation of the memory device
                        ;
0B07 A2 05      MEMMOVE LDX     #MEMORY
0B09 20 0CF0            JSR     INITDEV         Init. output memory
0B0C A9 01              LDA     #COMBYT1        Load commandbyte ACIA
0B0E 8D 35E7            STA     ACCMD
0B11 A9 01      LOOPMEM LDA     #COMBYT1        Enable RS232 device
0B13 8D 32E1            STA     ACICMD          Set command register
0B16 A2 03              LDX     #RS232          Load device number
0B18 20 09F0            JSR     INPDEV          Get char.
0B1B C5 A9              CMP     EOFCHA          Test eof char.
0B1D F0 0C              BEQ     EXITMEM
0B1F C9 0A              CMP     #LFEED          Suppress linefeed
0B21 F0 EE              BEQ     LOOPMEM
0B23 A2 05              LDX     #MEMORY         Load outputdevice number
0B25 20 06F0            JSR     OUTDEV          Save data into memory
0B28 4C 110B            JMP     LOOPMEM
                        ;
0B2B A0 00      EXITMEM LDY     #COMBYT2        Disable RS232 device
0B2D 8C 32E1            STY     ACICMD          Set Command register
0B30 A2 01              LDX     #DISPLAY        Load outputdevice number
0B32 20 0CF0            JSR     INITDEV         Init. output device
0B35 4C AD0A            JMP     EXIT1
                        ;
0B38 20 3BC0    RANERR  JSR     PRINT1
0B3B 071B692041         FCC     '\a\Ei Address range <$1000 or >=$A000 \En\r',0
0B63 60                 RTS                     Exit program
                        ;
0B64 A0 00      ERROR   LDY     #COMBYT2        Disable RS232 device
0B66 8C 32E1            STY     ACICMD          Set Command register
0B69 4C B7D0            JMP     ERMES1          Print error
                        ;
                        ; end main program
                        ;
0B6C 20 3BC0    INFO    JSR     PRINT1          Print help-info
0B6F 0C0D               FCC     '\f\r'
0B71 1B69204620         FCC     '\Ei F I L E  - T R A N S F E R \En\r\r'
0B93 5769746820         FCC     'With this program you can send a file from\r'
0BBE 616E6F7468         FCC     'another computer to your own computer via\r'
0BE8 7468652052         FCC     'the RS-232 interface.\r'
0BFE 43686F6963         FCC     'Choice for data to memory or disk.\r'
0C21 4D61782E20         FCC     'Max. address range $1000 to $A000\r'
0C43 0D                 FCC     '\r'
0C44 436F6D6D61         FCC     'Command: FILETRANS [-HMBE +m,n] FILE\r\r'
0C6A 1B69204F70         FCC     '\Ei Optiontabel \En\r\r'
0C7D 2D48202020         FCC     '-H     Help\r'
0C89 2D4D202020         FCC     '-M     Data to memory or disk (default disk)\r'
0CB6 2D42202020         FCC     '-B     Baudrate 1200/2400 b/s (default 2400b/s\r'
0CE5 2D45202020         FCC     '-E     $03 as eof char. (default @)\r'
0D09 2B6D202020         FCC     '+m     m = Startaddress for load in memory (default $3000)\r
'
0D44 206E202020         FCC     ' n     n = Endaddress for load in memory (default $9FFF)\r'
0D7D 46494C4520         FCC     'FILE   Filename for the datafile\r'
0D9E 0D00               FCC     '\r',0
0DA0 60                 RTS
                        ;
      0A00              END     FILETRANS
```

Trage printerspooler.

Onlangs werd ik met een vreemd probleem geconfronteerd. De Star gemini 10 printer van een van de DOS65 gebruikers werkte wel maar niet erg snel op DOS65. Werd output device 2 aangezet dan deed de printer zijn werk zoals dat hoort op de normale snelheid. Werd echter de printer spooler geactiveerd (het PRINT commando) dan duurde het erg lang voordat er een regel tekst naar de printer verstuurd werd. Na het normaal printen van een regel duurde het een aantal seconden voordat de volgende regel kwam. Eerst werd er aan een hardware fout gedacht, maar aangezien de printer correct werkte met via output device 2 moesten we dat idee maar snel laten varen.

Om de oorzaak te kunnen begrijpen moeten we even iets dieper in DOS65 duiken. Bij 1 MHz systemen komt er iedere 1/20e seconde een interrupt. Deze interrupt komt van een timer in een VIA. De DOS vangt deze interrupt eerst op om allerhande periodieke zaken af te handelen:

- De head-load tijd.
- De drive-on tijd.
- Copieren van IO65 tijd naar DOS (slechts iedere seconde)
- Printer spooler.

Daarna wordt de interrupt verder in IO65 afgehandeld om het uurwerk bij te houden.

Iedere 1/20e seconde komt dus even de printer spooler aan bod. De spooler krijgt dan de mogelijkheid om 25 karakters naar de printer te sturen. Voordat er een karakter verstuurd wordt, wordt eerst de status van de printer uitgelezen. Als de status ready is oftewel er is een acknowledge van de printer op het laatst verzonden karakter geweest, dan wordt pas het nieuwe karakter verstuurd. Omdat dit allemaal in een interrupt routine gedaan wordt mag er niet eeuwig op de ready van de printer gewacht worden anders ligt het hele systeem stil. In de standaard DOS configuratie test de spooler 10 keer of de printer ready is. Is dat niet het geval, dan wordt de printer spooler verlaten en worden er tijdens die interrupt periode geen pogingen meer gedaan om karakters naar de printer te sturen. Treft men nu een printer die na het ontvangen van een karakter niet binnen die 10 testen een acknowledge terug stuurt, dan wordt er per 1/20e seconde slechts 1 in plaats van 25 karakters verstuurd. Het duurt dan dus 4 seconden voordat er een regel van 80 karakters naar de printer gestuurd is. Dit kan echter met een snelheid van 0.16 seconden. Voor dat soort printers moet dus iets vaker (langer) op de ready getest worden. De variabele die in de DOS op 10 staat moet dan bijvoorbeeld op 20 gezet worden. De minimum waarde kan experimenteel vastgesteld worden. Aangezien er in de DOS een load Y immediate staat moet die plaats eerst opgezocht worden voordat er geexperimenteerd kan worden. Dit opzoeken is noodzakelijk omdat men verschillende DOS versies geinstalleerd kan hebben met het config programma zodat het te wijzigen adres niet bij iedereen op dezelfde plaats staat. Om het adres op te zoeken gaat men naar de monitor met het commando MON. Dan wordt er gezocht naar de volgende byte volgorde: A0,0A,88. Dat wordt gedaan met:

```
w cc00,cfff,/a0,0a,88/ <return>
```

De monitor komt dan terug met een adres waar de LDY staat. Een adres verder staat de 0A (decimaal 10). Dat adres is bijvoorbeeld $CD9A. Hier kan dan $14 (decimaal 20) in gezet worden met:

```
@ CD9B <return>
14 <return>
Q <return>
```

Men is nu terug in de DOS en er kan een print commando gegeven worden. Waarschijnlijk is het probleem dan al opgelost. Is dat nog niet het geval dan kan de waarde verder verhoogd worden. Ga echter niet te hoog daar anders de interrupt routine veel te lang zou kunnen gaan duren. Is een goede waarde gevonden, bv. $14 dan moet dit permanent in de DOS gezet worden. Om echter de standaard DOS niet aan te tasten wordt de 'patch' in de file login.com gezet. Deze file wordt namelijk altijd bij opstarten een keer doorlopen. Met de editor kan de volgende regel worden opgenomen in de file:

```
memfill cd9b,cd9b,14
```

Bij opstarten wordt dan op adres $cd9b de waarde $14 gezet.

Met deze oplossing was het probleem van de trage printer uit de wereld geholpen.

Software

Uiteraard staat de software ontwikkeling op het DOS65 front ook niet stil. Op dit moment worden er bij een aantal fanatieke DOS65 gebruikers modem programma's ontwikkeld. Dit is een zeer goede zaak zeker als we het artikel hebben gelezen over modems en bulletin boards in deze 6502 Kenner. Wat mij echter even van het hart moet is het volgende. Het valt mij op dat er binnen onze club meer DOS65 programmatuur ten tonele verschijnt dan ons bekend is. Daar is uiteraard niets op aan te merken. Alleen had die programmatuur er veel DOS65-vriendelijker uit kunnen zien. Ik denk bijvoorbeeld aan de 'help-tool' die bij ieder 'officieel' DOS65 programma (commando) zit ingebakken. Verder mis ik bij een heleboel van dat soort programma's het handige gebruik van de DOS65 entries. Verder blijkt dat lang niet al die programma's even betrouwbaar werken. Deze niet door de officiele DOS65 manier uitgegeven software (via de DOS65 coordinator) wordt niet door ons gesupport. Dat wil zeggen dat problemen die door die software ontstaan niet door ons onderzocht (hoeven te) worden. Als nu iemand een stuk software heeft geschreven en denkt dat dit goed bruikbaar is voor alle DOS65 gebruikers laat hij dat dan aan de DOS65 coordinator weten. Deze man bekijkt het stuk software op compatibiliteit met de rest van de DOS65 commando's en test het programma op fouten. Ook wordt op die manier de distributie en de bekendheid verzekerd. Ook als men van plan is iets 'groots' te gaan maken is het vaak raadzaam de DOS65 coordinator hiervan op de hoogte te stellen zodat niet twee keer het wiel hoeft te worden uitgevonden, mensen met elkaar in contact gebracht kunnen worden en van coordinators zijde eventueel ideeen meegenomen kunnen worden om bv. efficienter met de DOS te kunnen omspringen.

Aansluitend op de vorige alinea kan ik mededelen dat er ook al driftig aan DOS versie 3 gewerkt wordt. Deze versie brengt bepaalde restricties met zich mee voor wat betreft het ontwikkelen van de programmatuur. DOS versie 3 werkt namelijk met een multi-tasking omgeving, hetgeen inhoud dat er 8 taken (processen) tegelijkertijd kunnen lopen in het systeem. Het is dan mogelijk een file te compileren en te assembleren terwijl u zelf een andere file aan het editen bent. Als dan een task opgestart wordt moet aan de sceduler meegedeeld worden van welke stukken geheugen die taak gebruik maakt. Een andere taak moet daar dan vanaf blijven. Hier moet rekening mee gehouden worden als ook maar het kleinste programmaatje geschreven wordt. Toch klinkt het allemaal erger dan het eigenlijk is, maar over de werking en het gebruik van DOS versie 3 wordt nog voldoende gepubliceerd. Voor DOS versie 3 het wel noodzakelijk dat een virtual disk kaart in het systeem aanwezig is.

Manuals

C compiler documentatie.

Vanaf 1 november is er een nederlandstalige manual over de DOS65 C-compiler beschikbaar. Hierin staat niet wat de taal C is en hoe deze gebruikt moet worden maar een manual voor de specifieke eigenschappen van de DOS65 C-compiler. De in deze C-compiler geimplementeerde functies staan hierin beschreven evenals de beperkingen en verschillen met andere compilers. Tevens worden enkele functies van voorbeelden voorzien. Dit is een zeer handige (zo niet onmisbare) manual voor mensen die iets met de DOS65 C-compiler willen gaan doen. Ook uitgebreide voorbeelden en werkende programma's staan in deze manual. De programmeertaal C wordt wel eens een processor onafhankelijke machinetaal genoemd. De manual is voor F35,= exclusief F3,= verzendkosten verkrijgbaar bij het officiele DOS65 distributie-adres. Bij dit pakket hoort een diskette met daarop de C-compiler met al zijn libraries, een speciale uitvoering van de bekende AS assembler (voor snellere verwerking van library files) en diverse C-programma's die gebruikt en aangepast kunnen worden.

Hardware

- Eprommer.

Er zijn momenteel al twee feilloos werkende DOS65 eprommers. Het eprom programmeer pakket bestaande uit een eurokaart, een kleiner printje voor de socket en een stuk software wordt binnenkort aangeboden. Op dit moment worden de printen ontworpen. De software bestaande uit twee delen is al klaar. Het eerste stuk software is het testprogramma. Hiermee kan de complete hardware getest worden. Het is mogelijk alle pinnen van de eprom voet afzonderlijk aan te sturen en te meten. Het tweede stuk software is de operationele software van waaruit de eproms geprogrammeerd kunnen worden. Er zijn drie programmeer algorithmes aanwezig. Het 50 milliseconden, het Inteligent en het Quick pulse algorithme. Verder zijn diverse programmeerspanningen (25, 21 en 12.5 V) te kiezen evenals een 6 V voedingsspanning voor de Quick pulse en inteligent algorithmes. Als een epromtype wordt geselecteerd, wordt automatisch het algorithme, de programmeerspanning en de voedingsspanning ingesteld. De volgende types eproms zijn te programmeren:

- 2716	25 V	50 ms.
- 2732	25 V	50 ms.
- 2732A	21 V	50 ms.
- 2764	21 V	50 ms.
- 2764	21 V	Inteligent.
- 2764A	12.5 V	Inteligent.
- 2764A	12.5 V	Quick pulse.
- 27128	21 V	50 ms.
- 27128	21 V	Inteligent.
- 27128A	12.5 V	Inteligent.
- 27128A	12.5 V	Quick pulse.
- 27256	21 V	Inteligent.
- 27256A	12.5 V	Inteligent.
- 27256A	12.5 V	Quick pulse.
- 27512	12.5 V	Inteligent.
- 27512	12.5 V	Quick pulse.
- 2532	25 V	50 ms.
- 2532A	21 V	50 ms.
- 2564	25 V	50 ms.
- 2564A	21 V	50 ms.

N.B. Hierboven is inteligent opzettelijk met een '1' geschreven omdat het gebruikte algorithme een handelsmerk is van INTEL.

Zodra de printen van deze eprommer leverbaar zijn zal dit zo snel mogelijk bekend worden gemaakt in de 6502 Kenner. Naast de printen en de software wordt er uiteraard een manual bijgeleverd. Deze manual is ook in het engels beschikbaar.

- 1 Mbyte dram kaart.

De 1 Megabyte dynamische ram kaart die het mogelijk maakt een virtual disk te installeren op het DOS65 systeem is voor wat betreft de hardware ontwikkeling klaar. Het grote probleem is echter het ontwikkelen van een betrouwbare print voor dit ontwerp. Het ontwerp bestaat nu uit 50 IC's en die zijn zeer moeilijk op een print met kruisingsvrije bedrading te rangschikken. Hieraan wordt echter hard gewerkt maar een leverdatum is nog niet te voorspellen. De software voor het DOS65 systeem hoeft niet aangepast te worden als niet meer dan 512 KByte op de print gemonteerd wordt. Voor een 1 MB uitvoering moet er een nieuwe DOS (file BOOT) op de systeemschijf gezet worden. De mogelijkheden van een virtual disk zijn te veel en te groot om zo een twee drie te beschrijven. Momenteel gebruik ik mijn virtual disk (1 MB op een proef-print) als systeem disk. Bij het opstarten wordt dan de complete systeem disk van schijf naar virtual disk gecopieerd. Het voordeel is naderhand dat alle commando's die normaal van schijf moeten worden gelezen direct beschikbaar zijn. Verder zijn er twee drives leeg zodat het copieren van schijven zeer gemakkelijk gaat. Tevens gebruik ik een subdirectory van de virtual disk als gebruikers directory waardoor het compileren en assembleren van grote files veel sneller gaat. De 1 MB dram kaart is zo ontworpen dat de kaart ook op 2 MHz kan werken. U zult merken dat dan het hele systeem flitsend snel werkt. Daar kan geen IBM PC (clone) tegenop!!

```
EC-65   18:24:38 - 19/07/87

0005:                          TTL   EPROM-DISC SOFTWARE
0010: 3590                     ORG   $3590
0015:
0020:                  ********************************
0025:                  *   LOADER FOR E-PROM DISC.    *
0030:                  *     FOR OCTOPUS EC-65.       *
0035:                  *          21.01.87            *
0040:                  *     PETER LINDSTROEM         *
0045:                  *     SOLHAVEN 8               *
0050:                  *     DK 2990 NIVAA, DENMARK   *
0055:                  ********************************
0060:
0065:                  Version 1.2    14.02.1987
0070:                  This version intended for OCTOPUS / EC65.
0075:                  After idea of Mr. Andrew Gregory, England
0080:                  published in De Kenner No. 47
0085:
0090:                  For call from BASIC by:
0095:                  POKE232,X:POKE574,144:POKE575,53:Y=USR(Y)
0100:                  Where X=filenumber.
0105:
0110:                  Call from other programs by:
0115:                  LDAIM FILENUMBER
0120:                  STA $EB
0125:                  JSR START   (start program on $3590)
0130:
0135:                  DIRECTORY FORMAT:-
0140:
0145:                  byte 0  file ID number
0150:                  byte 1  eprom address   low
0155:                  byte 2  eprom address   high
0160:                  byte 3  load address    low
0165:                  byte 4  load address    high
0170:                  byte 5  length          low
0175:                  byte 6  length          high
0180:                  byte 7  exequte address low
0185:                  byte 8  exequte address high
0190:
0195:                  Up to 28 files are allowed. The directory
0200:                  fills first page of the "disc" eprom.
0205:                  If you don't want the file to be exequted,
0210:                  just write return address instead of exe-
0215:                  qute address in the directory. For example
0220:                  $FB11= SAMMON
0225:
0230:                  *** MONITOR ROUTINES ***
0235:
0240: F32F             RESET   EQU   $F32F   clear screen
0245: 2D73             STROUT  EQU   $2D73   print string until etx=$00
0250:
0255:                  *** EPROM DISC ADDRESSES ***
0260:
0265: E190             DRA     EQU   $E190   6821 registers
0270: E191             CRA     EQU   $E191
0275: E192             DRB     EQU   $E192
0280: E193             CRB     EQU   $E193
0285:
0290: E194             EPROM   EQU   $E194
0295:
0300:                  *** TEMPORARY ADDRESSES ***
0305:
0310: 00E0             EPADL   EQU   $E0     points to "disc" byte
0315: 00E1             EPADH   EQU   EPADL   +01
0320: 00E2             MEMADL  EQU   EPADL   +02 points to RAM byte
0325: 00E3             MEMADH  EQU   EPADL   +03
0330: 00E4             LENL    EQU   EPADL   +04 length of file
0335: 00E5             LENH    EQU   EPADL   +05
0340: 00E6             XQADL   EQU   EPADL   +06 exequte addr. lo
0345: 00E8             FILEID  EQU   EPADL   +08 hold the file ID number
0350: 00E9             NFIFLG  EQU   EPADL   +09 'NOFILE' flag
0355:
0360:                  *** MAIN PROGRAM ***
0365:
```

```
EC-65   18:24:39 - 19/07/87

0370: 3590 20 2F F3  START   JSR   RESET   clear screen
0375: 3593 A9 00             LDAIM $00
0380: 3595 85 E9             STA   NFIFLG  reset 'no-file flag'
0385: 3597 20 B6 35          JSR   LOAD    load the disc-file in memory
0390: 359A A5 E9             LDA   NFIFLG
0395: 359C F0 01             BEQ   STARTA
0400: 359E 60                RTS
0405:
0410: 359F 6C E6 00  STARTA  JMPI  XQADL   jump to program start
0415:
0420: 35A2 20 73 2D  NOFILE  JSR   STROUT
0425: 35A5 0D 0A             HEX   0D0A
0430: 35A7 4E 4F 20          ASC   NO FILE!
      35AA 46 49 4C
      35AD 45 21 20
0435: 35B0 0D 0A 00          HEX   0D0A00
0440: 35B3 C6 E9             DEC   NFIFLG  set no-file flag
0445: 35B5 60                RTS
0450:
0455:                  *** SUBROUTINES ***
0460:
0465: 35B6 A5 E8     LOAD    LDA   FILEID  'FILEID' must contain file no.
0470:                                      before calling.
0475: 35B8 48                PHA           save entry
0480:
0485:                  initialise PIA 6821
0490:
0495: 35B9 A9 04             LDAIM $04
0500: 35BB A0 FF             LDYIM $FF
0505: 35BD A2 00             LDXIM $00
0510: 35BF 8E 91 E1          STX   CRA     access direction reg. A
0515: 35C2 8E 93 E1          STX   CRB     access direction reg. B
0520: 35C5 8C 90 E1          STY   DRA     A is output
0525: 35C8 8C 92 E1          STY   DRB     B is output
0530: 35CB 8D 91 E1          STA   CRA     access data reg. A
0535: 35CE 8D 93 E1          STA   CRB     access data reg. B
0540:
0545: 35D1 8E 90 E1          STX   DRA     'disc' address = $0000
0550: 35D4 8E 92 E1          STX   DRB
0555:
0560: 35D7 68                PLA           retrieve entry
0565: 35D8 A8                TAY
0570:
0575:                  Search for correct directory entry
0580:
0585: 35D9 CC 94 E1  SEARCH  CPY   EPROM   is this it?
0590: 35DC F0 0F             BEQ   YES     branch if yes
0595: 35DE 30 02             BMI   NO
0600: 35E0 90 C0             BCC   NOFILE  branch if illegal id number
0605: 35E2 A9 09     NO      LDAIM $09     set up next address
0610: 35E4 18                CLC
0615: 35E5 6D 90 E1          ADC   DRA
0620: 35E8 8D 90 E1          STA   DRA
0625: 35EB 90 EC             BCC   SEARCH  branch if more entries
0630:
0635:
0640:                  Get eigth parameter bytes from
0645:                  directory and copy to RAM.
0650:
0655: 35ED 20 20 36  YES     JSR   PARAM
0660: 35F0 E0 08             CPXIM $08     more?
0665: 35F2 D0 F9             BNE   YES     branch if yes
0670:
0675:                  Transfer 'disc' start address to PIA
0680:
0685: 35F4 A5 E0             LDA   EPADL   low byte
0690: 35F6 8D 90 E1          STA   DRA
0695: 35F9 A5 E1             LDA   EPADH   high byte
0700: 35FB 8D 92 E1          STA   DRB
0705:
0710:                  Copy file from 'disc' to RAM
0715:
0720: 35FE E6 E4             INC   LENL
```

```
EC-65   18:24:40 - 19/07/87

3600 E6 E5              INC   LENH    adjust length-pointer
3602 A0 00              LDYIM $00
3604 AD 94 E1  MORE     LDA   EPROM   get byte
3607 91 E2              STAIY MEMADL  store in RAM

3609 EE 90 E1           INC   DRA     next 'disc' address
360C D0 03              BNE   POINT
360E EE 92 E1           INC   DRB
3611 C8        POINT    INY           increment ram pointer
3612 D0 02              BNE   LENGTH
3614 E6 E3              INC   MEMADH
3616 C6 E4     LENGTH   DEC   LENL    decrement length
3618 D0 EA              BNE   MORE
361A C6 E5              DEC   LENH
361C D0 E6              BNE   MORE    branch if more

361E 18                 CLC           clear carry as ok
361F 60                 RTS

3620 EE 90 E1  PARAM    INC   DRA     get next entry
3623 AD 94 E1           LDA   EPROM
3626 95 E0              STAX  EPADL
3628 E8                 INX
3629 60                 RTS



))) Error in 0000 statement(s)

))) Op-Code: $8000 - $8099 / $3590 - $3629 / 0154 Bytes / 01 Page(s)

))) Assembled by ASS114 / 3.4
```

<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

```
EPROM-DISC DIRECTORY

EC-65   18:14:30 - 19/07/87

                        TTL   EPROM-DISC DIRECTORY
CDF0                    ORG   $CDF0

                        ********************************
                        *         PRINT-OUT OF         *
                        *  DIRECTORY FOR E-PROM DISC.  *
                        *      FOR OCTOPUS EC-65.      *
                        *           22.06.87           *
                        *              by:             *
                        *      PETER LINDSTROEM        *
                        *      SOLHAVEN 8,             *
                        *      DK 2990  NIVAA,         *
                        *      D E N M A R K.          *
                        ********************************

                        This version intended for OCTOPUS / EC65.


                        DIRECTORY FORMAT:-

                        byte 0  file ID number
                        byte 1  eprom address    low
                        byte 2  eprom address    high
                        byte 3  load address     low
                        byte 4  load address     high
                        byte 5  length           low
                        byte 6  length           high
                        byte 7  exequte address  low
                        byte 8  exequte address  high

```

```
0155:                 Up to 28 files are allowed. The directory
0160:                 fills first page of the "disc" eprom.
0165:                 The program name must fill the first
0170:                 16 locations of the data file, if not,
0175:                 fill with spaces.
0180:
0185:                 *** MONITOR ROUTINES ***
0190:
0195: F32F            RESET   EQU   $F32F   clear screen
0200: 2D73            STROUT  EQU   $2D73   print string until etx=$00
0205: FA57            PRBYT   EQU   $FA57   convert byte to ASCII and print
0210: F7E2            PRCHA   EQU   $F7E2
0215: F9CB            CRLF    EQU   $F9CB
0220: F902            RECCHA  EQU   $F902   get keyboard
0225:
0230:                 *** EPROM DISC ADDRESSES ***
0235:
0240: E190            DRA     EQU   $E190   6821 registers
0245: E191            CRA     EQU   $E191
0250: E192            DRB     EQU   $E192
0255: E193            CRB     EQU   $E193
0260:
0265: E194            EPROM   EQU   $E194
0270:
0275:                 *** TEMPORARY ADDRESSES ***
0280:
0285: 2322            OUTDEV  EQU   $2322
0290: E7C8            PARFLG  EQU   $E7C8   printer flag
0295: CFE0            EPADL   EQU   $CFE0   points to "disc" byte
0300: CFE1            EPADH   EQU   EPADL   +01
0305: CFE2            MEMADL  EQU   EPADL   +02 points to RAM byte
0310: CFE3            MEMADH  EQU   EPADL   +03
0315: CFE4            LENL    EQU   EPADL   +04 length of file
0320: CFE5            LENH    EQU   EPADL   +05
0325: CFE6            XQADL   EQU   EPADL   +06 execute addr. lo
0330: CFE8            PROGNAM EQU   EPADL   +08 buffer for prog. name
0335: CFF8            ETX     EQU   EPADL   +18 end of text
0340: CFF9            POINTL  EQU   EPADL   +19 address pointer lo
0345: CFFA            POINTH  EQU   EPADL   +1A address pointer hi
0350: CFFB            TEMP    EQU   EPADL   +1B counter
0355: CFFC            FILEID  EQU . EPADL   +1C hold the file ID no.
0360:
0365:                 *** MAIN PROGRAM ***
0370:
0375: CDF0 20 2F F3   START   JSR   RESET   clear screen
0380: CDF3 20 73 2D           JSR   STROUT  print heading
0385: CDF6 0A 0A 20           HEX   0A0A20202020202020202020202020
      CDF9 20 20 20
      CDFC 20 20 20
      CDFF 20 20 20
      CE02 20 20
0390: CE04 45 20 2D           ASC   E - P R O M   D I R E C T O R Y:
      CE07 20 50 20
      CE0A 52 20 4F
      CE0D 20 4D 20
      CE10 20 20 44
      CE13 20 49 20
      CE16 52 20 45
      CE19 20 43 20
      CE1C 54 20 4F
      CE1F 20 52 20
      CE22 59 3A
0395: CE24 20 20 20           ASC             PRINTER (Y/N) ?
      CE27 20 20 20
      CE2A 20 20 20
      CE2D 50 52 49
      CE30 4E 54 45
      CE33 52 20 28
      CE36 59 2F 4E
      CE39 29 20 3F
      CE3C 20
0400: CE3D 00                 HEX   00
0405:
0410: CE3E 20 02 F9           JSR   RECCHA  y/n ?
0415: CE41 C9 59              CMPIM 'Y
0420: CE43 F0 07              BEQ   FLAG
```

```
0425: CE45 C9 4E          CMPIM 'N
0430: CE47 F0 0D          BEQ   CONT
0435: CE49 4C F0 CD       JMP   START
0440: CE4C A9 01    FLAG  LDAIM $01     set printer flag
0445: CE4E 8D C8 E7       STA   PARFLG
0450: CE51 A9 09          LDAIM $09
0455: CE53 8D 22 23       STA   OUTDEV
0460: CE56 20 73 2D CONT  JSR   STROUT
0465:
0470: CE59 0D 0A 0A       HEX   0D0A0A20202020202020
      CE5C 20 20 20
      CE5F 20 20 20
0475: CE62 46 49 4C       ASC   FILE ID# LOAD ADDR. START ADDR.   PROGRAM NAME
      CE65 45 20 49
      CE68 44 23 20
      CE6B 4C 4F 41
      CE6E 44 20 41
      CE71 44 44 52
      CE74 2E 20 53
      CE77 54 41 52
      CE7A 54 20 41
      CE7D 44 44 52
      CE80 2E 20 20
      CE83 20 50 52
      CE86 4F 47 52
      CE89 41 4D 20
      CE8C 4E 41 4D
      CE8F 45 20 20
      CE92 20 20
0480: CE94 45 2D 50       ASC   E-PROM ADDR.
      CE97 52 4F 4D
      CE9A 20 41 44
      CE9D 44 52 2E
0485: CEA0 0D 0A 0A       HEX   0D0A0A00
      CEA3 00
0490:
0495:               *** initialise PIA 6821 ***
0500:
0505: CEA4 A9 04          LDAIM $04
0510: CEA6 A0 FF          LDYIM $FF
0515: CEA8 A2 00          LDXIM $00
0520: CEAA 8E 91 E1       STX   CRA     access direction reg. A
0525: CEAD 8E 93 E1       STX   CRB     access direction reg. B
0530: CEB0 8C 90 E1       STY   DRA     A is output
0535: CEB3 8C 92 E1       STY   DRB     B is output
0540: CEB6 8D 91 E1       STA   CRA     access data reg. A
0545: CEB9 8D 93 E1       STA   CRB     access data reg. B
0550:
0555: CEBC 8E 90 E1       STX   DRA     'disc' address = $0000
0560: CEBF 8E 92 E1       STX   DRB
0565: CEC2 8E FB CF       STX   TEMP    reset counter
0570: CEC5 8E F8 CF       STX   ETX     set ETX =$00
0575:
0580:               *** Fetch directory ***
0585:
0590: CEC8 A9 00    FETCH LDAIM $00
0595: CECA 8D 92 E1       STA   DRB
0600: CECD AD FB CF       LDA   TEMP
0605: CED0 8D 90 E1       STA   DRA
0610: CED3 20 13 CF       JSR   GET8
0615: CED6 30 0B          BMI   FINAL   no more entry's
0620: CED8 A9 09          LDAIM $09
0625: CEDA 18             CLC
0630: CEDB 6D FB CF       ADC   TEMP    calculate next entry
0635: CEDE 8D FB CF       STA   TEMP
0640: CEE1 D0 E5          BNE   FETCH   branch always
0645: CEE3 A9 00    FINAL LDAIM $00
0650: CEE5 8D C8 E7       STA   PARFLG  reset printer flag
0655: CEE8 A9 01          LDAIM $01
0660: CEEA 8D 22 23       STA   OUTDEV
0665:
0670: CEED 20 73 2D       JSR   STROUT
0675: CEF0 0D 0A 0A       HEX   0D0A0A2020202020
      CEF3 20 20 20
      CEF6 20 20
0680: CEF8 52 45 41       ASC   READY? (PRESS A KEY)
```

```
      CEFB 44 59 3F
      CEFE 20 28 50
      CF01 52 45 53
      CF04 53 20 41
      CF07 20 4B 45
      CF0A 59 29 20
      CF0D 20
0685: CF0E 00              HEX   00
0690: CF0F 20 02 F9        JSR   RECCHA
0695: CF12 60              RTS
0700:
0705:                 Get eigth parameter bytes from
0710:                 directory and copy to RAM.
0715:
0720: CF13 AD 94 E1  GET8  LDA   EPROM
0725: CF16 30 0D           BMI   END     no more data
0730: CF18 8D FC CF        STA   FILEID
0735: CF1B 20 26 CF  YES   JSR   PARAM
0740: CF1E E0 08           CPXIM $08     more?
0745: CF20 D0 F9           BNE   YES     branch if yes
0750: CF22 20 75 CF        JSR   PRIPAR  print-out
0755: CF25 60        END   RTS
0760:
0765: CF26 EE 90 E1  PARAM  INC   DRA     get next entry
0770: CF29 AD 94 E1        LDA   EPROM
0775: CF2C 9D E0 CF        STAX  EPADL
0780: CF2F E8              INX
0785: CF30 60              RTS
0790:
0795:                 *** Print Y x space ***
0800:
0805: CF31 A9 20     YSPACE LDAIM $20
0810: CF33 20 E2 F7  FINISH JSR   PRCHA
0815: CF36 88              DEY
0820: CF37 C0 00           CPYIM $00
0825: CF39 D0 F8           BNE   FINISH
0830: CF3B 60              RTS
0835:
0840:                 *** DECPNT decrement addr. pointer by one ***
0845:
0850: CF3C 38        DECPNT SEC
0855: CF3D AD F9 CF        LDA   POINTL
0860: CF40 E9 01           SBCIM $01
0865: CF42 8D F9 CF        STA   POINTL
0870: CF45 B0 03           BCS   DECPZ
0875: CF47 CE FA CF        DEC   POINTH
0880: CF4A 60        DECPZ RTS
0885:
0890:                 *** GTFINA get file name ***
0895:                 ***    and copy to ram   ***
0900:
0905: CF4B A2 10     GTFINA LDXIM $10
0910: CF4D 20 3C CF  GTFINB JSR   DECPNT
0915: CF50 AD F9 CF        LDA   POINTL
0920: CF53 8D 90 E1        STA   DRA
0925: CF56 AD FA CF        LDA   POINTH
0930: CF59 8D 92 E1        STA   DRB
0935: CF5C CA              DEX
0940: CF5D AD 94 E1        LDA   EPROM
0945: CF60 9D E8 CF        STAX  PROGNAM
0950: CF63 E0 00           CPXIM $00
0955: CF65 D0 E6           BNE   GTFINB
0960: CF67 60              RTS
0965:
0970:                 *** SETPNT set addr. pointer ***
0975:
0980: CF68 AD E0 CF  SETPNT LDA   EPADL
0985: CF6B 8D F9 CF        STA   POINTL
0990: CF6E AD E1 CF        LDA   EPADH
0995: CF71 8D FA CF        STA   POINTH
1000: CF74 60              RTS
1005:
1010:                 *** PRINT PARAMETERS ***
1015:
1020: CF75 A0 09     PRIPAR LDYIM $09
1025: CF77 20 31 CF        JSR   YSPACE
```

```
1030: CF7A AD FC CF          LDA   FILEID
1035: CF7D 20 57 FA          JSR   PRBYT   print file #
1040: CF80 A0 06             LDYIM $06
1045: CF82 20 31 CF          JSR   YSPACE
1050: CF85 AD E3 CF          LDA   MEMADH
1055: CF88 20 57 FA          JSR   PRBYT
1060: CF8B AD E2 CF          LDA   MEMADL
1065: CF8E 20 57 FA          JSR   PRBYT   print memory load addr.
1070: CF91 A0 08             LDYIM $08
1075: CF93 20 31 CF          JSR   YSPACE
1080: CF96 AD E7 CF          LDA   XQADL   +01
1085: CF99 20 57 FA          JSR   PRBYT
1090: CF9C AD E6 CF          LDA   XQADL
1095: CF9F 20 57 FA          JSR   PRBYT   print start addr.
1100: CFA2 20 68 CF          JSR   SETPNT
1105: CFA5 20 4B CF          JSR   GTFINA
1110: CFA8 A0 06             LDYIM $06
1115: CFAA 20 31 CF          JSR   YSPACE
1120: CFAD 20 C5 CF          JSR   PRINAME print program-name
1125: CFB0 A0 05             LDYIM $05
1130: CFB2 20 31 CF          JSR   YSPACE
1135: CFB5 AD E1 CF          LDA   EPADH
1140: CFB8 20 57 FA          JSR   PRBYT
1145: CFBB AD E0 CF          LDA   EPADL
1150: CFBE 20 57 FA          JSR   PRBYT   print E-PROM addr.
1155: CFC1 20 CB F9          JSR   CRLF
1160: CFC4 60                RTS
1165:
1170:               *** PRINAME print the program-name ***
1175:
1180: CFC5 A0 00     PRINAME LDYIM $00
1185: CFC7 B9 E8 CF  PRINT   LDAY  PROGNAM
1190: CFCA C9 00             CMPIM $00     end of text?
1195: CFCC F0 06             BEQ   PRIEND
1200: CFCE 20 E2 F7          JSR   PRCHA
1205: CFD1 C8                INY
1210: CFD2 D0 F3             BNE   PRINT
1215: CFD4 60        PRIEND  RTS
1220:

))) Error in 0000 statement(s)

))) Op-Code: $B000 - $B1E4 / $CDF0 - $CFD4 / 0485 Bytes / 02 Page(s)

))) Assembled by ASS114 / 3.4
```

FILE ID#	LOAD ADDR.	START ADDR.	PROGRAM NAME	E-PROM ADDR.
01	C800	C800	COPIER 80-05.87	0110
02	0200	0200	ASS 114 V.3.4	0910
03	2200	0200	WORDPROC. V.3.0	2920
04	BC00	BC00	EDMO V.2.2	4920
05	CC00	CC00	REAL TIME INIT	5B20
06	CE00	CE00	EMPTY FILE $16F	5E40
07	C000	C000	LARGE CHARACTERS	5FC0
08	C800	C800	MORSE DECODER	6740
09	CD00	CD00	EPROM PROGRAMMER	6A90
10	CDF0	CDF0	E-PROM DIRECTORY	6D00
0A	CF00	CF00	SCREEN DUMP	6F00
0B	CE00	CE00	ZERO-PAGE LIST	6F40

```
10 REM ***************     PROGRAMM NUMINT     ***************
20 REM DIESES PROGRAMM DIENT ZUR NUMERISCHEN INTEGRATION
30 REM VON STETIGEN FUNKTIONEN NACH DER SIMPSONSCHEN REGEL
40 REM VEROEFFENTLICHUNG IN DER FUNKSCHAU 1980, HEFT14/83
50 REM AUTOR: DIETER SMODE 13-12-1978
60 REM EINGABE VON FUNKTIONSWERTEN NACH "FORMEL-INPUT
70 REM BEIM AIM-65 IN DER FUNKSCHAU 25/1980/97
80 REM
90 REM AUF DEN EC-65 ANGEPASST DURCH W.TIETSCH, APRIL 1986
100 REM ********************************************************
110 :
120 :
130 T1=PEEK(2888):T2=PEEK(8722):POKE2888,0:POKE8722,0
140 DISK!"GO F32F":REM CLEAR SCREEN AND HOME CURSOR
150 DISK!"IO ,01:REM SCREEN ONLY OUTPUT
160 REM ALLOW <CR> ONLY INPUT --> ""
170 FORI=1TO5:PRINT:NEXTI
180 PRINTTAB(20)"*******************************************"
190 PRINTTAB(20)"*                                         *"
200 PRINTTAB(20)"*            Dieses  Programm             *"
210 PRINTTAB(20)"*                                         *"
220 PRINTTAB(20)"*    dient zur numerischen Integration    *"
230 PRINTTAB(20)"*                 von                     *"
240 PRINTTAB(20)"*         stetigen Funktionen             *"
250 PRINTTAB(20)"*              nach der                   *"
260 PRINTTAB(20)"*                                         *"
270 PRINTTAB(20)"*  S i m p s o n ' s c h e n   R e g e l  *"
280 PRINTTAB(20)"*                                         *"
290 PRINTTAB(20)"*******************************************"
300 PRINT:PRINT:PRINTTAB(20)"Bitte warten, ich suche DEFFNY(X)"
310 DIMA$(23):RESTORE:PI=3.141592654
320 ST=0:IFPEEK(8999)=58THENST=14975:REM DOS V3.3
330 SD=0:IFPEEK(8999)=50THENST=12927:REM DOS V3.2
340 IF ST=0 THEN2020
350 PRINTTAB(20);
360 FORI=0TO23:READA$(I):NEXTI
370 FORI=STTO(ST+PEEK(ST-2)*2048)
380 IF(PEEK(I)=149)AND(PEEK(I+4)=88)AND(PEEK(I+7)=58)THENSD=I
390 PRINT".";:IFPOS(X)>52THENPRINTCHR$(13);:PRINTTAB(20);
400 IFSD<>0THEN420
410 NEXTI:PRINT:GOTO2000
420 SD=SD+7:SA=SD:PRINTCHR$(13);:PRINTCHR$(27);"5";
430 PRINTCHR$(27);"4";:PRINT
440 SD=SA:REM FORMEL-EINGABE
450 PRINTTAB(20)"Die zu integrierende Funktion bitte in"
460 PRINTTAB(20)"folgender Form mit maximal 55 Zeichen"
470 PRINTTAB(20)"nach dem '?' eingeben ( pi ist installiert )!"
480 PRINT:PRINTTAB(20)"Beispiel --> FNY(X)=X^2+(3*X)^(-2)"
490 PRINTTAB(20)"Eine Eingabe von <CR> beendet das Programm"
500 PRINT:PRINTTAB(14);:INPUT"FNY(X)=";W$
510 W=LEN(W$):IFW$=""THEN2060
520 IFW<56THEN570
530 PRINTTAB(20)"Ausdruck >55 Zeichen. Bitte neu eingeben !"
540 PRINTCHR$(13);:FORI=1TO2000:NEXTI
550 FORI=1TO11:PRINTCHR$(27);"5";:NEXTI
560 PRINTCHR$(27);"4";:PRINT:GOTO440
570 FORI=SDTOSD+55:POKEI,58:NEXTI:REM FILL WITH ":"
580 REM SEARCHING FOR THE MATH EXPRESSIONS AND SUBSTITUTION BY TOKENS
590 FORI=1TOW
600 W%=ASC(MID$(W$,I,1))
610 FORJ=0TO23
620 IFMID$(W$,I,LEN(A$(J)))=A$(J)THENW%=J+163
630 NEXTJ
640 IFW%>172ANDW%<187THENI=I+2
650 POKESD,W%:SD=SD+1:
660 NEXTI
670 :
680 GOTO 999
690 :
700 DATA +,-,*,/,^,#,#,#,#,#,SGN,INT,ABS,USR,#,#
710 DATA SQR,RND,LOG,EXP,COS,SIN,TAN,ATN
720 :
888 REM THE NEXT STATEMENT WILL BE SELF ADJUSTING
999 DEFFNY(X)=:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
1000 Z=0:DISK!"GO F32F:PRINT:REM CLEAR SCREEN
1010 PRINTTAB(20)"Berechnung mit halber Schrittweite und"
```

```
1020 PRINTTAB(20)"und anschliessender Fehlerabschaetzung"
1030 PRINTTAB(20);:INPUT"(J/N) ";W$
1040 IFW$<>"J"ANDW$<>"N"THENPRINTCHR$(27);"5";CHR$(27);"5":GOTO1030
1050 IFW$="N"THEN1080
1060 Z=1
1080 PRINT:PRINTTAB(20);:INPUT"Untere Grenze = ";U
1090 PRINTTAB(20);:INPUT"Obere  Grenze = ";O
1100 PRINTTAB(20);:INPUT"Schrittzahl    = ";S
1110 K=SAND1:IFK=0ANDS>=2THEN1160
1120 PRINT"          ------->  Die Schrittweite muss eine gerade ";
1130 PRINT"ganze Zahl sein !";
1140 FORI=1TO2000:NEXTI
1150 PRINTCHR$(13);:PRINTCHR$(27);"5";CHR$(27);"4";
1155 PRINTCHR$(27);"5";:PRINT:GOTO1100
1160 H=(O-U)/S:PRINT:PRINTTAB(20)"Die Schrittweite h = ";H
1170 F=FNY(U)+FNY(O)+4*FNY(U+H)
1180 IFS=2THEN1210
1190 FORK=1TOS/2-1:F1=FNY(U+(2*K+1)*H):F2=FNY(U+2*K*H)
1200 F=F+4*F1+2*F2:NEXTK
1210 F=F*H/3:PRINT:PRINTTAB(20)"Flaecheninhalt   F = ";F
1220 PRINT:PRINTTAB(20);:FORI=1TO40:PRINT"*";:NEXTI:PRINT
1230 IFZ=0THENPRINT:GOSUB1800:IFFL=0THEN440:REM NEW FUNCTION
1235 IFZ=0THEN1000
1240 IFZ=2THEN1260
1250 G=F:H=H/2:S=2*S:Z=2:GOTO1160
1260 F=(16*F-G)/15
1270 PRINT:PRINTTAB(20)"Flaeche nach Fehlersch. = ";F:Z=0
1280 PRINT:PRINTTAB(20);:FORI=1TO40:PRINT"*";:NEXTI:PRINT
1290 PRINT:PRINTTAB(20)"Erneute Berechnung mit den gleichen"
1300 PRINTTAB(20)"Grenzen aber anderer Schrittweite (J/N) "
1305 PRINTTAB(20)"Neue Funktion eingeben --> <CR> ";
1310 INPUTW$:IFW$="N"THEN1000
1315 IFW$=""THENDISK!"GO F32F":GOTO 440
1320 IFW$="J"THEN1340
1330 PRINTCHR$(13):PRINTCHR$(27);"5";CHR$(27);"5";:PRINT
1335 GOTO1300
1340 PRINTTAB(20);:INPUT"Schrittzahl (gerade) = ";S
1350 PRINTTAB(20);:INPUT"Fehlerschaetzung (J/N) ";W$
1360 IFW$="J"THENZ=1
1370 PRINT:GOTO1160
1800 :
1810 PRINTTAB(20);:INPUT"Weiter (J-CR) : Quit --> <CR> ";W$
1820 PRINTCHR$(6);CHR$(27);"8";:PRINT
1830 IF W$=""THEN FL=0:RETURN
1840 FL=1:RETURN
1998 :
1999 :
2000 DISK!"GO F32F":PRINTTAB(20)"PROG-ERR  --> NO DEFFN !"
2010 GOTO 2040
2020 DISK!"GO F32F":PRINTTAB(20)"There are buffers installed"
2030 PRINTTAB(20)"or an incompatible version of DOS/BASIC !"
2040 PRINTTAB(20)"PROGRAM WILL BE TERMINATED SOON 1!"
2050 FORI=1TO4000:NEXTI
2060 DISK!"GO F32F":POKE2888,T1:POKE8722,T2:DISK!"IO ,01":END
```

```
0001            **************************************
0002            LANGUAGE STUDY HELP PROGRAM
0003            **************************************
0004
0005
0006
0007            A program originally written for the KIM
0008            By G. Verkooy, Holland
0009            See Kim-kenner Nr 13
0010
0011            Adapted for the Junior
0012            By W. Van Pelt, Holland
0013
0014            Adapted for the Octopus
0015            By M. Lachaert, Belgium
0016
0017            Version 3.0 - 19.07.1987
0018            Last review - 21.07.1987
0019
0020
0021
0022            Type in the word of which you want a translation,
0023            followed by "=", and the program will search in
0024            its memory for the translation. If this is not a-
0025            vailable, the program will answer with "?". You
0026            can now type the translation yourself, followed
0027            by an monkeytail (@). If you do not wish so, type
0028            the "delete" key ($7F).
0029            If you want to remove information, then call it
0030            up, followed by a control-C.
0031
0032
0033
0034            **************************************
0035
0036 0000                .TIT 'LANGUAGE STUDY HELP PROGRAM V3.0'
0037
0038            SYSTEM VARIABLES
0039
0040 0000            WIJZER   = $10          ; ADDRESS POINTER IN FILE
0041 0000            SAVEY    = $12          ; SCRIBBLING ADDRESS Y-REGISTER
0042 0000            BUFFER   = $100         ; FIRST ADDRESS TEXT BUFFER
0043 0000            START    = $3FFF        ; START OF DATA FILE - 1
0044 0000            EIND     = $D000        ; END OF DATA FILE + 1
0045
0046            DOS ROUTINES
0047
0048 0000            INECHO   = $2340        ; INPUT A CHAR FROM A.I.D.
0049 0000            PRINT    = $2343        ; PRINT CHARACTER ON A.O.D.
0050 0000            CRLF     = $2D6A        ; PRINT CR/LF ON A.O.D.
0051
0052            SPECIAL CHARACTERS
0053
0054 0000            CTLC     = $03          ; CONTROL C = BREAK
0055 0000            BS       = $08          ; BACKSPACE
0056 0000            EOW      = $40          ; WORD DELIMITER
0057 0000            DEL      = $7F          ; DELETE (RUBOUT)
0058
0059
0060 0000                     .OPT SYM
0061
0062
0063 0000                     * = $0200
0064
0065            START OF PROGRAM
0066
0067 0200 206A2D TAALST  JSR CRLF        ; CR/LF
0068 0203 A940           LDA #EOW        ; SET END OF FILE
0069 0205 8DFFCF         STA EIND-1
0070 0208 A200           LDX #$00        ; PRESET X TO READ AND...
0071
0072            ENTER WORD IN LANGUAGE 1 IN BUFFER
0073
0074 020A 204023 NEXTIN  JSR INECHO      ; ...TO FILL THE BUFFER
0075 020D C97F           CMP #DEL        ; DELETE CHAR?
0076 020F F0EF           BEQ TAALST      ; => YES, START AGAIN
0077 0211 C903           CMP #CTLC       ; CONTROL C?
0078 0213 F056           BEQ JCLEAR      ; => YES, REMOVE DATA
```

```
0079 0215 C908          CMP #BS          ; BACKSPACE?
0080 0217 F049          BEQ BACKSP       ; => YES, DO IT!
0081 0219 9D0001        STA BUFFER,X     ; STORE IN BUFFER
0082 021C E8            INX              ; READY FOR NEXT CHARACTER
0083
0084 021D C93D   BSIN   CMP #'='         ; '='?
0085 021F D0E9          BNE NEXTIN       ; => NO, NEXT CHAR IN
0086
0087          SEARCH IN FILE FOR WORD IN BUFFER
0088
0089 0221 A9FF          LDA #<START      ; POINTER = START OF FILE
0090 0223 8510          STA WIJZER
0091 0225 A93F          LDA #>START
0092 0227 8511          STA WIJZER+1
0093
0094 0229 A000   ZOEK   LDY #$00
0095
0096 022B E610   LDAADW INC WIJZER       ; INCREMENT POINTER
0097 022D D002          BNE SKIP
0098 022F E611          INC WIJZER+1
0099
0100 0231 A9D0   SKIP   LDA #>EIND       ; POINTER ON END?
0101 0233 C511          CMP WIJZER+1
0102 0235 F037          BEQ NIETGV       ; => YES, ITEM NOT FOUND
0103 0237 B110          LDA (WIJZER),Y
0104 0239 C940          CMP #EOW         ; WORD DELIMITER?
0105 023B D0EE          BNE LDAADW       ; => NO, CONTINUE SEARCHING
0106
0107 023D C8     NXTWRD INY
0108 023E B110          LDA (WIJZER),Y   ; COMPARE CHAR IN BUFFER...
0109 0240 D9FF00        CMP BUFFER-1,Y   ; ...TO THE ONE IN BUFFER
0110 0243 D0E4          BNE ZOEK         ; => NOT THE SAME
0111 0245 C93D          CMP #'='         ; '='?
0112 0247 F002          BEQ GEVOND       ; => YES, FOUND!
0113 0249 D0F2          BNE NXTWRD       ; => NO, TEST NEXT WORD
0114
0115 024B E612   GEVOND INC SAVEY        ; INCREMENT SCRIBLING POINTER
0116 024D C8            INY
0117
0118          TRANSPORT WORD IN FILE TO BUFFER
0119
0120 024E B110   HERVUL LDA (WIJZER),Y   ; TRANSPORT CHAR IN FILE...
0121 0250 99FF00        STA BUFFER-1,Y   ; ...TO BUFFER
0122 0253 C940          CMP #EOW         ; WORD DELIMITER?
0123 0255 F00F          BEQ INCSY        ; => YES, BRANCH
0124 0257 8412          STY SAVEY        ; SAVE Y-REG
0125 0259 204323        JSR PRINT        ; PRINT CHAR
0126 025C A412          LDY SAVEY        ; RESTORE Y-REG
0127 025E C8            INY
0128 025F 4C4E02        JMP HERVUL       ; PROCEED WITH NEXT CHAR
0129
0130       DECREMENT SCRIBLING POINTER, DO BACKSPACE
0131
0132 0262 CA     BACKSP DEX
0133 0263 4C1D02        JMP BSIN
0134
0135 0266 E612   INCSY  INC SAVEY        ; INCREMENT SCRIBLING POINTER
0136 0268 4C9802        JMP SCHEEN
0137
0138 026B 4CC302 JCLEAR JMP CLEAR
0139
0140          WORD NOT FOUND, PRINT ? AND GIVE TRANSLATION
0141
0142 026E A93F   NIETGV LDA #'?'
0143 0270 204323        JSR PRINT        ; PRINT '?'
0144
0145          ENTER NEW TRANSLATION IN BUFFER
0146
0147 0273 204023 INVOER JSR INECHO       ; FETCH AND PRINT CHAR
0148 0276 C97F          CMP #DEL         ; DELETE CHAR?
0149 0278 F0B6          BEQ TAALST       ; => YES, START AGAIN
0150 027A 9D0001        STA BUFFER,X
0151 027D C940          CMP #EOW         ; WORD DELIMITER?
0152 027F D003          BNE NOEND        ; => NO, CONTINUE INPUT
0153 0281 4C8802        JMP STORE        ; YES, STORE THE WORD
0154
0155 0284 E8     NOEND  INX              ; INCREMENT OFFSET IN BUFFER
0156 0285 4C7302        JMP INVOER       ; GO FOR NEXT CHAR INPUT
```

```
0157
0158                STORE NEW WORD IN FILE
0159
0160 0288 E8        STORE   INX                 ; INCREMENT OFFSET IN BUFFER
0161 0289 8612              STX SAVEY           ; PUT IN SCRIBLING POINTER
0162 028B A900              LDA #<EIND          ; FILE POINTER = END OF FILE...
0163 028D 18                CLC                 ; ...MINUS WORD LENGHT IN BUFFER
0164 028E E512              SBC SAVEY
0165 0290 8510              STA WIJZER
0166 0292 A9D0              LDA #>EIND
0167 0294 E900              SBC #$00
0168 0296 8511              STA WIJZER+1
0169
0170                MAKE PLACE FOR NEW WORD
0171
0172 0298 A2FF    SCHEEN    LDX #$FF
0173
0174 029A A000    SCHUIF    LDY #$00
0175 029C B110              LDA (WIJZER),Y    ; TRANSPORT CHAR IN POINTER
0176 029E A412              LDY SAVEY         ; ...TO POINTER + WORD LENGHT
0177 02A0 9110              STA (WIJZER),Y
0178 02A2 C610              DEC WIJZER        ; DECREMENT POINTER
0179 02A4 E410              CPX WIJZER
0180 02A6 D002              BNE NODEC
0181 02A8 C611              DEC WIJZER+1
0182
0183 02AA A93F    NODEC     LDA #>START       ; BOTTOM REACHED?
0184 02AC C511              CMP WIJZER+1
0185 02AE D0EA              BNE SCHUIF => NO
0186 02B0 A412              LDY SAVEY
0187
0188                INSERT WORD
0189
0190 02B2 B9FFD0  VUL       LDA BUFFER-1,Y    ; FILL OPEN SPACE...
0191 02B5 990040            STA START+1,Y     ; ...WITH WORD IN BUFFER
0192 02B8 88                DEY
0193 02B9 D0F7              BNE VUL           ; => NOT COMPLETE
0194 02BB A940              LDA #EOW          ; PUT WORD DELIMITER
0195 02BD 8D0040            STA START+1
0196 02C0 4C0002            JMP TAALST        ; RESTART
0197
0198                REMOVE WORD FROM FILE
0199
0200 02C3 A000    CLEAR     LDY #$00
0201
0202 02C5 C8      CLOOP     INY
0203 02C6 B90040            LDA START+1,Y
0204 02C9 C940              CMP #EOW          ; WORD DELIMITER?
0205 02CB D0F8              BNE CLOOP         ; => NO
0206 02CD C8                INY
0207 02CE 8412              STY SAVEY         ; SCRIBLING POINTER = Y-REG
0208 02D0 A9FF              LDA #<START       ; SET POINTER ON START
0209 02D2 8510              STA WIJZER
0210 02D4 A93F              LDA #>START
0211 02D6 8511              STA WIJZER+1
0212
0213 02D8 A412    SCHSCH    LDY SAVEY         ; DELETE WORD
0214 02DA B110              LDA (WIJZER),Y
0215 02DC A001              LDY #$01
0216 02DE 9110              STA (WIJZER),Y
0217 02E0 E610              INC WIJZER        ; INCREMENT POINTER
0218 02E2 D002              BNE NOINC
0219 02E4 E611              INC WIJZER+1
0220
0221 02E6 A9D0    NOINC    LDA #>EIND        ; END OF FILE?
0222 02E8 C511              CMP WIJZER+1
0223 02EA D0EC              BNE SCHSCH        ; => NO
0224 02EC 4C0002            JMP TAALST        ; => YES, RESTART
0225
0226
0227 02EF                   .END

END OF ASSEMBLY        ERRORS DETECTED: 0000                    <0000>
```

Maze for Junior 27-Aug-87 22:51 Page 0

```
                00D0            org     $00d0
00D0            length  res     1       length of maze (bytes)
00D1            lenmin  res     1       length minus 1
00D2            hbyte   res     1       hor. position (bytes)
00D3            hbit    res     1       hor. position (bit)
00D4            cur     res     1       low byte address of cursor
00D5            disone  res     1       pattern of display 1
00D6            distwo  res     1       pattern of display 2
00D7            disthr  res     1       pattern of display 3
00D8            flash   res     1       flashcounter
00D9            savx    res     1       temporary storage for x
00DA            bit     res     1       used in caldis
00DB            dis     res     1       used in caldis

                0100    data    equ     $0100   shape of maze
                1A80    pad     equ     $1a80
                1A81    padd    equ     pad+1
                1A82    pbd     equ     pad+2
                1C1D    reset   equ     $1c1d   monitor entry
                1DAC    scdsb   equ     $1dac   scan keyboard
                1DF9    getkey  equ     $1df9   get value of depressed key

                0200            org     $0200

0200 78         maze    sei
0201 D8                 cld
0202 A2 FF              ldx     #$ff
0204 9A                 txs
0205 AE 0001            ldx     data
0208 F0 0A              beq     search  length already initialized ?
020A 86 D0              stx     length
020C CA                 dex
020D 86 D1              stx     lenmin
020F A2 00              ldx     #$00
0211 8E 0001            stx     data
0214 BD 0001    search  lda     data,x  search entrance
0217 30 07              bmi     found
0219 8A                 txa
021A 18                 clc
021B 65 D0              adc     length
021D AA                 tax
021E D0 F4              bne     search  try next line
0220 86 D4      found   stx     cur     entrance found
0222 A9 00              lda     #$00    mostleft position
0224 85 D2              sta     hbyte
0226 A9 80              lda     #$80
0228 85 D3              sta     hbit
022A A6 D4      main    ldx     cur     main program
022C A5 D3              lda     hbit    calculate pattern of display 1
022E 0A                 asla
022F 90 07              bcc     calone
0231 A5 D2              lda     hbyte
0233 F0 06              beq     strone  leftmost pos then blank display
0235 A9 01              lda     #$01
0237 CA                 dex
0238 20 DC02    calone  jsr     caldis
023B 85 D5      strone  sta     disone
023D A6 D4              ldx     cur     calculate pattern of display 2
023F A5 D3              lda     hbit
0241 20 DC02            jsr     caldis
0244 85 D6              sta     distwo
0246 A6 D4              ldx     cur     calculate pattern of display 3
0248 A5 D3              lda     hbit
024A 4A                 lsra
024B 90 09              bcc     calthr
024D A5 D2              lda     hbyte
024F E5 D1              sbc     lenmin
0251 F0 06              beq     strthr  rightmost pos then blank display
0253 A9 80              lda     #$80
0255 E8                 inx
```

```
Maze for Junior                 27-Aug-87 22:51          Page 1

0256 20 DC02     calthr  jsr     caldis
0259 85 D7       strthr  sta     disthr
025B 20 0503     shget   jsr     show        show display and wait for key
025E D0 FB               bne     shget
0260 20 0503     shgeta  jsr     show
0263 F0 FB               beq     shgeta
0265 20 0503             jsr     show
0268 F0 F6               beq     shgeta
026A 20 F91D             jsr     getkey
026D C9 04               cmp     #$04
026F B0 EA               bcs     shget       only keys 0123 valid
0271 A6 D4               ldx     cur
0273 C9 00               cmp     #$00
0275 D0 1F               bne     up
0277 A5 D3               lda     hbit        0 is left
0279 0A                  asla
027A 90 07               bcc     lefta
027C A5 D2               lda     hbyte
027E F0 13               beq     maina       once started there is no returning
0280 A9 01               lda     #$01
0282 CA                  dex
0283 A8          lefta   tay
0284 3D 0001             and     data,x
0287 F0 0A               beq     maina       wall ?
0289 84 D3               sty     hbit        move is correct
028B 86 D4               stx     cur
028D C0 01               cpy     #$01        if cur decremented then also hbyte
028F D0 02               bne     maina
0291 C6 D2               dec     hbyte
0293 4C 2A02     maina   jmp     main
0296 C9 01       up      cmp     #$01
0298 D0 06               bne     down
029A 8A                  txa                 1 is up
029B E5 D0               sbc     length
029D 4C A802             jmp     downa
02A0 C9 02       down    cmp     #$02
02A2 D0 14               bne     right
02A4 18                  clc                 2 is down
02A5 8A                  txa
02A6 65 D0               adc     length
02A8 AA          downa   tax
02A9 A5 D3               lda     hbit
02AB A8                  tay
02AC 3D 0001             and     data,x
02AF F0 04               beq     mainb       wall ?
02B1 84 D3               sty     hbit
02B3 86 D4               stx     cur
02B5 4C 2A02     mainb   jmp     main
02B8 A5 D3       right   lda     hbit        3 is right
02BA 4A                  lsra
02BB 90 09               bcc     righta
02BD A5 D2               lda     hbyte
02BF C5 D1               cmp     lenmin      mostright position ?
02C1 F0 16               beq     mon         then ready
02C3 A9 80               lda     #$80
02C5 E8                  inx
02C6 A8          righta  tay
02C7 3D 0001             and     data,x
02CA F0 0A               beq     mainc       wall ?
02CC 84 D3               sty     hbit
02CE 86 D4               stx     cur
02D0 C0 80               cpy     #$80        if cur incremented then also hbyte
02D2 D0 02               bne     mainc
02D4 E6 D2               inc     hbyte
02D6 4C 2A02     mainc   jmp     main
02D9 4C 1D1C     mon     jmp     reset

02DC 85 DA       caldis  sta     bit         calculate pattern of display
02DE A9 00               lda     #$00        in: A=bitno, X points to middle segment
02E0 85 DB               sta     dis         out: X=pattern (inverted)
02E2 A0 02               ldy     #$02        start with bottom segment (d)
02E4 18                  clc
```

```
02E5 8A              txa
02E6 65 D0           adc     length
02E8 AA              tax
02E9 BD 0001 oneseg  lda     data,x      calculate one segment
02EC 25 DA           and     bit
02EE D0 07           bne     nxtseg      segment remains blank
02F0 A5 DB           lda     dis         set segment
02F2 19 0203         ora     pat,y
02F5 85 DB           sta     dis
02F7 38      nxtseg  sec
02F8 8A              txa
02F9 E5 D0           sbc     length
02FB AA              tax
02FC 88              dey
02FD 10 EA           bpl     oneseg      not yet done all segments ?
02FF A5 DB           lda     dis         result pattern
0301 60              rts

0302 01 40 08 pat    fcb     $01,$40,$08 segments a (upper), g (middle), and d (bottc

0305 C6 D8    show   dec     flash       show display
0307 D0 06           bne     showa
0309 A5 D6           lda     distwo      invert cursor
030B 49 40           eor     #$40
030D 85 D6           sta     distwo
030F A9 7F    showa  lda     #$7f        pa is output
0311 8D 811A         sta     padd
0314 A0 08           ldy     #$08
0316 A2 00           ldx     #$00
0318 86 D9    shwone stx     savx        show one display
031A B5 D5           lda     disone,x
031C 49 7F           eor     #$7f        inverted pattern is stored
031E 8D 801A         sta     pad
0321 8C 821A         sty     pbd
0324 A2 7F           ldx     #$7f
0326 CA       delay  dex
0327 10 FD           bpl     delay
0329 8E 801A         stx     pad
032C A2 06           ldx     #$06
032E 8E 821A         stx     pbd
0331 C8              iny
0332 C8              iny
0333 A6 D9           ldx     savx
0335 E8              inx
0336 E0 03           cpx     #$03        only three displays shown
0338 D0 DE           bne     shwone
033A 4C AC1D         jmp     scdsb       scan keyboard

         0200        end     maze
```

Errors detected: 0

MAZE Pieter de Visser

Needed: Standard Junior

The intention of this game is to find the exit from the maze. You can see only a few blocks around you on the display. The following keys are used to walk around:
0-left 1-up 2-down 3-right

Any maze must satisfy these characteristics:

First byte of data must contain the length of the maze in bytes (so length must be an octuple). Entrance must be left, exit(s) right. No holes should appear in upper or bottom lines. A zero bit means a wall, a one is a hole.

Example:

```
0100- 04 00 00 00  so the length is 4
0104- 6F 97 FC FE  bytes = 32 bits
0108- C8 DC 07 82
010C- 7C 45 F1 36
0110- 55 F4 1D E0
0114- 44 67 54 3F
0118- 6F CD F7 82
011C- 68 18 04 18
0120- 4E 4B 9C FE
0124- 6B 7E F6 82
0128- 49 20 A3 FA
012C- 0B 8E 28 28
0130- 7A 3B 6E EE
0134- 40 A1 08 02
0138- 7F FD FF FE
013C- 00 00 00 00
```

De SmartWatch: 'n slim klokje.

Van tijd tot tijd verschijnen er IC's op de markt die in de categorie ei van Columbus vallen. De Dallas Semiconductor DS1216 is zo'n IC. Het is een real time clock. Nu niet eens kaal in de vorm van een IC, maar compleet met batterij, kristal en power-down logica. Geen kopzorgen dus. Het ei van Columbus wordt gevormd door de mechanische uitvoering van het geheel. Aan de onderzijde hebben we te maken met een 28-polige DIL aansluiting. Deze heeft een zogenaamde JEDEC-pinout, hetgeen inhoudt, dat de klok op de plaats van een 28-pins EPROM, EEPROM of SRAM gestoken kan worden. Aan de bovenkant van de 9 millimeter hoge behuizing vinden we een 28-polige DIL voet. Alle pinnen aan de onderkant zijn doorverbonden met de overeenkomstige pinnen aan de bovenkant, op 3 na: de CS op pin 20 en de voeding op de pinnen 26 en 28. Deze worden door de SmartWatch bestuurd. Men kan dus eenvoudig een (E)EPROM of SRAM uit zijn voetje halen, de SmartWatch in dat voetje steken en tenslotte diezelfde (E)EPROM of SRAM in de SmartWatch steken. De SRAM of (E)EPROM blijft hierbij volledig aanspreekbaar. Betreft het een CMOS SRAM, dan houdt de batterij in de SmartWatch niet alleen de klok, maar ook het geheugen onder spanning als de computer is uitgeschakeld. De communicatie met de klok verloopt serieel. De klok kijkt naar de lijnen CS (20), DO (11) en WE (27). Normaal blijft het RAM/EPROM op de klok volledig bereikbaar. Men activeert de klok door 64 achtereenvolgende schrijfcycli uit te voeren. Hierbij moet databit DO achtereenvolgens de volgende 8 bytes 'zien': $C5, $3A, $A3, $5C, $C5, $3A, $A3, $C5. Deze reeks mag onderbroken worden door willekeurige lees-cycli, doch niet door willekeurige schrijfoperaties. De kans dat een dergelijke reeks bij toeval, en dus ongewild optreedt is 1 op 10 tot de macht 19. Wordt de bovengenoemde reeks op DO herkend, dan onderbreekt de SmartWatch de CS naar het RAM, en kan er naar keuze naar de klok geschreven of uit de klok gelezen worden. De klok zelf bestaat ook uit 8 bytes die serieel via DO worden gelezen of geschreven. Nadat het 64e bit gelezen/geschreven is, wordt de toegang tot het RAM/EPROM weer automatisch hersteld.
De klok zelf bestaat uit 8 bytes die als volgt zijn ingedeeld:

Byte	Functie	Bijzonderheden
0	10den/100sten seconden	BCD formaat
1	10tallen/eenheden sec.	BCD formaat, bit 7 altijd nul
2	10tallen/eenheden min.	BCD formaat, bit 7 altijd nul
3	10tallen/eenheden uren	BCD formaat Bit 7: 12 of 24 uurs formaat Bit 6: altijd nul Bit 5: AM/PM vlag bij 12h formaat
4	Dagnummer	In bits 0-2 Bit 3: altijd nul Bit 4: RESET enable Bit 5: Oscillator aan/uit Bits 6 en 7: altijd nul
5	Datum	BCD, in bits 0-5 Bits 6 en 7: altijd nul
6	Maandnummer	BCD, in bits 0-4 Bits 5, 6 en 7: altijd nul
7	Jaar (10tallen/eenh)	BCD formaat

Net als de 64-bit toegangscode, dienen deze bytes serieel via D0 gelezen of geschreven te worden. Een aantal bits verdienen een nadere uitleg. Bit 7 van byte 3 bepaalt of de klok in 24-uurs of 12-uurs formaat loopt. Indien voor 12-uurs formaat is gekozen (bit 7 is dan 1), dan stelt bit 5 van dit byte de voor/namiddag vlag voor. Na 12 uur 's middags is dit bit gezet. Bit 4 van byte 4 is de RESET enable. Pin 1 doet namelijk dienst als RESET-ingang. Met deze pin kan een eenmaal gestarte toegangscyclus tot de klok worden afgebroken zonder dat de klokregisters gewijzigd worden. Deze functie kan met bit 4 aan of uit gezet worden. Als bit 4 een 1 is, dan is de RESET-functie gedisabled. Met bit 5 van byte 4 kan de interne 32 kHz oscillator aan- en uitgezet worden. De oscillator is namelijk de grootste stroomconsument en bepaalt voornamelijk de levensduur van de ingebouwde batterijen. De SmartWatch wordt standaard geleverd met de oscillator uitgeschakeld. Dan is bit5 een 1. Een nul schakelt de oscillator in. De klok houdt rekening met het verschillend aantal dagen in de diverse maanden. Ook schrikkeljaren worden correct verwerkt. Men kan helaas niet een willekeurig SRAM in het systeem nemen om de klok onder te steken. Men dient een zodanige RAM te kiezen, dan hierin geen schrijfcycli in op kunnen treden door periodieke interrupts (toetsenbord, software clock, of videogeheugen). Dergelijke cycli zouden namelijk de toegangscyclus onderbreken. In 6502 systemen betekent dit, dat het RAM waarin de zero-page en de stack in opgeslagen zijn niet bruikbaar is. In een DOS65 systeem kan bovendien het SRAM op de CPU-kaart niet gebruikt worden: hierin loopt onder andere de softwareklok en wordt de systeem administratie bijgehouden. Voor DOS65 is daarom adres $2000 gekozen als toegangslocatie. Om met de klok te kunnen communiceren is een utility geschreven dat de klok kan uitlezen en gelijk zetten. De naam van deze utility is SETSW. Er zijn in totaal drie verschillende versies van de SmartWatch verkrijgbaar. De eerste is de DS1216. Deze versie is bedoeld voor 2k byte en 8 kbyte CMOS SRAMs. De pinnen 26 en 28 van de bovenste voet zijn doorverbonden. De tweede versie is de DS1216C. Deze is bedoeld voor 8k byte en 32k byte CMOS SRAMs. Van deze versie is pin 26 gewoon doorverbonden en treedt alleen battery-backup via pin 28 op. De laatste versie is de DS1216E. Deze is bedoeld voor EPROMs. De battry-backup werkt in de DS1216E alleen voor de interne klok. Verder kan men in een EPROM niet schrijven. De toegangscyclus en de uitlees/inleescycli zijn dan ook iets anders gedefinieerd. A0 doet nu dienst als data-ingang, terwijl A2 bepaalt of er in de klok gelezen of geschreven wordt. Toegang tot de klok wordt verkregen door 64 leescycli uit te voeren met A2 nul (=schrijven) waarbij A0 serieel de toegangscode voert. Hierna kan de klok worden geschreven door de te schrijven data op A0 aan te bieden en A2 steeds nul te houden. Dit geschiedt door de sturen microprocessor leescycli in de EPROM te laten doen. De adresvolgorde wordt bepaald door de toegangscode of de te schrijven data. Het lezen van de klok na het aanbieden van de toegangscode is relatief eenvoudiger: op D0 verschijnt de klokdata (wederom serieel). De klok kan dus gelezen worden door 64 leesoperaties te doen en D0 uit te filteren. In tegenstelling tot de RAMversies kan bij de EPROMversies de toegangscyclus niet onderbroken worden omdat de EPROM slechts leescycli kent.

Dallas Semiconductor wordt in Nederland vertegenwoordigd door:
Alcom Electronics B.V. Postbus 358
2900 AJ Capelle aan den IJssel
Telefoon 010-4519533
Dit bedrijf is een groothandel en levert niet aan particulieren.

****

Tips, of: leve de Taiwan-klonen!

De komst van de Taiwan-klonen van de diverse IBM-computers maakt niet iedereen

even blij, zeker IBM niet. Ook is er een enorme verschraling waar te nemen in het niveau van de gemiddelde computer-'gek' en in de keuze van het publiek voor een bepaalde computer. Toch is niet alles negatief. De moordende concurrentie, gecombineerd met de enorme aantallen verkochte computers maakt dat ze erg goedkoop geworden zijn. Zo kun je bijvoorbeeld al een PC-XT kopen voor ca. fl. 1300.-. Voor dat geld kun je nauwelijks een Octopus of DOS65 computer bouwen. De lage prijzen gelden niet alleen voor de complete computers, maar natuurlijk ook voor de onderdelen waaruit ze zijn samengesteld. En hiermee kunnen we ons voordeel doen!

- De voeding, of power supply.
Alle klonen werken met een geschakelde voeding. De XT-machines hebben bijna altijd een voeding van 135 Watt, de AT-compatibles doen het met 180, 200 of 220 Watt voedingen. Voor ons zijn de 135 XT-voedingen het aantrekkelijkst. Alle voedingen zien eruit als een blikken doos met daarin een ventilator, een netentree, een netuitgang, de netschakelaar en wat draden met connectors. Halen we de blikken doos om de voeding weg, dan blijkt deze een print te bevatten ter grootte van een Eurokaart met daarop de complete geschakelde voeding. De ventilator blijkt zonder uitzondering voor 12 Volt DC géschikt te zijn. We houden dus een prima inbouwvoeding over, compleet met fan voor de koeling van het systeem. De aansluitingen bestaan vrijwel altijd uit het volgende. Eerst zijn er vier aansluitingen voor diskdrives, compleet met die lastig te verkrijgen vierpolige stekker. Verder zijn er nog een of twee connectors, bedoeld voor het moederbord van de PC. De draden hierin zijn meestal als volgt gecodeerd:

Kleur	Spanning	Max. Stroom
Rood	+ 5 Volt	15 Ampere
Geel	+ 12 Volt	4.5 Ampere
Blauw	- 5 Volt	0.5 Ampere
Bruin	- 12 Volt	0.5 Ampere
Wit	'Power Good'	TTL niveau
Zwart	Ground	

De witte Power-Good lijn kan eventueel met de RES-aansluiting van de bus verbonden worden. De voeding is zodanig geconstrueerd, dat overschrijding van de maximum stroom op een uitgang automatisch afschakelen inhoudt. De voeding wordt weer ge-RESET, door de netspanning uit- en weer in te schakelen. Hetzelfde geldt als het maximum vermogen wordt overschreden. De voeding is in staat de hoge startstromen die op de +12 Volt optreden bij het starten van winchesters te leveren. Tenslotte is alleen de + 5 Volt gestabiliseerd (en soms instelbaar). De andere uitgangen lopen met deze spanning mee. Een goedkope, doch voldoende grote voeding voor uw nieuwe systeem!

- De kast.
De PC-XT klonen worden meestal gehuisvest in een kast waarvan de bovenkant opgeklapt kan worden. Hierdoor is de inhoud van de kast goed bereikbaar. De kast is hoog genoeg om er een Eurokaartrekje in te monteren. Verder heeft de kast standaard openingen voor 4 slim-line drives en worden er afdekplaatjes meegeleverd voor de niet gebruikte gaten. Het monteren van drives in zo'n kast is een fluitje van een cent. Aan de achterkant is de kast voorzien van alle gaten voor de montage van een voeding (zie boven). De maten van de voeding en de bevestiging zijn redelijk genormaliseerd. De voeding wordt vastgezet met vier schroeven, die meestal met de voeding worden meegeleverd. De meeste kasten hebben vaak ook al gaten voor RS-232 connectors en dergelijke. Het enige nadeel aan de kast is dat hij van staalplaat gemaakt is. Dit bewerkt aanzienlijk moeilijker dan aluminiumplaat.

- Het toetsenbord.
Het toetsenbord van de klonen is van een bijzonder type. Het wordt met de computer verbonden met een vijfpolige DIN-steker. 1 lijn wordt niet gebruikt. De verdere lijnen zijn: +5 Volt, GND, keyboard clock en keyboard-data. De tweesignaallijnen zijn bidirectioneel. Worden ze beide gedurende 20 ms laag gehouden, dan voert het toetsenbord een selftest uit. Het keyboard stuurt geen ASCII-tekens naar de computer, maar toetsnummers. Wordt een toets ingedrukt, dan wordt het toetsnummer serieel over de lijn keyboard-data gestuurd. Er worden steeds acht bits (1 byte) verzonden. Verder wordt bij ieder bit een clockflank gegenereerd waarmee

een ontvangend schuifregister geclockt kan worden. Wordt een toets weer losgelaten, dan wordt wederom het toetsnummer verzonden, maar nu met het 8ste bit gezet. Als een toets wordt vastgehouden, dan zal het toetsenbord uit zichzelf de code voor loslaten en drukken gaan versturen, waardoor de ontvangende computer denkt, dat de toets reperteert. Dit laatste geldt niet voor de beide shift toetsen, en de Alt, Ctl, en Lock toetsen. Het is niet moeilijk het toetsenbord te interfacen met een VIA. Voor DOS65 is bijvoorbeeld een driver voor een XT-toetsenbord beschikbaar. Let op: AT-toetsenborden hebben een andere toetsnummering dan XT-toetsen borden. Wel blijft binnen een familie de nummering gelijk: de Esc op een XT-toetsenbord heeft altijd toetsnummer 1, onafhankelijk waar deze toets zich in het toetsenbord bevindt.

- Diskdrives.

Ook diskdrives profiteren van de grote aantallen en worden goedkoper. Dit geldt echter niet voor alle typen. De standaard drive voor klonen is de 40-track dual sided drive. In AT-klonen worden weliswaar 80-track drives toegepast, maar deze hebben een hogere spindelsnelheid en zijn voor DOS65 en OHIO-OS65D niet te gebruiken. Bedenk, dat de enorme capaciteit van een 80 track drive maar zelden volledig wordt benut, dat zo'n drive duurdere schijfjes vereist, en dat de data kwetsbaarder is. 40-track dual sided is zo gek nog niet.

- Hard-disks.

Ook deze worden steeds goedkoper: er zijn al aanbiedingen van 20 Mbyte drives met controller voor minder dan fl. 650.-. Aan de controller heb je niets: niet kopen dus. Alle drives voor klonen zijn voorzien van de Seagate ST506 interface. Er bestaan OHIO- schema's om zo'n drive aan te sturen. Voor DOS65 komt er een SASI controller board bij te pas. Deze zijn niet gemakkelijk goedkoop te krijgen.

Wellicht kunt u uw voordeel doen met deze ontwikkelingen in de markt als u uw systeem nog moet bouwen of het weer eens op de helling zet. Tijdens de komende HCC-dagen worden de prijzen nog lager, let maar op!

NIEUW VAN ACORN COMPUTERS : DE ARCHIMEDES

Wil je als computerfabrikant, midden tussen het MS-DOS geweld, iets nieuws brengen, dan moet dat ook wel iets bijzonders zijn. De mensen van ACORN hebben zich dat wellicht goed gerealiseerd toen ze begonnen met de ontwikkeling van hun nieuwste computer, de Archimedes. Welnu, ze hebben duidelijk hun best gedaan. De Archimedes is een veelbelovende ontwikkeling, die de konkurrentie zeker tot nadenken zal stemmen.

Het hart van de Archimedes bestaat niet uit een gangbare processor, zoals de 6502, 8086 enz. De mensen van Acorn ontwikkelden zelf een geheel nieuwe CPU, die men R.I.S.C. gedoopt heeft. R.I.S.C. staat voor Reduced Instruction Set Computer. De filosofie die hier achter schuilt, is het gegeven, dat veel CPU's een aantal weinig of helemaal niet gebruikte instrukties bevatten. (wie gebruikt bijv. Indirect Indexed X van de 6502 ?) Door de instruktieset te beperken tot het hoogst noodzakelijke, werd het mogelijk deze instrukties relatief snel te laten uitvoeren. Bovendien is de R.I.S.C. een 32 bits CPU. Dit resulteert in een snelheid van 4 MIPS (miljoen instrukties per seconde). Volgens de gegevens van Acorn nog lang niet de limiet; in het laboratorium schijnt reeds 18 MIPS gehaald te zijn.

Van alle eigenschappen zijn er twee wel zeer vermeldenswaardig: de machine kan zowel MS-DOS (in software) als de 6502 (in hardware!) emuleren. Volgens opgave emuleert de Archimedes MS-DOS acht keer zo snel dan de standaard MS-DOS machines. De 6502 schijnt drie keer sneller dan normaal geemuleerd te worden. Waarschijnlijk wordt dan wel uitgegaan van een 1 MHz 6502.

Van de Archimedes verschijnen twee series: de 300 en de 400 serie. De 300 serie is de standaard serie, en bestaat althans voorlopig uit de 300 en de 310. De 300 heeft 512 Kbyte RAM, de 310 heeft 1025 Kbyte RAM. Beide machines zijn voorzien van 512 Kbyte ROM, en beschikken over een 1 Mbyte (ongeformatteerd) 3,5 inch disk drive. Beide machines

beschikken over uitbreidingsmogelijkheden, Of hiermee een uitbreiding naar de 400 serie bedoeld wordt, is echter niet geheel duidelijk. De 400 serie bestaat uit de 410 en de 440. De 410 heeft dezelfde eigenschappen als de 310, maar is bovendien uitgebreid met een hardware floating point unit optie, een co-processor bus, een hard disk controller, en een backplane (bus) met vier slots. De 440 is het topmodel. Eigenschappen identiek aan de 410, echter voorzien van 4 Mbyte RAM en een 20 Mbyte hard disk.

Voor alle machines zijn er een aantal uitbreidingskaarten :
- hard disk controller (voor de 300 serie)
- ROM expansie module
- MIDI interface
- MS-DOS co-processor
- SCSI
- hardware floating point unit
- Ethernet interface

In de toekomst zullen nog meerdere interfaces en uitbreidingen beschikbaar komen.

De software: Het operating systeem van de Archimedes heet ARTHUR, en is een verbeterde versie van het BBC Micro operating systeem. De BASIC die wordt meegeleverd heet BBC BASIC V, en is een verbeterde versie van de bekende BBC Basic. Ook andere programmeertalen, ondermeer C, ISO-PASCAL, FORTRAN, Lisp en Prolog zijn al leverbaar. Op korte termijn komen extra utilities beschikbaar, zoals tekstverwerkers, development tools, databases enz.

De prijzen van de Archimedes liegen er niet om; afhankelijk van model en konfiguratie dient men te rekenen op plm. f 3500,- tot circa f 6000,-

Dan nog enkele specificaties:

CPU	:	ARM (Acorn RISC Machine). Klokfrekwentie 4 of 8 MHz.
RAM	:	0,5 tot 4,0 Mbyte, afhankelijk van de uitvoering.
ROM	:	512 KByte
DISPLAY	:	Totaal 18 modes, tekst maximaal 132 kolommen, 32 regels, oplossend vermogen maximaal 640 x 256 pixels, maximaal 256 kleuren.
GELUID	:	2 geluidskanalen.
DRIVES	:	Een 3,5 inch drive, kapaciteit 1 Mbyte. Aansluiting 20 Mbyte winchester.
I/O	:	RS432. Baudrate 75 tot 19200. Centronics parallel.
KEYBOARD:		IBM-achtig toetsenbord, 103 toetsen, aansluiting voor muis.

De importeur van de Archimedes is :
Eeckhorn B.V.
Poortweg 6,
2612 PA Delft
tel. 015-569365

****

```
/*************************************************************************/
/*                                                                       */
/*      Convert normal ASCII string into big characters in a file        */
/*                                                                       */
/*************************************************************************/

#define EOL 10
#define EOF -1
#define NULL 0

#define lenlin 80
#define maxch 61        /* at this moment max. defined char. */
  #define chstr "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890[]=!?+#-^.,$ <>&:;*/'\\()%"

int sh;                 /* height of one dot                             */
int sd;                 /* width of one dot                              */
int hv;                 /* horizontal/vertical flag, 1=hor, 0=ver        */
int fo;                 /* output file pointer                           */
int maxlen;             /* maximum length of line                        */
int lxstring[7];        /* contains pointers to text strings             */
int hpstring[maxch];    /* contains pointers to character strings        */
char txtstring[lenlin]; /* string to convert                             */
char fcstring[lenlin];  /* contains fill out characters                  */

main()
{ char fname[21];
  char tp0[lenlin],tp1[lenlin],tp2[lenlin],tp3[lenlin],tp4[lenlin],
       tp5[lenlin],tp6[lenlin];

  lxstring[0] = tp0;
  lxstring[1] = tp1;
  lxstring[2] = tp2;
  lxstring[3] = tp3;
  lxstring[4] = tp4;
  lxstring[5] = tp5;
  lxstring[6] = tp6;

  data();

  *txtstring = NULL;            /* fill in defaults */
  *fcstring = NULL;
  sh = 1;
  sd = 2;
  hv = 1;
  maxlen = 80;

  while (strlen(txtstring) == 0)
  {     printf("Text (max. 80 characters)     :");
        gets(txtstring);
  }
  printf("Fill out char's       [    text]:");gets(fcstring);
  printf("Height of one dot     [%8d]:",sh);scanf("%d",&sh);
  printf("Width of one dot      [%8d]:",sd);scanf("%d",&sd);
  printf("Hor/Ver = 1/0         [%8d]:",hv);scanf("%d",&hv);
  hv = hv & 1;
  if (hv)
  {     printf("Max. line length      [%8d]:",maxlen);
        scanf("%d",&maxlen);
  }
  else   maxlen = lenlin;

  if (maxlen <= 0)      maxlen = lenlin;
  if (maxlen > lenlin)  maxlen = lenlin;
  if (sh <= 0) sh = 1;
  if (sd <= 0) sd = 1;

  strcpy(fname,"text.tmp");
  printf("Filename              [%s]:",fname);scanf("%s",fname);
  fo = fopen(fname,"w");
  if (fo == NULL)
  {     printf("Can't open/write file %s\n",fname);
        exit(-1);
  }
```

```
  convert();

  printf("File %s filled with big characters\n",fname);
  fclose(fo);
  exit(0);
}

convert()
{ int ind,h1,h2,h3,h4,h5,linp;
  char chs,*pt,*rf,*pf,*hp,*pl;

  pt = txtstring;
  pf = fcstring;
  rf = chstr;
  initfc();
  linp = 0;
  while(*pt)
  {     *pt = toupper(*pt); /* convert to upper case */
        if ((ind = index(rf,*pt)) == -1)
        {       printf("Character '%c' not implemented\n",*pt++);
                pf++;
                continue;
        }
        hp = hpstring[ind];
        while (*hp)
        {       h1 = ((((*hp++) & 7) * 16) + ((*hp++) & 15));
                for (h4=1;h4<=sd;h4++)
                {       h3 = 64;
                        for (h2=0;h2<7;h2++)
                        {       if (h3 & h1)    chs = toupper(*pf);
                                else            chs = ' ';
                                if (hv) *(lxstring[h2] + linp) = chs;
                                else    for (h5=1;h5<=sh;h5++) putc(chs,fo);
                                h3 = h3 / 2;
                        }
                        linp++;
                        if (!hv) putc(EOL,fo);
                }
        }
        if (hv)
        {       if (linp > (maxlen - ( 7 * sd)))
                {       *(lxstring[h2] + linp + 1) = NULL;
                        prtlxs();
                        linp = 0;
                }
                else
                {       for (h5=1;h5 <= sd;h5++)
                        {       for (h3=0;h3 <= 6;h3++)
                                        *(lxstring[h3] + linp) = ' ';
                                linp++;
                        }
                }
        }
        else for (h5=1;h5 <= (2 * sd);h5++) putc(EOL,fo);
        pt++;
        pf++;
   }
   if ((linp > 0) & hv)
   {    *(lxstring[h2] + linp + 1) = NULL;
        prtlxs();
   }
}

prtlxs()
{ char *pt;
  int  n,i;
  for (n=6 ;n >= 0;n--)
  {   for (i=1;i <= sh;i++)
      {    pt = lxstring[n];
           while(*pt) putc(*pt++,fo);
           putc(EOL,fo);
      }
  }
  for (i=1;i <= sh;i++) putc(EOL,fo);
}
```

```
initfc()
{ char *pt,*pf;
  pt = txtstring;
  pf = fcstring;
  if ((pf[1] == NULL) & (*pf != NULL))
  {    while(*pt++) *pf++ = *fcstring; /* fill out character */
       return;
  }

  if (strlen(pt) != strlen(pf))

     while(*pt) *pf++ = *pt++;          /* else fill out with text */
}

/* return index of character t in string s or -1 if none */
index(s,t)
char s[],t;
{ int i;
  for (i = 0 ;s[i] != NULL; i++) { if (s[i]==t) return(i); }
  return(-1);
}

data()
{

  /* HOOFDLETTERS DATA */

  hpstring[ 0]="7>09097>";    /* A */    hpstring[ 1]="7?494936";    /* B */
  hpstring[ 2]="3>414141";    /* C */    hpstring[ 3]="7?41413>";    /* D */
  hpstring[ 4]="7?494941";    /* E */    hpstring[ 5]="7?090901";    /* F */
  hpstring[ 6]="3>41414979"; /* G */    hpstring[ 7]="7?080887?";   /* H */
  hpstring[ 8]="417?41";      /* I */    hpstring[ 9]="60404037";    /* J */
  hpstring[10]="7?083641";    /* K */    hpstring[11]="7?404040";    /* L */
  hpstring[12]="7?02040277"; /* M */    hpstring[13]="7?061<307?"; /* N */
  hpstring[14]="3>41413>";    /* O */    hpstring[15]="7?090906";    /* P */
  hpstring[16]="3>51617>";    /* Q */    hpstring[17]="7?090976";    /* R */
  hpstring[18]="26494932";    /* S */    hpstring[19]="01017?0101"; /* T */
  hpstring[20]="3?40403?";    /* U */    hpstring[21]="031<601<03"; /* V */
  hpstring[22]="3?4020403?"; /* W */    hpstring[23]="4136083641"; /* X */
  hpstring[24]="01027<0201"; /* Y */    hpstring[25]="6151494543"; /* Z */

  /* DATA VAN DE GETALLEN */

  hpstring[26]="02037?";      /* 1 */    hpstring[27]="6251494542"; /* 2 */
  hpstring[28]="22414936";    /* 3 */    hpstring[29]="3>207020";    /* 4 */
  hpstring[30]="2?494931";    /* 5 */    hpstring[31]="3<4:4930";    /* 6 */
  hpstring[32]="61110907";    /* 7 */    hpstring[33]="36494936";    /* 8 */
  hpstring[34]="0649291>";    /* 9 */    hpstring[35]="3>51493>";    /* 0 */

  /* DATA VAN OVERIGE TEKENS */

  hpstring[36]="7?41";        /* [ */    hpstring[37]="417?";        /* ] */
  hpstring[38]="14141414";    /* = */    hpstring[39]="5?5?";        /* ! */
  hpstring[40]="02015906";    /* ? */    hpstring[41]="08083>0808"; /* + */
  hpstring[42]="147?147?14"; /* # */    hpstring[43]="08080808";    /* - */
  hpstring[44]="04067?0604"; /* ^ */    hpstring[45]="6060";        /* . */
  hpstring[46]="2060";        /* , */    hpstring[47]="26497?4932"; /* $ */
  hpstring[48]="00000000";    /*   */    hpstring[49]="08142241";    /* < */
  hpstring[50]="41221408";    /* > */    hpstring[51]="3649552250"; /* & */
  hpstring[52]="6<6<";        /* : */    hpstring[53]="2<6<";        /* ; */
  hpstring[54]="22147?1422"; /* * */    hpstring[55]="2010080402"; /* / */
  hpstring[56]="0307";        /* ' */    hpstring[57]="0204081020"; /* \ */
  hpstring[58]="1<2241";      /* ( */    hpstring[59]="41221<";      /* ) */
  hpstring[60]="2616083432"; /* % */
```

NIEUWE CHIPS VAN GTE

Wie denkt, dat de 65xx familie inmiddels tot de verouderde processoren behoort, heeft het goed mis ! Dat blijkt onder andere uit het feit, dat GTE Micro-circuits onlangs een aantal nieuwe 'bouwstenen' introduceerde, onder het motto 'still going strong'. Dit bewijst maar weer eens, dat er wat betreft de 65xx familie geen sprake is van veroudering en/of achterhaald zijn.

Op zich is dat niet zo verwonderlijk; de 65xx familie, met de 6502 als middelpunt, heeft zijn kwaliteiten duidelijk bewezen. Zo langzamerhand mag je wel konkluderen, dat de 6502 de meest efficiente 8 bits processor is.
Enkele voorbeelden : Een BASIC programma draait op een 2 MHz 6502 sneller dan op een IBM PC. Sterker nog: een 6502 op 3 MHz benadert de performance van de 68000. In veel industriele toepassingen geniet de 6502 de voorkeur boven tal van andere 8 en zelfs 16 bits processoren !

Terug naar het onderwerp: nieuwe chips van GTE :

De standaard 65xx familie (65C02, 65C22, 65C51 enz.) is nu ook beschikbaar in 4 MHz versies. Er schijnt zelfs sprake te zijn, dat er volgend jaar 6 MHz en 8 Mhz versies op de markt komen!

G65SC37 CMOS Timing and Keyboard/Display Interface (TKDI)

Deze 68 pens chip is in staat om geheel autonoom 64 toetsen te scannen, en bovendien 64 display indicators (bijv. LED's) aan te sturen. De chip beschikt over een tweetal tussengeheugens, elk van 8 bytes. Bovendien genereert de chip een nauwkeurig timing signaal, bijvoorbeeld voor Real Time Clock toepassingen. Data-overdracht kan zowel parallel als serieel plaats vinden. De interfacing vindt plaats met behulp van de ons welbekende signalen; D0-D7, A0-A1, CS, R/W, phi2, RES enz. De timing kan plaats vinden door de systeemklok, maar ook door een externe klok.

G65SC150 CMOS Communications Terminal Unit (Telecommunication Microcomputer)

Een opmerkelijke chip, eigenlijk een single chip micro-processor, maar wat betreft de struktuur identiek aan de 65xx interface.
De mogelijkheden van deze interessante chip:

* genereert signalen compatible met 'switched telephone networks' of 'packet switched data networks'.
* Modem faciliteiten tot 1200 baud
* Voorzien van DP (dial pulse) en DT (dial tone) mogelijkheden.
* In low power mode is het stroomverbruik slechts 300 micro-ampere. Dit maakt voeding vanuit het telefoonnet mogelijk
* On chip geheugen : 2K ROM, 64 bytes RAM, extern uit te breiden tot 64 K
* Twee sinus generatoren
* 65Cxx programmeer struktuur
* 27 I/O lijnen

Van beide chips zijn alleen nog maar voorlopige gegevens beschikbaar. Wanneer nadere gegevens bekend zijn, zullen deze gepubliceerd worden.

##########################

65C02 Tip(je).

Bij studie van het datasheet blijkt, dat alle aansluitingen van onze grote vriend, de 65(C)02 CPU, bedoeld zijn voor TTL. Er is echter 1 uitzondering: de clockingang phi0 (pin 37). Deze pin blijkt andere logische niveaus te vereisen. Bij het nul-niveau gaat alles goed bij aansturing vanuit een TTL-uitgang, maar bij een logische 1 wordt de spanning in theorie niet hoog genoeg. Door de TTL-uitgang een beetje te helpen met een weerstandje van 1 kohm naar de plus, wordt het 1-niveau veel mooier. En hiermee wordt de onderlinge verhouding tussen phi1 en phi2 verbeterd. Dit probleem blijkt zich vooral voor te doen bij de 65C02. Dus: even een pull-up weerstandje van 1kohm naar de plus (5 Volt) als de phi0 pin vanuit een TTL-uitgang wordt gestuurd.
(Opgemerkt door Ad Brouwer)

VERZEKERING VAN PERSONAL COMPUTERS

Verzekerd kunnen worden dé micro-computers, bestemd voor prive gebruik.

Deze apparatuur is ook buiten de woning of de bedrijfsruimte verzekerd.
De verzekering is namelijk van kracht in de Benelux en West-Duitsland.

VERGOED WORDEN:

1. schaden aan de apparatuur veroorzaakt door:
   - brand, ontploffing en kortsluiting (incl. eigen gebrek)
   - een van buiten komend onheil.

2. verlies van de apparatuur door diefstal.
   - diefstal van de apparatuur buiten het in de polis genoemde pand is verzekerd na braak.

3. de bereddingskosten die zijn gemaakt tot vermindering van schade aan de apparatuur.

4. de opruimingskosten tot 10 % van de op apparatuur verzekerde som.

SCHADEVERGOEDING

De reparatiekosten worden vergoed:
Bij totaal verlies wordt de nieuwwaarde uitgekeerd. De vergoeding geschiedt echter naar dagwaarde indien de waarde minder dan 40 % van de nieuwwaarde is.

NADERE BIJZONDERHEDEN

De verzekering wordt gesloten voor de duur van 1 jaar en wordt stilzwijgend verlengd voor dezelfde termijn.

De premie voorwaarden worden op aanvraag verstrekt voor:
- apparatuur die in beroep of bedrijf voor andere dan administratieve doeleinden wordt gebruikt.
- apparatuur in gebruik bij onderwijsinstellingen en verenigingen.

LET OP !!

Leden van de Kim gebruikers club nederland betalen geen poliskosten.

INLICHTINGEN: ADMINISTRATIEKANTOOR A. METZLAR
VOORBURGGRACHT 479
1724 RH OUDKARSPEL
TEL.02260-16889

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