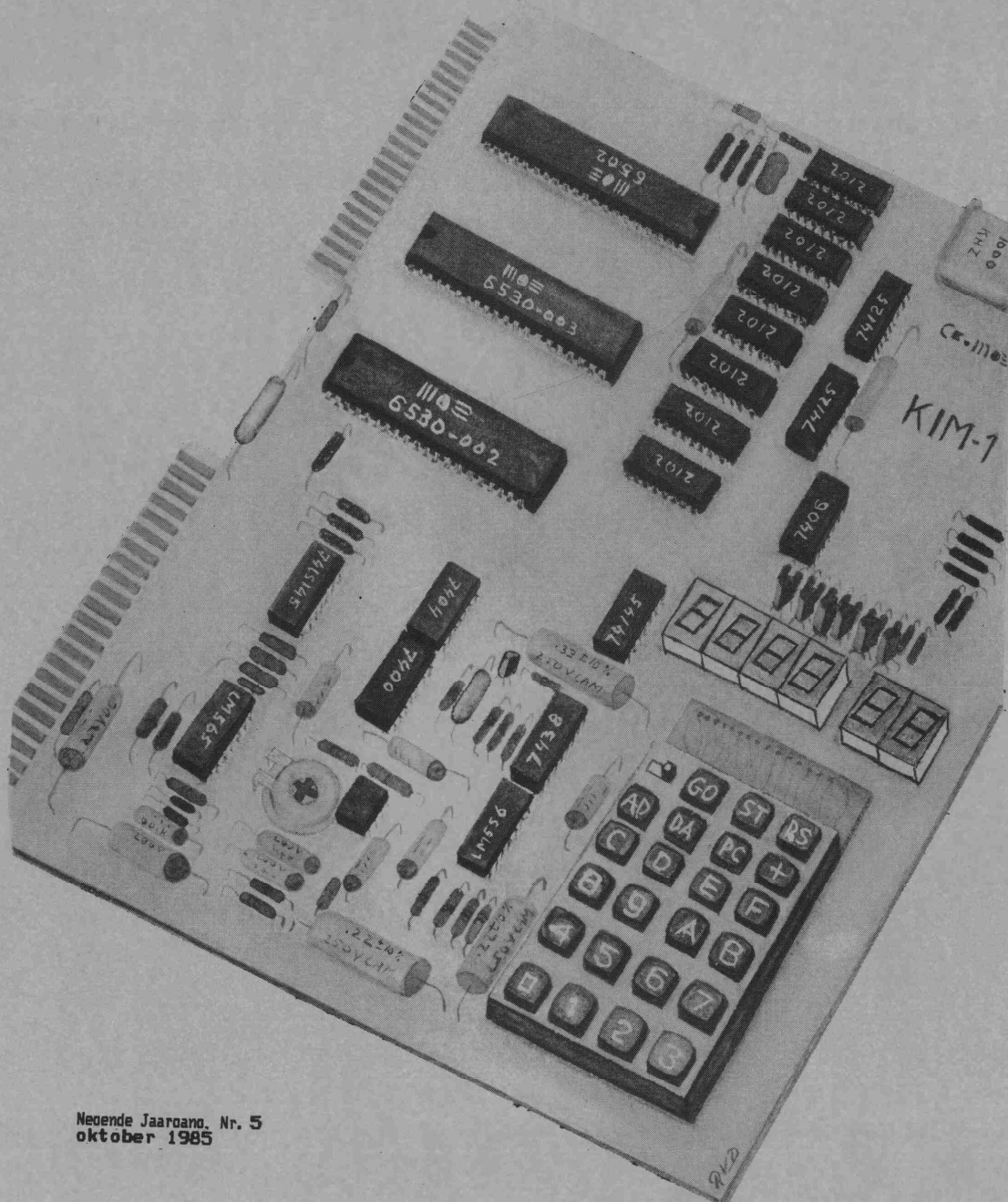




DE 6502 KENNER

**** DE 6502 KENNERS ** — EEN CLUB VOOR 65xx GEBRUIKERS**

De vereniging heeft leden in Nederland, België, Duitsland, Frankrijk, Spanje, Portugal, Amerika, Zambia. Het doel van de vereniging is: het bevorderen van de kennisuitwisseling tussen gebruikers van 65xx-computers, zoals KIM, JUNIOR, COMMODORE-64, APPLE, CHE-1, PEARCOM, AIM-65, SYN, PET, BBC ATARI, VIC-20, BASIS 108, PROTON-computers, ITT-2020, OSI, ACC 8000, ACCORN ELECTRON, SYSTEM 65, PC-100, PALLAS, MINTA FORMOSA, ORIC-1, STARLIGHT, CV-777, ESTATE 111, SBC65/68, MCS 6502, KEMPAC System-4, Elektuur SANSON-65 DOS computer, LASER, etc., etc.

De kennisuitwisseling wordt o.a. gerealiseerd door 5 maal per jaar DE 6502 KENNER te publiceren (1984 en 1985 6 maal zonder kontributieverhoging), door het houden van clubbijeenkomsten, door het instandhouden van een cassette-bibliotheek en door het verlenen van paperware-service. Regionale bijeenkomsten worden door leden georganiseerd.

Verschijningsdata DE 6502 KENNER 1985

derde zaterdag van
februari, april, juni,
augustus, oktober, december.

Inlichtingen over de regio- bijeenkomsten:

Gerard van Roekel
Van der Paleestraat 11 - C
3135 LK Vlaardingen
Tel.: 010 - 351101

Bijeenkomsten van de club

derde zaterdag van
januari, maart, mei,
september, november.

Redactie-adres en informa- ties over paperware etc.

Willeen L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpden/IJssel.
Tel.: 01807 - 19881

De vereniging is volledig onafhankelijk, is statutair ogericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Voorzitter:
Rinus Vleesch-Dubois
Fl. Nightingalestraat 212
2037 NS Haarlem
Tel.: 023 - 330993

Sekretaris:
Gert Klein
Diedenweg 119
6706 CM Wageningen
Tel.: 08370 - 23646

Pemningmeester:
John F. van Sprang
Tulp 71
2925 EW Krimpden/IJssel.
Tel.: 01807 - 20589

Redactie DE 6502 KENNER:
Willeen L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpden/IJssel.
Tel.: 01807 - 19881

Leden:
Adri Hankel (05490 - 51151) Hardware/software
Erwin Visschedijk (05490 - 71416) Hardware/software
— Promotie

Nico de Vries (010 - 502239) Hardware/software/PET
Erevoorzitter: Siep de Vries
Ereleden: Mm. H. de Vries - Van der Winden
Anton Mueller

Lidmaatschap: Hfl. 45,- per kalenderjaar, postrekening 3757649 t.n.v. Pemningmeester KIM Gebruikers Club Ned., Krimpden/IJssel.

Lidmaatschap 86: Te voldoen uiterlijk in december 1985.
Advertenties: Tarieven op aanvraag bij de redactie.

**** DE 6502 KENNER ** — EEN BLAD VOOR 65xx GEBRUIKERS**

DE 6502 KENNER is een uitgave van de KIM Gebruikers Club Nederland. Het blad wordt verstrekt aan leden van de club. DE 6502 KENNER wordt van kopij voorzien door leden van de club, bij de opmaak van een publikatie bijgestaan door de redactie. De inzendingen van programma's dienen voorzien te zijn van commentaar in de listings en zo mogelijk door een inleiding voorafgegaan. Publikatie van een inzending betekent niet dat de redactie of het bestuur enige aansprakelijkheid aanvaardt voor de toepassing ervan. De inzendingen kunnen geschieden in assembly-source-listings, in Basic, in Basicode, Forth, Focal, Comal, Pascal, Fortran, Cobol, Logo Elan, etc. etc.

De leden schrijven ook artikelen over de door hen ontwikkelde hardware en/of aanpassingen daarop. Zij schrijven tevens artikelen van algemene aard of reageren op publikaties van andere inzenders.

DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

Micro-ADE Assembler/Disassembler/Editor is een produkt van Micro Ware Ltd., geschreven door Peter Jennings en bestemd voor alle 6502-computers. De KIM Gebruikers Club Ned. heeft de copyrights verworven nadat ons lid Sebo Woldringh de 4 K KIM-1 versie uitbreidde tot 8 K KIM-1 versie. Adri Hankel paste deze aan voor de JUNIOR. Willeen L. van Pelt stelde een nieuwe 8 K source-listing voor de JUNIOR samen.

De implementatie op andere systemen dan de KIM-1 en JUNIOR kan eenvoudig gebeuren door het aanpassen van de I/O-adressen, welke in de source-listing gemakkelijk te vinden zijn.

FATE Format-listener/cond. Assembler/Tape-utilities/Editor is de door ons lid Rob Banen geschreven source-listing van een 12 K universeel systeem voor de JUNIOR-computer aan de hand van het universele disk operating systeem van de fa. Proton Electronics te Naarden, nu geschikt voor werken met tapes. FATE wordt beschikbaar gesteld met toestemming van Proton.

In de edities van DE 6502 KENNER worden regelmatig mededelingen gedaan over de door de club georganiseerde bijeenkomsten. Ook worden bestuurlijke mededelingen gedaan, naast informatie over hetgeen in de handel te koop is. Leden die iets te koop hebben of iets zoeken kunnen dit in de edities van DE 6502 KENNER bekend maken. Ook worden wel brieven aan redactie gepubliceerd, evenals specifieke vragen van leden. De edities worden samengesteld op basis van een groot aantal prioriteiten, welke door een redactievergadering worden gehanteerd. Deze vergadering bestaat uit de vaste medewerkers zoals in de colofon vermeld. Het aantal inzendingen is groter dan in een enkele editie van minimaal 48 pagina's is te verwerken. Hierdoor kan het voorkomen dat een inzending eerst na enige tijd kan worden gepubliceerd.

DE CLUB HEEFT BEHOEFTE AAN MEER LEDEN. WIJ WILLEN MEER AAN KUNNEN BIEDEN DAN NU AL HET GEVAL IS. MERF DAAROM EEN LID!

WILT U EEN PRIJSLIJST? STUUR EEN GEFRANKEERDE ENVELOP AAN HET REDAKTIE-ADRES.

Een onafhankelijke jury kent jaarlijks een aantal aanmoedigingspremie toe aan auteurs van gepubliceerde artikelen in DE 6502 KENNER.

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

Adres voor het inzenden van en reacties op artikelen voor DE 6502 KENNER:
Willeen L. van Pelt
Jacob Jordansstraat 15
2923 CK Krimpen a/IJssel
Tel.: 01807 - 19881

Vaste medewerkers:

Willeen L. van Pelt
Gerard van Roekel
Frans Smeethuizen
Jaap van Toledo

Freelance medewerkers:

Frans Bakx
Rob Banen
Fridus Jonkean
Gert Klein
Roger Langeveld
Anton Mueller
Gert van Oobroek
Rud Uhoff

Gehele of gedeeltelijke overname van de inhoud van DE 6502 KENNER zonder toestemming van het bestuur is verboden. Toesending van gepubliceerde programma's, hardware etc. is alleen toegestaan voor persoonlijk gebruik.

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

Copyright (C) 1985 KIM Gebruikers Club Nederland.

De voorpagina is een aquarel van een KIM, geschilderd door:
Rinus Vleesch Dubois.

In verband met auteurswetgeving en andere maatregelen op het gebied van bescherming van software kan de redactie geen aansprakelijkheid aanvaarden voor inzendingen. Inzendingen dienen afkomstig te zijn van de inzender, tenzij anders aangegeven.

INHOUDSOPGAVE DE 6502 KENNER NR. 40 OKTOBER 1985

1. Van de redactie	3.
2. SANSON 65 (OCTOPUS) ... Eric Verkuijlen	2.
3. Uitnodiging Ledenvergadering/Bijeenkomst Rijswijk	4.
4. DOS 65 Nieuw voor Uw 6502 systeem: DOS 65 ... HAVISOFT Adri Hankel/Erwin Visschedijk	5.
5. APPLE Mergen van Applesoft programma's ... Pieter de Visser Type-Ahead and Printer Buffer ... Marcel Visser Pascal Peek en Poke ... Walter Linoens, Belgie JUNIOR in APPLE ... Frans Verberkt	6. 12. 31. 39.
6. COMMODORE 64 Uni Lister (grafische tekens vertaald naar tekst) ... R. Franssen Cassette Inlegvel Plotten ... Fer Weber	7. 37.
7. JUNIOR C64JUN: JUNIOR leest C-64 tapes ... R.A.F. Bens	42.
8. PROTON PC-2 Hex/Ascii-demo voor de Proton PC-2 computer ... Simon Voortaan	28.
9. VIC-20 Recovering BASIC Programs when All Seems Lost ... Fred Behringer, Duitsland Het verbinden van programma's met de VIC-20 ... Fred Behringer, Duitsland	29. 38.
10. ALGEMEEN Interruot in de 6502 ... Rud Uhoff Description of Copying Journal-Layouts ... Siegfried Losensky, Duitsland	26. 41.
11. BASIC Tokenized Microsoft Basic Keywords and Addresses AIM-65 ... Willeen L. van Pelt/Phons Bloemen	45.
12. FORTH Patches for PDS-65 FORTH ... Gert van Oobroek LIST zonder regelnummers, etc. ... Frans Bakx	48. 49. 41.
13. Vragen van Leden	2, 11, 23, 41, 47.
14. Vraag en Aanbod	2, 3, 41, 47.
15. Diversen	6, 47.
16. HCC-computerdagen 22/23 november 1985	24, 49.
17. Boekinformatie	
18. HARDWARE Bouw eens een Miljonoer ... Jan Vernimmen Basicode Switch-Unit ... B. de Bruine	24. 35.

DE 6502 KENNER

FATE 65 is a 12K program package, originally written for the Elektor JUNIOR-computer, but easy to install on other computers for those who are familiar with programming in machine language.

For FATE 65 you need following configuration:

- * Extended Elektor JUNIOR-computer (motherboard + interfaceboard) with Tape Monitor (TM) and Printer Monitor (PM).
- * At least 16K free RAM memory, better 32K. Extension with addresses from \$2000 and up.
- * The Elekterminal.
- * Two tape recorders with motor control (or one).

The heart of FATE 65 consist of:

- * The central editor with 27 different In- and Outout commands, with 14 commands to change, to delete and to search, with 9 function keys and with 9 program control commands.
- * The two-pass resident assembler, conform specifications of the standard MOS Technology 6502 language, which works completely independent, with 16 commands to serve the value definition, listing control, in- and outout control, conditional assembly, and to make a block structure. For instance, you are able now to give blank lines in the listing, to give your own title on top of the page, to generate a headline in the listing, to force a jump to a new page, to link files, etc. etc. With the assembler you can create object code from cassette to cassette, from memory to memory, from cassette to memory and from memory to cassette.
- * The format lister to print textfiles and to create a page concept automatically. The software contains 12 commands in OKI Microline 80 code, easy to change if you own another printer.
- * The object loader to load and save files from and to cassette.
- * The merge and solit routines to merge two source files into one or to solit one file into two.

Memory map of FATE 65:

0000 - 00CE	Page zero working storage	5000 - 8FFF	Source buffer
0100 - 0164	Page one working storage	9000 - 9FFF	Symbol table
0200 - 02FF	Tape input buffer		
0300 - 03FF	Tape output buffer		
0400 - 043A	Working storage		
2000 - 4FFF	Program		

Our member Rob Banen developed the complete heavily commented source listing and manual, published with permission of Proton Electronics, Naarden, The Netherlands.

MANUAL, Dutch version	Hfl.	35,00
MANUAL, English version	Hfl.	35,00

SOURCE LISTING, English version only	Hfl.	110,00
--------------------------------------	------	--------

MANUAL Dutch or English version + Cassette only for JUNIOR computer (KIM/JUNIOR hypertape format)	Hfl.	47,50
---	------	-------

Send Eurocheque to Mr. W.L. Pelt, Jacob Jordaensstraat 15, 2923 CK Krimpden aan den IJssel, The Netherlands.

Foreign countries: IF NOT PAYING WITH EUROCHEQUE, YOU HAVE TO PAY HFL. 7,50 EXTRA TRANSFERS !!!

TE KOOP AANGEBODEN.		
8" FDOS controllerkaart met FD 1771	Hfl.	90.00
2 * 8K RAM/ROM kaart met 2 * 8K RAM	Hfl.	100.00
en in Ecom's FDOS 1.2 voor 8"		
Verkoop ten behoeve van een Belgisch		
lid. Voor informatie of bezichtiging		
bel 01807-19881 (redactie).		
Verzendkosten zijn niet in de prijs		
inbegrepen.		

S_A_M_S_O_N_6_5_ (O_C_T_O_P_U_S_)_

With the OSI-computer one may stop the computer during outout to screen by means of the command (CTRL-S), and go on with the outout by (CTRL-Q). These commands are very handy during listing on screen of a Basic program, or during outout to the printer. These commands may be added to the Elektor SAMSON65 (= OCTOPUS) computer after initializing program below on track 0.

\$24CD	48	PHA		
\$24CE	AD 0D E1	LDA VAIFR		
\$24D1	29 02	AND #02	KEY	
			PRESSED?	
\$24D3	F0 19	BEQ #24EE	NO? BACK	
\$24D5	AD 01 E1	LDA VAPAD		
\$24D8	29 7F	AND #7F	REMOVE	
			HIGH BIT	
			(CTRL-C)?	
\$24DA	C9 03	CMP #03		
\$24DC	D0 05	BNE #24E3		
\$24DE	8D 25 23	STA KPDD		
\$24DF	68	PLA		
\$24E2	60	RTS		
\$24E3	C9 13	CMP #13	(CTRL-S)?	
\$24E5	D0 07	BNE #24EE	NO? BACK	
\$24E7	20 1D F7	JSR RECCHA		
\$24EA	C9 11	CMP #11	(CTRL-Q)?	
\$24EC	D0 F9	BNE #24E7		
\$24EE	68	PLA		
\$24EF	8D 63 23	STA AHOLD		
\$24F2	20 00 F0	JSR VIDEO		
\$24F5	60	RTS		

CHANGE OUTPUT TABLE ON ADDRESS 2311-2314 !!!

\$2311	CC
\$2312	24
\$2313	CC
\$2314	24

With the command (CTRL-G) you may switch (Centronics)printer on and off on each moment you wish. Only change two addresses below on track 0.

\$258A	C9
\$258B	07

Eric Verkuijlen
Hoofdstraat 86
5473 AT HEESWIJK-DINTHER
PHONE: 04139-1019
THE NETHERLANDS.

PLEASE SEND YOUR OWN PROGRAMS TO THE EDITORS OFFICE
C/O WILLEM L. VAN PELT
JACOB JORDAENSSTRAAT 15
2923 CK KRIMPEN A.D. IJSSEL
PHONE: 01807-19881

Het effect van ons aller inspanningen om nieuwe leden te werven is moeilijk te meten. Een paar dingen zijn in de loop van de tijd wel heel duidelijk geworden. Het verloop van leden is een jaarlijks terugkerende zorg voor bestuur en redactie. Hoewel het aantal jaarlijkse onzeggingen omlaand genoeg zeer stabiel blijft, het aanboren van bronnen om nieuwe leden te werven wordt steeds moeilijker. Tot voor kort zagen we steeds kans het aantal leden ook op het zelfde peil te houden. De les die hieruit geleerd kan worden is, dat de club nog steeds een enorme aantrekkingskracht uitoefent, maar dat de toenemende geavanceerde ontwerpen van andere computers een behoorlijke concurrentie veroorzaken. Hobbyisten als wij willen steeds meer geheugen, steeds meer mogelijkheden, steeds intelligenter programma's. Het is het bekende effect: als de detailhandel maar veel schreeuwt, dan zal de uiteindelijk weinige wel die feitelijk wordt aangeboden niet zo direct in het oog lopen. Radio's, televisies, video, recorders, alles wordt aangeboden met zoveel mogelijk knopjes, lampjes, displays, stations, scammers, auting, direct-drives, dubbing, en ten laatste ontdekt men dat het meeste of optioneel is of niet gebruikt zal worden. Men heeft bij lange na nog niet het eigen apparaat uitgebaat. Als het nu maar in de sfeer van de 65XX bleef, dan maakt het voor het aantal leden niet zo uit. We bezinnen ons daarom op de toekomst. En intussen doen we een beroep op de leden om mee te helpen het aantal leden van de club op te voeren. Nieuwe leden zijn van het grootste belang om de informatie-voorziening op peil te houden. Dat is immers het belang dat we erbij hebben. Het bestuur en de redactie doen natuurlijk als voorheen gewoon mee. En dat heeft in het verleden bewezen effect opgeleverd.

Zo legden we bij diverse handelaren informatie over onze club neer. Dat kost natuurlijk geld. Het effect was er wel maar toen het nieuwtje eraf was zag men het belang niet meer zo zitten. De redactie is er mee gestopt.

Leden die contact hebben met de redactie, ontvangen steeds een informatie-formulier met aanmeldingstroom. In de afgelopen 9 maanden leverde dat 5 nieuwe leden op. Met een beetje meer inspanning zouden dat er meer kunnen zijn. Het blijkt positief uit te werken.

Op de MCC-dagen schreven we meteen 10 nieuwe leden in, daarna nog een onbekend aantal, in de eerste 9 maanden van 1985 nog eens drie. Tegenover de kosten lijkt dit weinig, maar het uitdragen van de naam van onze club, of liever! het bestaan en de aard van onze club heeft ook zo zijn positieve gevolgen.

Zo zijn er mensen togetreden die het bestaan van onze club pas ontdekten nadat zij via de NOS Hobbyscoop een Basiccode-programma uitzonden, of hun eigen bestaan op andere wijze kenbaar maakten en van ons informatie ontvingen.

Elektuur heeft op de instroom van leden ook invloed. Toen zij in Duitsland met de SAMSONS-computer uitkwamen, maakten zij het bestaan van onze club bekend. Tot nu toe leverde dat 9 Duitse, 1 Oostenrijks, 2 Belgische en 1 in Zambia woonachtig lid op.

In 1985, tot september, verhoudt de instroom van leden zich als volgt: bij elke drie inschrijvingen is er nu een buiten Nederland wonende.

Het jaar is nog niet o.a. Als club zijn we nog niet groter geworden. We hebben dat echter wel hard nodig. Het bestuur en de redactie willen meer doen voor de leden. Daar is meer geld voor nodig. Meer geld kun je bijvoorbeeld krijgen door meer leden. Het bestuur en de redactie zetten hun beste beentje al jaren voor. Zij vragen nu aan U te helpen, meer dan voorheen al het geval was, bij het werven van nieuwe leden. In de editie is een los informatieblad toegevoegd. U kent vast wel in Uw directe omgeving iemand met een 65XX computer. Geef hem/haar het blad. Wilt U als tussenpersoon werken? Geef naam en adres van de redactie op, met eigen naam en adres, aan de gemeentegids van de gemeente waar U woont.

Deze editie is een beetje bijzonder. Het is nummer 40, en doet me denken aan editie 20. Een editie die uitroeuen werd als dubbele editie in verband met de viering van het lustrum en die geheel uitverkocht werd. We hebben nu geen lustrum te vieren, maar we weten hieruit wel op te maken dat evenveel edities werden uitgegeven in einder tijd. We hopen nog steeds het volgende jaar uit te kunnen komen met een zeede editie, zoals dat de afgelopen paar jaar het geval is geweest, zonder dat de lidmaatschapsbijdrage omlaand moest. Rustig van geld natuurlijk, maar U weet inmiddels hoe U daaraan mee kunt helpen.

Het werven van leden mag dan een zorg zijn, het werven van materiaal voor publicatie in DE 6502 KENNER heeft ook de voortdurende aandacht van bestuur en redactie. Velen hebben inmiddels op de redactie zelf met eigen ogen kunnen waarnemen wat er zo allemaal moet kijken voor er gepubliceerd kan worden. Ik heb wel eens iemand met grijze haren hier weg zien gaan... Maar het moet gezegd: men laat mij niet gemakkelijk zitten met mijn onontbeerbare honger naar nog meer materiaal. De mensen met andere computers dan JUNIOR blijken toch wel degelijk auteurs in soe. Eensmaal over de drempel, dan blijkt hoe enthousiast men kan worden. Zo erg zelfs dat het hier soms een onoverzichtelijke puinhoop voor de buitenstaander lijkt en ik moeite heb het met enig slaperig wordende oogleden te verwerken. Maar het werkt! Als de mensen maar geloven dat het echt niet zo veeleisend is en dat het niet aan tijd is gebonden, dat het een kwestie is van gewoon doen waar je zin in hebt, als je de redactie maar niet vergeet. Die zit immers te springen om je materiaal te verwerken en aan de copy-buffer toe te voegen en een goed contact met je op te bouwen dat soms veel verder gaat dan alleen maar helpen bij het totstandkomen van een publicatie.

In deze veertigste editie ligt het resultaat van de positieve medewerking die er uit de hoek van APPLE-bezitters kwam, en waarvan nog veel meer te verwachten is. Maar het is extra bijzonder als we ook kunnen vaststellen dat we meteen aan kunnen sluiten bij de komst van de OCTOPUS, en het feit dat daarbij de aandacht ook wel zal worden gericht op PASCAL. Daar moet nog bij dat de leden in Duitsland zich ook niet onbetuigd laten. Wellicht dat we van hen nog meer mogen verwachten. Het zal wel duidelijk zijn dat ik hoop ook van de andere computers software of hardware te mogen ontvangen, waarbij ik vooral denk aan de BBC en de ATARI. Tenslotte nog een opmerking over het door de club ondersteunde systeem MONS/D0865, door Adri Hinkel en Erwin Visschedijk, Nico de Vries en Ad Brauner mogelijk gemaakt. De redactie zal dit systeem ten volle ondersteunen. Het is uniek van aard en geeft de zelfbouwer zeker een klap voor de komende jaren. Ik heb vernomen dat er het nodige aan publiceerbaar materiaal zit aan te komen. Het is wel aan te bevelen het systeem in z'n geheel aan te schaffen, oadat, vooral waar het de volledige documentatie betreft, niet kan worden gegarandeerd dat de prijs hetzelfde zal blijven. Men moet bovendien slechts eenmaal in aanmerking voor deze aanbieding. Voor de totaalrijke heeft een het niet te laten.

Willeen L. van Pelt.

In 1986 wilt U vast de edities van DE 6502 KENNER blijven ontvangen. Betaal daarom op tijd, d.w.z. uiterlijk in de maand december a.s. Uw lidmaatschap ad Hfl. 45,00 aan de penningmeester van de KIM Gebruikers Club Nederland te Krienen a.d. IJssel (Johm van Sorang), door overschrijving of storting op postrekening 3757649.

Leden in België of elders buiten Nederland gebruiken bij voorkeur een Eurocheque. Indien niet oer Eurocheque betaald wordt, dient men Hfl. 7,50 extra transfers te betalen.

PLEASE PAY YOUR 1986 SUBSCRIPTION ON EUROCHEQUE IN DECEMBER AT THE LATEST. IF NOT PAYING WITH EUROCHEQUE, YOU HAVE TO PAY HFL. 7,50 EXTRA TRANSFERS.

UITNODIGING BIJeenkomst

Datum : zaterdag 16 november 1985
 Lokatie : R.K. HTS "Rijswijk". Tel.: 070 - 907839
 Lange Kleiweg 4 te RIJSWIJK.

Route :

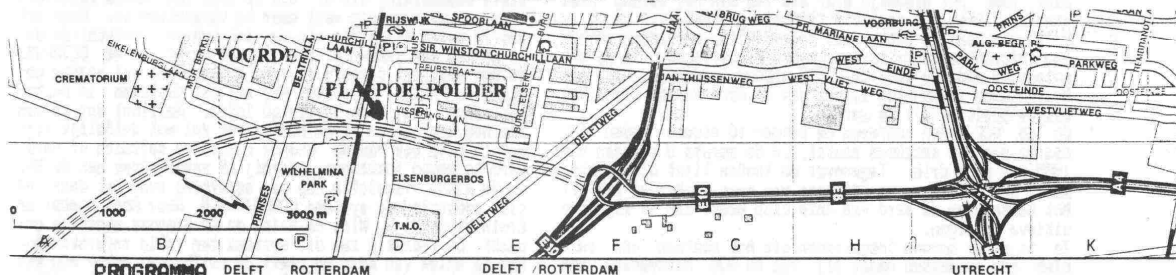
- per auto - komende uit de richting Utrecht
 Volg autoweg E8 Utrecht-Den Haag. Knooppunt Leidschendam via de hoge rijbaan linksaf richting Den Haag Zuid/Rijswijk. Aan het eind linkerbanen aanhouden en meebuigen naar rechts. Rechts blijven aanhouden, en met trambaan mee over Hoornbrug. Dan scherpe draai naar rechts en onder brug door, weg blijven volgen tot T-splitsing. Linksaf en richting Wateringen aanhouden via de Winston Churchilllaan. Laatste stoplichten voor de spoorwegovergang linksaf Huis te Landelaan op. Deze uitrijden tot het eind en dan rechtsaf. Uitrijden tot schoololein.

komende uit de richting Amsterdam
 Volg A4 A'dam-Rotterdam. Knooppunt Leidschendam passeren. Verder als hierboven.

komende uit de richting Rotterdam
 Volg E10 Rotterdam-Den Haag Zuid en passeer Hoornbrug als hierboven aangegeven. Verder als hierboven.

- per trein - Station Rijswijk uitgang voor de richting Steenvoordelaan/Winkelcentrum De Boogaard nemen. Bij uitgang direkt linksaf tot aan spoorwegovergang. Deze oversteken. Men kan dan even verder parallel aan spoorbaan weg volgen tot de HTS "Rijswijk".

Lunchoakket zelf meenemen
 Consumpties tegen betaling



PROGRAMMA DELFT ROTTERDAM

DELFT / ROTTERDAM

UTRECHT

10.00 OPENING LEDENVERGADERING 1985.

10.15 CONCEPT- BEGROTING 1986.

VERKIEZING KASCONTOLECOMMISSIE.

VERKIEZING BESTUURSLEDEN:

afredend en herkiesbaar: voorz. Rinus Vleesch Dubois
 pennm. John van Sprang
 lid Adri Hankel
 lid Erwin Visschedijk

verkiezing 1 bestuurslid (kandidaten kunnen schriftelijk worden aangemeld bij het sekretariaat of mondeling voor de aanvang van de vergadering).

MEDEDELING over c.d. uitreiking publikatie-aanmoedigingspremie(s).

RONDVRAAG en SLUITING.

12.00 LUNCH.

13.00 LEZING Nico de Vries. Behandeling software mogelijkheden van de 16-bitter 65816.

14.00 INFORMEEL GEDEELTE.

In het informeel deel wordt ieder in de gelegenheid gesteld kennis met elkaars systemen te maken. Breng daarom Uw systeem mee!

Heeft U hardware aan te bieden? Doe dat op een eigen tafel.

Heeft U software zelf ontwikkeld? Geef dat aan de redaktie mee.

Kopieren van software waarop enige vorm van auteursrechten rusten is binnen onze geleiden geen gebruik. Doe dat ook niet op de bijeenkomst.

17.00 Sluiting.

DE 6502 KENNER

```
*****
*
*      NIEUW VOOR UW 6502 SYSTEEM:
*
*      D O S 6 5
*      -----
*
*****
```

Wat is DOS65:

DOS65 is een operating-system, in principe geschikt voor elk 6502-systeem. Het is ontwikkeld door A.B.M. Brouwers, ten behoeve van implementatie op de zogenaamde uitgebreide Junior. Door E.J.M. Visschedijk en A.S. Hankel werd het aangepast voor implementatie op een configuratie van Elektuur's CPU/VDU kaarten.

DOS65 bestaat uit een 'Kernel', 8Kbyte groot, en een groot aantal hulproutine's (utility's), waaronder een full screen editor/word processor. DOS65 ondersteunt de programmeertalen BASIC, FORTH, alsmede Micro-Ade en de Moser assembler, en in de toekomst wellicht nog andere, wo. PASCAL en 'C'. DOS65 is geschikt voor alle type's disk-drives met 40 tracks per kant, dus zowel single- als double-sided, single- als double density. De maximale schijfkapaciteit (geformatteerd) is 304 Kbyte. DOS65 wordt, afhankelijk van de implementatie, ondersteund door de monitor van A.B.M. Brouwer. of de MON65 monitor van HaViSOFT.

Wat er aan hardware nodig is:

Wordt DOS65 ge-implementeerd voor de CPU/VDU configuratie, dan dient de volgende hardware aanwezig te zijn:

- Elektuur's CPU-kaart inkl. 2 Kbyte RAM en 8 Kbyte ROM (Deze ROM bevat MON65).
- Elektuur's VDU-kaart met speciale karakter-ROM.
- DOS65 floppy-controller-kaart.
- 48 Kbyte RAM (van \$0000 tot \$C000).

De print voor de floppy-controller, de monitor-ROM, de speciale karakter-ROM, en DOS65 zijn leverbaar via de software-service van onze club. Alle beschikbare handleidingen zijn geschreven in het nederlands.

Wat het systeem kan:

Verkort kommando overzicht DOS65.

APPEND	plaats een file achter een andere
CAT	geef catalog van een schijf
COPY	kopieer file of file's

CREATE	kreeer een ascii-file
DELETE	verwijder file of file's
DIR	geef de directory van een schijf
DUMP	laat file als hexdump zien
EDITOR	start de full screen editor
FORMAT	formateer een schijf
LIST	laat een ascii-file zien
LOAD	laad een file in het geheugen
MEMFILL	vul een stuk geheugen
MEMMOVE	verplaats een stuk geheugen
PLIST	laat ascii-file zien op pagina formaat
PRINT	print een file op de printer
RENAME	geef een file een andere naam
RUN	executeer een file
SAVE	schrijf geheugen naar schijf
SDIR	laat een gesorteerde directory zien
SETDRIVES	zet motor-on en headload tijd
SETMODE	zet de mode van een file
TIME	definieer tijd en datum

Verkort mogelijkheden-overzicht full screen editor.

- Tekst invoeren, wijzigen en verwijderen.
- Door de tekst stappen:
 - per karakter, woord, tab positie, regel of sectie, zowel voor- als achteruit.
- Zoeken naar een string, zowel voor- als achteruit. De gevonden string kan eventueel worden vervangen door een andere (zoek & vervang).
- Plaats een gedeelte van de tekst in een tijdelijke buffer.
- Voeg de buffer toe aan, of tussen, de tekst.
- Verander hoofdletters in kleine letters vice versa.
- Vul de tekst uit met spaties, zodat een rechte rechterkantlijn ontstaat.
- Verwijder overtollige spaties.
- Voer een DOS65 kommando uit.
- Lees een file in.
- Schrijf de tekst naar een file.
- Definieer/executeer een macro.
- Geef een kort mogelijkheden-overzicht (help-functie).

Kommando overzicht MON65 (HaViSOFT)

C	open een adres, laat de inhoud zien
A	laat datum zien / wijzig datum
B	zet breakpoint
C	controleer een geheugengebied
D	disassembleer een geheugengebied
E	executeer vanaf een adres
F	vul een geheugengebied
H	hexdump
L	laat de CPU registers zien
M	verplaats een geheugengebied
P	maak het scherm schoon
Q	verlaat MON65 middels een RTS instructie
S	bepaal de checksum van een geheugengebied
T	laat tijd zien / wijzig tijd
V	vergelijk twee geheugengebieden
W	zoek naar een data-patroon
X	door systeemgebruiker te definieren
Y	door systeemgebruiker te definieren
+	tel twee hex getallen bij elkaar op
-	trek twee hex getallen van elkaar af
\$	reken om van decimaal naar hex
#	reken om van hex naar decimaal
<	herhaal het kommando / de kommando's

Verder beschikt MON65 over een aantal I/O-routines, waaronder:

- Keyboard (parallel)
- RS 232
- Centronics
- Viacom

Viacom is een routine, waarmee twee computers, (die beiden over Viacom beschikken), razendsnel data uit kunnen wisselen.

Wat DOS65 kost:

DOS65 en alles wat erbij hoort, is door clubleden ontwikkeld, zonder winst oogmerk. DOS65 is dan ook tegen kostprijs verkrijgbaar. Ter indicatie:

- floppy-controller print: ca. f 50,- (dubbelzijdig, doorgemetaliseerd, met tekstopdruk)
- ROM's, schijven: bestaande verkoopprijzen
- handleidingen: ca 15 ct. per pagina

Beschikbare handleidingen:

- DOS65 handleiding
- Editor handleiding
- Technische handleiding (hardware)
- Bouwbeschrijving floppy-controller
- MON65 handleiding
- Sources MON65
- Binnenkort: Sources DOS65

Prijzen van het MON65/DOS65 systeem:

- Handleiding MON65
 - Handleiding DOS65
 - Handleiding editor
 - Hardware beschrijving
 - Samen ongeveer 110 pagina's FL. 50,-
 - 2764 Monitor Eeprom FL. 35,-
 - 2732 Karaktergenerator Eeprom FL. 25,-
 - Diskette met o.a. Micro-ADE, FORTH, Editor, en diverse utilities FL. 15,-
 - Floppy Disk Controller kaart FL. 50,-
 - Source MON65 FL. 25,-
 - Source DOS65 FL. 25,-
 - Source utilities FL. 25,-
- Deze prijzen zijn exclusief verzendkosten en gelden alleen voor leden van de KIM Gebruikers Club Ned. Ieder lid komt slechts eenmaal voor deze prijzen in aanmerking. Bestellingen moet niet zoals gebruikelijk bij de redactie worden gedaan, maar als volgt:

E.J.M. Visschedijk (Havisoft)
Drakesteyn 299
7608 TR ALMELO
Postrekening 225746
of per eurocheque
(Voor buitenland geldt: indien niet per eurocheque wordt betaald, dan moet het bedrag met Fl. 7,50 extra transfers worden verhoogd)

Vermeldt duidelijk wat gewenst wordt.

MERGEN VAN APPLESOFT PROGRAMMA'S

Heeft U niet de beschikking over een routine om Applesoft Basic programma's te mergen (= aan elkaar plakken), dan kan dat blijkbaar ook met de volgende truc:

1. Laadt het eerste programma
 2. CALL-151
 3. Maak pointer op \$67-\$68 gelijk aan pointer op \$69-\$6A minus 3
 4. Terug naar Basic en laadt tweede programma
 5. CALL-151 en geef pointer op \$67-\$68 de oude waarde terug
 6. Terug naar Basic door CTRL-C of door E0036
- Aldus Pieter de Visser uit Veldhoven.

1 BIT/DRUPPEL geeft op regenachtige dagen minstens een subroutine.

Fr. Verberkt

HCC-Comouterdagen

De in de vorige editie afgedrukte reductiebon blijkt een vergissing te zijn welke niet meer te herstellen bleek. Door het tijdstip waarop de edities moeten worden samengesteld is herstel niet meer mogelijk. Excuses.

=====

UNI L I S T E R 2

=====

DOOR R. FRANSEN

VOOR DE CBM 64

DIT PROGRAMMA VERTAALT DE GRAFISCHE
TEKENS DIE IN DE PRINT-OPDRACHTEN
STAAN NAAR DUIDELIJKE TEKST.
BIJVOORBEELD:

```
100 PRINT "V000"          OUD"
110 PRINT "{CLR,DOWN*2,RED}" NIEUW"
```

```
NIEUW LIST IS      SYS 49152
NORMALE LIST IS    SYS 49169
```

=====

UNI L I S T E R 2

=====

*** ASSEMBLING ***

2

```

; FILENAME=      UNI 16
; AUTHOR=        R.FRANSEN
; DATE=          9 MEI 1985
; PROGRAMNAME=   UNI LISTER 2
; VERSION=       CBM V1.1
;
240:  C000          .OPT 00,P
;
; SCRATCHPAD DEFINITIONS
270:  033C          *= $033C
280:  033D          CHAR  *= ++1      ; PREVIOUS CHAR. BUF.
290:  033E          NUMBER *= ++1      ;
300:  033F          COMMA  *= ++1      ;
;
320:  C000          *= $C000
;
; SET 'LIST' VECTOR AND INIT. POINTERS
;
360:  C000 A9 1C     INIT   LDA #< MAIN ; NEW LIST POINTER
370:  C002 8D 06 03     STA  $0306 ; BASIC LIST VECTOR
380:  C005 A9 C0     LDA  #>MAIN
390:  C007 8D 07 03     STA  $0307
400:  C00A 20 7D C0     JSR  CLEAR ; INIT POINTERS
410:  C00D EA         NOP      ; USER INIT
420:  C00E EA         NOP
430:  C00F EA         NOP
440:  C010 60         RTS      ; RETOUR CALLER
```

```

;
; RESET OLD 'LIST' VECTOR
;
480:  C011 A9 1A      RESET      LDA  #$1A      ; OLD LIST VECTOR
490:  C013 8D 06 03      STA  $0306      ; BASIC LIST VECTOR
500:  C016 A9 A7      LDA  #$A7
510:  C018 8D 07 03      STA  $0307
520:  C01B 60      RTS      ; RETOUR CALLER
;
; MAIN PROGRAM
;
560:  C01C 24 0F      MAIN      BIT  $0F      ; QUOTATION (") MODE
570:  C01E 30 0C      BMI  SEARCH      ; YES, TEST FOR SPECIAL CHAR.
580:  C020 C9 FF      CMP  #$FF      ; CODE FOR PHI
590:  C022 D0 03      BNE  AAA      ;
600:  C024 4C F3 A6 PRNXT      JMP  $A6F3      ; TYPE IT AND CONTINUE THE LISTING
610:  C027 10 FB      AAA      BPL  PRNXT      ; NO TOKEN, SO TYPE IT
620:  C029 4C 24 A7      JMP  $A724      ; FIND BASIC TOKEN
630:  C02C 20 CD C0 SEARCH      JSR  COMP      ; TEST FOR SPECIAL CHAR.
640:  C02F B0 15      BCS  FOUND      ; C=1 --> FOUND
650:  C031 AE 3C 03      LDX  CHAR
660:  C034 F0 EE      BEQ  PRNXT      ; NO SPECIAL CHAR.
670:  C036 48      PHA      ; SAVE CHAR.
680:  C037 20 8B C0      JSR  TYPE      ; TYPE SPECIAL CHAR. TEXT
690:  C03A A9 5D      LDA  #$5D      ; END CHAR.
700:  C03C 20 47 AB      JSR  $AB47      ; OUTPUT CHAR.
710:  C03F 20 7D C0      JSR  CLEAR      ; RESET POINTERS
720:  C042 68      PLA
730:  C043 4C 24 C0      JMP  PRNXT      ; TYPE CHAR. AND CONTINUE
;
750:  C046 AD 3C 03 FOUND      LDA  CHAR      ; IS IT THIS THE FIRST
760:  C049 D0 09      BNE  SAME      ; NO, TEST FOR SAME
770:  C04B 8E 3C 03      STX  CHAR      ; FIRST SPECIAL CHAR
780:  C04E 8D 3D 03      STA  NUMBER      ; CLEAR COUNTER
790:  C051 4C F6 A6 CONT      JMP  $A6F6      ; CONTINUE WITHOUT TYPING
;
810:  C054 EC 3C 03 SAME      CPX  CHAR      ; IS IT THE SAME CHAR.
820:  C057 D0 0E      BNE  NEW
830:  C059 18      CLC      ; SAME CHAR. SO
840:  C05A F8      SED      ; INCREMENT COUNTER (DECIMAL)
850:  C05B A9 01      LDA  #$01
860:  C05D 6D 3D 03      ADC  NUMBER
870:  C060 8D 3D 03      STA  NUMBER
880:  C063 D8      CLD
890:  C064 4C 51 C0      JMP  CONT      ; CONTINUE LISTING
900:  C067 8A      NEW      TXA
910:  C068 48      PHA
920:  C069 20 8B C0      JSR  TYPE      ; TYPE SPECIAL CHAR. TEXT
930:  C06C A9 00      LDA  #$00
940:  C06E 8D 3D 03      STA  NUMBER      ; CLEAR COUNTER
950:  C071 A9 2C      LDA  #", "      ; SET , IN COMMA
960:  C073 8D 3E 03      STA  COMMA
970:  C076 68      PLA
980:  C077 8D 3C 03      STA  CHAR      ; NEW SPECIAL CHAR
990:  C07A 4C 51 C0      JMP  CONT      ; CONTINUE LISTING

```

```

;
1010: C07D A9 00 CLEAR LDA #00
1020: C07F 8D 3C 03 STA CHAR ; CLEAR CHAR. BUFFER
1030: C082 8D 3D 03 STA NUMBER ; CLEAR COUNTER
1040: C085 A9 5B LDA #$5B ; START CHAR.
1050: C087 8D 3E 03 STA COMMA
1060: C08A 60 RTS

;
1080: C08B AD 3E 03 TYPE LDA COMMA ; TYPE , OR +
1090: C08E 20 47 AB JSR $AB47 ; OUTPUT CHAR.
1100: C091 AE 3C 03 LDX CHAR
1110: C094 BD EE C0 LOOP1 LDA TABLE,X ; READ TEXT
1120: C097 F0 07 BEQ NUM ; EOL = 00
1130: C099 20 47 AB JSR $AB47 ; OUTPUT CHAR.
1140: C09C E8 INX
1150: C09D 4C 94 C0 JMP LOOP1 ; NEXT CHAR.
1160: C0A0 AD 3D 03 NUM LDA NUMBER
1170: C0A3 F0 27 BEQ RET ; NUMBER = 0
1180: C0A5 A9 2A LDA #$2A ; TYPE "*"
1190: C0A7 20 47 AB JSR $AB47 ; OUTPUT CHAR.
1200: C0AA 18 CLC ; INCREMENT COUNTER
1210: C0AB F8 SED
1220: C0AC A9 01 LDA #$01
1230: C0AE 6D 3D 03 ADC NUMBER
1240: C0B1 8D 3D 03 STA NUMBER
1250: C0B4 D8 CLD
1260: C0B5 4A LSR ; CONVERT DECIMAL NUMBER
1270: C0B6 4A LSR ; TO A TWO DIGIT ASCII
1280: C0B7 4A LSR ; VALUE
1290: C0B8 4A LSR
1300: C0B9 F0 06 BEQ LOW ; NO HIGH BYTE
1310: C0BB 18 CLC
1320: C0BC 69 30 ADC #$30
1330: C0BE 20 47 AB JSR $AB47 ; HIGH BYTE OUTPUT CHAR.
1340: C0C1 AD 3D 03 LOW LDA NUMBER
1350: C0C4 29 0F AND #$0F
1360: C0C6 18 CLC
1370: C0C7 69 30 ADC #$30
1380: C0C9 20 47 AB JSR $AB47 ; LOW BYTE OUTPUT CHAR.
1390: C0CC 60 RET

;
1410: C0CD A2 00 COMP LDX #$00
1420: C0CF DD EE C0 LOOP2 CMP TABLE,X
1430: C0D2 F0 14 BEQ OK ; SPECIAL CHAR. FOUND IN TABLE
1440: C0D4 E8 INX
1450: C0D5 48 PHA
1460: C0D6 BD EE C0 LOOP3 LDA TABLE,X ; FIND EOL
1470: C0D9 D0 05 BNE END
1480: C0DB E8 INX
1490: C0DC 68 PLA
1500: C0DD 4C CF C0 JMP LOOP2
1510: C0E0 C9 FF END CMP #$FF ; END OF TABLE
1520: C0E2 F0 07 BEQ NOT
1530: C0E4 E8 INX
1540: C0E5 4C D6 C0 JMP LOOP3

```

```

;
1560: C0E8 E8      OK      INX      ; SET FOR FOUND
1570: C0E9 38      SEC
1580: C0EA 60      RTS
1590: C0EB 68      NOT      PLA      ; RESET STACK
1600: C0EC 18      CLC      ; CLR FOR NOT FOUND
1610: C0ED 60      RTS

;
; TABLE FORMAT (CHAR, TEXT, 00)
;
; COLOR KEY TABLE
;
1670: C0EE 90      TABLE .BYTE$90
1670: C0EF 42 4C 41 .ASC "BLACK"
1680: C0F4 00 05 .BYTE$00, $05
1680: C0F6 57 48 49 .ASC "WHITE"
1690: C0FB 00 1C .BYTE$00, $1C
1690: C0FD 52 45 44 .ASC "RED"
1700: C100 00 9F .BYTE$00, $9F
1700: C102 43 59 41 .ASC "CYAN"
1710: C106 00 9C .BYTE$00, $9C
1710: C108 50 55 52 .ASC "PURPLE"
1720: C10E 00 1E .BYTE$00, $1E
1720: C110 47 52 45 .ASC "GREEN"
1730: C115 00 1F .BYTE$00, $1F
1730: C117 42 4C 55 .ASC "BLUE"
1740: C11B 00 9E .BYTE$00, $9E
1740: C11D 59 45 4C .ASC "YELLOW"
1750: C123 00 81 .BYTE$00, $81
1750: C125 4F 52 41 .ASC "ORANGE"
1760: C12B 00 95 .BYTE$00, $95
1760: C12D 42 52 4F .ASC "BROWN"
1770: C132 00 96 .BYTE$00, $96
1770: C134 4C 2E 52 .ASC "L. RED"
1780: C139 00 97 .BYTE$00, $97
1780: C13B 44 2E 47 .ASC "D. GRAY"
1790: C141 00 98 .BYTE$00, $98
1790: C143 4D 2E 47 .ASC "M. GRAY"
1800: C149 00 99 .BYTE$00, $99
1800: C14B 4C 2E 47 .ASC "L. GREEN"
1810: C152 00 9A .BYTE$00, $9A
1810: C154 4C 2E 42 .ASC "L. BLUE"
1820: C15A 00 9B .BYTE$00, $9B
1820: C15C 4C 2E 47 .ASC "L. GRAY"

;
; CURSOR KEY TABLE
;
1860: C162 00 12 .BYTE$00, $12
1860: C164 52 56 53 .ASC "RV5 ON"
1870: C16A 00 92 .BYTE$00, $92
1870: C16C 52 56 53 .ASC "RV5 OFF"
1880: C173 00 13 .BYTE$00, $13
1880: C175 48 4F 4D .ASC "HOME"
1890: C179 00 93 .BYTE$00, $93

```

```

1890: C17B 43 4C 52 .ASC "CLR"
1900: C17E 00 14 .BYTE$00,$14
1900: C180 44 45 4C .ASC "DEL"
1910: C183 00 94 .BYTE$00,$94
1910: C185 49 4E 53 .ASC "INST"
1920: C189 00 11 .BYTE$00,$11
1920: C18B 44 4F 57 .ASC "DOWN"
1930: C18F 00 91 .BYTE$00,$91
1930: C191 55 50 .ASC "UP"
1940: C193 00 1D .BYTE$00,$1D
1940: C195 52 49 47 .ASC "RIGHT"
1950: C19A 00 9D .BYTE$00,$9D
1950: C19C 4C 45 46 .ASC "LEFT"

```

; FUNCTION KEY TABLE

```

1990: C1A0 00 85 .BYTE$00,$85
1990: C1A2 46 31 .ASC "F1"
2000: C1A4 00 86 .BYTE$00,$86
2000: C1A6 46 33 .ASC "F3"
2010: C1A8 00 87 .BYTE$00,$87
2010: C1AA 46 35 .ASC "F5"
2020: C1AC 00 88 .BYTE$00,$88
2020: C1AE 46 37 .ASC "F7"
2030: C1B0 00 89 .BYTE$00,$89
2030: C1B2 46 32 .ASC "F2"
2040: C1B4 00 8A .BYTE$00,$8A
2040: C1B6 46 34 .ASC "F4"
2050: C1B8 00 8B .BYTE$00,$8B
2050: C1BA 46 36 .ASC "F6"
2060: C1BC 00 8C .BYTE$00,$8C
2060: C1BE 46 38 .ASC "F8"

```

; SPECIAL KEY TABLE

```

2100: C1C0 00 0D .BYTE$00,$0D
2100: C1C2 52 45 54 .ASC "RETURN"
2110: C1C8 00 0E .BYTE$00,$0E
2110: C1CA 4C 2E 43 .ASC "L.CASE"
2120: C1D0 00 0E .BYTE$00,$0E
2120: C1D2 55 2E 43 .ASC "U.CASE"
2130: C1D8 00 08 .BYTE$00,$08
2130: C1DA 44 49 53 .ASC "DIS"
2140: C1DD 00 09 .BYTE$00,$09
2140: C1DF 45 4E 41 .ASC "ENA"
2150: C1E2 00 03 .BYTE$00,$03
2150: C1E4 53 54 4F .ASC "STOP"
2160: C1E8 00 FF FF .BYTE$00,$FF,$FF ; END OF TABLE

```

+C000-C1EB

READY.

Te koop:

=====
 PROTON PC-2, 4k RAM, 6k Monitor,
 10k Basic interpreter/compiler
 Nieuwwaarde f 1000,- voor f 500,-
 Te bevragen bij: Simon Voortman,
 Beatrixweg 28, 3253 BB Ouddorp

ASM

```

1  *
2  ****
3  **
4  **  TITLE: TYPE-AHEAD AND PRINTER BUFFER
5  **
6  **  THIS PROGRAM CONSISTS OF TWO PARTS:
7  **    1) THE LAUNCHER
8  **    2) THE ROUTINES
9  **
10 **
11 **    -> WARNING <-
12 **
13 **  DO NOT BRUN THIS PROGRAM TWICE! THE RESET-KEY
14 **  WILL RUN MAD!
15 **  THIS ROUTINE PATCHES DOS, SO YOU CANNOT REMOVE
16 **  IT WITHOUT REBOOTING DOS.
17 **
18 **  COMPUTER: APPLE II WITH DOS 3.3 OR DIVERSI-DOS
19 **
20 **  AUTHOR: M.J. VISSER
21 **    PASTOOR KONIJNSTRAAT 48
22 **    1616 BX HOOBKARSPEL
23 **
24  *
25  *  GLOBAL ADDRESSES
26  *
27  BEGIN    EQU  $1000    ;START OF ROUTINES
28  *
29  ****
30  **
31  **  LAUNCHER:
32  **
33  **    - CHECK FOR RIGHT DOS (DOS 3.3 OR DIVERSI-DOS)
34  **    - MAKE ROOM BETWEEN DOS AND ITS BUFFERS
35  **    - RELOCATE THE ROUTINES INTO THIS NEW SPACE
36  **    - PATCH DOS COMMANDS PR# AND IN#
37  **    - PATCH RWT$
38  **    - PATCH RESET
39  **    - ACTIVATE INPUT- AND OUTPUTHOOKS
40  **
41  ****
42  *
43  *  LOCAL ADDRESSES
44  *
45  LENGTH    EQU  $2F    ;LENGTH OPCODE
46  TEMP      EQU  $3C
47  SRCPTR    EQU  $3C    ;SOURCE POINTER
48  SRCEND    EQU  $3E    ;END OF SOURCE POINTER
49  RELPOS    EQU  $40    ;REL. POSITION OF ADDRESS
50  TRGTPTR   EQU  $42    ;TARGET POINTER
51  TARGET    EQU  $44    ;ABSOLUTE TARGET ADDRESS
52  BYTES     EQU  $46    ;BUFFER FOR OPCODE
53  STACK     EQU  $100

```

```

54 WRMSTRT EQU $3D0 ;DOS WARMSTART ADDRESS
55 RWTS EQU $3DA ;DOS RWTS ADDRESS
56 RESET EQU $3F2 ;AUTOSTART RESET HANDLER
57 PWRUP EQU $3F4 ;POWER-UP BYTE
58 MAKEBUF EQU $A7D4 ;BUILDS DOS BUFFERS
59 INSDS2 EQU $F88E ;DETERMINES LENGTH OF OPCODE
60 NXTA4 EQU $FCB4 ;INC SRCPTR & TRGTPTR AND CHECK END
61 *
62 * CHECK FOR DIVERSI-DOS OR DOS 3.3
63 * IF NOT FOUND THEN EXIT LAUNCHER
64 *
1000: 18 65 ENTRY CLC ;VERSION # IS AT OFFSET $16BE
1001: A9 BE 66 LDA #$BE
1003: 85 3C 67 STA TEMP
1005: AD D2 03 68 LDA WRMSTRT+2
1008: 69 16 69 ADC #$16
100A: 85 3D 70 STA TEMP+1
100C: A0 00 71 LDY #$00
100E: B1 3C 72 LDA (TEMP),Y ;VERSION
1010: C9 03 73 CMP #3 ;IF DOS 3.3 THEN CONTINU
1012: F0 01 74 BEQ MAKEROOM
1014: 60 75 RTS ; ELSE EXIT
76 *
77 * MAKE ROOM FOR THE ROUTINES BETWEEN
78 * DOS AND ITS BUFFERS AND INITIALIZE
79 * THE TARGET POINTER.
80 *
1015: 98 81 MAKEROOM TYA ;Y=0
1016: 85 3C 82 STA TEMP ;GET POINTER TO FIRST BUFFER
1018: AD D2 03 83 LDA WRMSTRT+2 ;THIS POINTER IS LOCATED AT $9D00
101B: 85 3D 84 STA TEMP+1 ; (48K APPLE)
101D: 38 85 SEC ;POINTER := POINTER - LEN ROUTINES
101E: B1 3C 86 LDA (TEMP),Y
1020: E9 A1 87 SBC #<END-BEGIN
1022: 91 3C 88 STA (TEMP),Y ;PUT THE NEW POINTER ON $9D00
1024: AA 89 TAX ;AND SAVE IT IN THE X & Y REGISTERS
1025: C8 90 INY
1026: B1 3C 91 LDA (TEMP),Y
1028: E9 03 92 SBC #>END-BEGIN
102A: 91 3C 93 STA (TEMP),Y
102C: A8 94 TAY
102D: 18 95 CLC ;INITIALIZE THE ABSOLUTE TARGET
102E: 8A 96 TXA ; ADDRESS AND THE TARGET POINTER
102F: 69 26 97 ADC #38 ; 38 BYTES ABOVE THE FIRST DOS
1031: 85 44 98 STA TARGET ; BUFFER.
1033: 85 42 99 STA TRGTPTR
1035: 98 100 TYA
1036: 69 00 101 ADC #00
1038: 85 45 102 STA TARGET+1
103A: 85 43 103 STA TRGTPTR+1
103C: 20 D4 A7 104 NWBUF JSR MAKEBUF ;REBUILD THE DOS BUFFERS
105 *
106 * RELOCATE THE ROUTINES INTO THE
107 * NEWLY CREATED SPACE.
108 *

```

B03F: 18	109	CLC	; WHERE AM I?
B040: BA	110	TSX	; RETURN ADDRESS STILL ON STACK
B041: BD	111	DFB	\$BD ; LDA STACK-1,X
B042: FF 00	112	DFB	\$FF,\$00
B044: 69 E0	113	ADC	#<ENDIT-NWBUF+1 ; CALC THE POSITION OF THE
B046: 85 3C	114	STA	SRCPTR ; ROUTINES BY ADDING BYTES TO THE
B048: BD 00 01	115	LDA	STACK,X ; RETURN ADDRESS AND STORE THIS
B04B: 69 00	116	ADC	#>ENDIT-NWBUF+1 ; ADDRESS IN SRCPTR
B04D: 85 3D	117	STA	SRCPTR+1
B04F: 18	118	CLC	; CALCULATE THE END POSITION OF THE
B050: A5 3C	119	LDA	SRCPTR ; ROUTINES BY ADDING THE LENGTH TO
B052: 69 A1	120	ADC	#<END-BEGIN ; SRCPTR AND STORE THIS ADDRESS
B054: 85 3E	121	STA	SRCEND ; IN SRCEND
B056: A5 3D	122	LDA	SRCPTR+1
B058: 69 03	123	ADC	#>END-BEGIN
B05A: 85 3F	124	STA	SRCEND+1
B05C: A0 02	125	RELOCATE LDY	##02 ; MOVE 3 BYTES FROM SOURCE INTO
B05E: B1 3C	126	TAKE3BYT LDA	(SRCPTR),Y ; BUFFER
B060: 99 46 00	127	STA	BYTES,Y
B063: 88	128	DEY	
B064: 10 F8	129	BPL	TAKE3BYT
B066: 20 8E F8	130	JSR	INSDS2 ; DETERMINE THE LENGTH OF THE OPCODE
B069: A6 2F	131	LDX	LENGTH ; LENGTH = LENGTH -1
B06B: E0 02	132	CPX	##02 ; IF ABSOLUTE ADDRESSING THEN
B06D: D0 25	133	BNE	MOVEBYTES
B06F: A9 A1	134	LDA	#<END ; IF ADDRESS > END THEN MOVE
B071: C5 47	135	CMF	BYTES+1 ; (NOTE THE COMPACTNESS OF THIS
B073: A9 13	136	LDA	#>END ; UNSIGNED INTEGER COMPARISON!)
B075: E5 48	137	SBC	BYTES+2
B077: 90 1B	138	BCC	MOVEBYTES
B079: A5 47	139	LDA	BYTES+1 ; IF ADDRESS < BEGIN THEN MOVE
B07B: E9 00	140	SBC	#<BEGIN
B07D: 85 40	141	STA	RELPOS
B07F: A5 48	142	LDA	BYTES+2
B081: E9 10	143	SBC	#>BEGIN
B083: 85 41	144	STA	RELPOS+1
B085: 90 0D	145	BCC	MOVEBYTES
B087: 18	146	CLC	; ELSE
B088: A5 40	147	LDA	RELPOS ; RELOCATE THE ABSOLUTE ADDRESS
B08A: 65 44	148	ADC	TARGET
B08C: 85 47	149	STA	BYTES+1
B08E: A5 41	150	LDA	RELPOS+1
B090: 65 45	151	ADC	TARGET+1
B092: 85 48	152	STA	BYTES+2
B094: A2 00	153	MOVEBYTES LDX	##00 ; MOVE LENGTH BYTES TO TARGET
B096: B5 46	154	MOVE LDA	BYTES,X
B098: 91 42	155	STA	(TRGTPTR),Y
B09A: E8	156	INX	
B09B: 20 B4 FC	157	JSR	NXTA4 ; INC SOURCE AND TARGET POINTERS,
B09E: C6 2F	158	DEC	LENGTH ; AT END CARRY IS SET
B0A0: 10 F4	159	BPL	MOVE
B0A2: 90 B8	160	BCC	RELOCATE ; UNTIL AT END
	161	*	
	162	*	PATCH DOS-COMMANDS PR# AND IN#
	163	*	

B0A4: 18	164	CLC	;PR# HANDLER IS AT OFFSET \$052C
B0A5: A9 2C	165	LDA ##2C	
B0A7: 85 3C	166	STA TEMP	
B0A9: AD D2 03	167	LDA WRMSTRT+2	
B0AC: 69 05	168	ADC ##05	
B0AE: 85 3D	169	STA TEMP+1	
B0B0: A0 00	170	LDY ##00	
B0B2: 18	171	CLC	;PATCH THIS ADDRESS WITH OWN PR#
B0B3: A9 31	172	LDA #<PR-BEGIN	;HANDLER (RELOCATED)
B0B5: 65 44	173	ADC TARGET	
B0B7: 91 3C	174	STA (TEMP),Y	
B0B9: C8	175	INY	
B0BA: A9 00	176	LDA #>PR-BEGIN	
B0BC: 65 45	177	ADC TARGET+1	
B0BE: 91 3C	178	STA (TEMP),Y	
B0C0: A0 05	179	LDY ##05	;IN# HANDLER IS AT OFFSET \$0531
B0C2: 18	180	CLC	;PATCH THIS ADDRESS WITH OWN IN#
B0C3: A9 76	181	LDA #<IN-BEGIN	;HANDLER (RELOCATED)
B0C5: 65 44	182	ADC TARGET	
B0C7: 91 3C	183	STA (TEMP),Y	
B0C9: C8	184	INY	
B0CA: A9 00	185	LDA #>IN-BEGIN	
B0CC: 65 45	186	ADC TARGET+1	
B0CE: 91 3C	187	STA (TEMP),Y	
	188	*	
	189	* PATCH RWTS-ROUTINE	
	190	*	
B0D0: AD DA 03	191	LDA RWTS	;START OF RWTS SUBROUTINE
B0D3: 85 3C	192	STA TEMP	
B0D5: AD DB 03	193	LDA RWTS+1	
B0D8: 85 3D	194	STA TEMP+1	
B0DA: A0 04	195	LDY #4	;OFFSET IS \$04
B0DC: B1 3C	196	LDA (TEMP),Y	;SAVE OLD ADDRESS
B0DE: 48	197	PHA	
B0DF: 88	198	DEY	
B0E0: B1 3C	199	LDA (TEMP),Y	
B0E2: 48	200	PHA	
B0E3: 18	201	CLC	;PUT ADDRESS OF OWN RWTS THERE
B0E4: A9 97	202	LDA #<NWRWTS-BEGIN	; (RELOCATED)
B0E6: 65 44	203	ADC TARGET	
B0E8: 91 3C	204	STA (TEMP),Y	
B0EA: C8	205	INY	
B0EB: A9 00	206	LDA #>NWRWTS-BEGIN	
B0ED: 65 45	207	ADC TARGET+1	
B0EF: 91 3C	208	STA (TEMP),Y	
B0F1: A0 98	209	LDY #<NWRWTS+1	;AND PUT OLD ADDRESS IN OWN RWTS
B0F3: 68	210	PLA	
B0F4: 91 44	211	STA (TARGET),Y	
B0F6: C8	212	INY	
B0F7: 68	213	PLA	
B0F8: 91 44	214	STA (TARGET),Y	
	215	*	
	216	* PATCH RESET	
	217	*	
B0FA: A0 14	218	LDY #<OLDRST+1	;NEW RESET SHOULD CONTINU IN

```

0FC: AD F2 03 219      LDA RESET      ;OLD RESET
OFF: 91 44 220      STA (TARGET),Y ;PUT ADDRESS OF OLD RESET IN
101: C8 221          INY      ; RELOCATED ROUTINE
102: AD F3 03 222      LDA RESET+1 /
105: 91 44 223      STA (TARGET),Y
107: 18 224          CLC      ; PATCH RESET TO OWN RESET-ROUTINE
108: A9 00 225      LDA #<NEWST-BEGIN ; (RELOCATED)
10A: 65 44 226      ADC TARGET
10C: 8D F2 03 227      STA RESET
10F: A9 00 228      LDA #>NEWST-BEGIN
111: 65 45 229      ADC TARGET+1
113: 8D F3 03 230      STA RESET+1
116: 49 A5 231      EOR #$A5    ;ADJUS POWER-UP BYTE
118: 8D F4 03 232      STA PWRUP
233 *
234 * ACTIVATE INPUT- AND OUTPUTHOOKS
235 *
11B: 6C F2 03 236  ENDIT      JMP (RESET)    ;SIMULATE A RESET
237 *
238 *****
239 **
240 ** THE ROUTINES:
241 **
242 ** - NEW RESET
243 ** - CHARACTER GOT
244 ** - PR#
245 ** - IN#
246 ** - REPATCH CHARACTER GOT
247 ** - TYPE-AHEAD & PRINTER BUFFER
248 ** - KEYIN
249 ** - SCREEN CHARACTER OUT
250 ** - PRINTER CHARACTER OUT
251 **
252 *****
253 *
254 * LOCAL ADDRESSES
255 *
256 CH      EQU $24      ;HTAB POSITION
257 BASL     EQU $28      ;SCREEN BASE ADDRESS
258 CSWL     EQU $36      ;OUTPUT POINTER
259 KSWL     EQU $38      ;INPUT POINTER
260 RND      EQU $4E      ;RANDOM NUMBER
261 CHRGET   EQU $B1      ;BASIC CHARACTER-GET ROUTINE
262 NWCHRGOT EQU $BA      ;PATCH IN CHARACTER-GET ROUTINE
263 SCRNBIT  EQU $0779    ;PRINTER & SCREEN OUTPUT
264 KEYBRD   EQU $C000    ;KEYBOARD
265 KBDSTRB  EQU $C010    ;KEYBOARD STROBE
266 L1      EQU $C08C    ;PRINTER LINE 1
267 L2      EQU $C082    ;PRINTER LINE 2
268 DATA    EQU $C080    ;PRINTER DATA LINE
269 BUSY     EQU $C08D    ;PRINTER BUSY LINE
270 COUT     EQU $FDF0    ;SCREEN OUTPUT
271 INPORT   EQU $FE8B    ;SETS INPUT POINTER
272 OUTPORT  EQU $FE95    ;SETS OUTPUT POINTER
273 BELL     EQU $FF3A    ;RING BELL

```

```

274 PRSLOT EQU $01 ; PRINTER SLOT
275 BREAK EQU $83 ; BREAK KEY (^C)
276 LF EQU $8A ; LINE FEED (^J)
277 CR EQU $8D ; CARRIAGE RETURN (^M)
278 STOP EQU $93 ; STOP KEY (^S)
279 FLUSH EQU $98 ; FLUSH KEY (^X)
280 *
281 * ORG BEGIN ; RELATIVE ORIGIN
282 *
283 *
284 * NEW RESET FLUSHES THE TYPE-AHEAD BUFFER
285 * AND TURNS OFF THE PRINTER BUFFER.
286 * YOU CAN EMPTY THE PRINTER BUFFER BY
287 * SETTING THE HIGH BIT OF PRON (RELOCATED!)
288 * RESET ALSO REPATCHES CHRGET, ACTIVATES
289 * ITS OWN INPUT- AND OUTPUT ROUTES AND
290 * JUMPS TO THE OLD RESET HANDLER (DOS).
291 *
1000: AD 9D 11 292 NEWRST LDA KPUT ; FLUSH TYPE-AHEAD
1003: 8D 9C 11 293 STA KGET
1006: A9 00 294 LDA #$00
1008: 8D A0 11 295 STA PRON ; PRINTER & STOP-COMMAND OFF
100B: 20 76 10 296 JSR IN ; KEYBOARD INPUT
100E: A9 00 297 LDA #$00
1010: 20 31 10 298 JSR PR ; SCREEN OUTPUT
1013: 4C 00 00 299 OLDRST JMP $0000 ; JUMP TO OLD RESET
300 *
301 * CHRGET IS A COPY OF THE CHRGET ROUTINE
302 * AT $00BA + A CALL TO INKEY.
303 *
1016: 4C B1 00 304 CHRGT JMP CHRGET ; INC PROGRAM COUNTER
1019: 20 9A 10 305 CHRGOT JSR INKEY
101C: C9 3A 306 CMP #$3A
101E: B0 0A 307 BCS CHRTS
1020: C9 20 308 CMP #$20
1022: F0 F2 309 BEQ CHRGT
1024: 38 310 SEC
1025: E9 30 311 SBC #$30
1027: 38 312 SEC
1028: E9 D0 313 SBC #$D0
102A: 60 314 CHRTS RTS
315 *
316 * PR#-HANDLER
317 * THE ACCUMULATOR CONTAINS THE SLOTNUMBER
318 * TO WHICH THE OUTPUT SHOULD GO.
319 * IF OUTPUT TO SCREEN OR PRINTER THEN
320 * CONNECT OWN ROUTINE
321 * ELSE GO TO OUTPUT.
322 *
102B: 4C 2B 11 323 SOUTPUT JMP SCOUT ; SCREEN OUTPUT
102E: 4C 31 11 324 POUTPUT JMP PCOUT ; PRINTER OUTPUT
1031: C9 00 325 PR CMP #00 ; SCREEN OUTPUT?
1033: F0 07 326 BEQ PRSCREEN
1035: C9 01 327 CMP #PRSLOT ; PRINTER OUTPUT?
1037: F0 0F 328 BEQ PRPRNTR
1039: 4C 95 FE 329 JMP OUTPORT ; MONITOR HANDLER

```

```

103C: A2 01 330 PRSCREEN LDX  #01 ;SET OUTPUT POINTER
103E: BD 2C 10 331 PRLOOPA LDA  SOUTPUT+1,X
1041: 95 36 332 STA  CSWL,X
1043: CA 333 DEX
1044: 10 F8 334 BPL  PRLOOPA
1046: 30 44 335 BMI  REPATCH ;REPATCH CHRGT
1048: 2C A0 11 336 PRFRNTR BIT  PRON ;PRINTER BUFFER EMPTY?
104B: 30 15 337 BMI  PRNTR2 ;NO
338 * THIS CODE IS INTERFACE DEPENDENT!
339 * TO ACTIVATE THE BUSY SIGNAL A NULL
340 * CHARACTER IS PRINTED FIRST.
341 * WHEN YOUR PRINTER IS NOT PRINTING
342 * TRY TURNING IT OFF AND ON.
104D: A9 A0 343 LDA  #A0 ;ACTIVATE BUSY SIGNAL
104F: BD 9C C0 344 STA  L1+16*PRSL0T
1052: A9 FF 345 LDA  #FF
1054: BD 92 C0 346 STA  L2+16*PRSL0T
1057: A9 00 347 LDA  #00
1059: BD 90 C0 348 STA  DATA+16*PRSL0T
105C: AE 9E 11 349 LDX  PGET ;PUT := GET, SO BUFFER IS
105F: 8E 9F 11 350 STX  PPUT ; EMPTY NOW
1062: A5 24 351 PRNTR2 LDA  CH ;SAVE CURSOR POSITION
1064: 8D 9B 11 352 STA  GLDCH
1067: A2 01 353 LDX  #01 ;SET OUTPUT POINTER
1069: BD 2F 10 354 PRLOOPB LDA  POUTPUT+1,X
106C: 95 36 355 STA  CSWL,X
106E: CA 356 DEX
106F: 10 F8 357 BPL  PRLOOPB
1071: 30 19 358 BMI  REPATCH ;REPATCH CHRGT
359 *
360 * IN#-HANDLER
361 * THE ACCUMULATOR CONTAINS THE SLOTNUMBER
362 * TO WHICH THE INPUT SHOULD GO.
363 * IF INPUT TO KEYBOARD THEN
364 * CONNECT OWN ROUTINE
365 * ELSE GO TO INPUT.
366 *
1073: 4C 78 11 367 INPUT JMP  KEYIN
1076: C9 00 368 IN CMP  #00 ;KEYBOARD INPUT?
1078: F0 03 369 BEQ  INKBD
107A: 4C 8B FE 370 JMP  INPORT ;MONITOR HANDLER
107D: A2 01 371 INKBD LDX  #01 ;SET INPUT POINTER
107F: BD 74 10 372 INLOOP LDA  INPUT+1,X
1082: 95 38 373 STA  KSWL,X
1084: CA 374 DEX
1085: 10 F8 375 BPL  INLOOP
1087: 30 03 376 BMI  REPATCH ;REPATCH CHRGT
377 *
378 * WHENEVER YOU ISSUE A CONTROL B WITHIN
379 * THE MONITOR OR YOU CALL $E000, THE
380 * NEW CHRGT ROUTINE IS DISCONNECTED.
381 * YOU CAN RECONNECT THE CHRGT ROUTINE
382 * BY PRESSING RESET OR GIVING THE DOS
383 * COMMAND IN#0 OR PR#0.
384 *

```

```

1089: 4C 19 10 385 PATCH JMP CHRGOT
108C: A2 02 386 REPATCH LDX #$02 ; COPY THE RELOCATED JUMP
108E: BC 89 10 387 REPATCH2 LDY PATCH,X ; INSTRUCTION INTO CHRGOT
1091: 94 BA 388 STY NWCHRGOT,X
1093: CA 389 DEX
1094: 10 F8 390 BFL REPATCH2
1096: 60 391 RTS
      392 *
      393 * NEW READ/WRITE TRACK/SECTOR
      394 * FIRST GO TO OLD RWTS, THEN
      395 * GO TO INKEY.
      396 *
1097: 20 00 00 397 NWRWTS JSR $0000
      398 *
      399 * INKEY IS THE MAIN ROUTINE. IT CHECKS
      400 * FOR A CHARACTER AND WHEN ONE IS FOUND
      401 * IT STORES IT IN A BUFFER. SPECIAL
      402 * KEYS ARE:
      403 * BREAK FLUSHES TYPE-AHEAD, STORES
      404 * CONTROL C IN BUFFER, BUT DOES
      405 * NOT CLEAR THE KEYBOARDSTROBE
      406 * SO BASIC WILL SEE IT.
      407 * STOP STOPS THE EXECUTION OF THE
      408 * PROGRAM, UNTIL THE STOP-KEY
      409 * IS PRESSED A SECOND TIME.
      410 * FLUSH FLUSHES THE TYPE-AHEAD BUFFER
      411 *
      412 * PRINT PRINTS A CHARACTER WHEN THE
      413 * BUSY SIGNAL IS OFF.
      414 *
109A: 08 415 INKEY PHP ;SAVE P-, A- & X-REGISTER
109B: 49 416 PHA
109C: 8A 417 TXA ;Y-REGISTER IS NOT
109D: 48 418 PHA ; USED.
109E: E6 4E 419 KEY1 INC RND ;INC RANDOM NUMBER
10A0: D0 02 420 BNE KEY2
10A2: E6 4F 421 INC RND+1
10A4: AD 00 C0 422 KEY2 LDA KEYBRD ;GET CHARACTER
10A7: 10 47 423 BPL PRINT ;NO CHAR AVAILABLE
10A9: C9 83 424 CMP #BREAK ;IF BREAK THEN
10AB: D0 0F 425 BNE KEY3
10AD: AE 9C 11 426 LDX KGET ; FLUSH TYPE AHEAD
10B0: AD A0 11 427 LDA PRON ; DEACTIVATE STOP-CYCLE
10B3: 29 BF 428 AND #%10111111
10B5: 8D A0 11 429 STA PRON
10B8: A9 83 430 LDA #BREAK ; AND STORE ^C
10BA: D0 1D 431 BNE KEY5 ; ALWAYS
10BC: 2C 10 C0 432 KEY3 BIT KBDSTRB ;CLEAR KEYBOARDSTROBE
10BF: AE 9D 11 433 LDX KPUT ;PUT-POINTER
10C2: C9 98 434 CMP #FLUSH ;IF FLUSH THEN
10C4: D0 05 435 BNE KEY4 ; IF EMPTY BUF THEN
10C6: EC 9C 11 436 CPX KGET ; ^X = DEL LINE -> STORE ^X
10C9: D0 1C 437 BNE EMPTYBUF ; ELSE FLUSH TYPE-AHEAD
10CB: C9 93 438 KEY4 CMP #STOP ;IF STOP THEN
10CD: D0 0A 439 BNE KEY5

```

```

10CF: AD A0 11 440      LDA PRON          ; SET STOP-BIT
10D2: 49 40 441      EOR  #01000000
10D4: 8D A0 11 442      STA PRON
10D7: B0 17 443      BCS PRINT          ; ALWAYS
10D9: E9 444      INX          ; INC PUT-POINTER
10DA: EC 9C 11 445      CPX KGET          ; BUFFER FULL?
10DD: F0 0B 446      BEQ KBUFFFULL        ; YES -> RING BELL
10DF: 9D A1 11 447      STA KBUF,X        ; STORE CHARACTER
10E2: 8E 9D 11 448      STX KPUT          ; SAVE PUT-POINTER
10E5: D0 09 449      BNE PRINT          ; ALWAYS
10E7: 8E 9C 11 450      EMPTYBUF STX KGET    ; GET := PUT -> EMPTY BUFFER
10EA: 20 3A FF 451      KBUFFFULL JSR BELL    ; RING BELL
10ED: B8 452      CLV          ; BRANCH LINK FOR STOP-
10EE: 70 AE 453      KEY6 BVS KEY1        ; CYCLE
10F0: 2C A0 11 454      PRINT BIT PRON      ; PRINTER ON?
10F3: 10 2C 455      BPL EXIT          ; NO -> EXIT
      456      * THIS CODE IS INTERFACE DEPENDENT!
10F5: AD 9D C0 457      LDA BUSY+16*PRSL0T
10F8: 29 10 458      AND  #10          ; CHECK BUSY SIGNAL
10FA: F0 25 459      BEQ EXIT
10FC: EE 9E 11 460      INC PGET          ; INC GET-POINTER
10FF: AE 9E 11 461      LDX PGET
1102: A9 A0 462      LDA  #A0          ; TELL THE PRINTER A
1104: 8D 9C C0 463      STA L1+16*PRSL0T ; CHARACTER IS COMING.
1107: A9 FF 464      LDA  #FF
1109: 8D 92 C0 465      STA L2+16*PRSL0T
110C: BD A1 12 466      LDA PBUF,X        ; GET CHARACTER
110F: 29 7F 467      AND  #01111111 ; CLEAR HIGH BIT
1111: 8D 90 C0 468      STA DATA+16*PRSL0T
1114: EC 9F 11 469      CPX PPUT          ; END OF BUFFER?
1117: D0 0B 470      BNE EXIT
1119: AD A0 11 471      LDA PRON          ; PUT PRINTER OFF
111C: 29 7F 472      AND  #01111111 ; BY CLEARING PRINTER-ON BIT
111E: 8D A0 11 473      STA PRON
1121: 2C A0 11 474      EXIT BIT PRON      ; CHECK FOR STOP-CYCLE
1124: 70 C8 475      BVS KEY6          ; YES -> BRANCH BACK
1126: 68 476      PLA          ; RESTORE REGISTERS
1127: AA 477      TAX
1128: 68 478      PLA
1129: 28 479      PLP
112A: 60 480      RTS
      481      *
      482      * SCOUT IS THE SCREEN OUTPUT ROUTINE.
      483      * IT CHECKS THE KEYBOARD- AND PRINTER-
      484      * BUFFER AND THEN PRINTS THE CHARACTER
      485      * ON THE SCREEN. THIS ROUTINE GARANTIES
      486      * TYPE-AHEAD DURING THE MONITOR LIST-
      487      * COMMAND.
      488      *
112B: 20 9A 10 489      SCOUT JSR INKEY
112E: 4C F0 FD 490      JMP  COUT

```

```

491 *
492 * PCOUT IS THE PRINTER OUTPUT ROUTINE.
493 * IT STORES THE CHARACTER IN THE PRINTER-
494 * BUFFER. WHEN THE BUFFER IS FULL, IT
495 * WAITS UNTIL A CHARACTER IS SEND TO THE
496 * PRINTER. A CARRIAGE RETURN IS SUBSTITUTED
497 * IN A CARRIAGE RETURN + LINEFEED.
498 * THE PRINTERSTATUS IS SET.
499 *
1131: 8E 9A 11 500 PCOUT STX XSAVE ;SAVE X-REGISTER
1134: 2C 79 07 501 BIT SCRNBIT ;IF SCREEN OUTPUT THEN
1137: 50 1B 502 BVC PCOUT1
1139: AE 9B 11 503 TAB LDX OLDCH ; IF 'PRINT,'-COMMAND HAS BEEN
113C: E4 24 504 CPX CH ; EXECUTED (CH IS ALTERED) THEN
113E: B0 0C 505 BCS NOTAB
1140: 48 506 PHA ; SAVE CHARACTER
1141: A9 A0 507 LDA #$A0 ; PRINT SPACE
1143: 20 54 11 508 JSR PCOUT1 ; (BUT NOT ON SCREEN)
1146: 68 509 PLA ; RESTORE CHARACTER
1147: EE 9B 11 510 INC OLDCH
114A: D0 ED 511 BNE TAB ; UNTIL TAB-POSITION
114C: 20 F0 FD 512 NOTAB JSR COUT ;OUTPUT CHAR TO SCREEN
114F: A4 24 513 LDX CH ;SAVE NEW CURSOR POSITION
1151: 8E 9B 11 514 STX OLDCH
1154: EE 9F 11 515 PCOUT1 INC PPUT ;INC PUT-POINTER
1157: AE 9F 11 516 LDX PPUT
115A: 20 9A 10 517 PCOUT2 JSR INKEY
115D: EC 9E 11 518 CPX PGET ;IF BUFFER FULL THEN
1160: F0 F8 519 BEQ PCOUT2 ; WAIT
1162: 9D A1 12 520 STA PBUF,X ;STORE CHARACTER
1165: C9 BD 521 CMP #CR ;IF CR THEN
1167: D0 04 522 BNE PCOUT3
1169: A9 BA 523 LDA #LF ; STORE A LF TOO
116B: D0 E7 524 BNE PCOUT1
116D: AE 9A 11 525 PCOUT3 LDX XSAVE
1170: 0E A0 11 526 ASL PRON ;SET PRINTER-ON BIT
1173: 38 527 SEC
1174: 6E A0 11 528 ROR PRON
1177: 60 529 RTS
530 *
531 * KEYIN IS A SUBSTITUTE FOR THE MONITOR
532 * KEYIN ROUTINE. IT GETS A CHARACTER
533 * FROM THE TYPE-AHEAD BUFFER. IF THERE
534 * IS NO CHARACTER, IT WAITS UNTIL THERE
535 * IS ONE.
536 *
1178: 8E 9A 11 537 KEYIN STX XSAVE ;SAVE X-REGISTER
117B: AE 9C 11 538 LDX KGET ;GET-POINTER
117E: 20 9A 10 539 KEYIN2 JSR INKEY
1181: EC 9D 11 540 CPX KPUT ;IF GET=PUT THEN
1184: F0 F8 541 BEQ KEYIN2 ; WAIT FOR CHARACTER
1186: E9 542 INX ;INC GET-POINTER
1187: 91 28 543 STA (BASL),Y
1189: BD A1 11 544 LDA KBUF,X ;GET CHARACTER
118C: C9 B3 545 CMP #BREAK ;BREAK KEY?
118E: D0 03 546 BNE KEYIN3

```

```

190: 2C 10 C0 547      BIT KBDSTRB      ;YES -> CLEAR STROBE AFTER ALL
193: 8E 9C 11 548      KEYIN3 STX KGET      ;UPDATE GET-POINTER
196: AE 9A 11 549      LDX XSAVE
199: 60                550      RTS
                    551      *
                    552      * VARIABLES
                    553      *
19A: 00                554 XSAVE      DFB $00
19B: 00                555 OLDCH      DFB $00      ;OLD HTAB POSITION
19C: 00                556 KGET      DFB $00      ;KEYBOARD GET-POINTER
19D: 00                557 KPUT      DFB $00      ;KEYBOARD PUT-POINTER
19E: 00                558 PGET      DFB $00      ;PRINTER GET-POINTER
19F: 00                559 PPUT      DFB $00      ;PRINTER PUT-POINTER
1A0: 00                560 PRON      DFB $00      ;PRINTER- & STOPSTATUS
1A1: 00                561 KBUF      DFB $00      ;KEYBOARD BUFFER
                    562      ORG $+255      ; (256 BYTES LONG)
2A1: 00                563 PBUF      DFB $00      ;PRINTER BUFFER
                    564      ORG $+255      ; (256 BYTES LONG)
3A1: 00                565 END      DFB $00

```

-END ASSEMBLY--

RRORS: 0

06 BYTES

Hier volgen nog enige omerkingen over de printerbuffer:

- Bij iedere programma byte en ieder indevoerd (KEYIN) en uitgevoerd (COUT) karakter wordt de INKEY-routine aangeroepen. Dit kost natuurlijk tijd. Een programma wordt ongeveer een kwart minder snel uitgevoerd. Wanneer er veel geprint wordt, wordt dit tijdverlies echter ruimschoots gecompenseerd. Wordt de type-ahead-buffer niet nodig bevonden, dan kan de volgende modificatie voor aanzienlijke tijdwinst zorgen.

regel	modificatie	
289-290		: WEG
291-292	NEWST LDA #\$60	: ZET RTS OP INKEY
	STA INKEY	: EN ONTKOPPEL HEM ZO
	LDA #\$00	: INVOER VANUIT SLOT 0 (TOETSENBORD)
333-334	PRPRNTR BIT INKEY	: PRINTERBUFFER LEEG?
	BVC PRNTR2	: NO -> PHP GEEFT OVERFLOW CLEAR
416-452		: WEG
468-470	LDA #\$60	: ZET RTS OP INKEY
	STA INKEY	: ZODAT DE ROUTINE NIET MEER
	EXIT ...	: DOORLOPEN WORDT
471-472		: WEG
522-525	PCOUT3 LDX #\$08	: ZET PHP OP INKEY
	STX INKEY	: EN KOPPEL HEM ZO WEER AAN
	LDX XSAVE	

DE 6502 KENNER

534-547 KEYIN JSR INKEY : PRINT EEN KARAKTER UIT DE BUFFER

INC RND
BNE KEY1
INC RND+1
KEY1 BIT KEYBRD
BPL KEYIN
STA (BASL),Y
LDA KEYBRD
BIT KBDSTRB
RTS

553-554 : WEG

557-559 : WEG

- De printerbuffer is 256 bytes lang. Deze buffer is vrij gauw vol, zodat je toch moet wachten op de printer. Een remedie is de buffer te vergroten. Eventueel kan een 16K language kaart als buffer dienen. Deze kan ongeveer 4 pagina's tekst bevatten.
- De routine maakt geen gebruik van de routines op de interface kaart. Speciale commando's gericht op deze routine (ctrl-IBON etc.) werken dus niet!
- Het moet mogelijk zijn het 'ready'-signaal van de printer te koppelen aan een interrupt lijn. Iets dergelijks kan waarschijnlijk ook met het toetsenbord. Op deze manier kan men een interrupt-gedreven printer- en type-ahead-buffer maken.

PAPERWARE-SERVICE

Send your orders to Editors Office DE
6502 KENNER, c/o Willem L. van Pelt,
Jacob Jordaensstraat 15, 2923 CK Krim-
pen a.d. IJssel, The Netherlands.
Foreign countries: If not paying with
Eurocheque, you have to pay HF1. 7.50
extra transfers!
Postal account: 841433.
Bankaccount : 44.11.06.471 AMRO-bank
Krimpen a.d. IJssel.
Postal account
AMRO-bank : 3050

MICRO-WARE Ltd. Toronto, Canada.
Assembler/Disassembler/Editor
Micro-ADE for the 6502, written by
Peter Jennings (c) 1977.
(c) 1982 by KIM Users Club The Nether-
lands.
System Description.

The Micro-ADE system is designed for
use with any 6502 microcomputer and
consists of three major programs as
well as a number of utility programs.
The major programs are an assembler, a
disassembler, and a text editor. The
assembler is used to create machine
executable code for the 6502 from a
symbolic input source program. Small
programs can be created and tested
directly in memory. Larger programs
may be written using cassette tapes
for source input and object output.
The disassembler is used to list

executable 6502 machine code in the
symbolic assembler source format.
Symbols are generated if they are
defined in the symbol table. The text
editor is used to create source
programs in the format required for
the assembler. It contains the neces-
sary routines for easy manipulation of
text data in memory or from cassette
files. The minimum system configura-
tion for full use of all Micro-ADE
features consists of a 6502 CPU, 8K of
random access memory, 2 cassette re-
corders with start/stop control, and
an ASCII input/output device. It is
possible to use all parts of the
system in a restricted way with less
memory and a single manually operated
recorder.

Manual English version for KIM-computer	HF1. 30.00
Manual English version + command review for Elektor's JUNIOR- computer with 8K Micro ADE	HF1. 35.00
Assembly source-listing original 4K version English commented for KIM-computer	HF1. 30.00
Assembly source-listing 8K version English commented for JUNIOR-computer	HF1. 65.00

```

*****
****          BOUW EENS EEN MILJONOR          ****
****          *****                          ****
****          Elektuur JUNIOR-computer met 65C02          ****
****          Door : Jan Vernimmen, Purmerend          ****
****          ****                                  ****
*****

```

Doel : Een flexibel ontwikkelsysteem met een geheugen-
bereik aanzienlijk groter dan 64 K.

Het geheugenbereik van de 65(C)02 is 64K vanwege zijn 16 adreslijnen. Het Miljo-
nor-systeem nu is erop gebaseerd dat alvast meerdere adreslijnen worden getrok-
ken en dat bij opstarten en, indien gewenst, op run-time wordt bepaald welke
adreslijnen nu adreslijn A13, A14 en A15 vormen. Het gedeelte van het geheugen
boven de 64K is nu aan te spreken door software-matig een 8K-blok binnen de
adresseerbare 64K te vervangen door een ander blok dat normaal niet adresseer-
baar is.

Met IC2 kunnen we 8 uit 16 blokken kiezen (4 uitgaande lijnen). De 16 blokken
van 8K vormen nu 1 bank (128K dus). Met IC1 kunnen we de bank kiezen. Daar IC1
niet via adreslijnen is te benaderen, kunnen we geen 64K samenstellen uit 2 of
meer verschillende banks. Wel mogen blokken zo worden uitgepoord, dat ze in
verschillende banks voorkomen, aan te bevelen voor blok 0 (adres 0000 - 2000)
met de I/O-routines.

Porten we de adreslijnen A17, A18, A19 en A20 uit door middel van een 74154,
dan kunnen we 16 banks van 128K elk adresseren, ofwel meer dan 2 miljoen bytes!!

```

Adres bankselect FFF8 ; write-only
Adres bankselect FFF9 ; eveneens write-only

```

De Miljo-nor maakt het mogelijk met gestructureerde programma's te werken welke
groter zijn dan 64K !

Opstarten dient te geschieden door de Boot-switch in de stand "BOOT" te zetten.
In dat geval kan uitsluitend blok 0 met de I/O-routines geadresseerd worden.
Eerst wanneer het bank-switch en blok-switchregister met een minimum aan logi-
sche waarden is gevuld kan de schakelaar naar de stand "EXTERN" worden omgezet
en is het gehele systeem toegankelijk.

Het verschil tussen dit schema en eerdere publikaties zit in de toevoeging van
IC 10. Dit IC zorgt voor de write-puls van IC2, de 74S189. Met de opgegeven com-
ponenten begint deze write-puls circa 370 nSec. na de opwaartse flank van J2,
en duurt 50 nSec. (35 nSec. zou genoeg zijn), terwijl na beëindiging van de
write-puls de data nog enige tijd ongewijzigd blijft aangeboden, vanwege de ver-
eiste write-recovery-time (vereist minimaal 35 nSec.).

Voor uitgebreidere informatie raadplege men editie nummer 30 van DE 6502 KENNER.

Zie voor het schema de volgende pagina.

Boekeninformatie

6502 Assembleertaal en machinecode
voor beginners

A.P. Stephenson
1984

206 pagina's fl. 37,50

ISBN 90 6233 123 8

Academic Service Den Haag

Veel hobbyisten werken uitsluitend in
Basic, het is de enige manier waarvoor
zij met hun computer kunnen communi-
cieren. Dit is heel gebruikelijk gezien
het relatief gebruikersvriendelijke
karakter van deze taal en het gemak
waarmee binnen een uur een bruikbaar
programma gemaakt kan worden. Alle
computers hebben echter nog een ver-

borgen, alternatieve taal: machinetaal.
Deze taal is 'gratis', werkt met licht-
snelheid en is uiterst efficiënt in het
gebruik van het geheugen.

Het leren programmeren in machinetaal
is niet eenvoudig, vandaar dat de voor-
beelden in dit boek eenvoudig gehouden
zijn. Een hulp bij dit programmeren in
machinetaal is het gebruik van een as-
sembleertaal (zoals Micro-ADE, CAT en
FATE). In dit boek komt uitsluitend de
microprocessor met het type-nummer 6502
(en 6502A) aan de orde. Deze micropro-
cessor vinden we in de APPLE, de PET, de
ACORN ATOM, de ATARI, de BBC, de ELEC-
TRON, en nog vele andere. Bovendien is
deze chip basis voor de 6510-processor
in de COMMODORE 64.

DE LANDELIJKE BIJEENKOMSTEN

Op de derde zaterdag van de maanden januari, maart, mei, september en november is er een clubbijeenkomst. De lokatie wordt steeds tijdig in DE 6502 KENNER bekend gemaakt. De toegangsprijs is vastgesteld op f. 10,-- per persoon. Lunch en konsumpties niet inbegrepen. In den Haag en Arnhem is regelmatig een regionale bijeenkomst. Regionale bijeenkomsten worden georganiseerd op initiatief van de leden in de regio. Deze bijeenkomsten vragen een toegangsprijs welke de uitnodigingen aan de leden kunnen bekostigen. Op de landelijke bijeenkomsten zijn vaste programmapunten zoals lezingen, de markt en het forum. De leden kunnen hun overvloedige spullen aanbieden. Commerciële handel is alleen toegestaan met toestemming van het bestuur. Tijdens het forum worden Uw problemen op hardware of software gebied (hopelijk) opgelost. Het laatste deel van de bijeenkomsten bestaat uit het "gezellig" bomen over de gemeenschappelijke hobby, al dan niet met demonstraties van meegebrachte hardware en/of software. In de meeste gevallen is ook de redactie van DE 6502 KENNER vertegenwoordigd.

DE 6502 KENNER

Het verenigingsblad verschijnt op de derde zaterdag van de maanden februari, mei, augustus, oktober en december. De mogelijkheid van een zesde editie is in studie.* De 48 redactionele pagina's worden door de leden met artikelen gevuld. Onze publikaties verschillen aanzienlijk van die in andere bladen. Komplete listings en schema's waar anderen van kunnen leren. Zogenaamde "hexdumps" behoren daar niet toe. Het gaat om de programmeertechnieken ! Ook hogere programmeertalen worden ingezonden en gepubliceerd. Studies m.b.t. assembleertalen en hogere programmeertalen leveren bovendien allerlei inzichten op die tot een beter programmeren kunnen leiden.

HET LIDMAATSCHAP

Het lidmaatschap staat open voor "natuurlijke personen". Geen bedrijven of instellingen dus ! De kontributie bedraagt sedert 1983 f. 45,-- per kalenderjaar. U kunt lid worden door bijgaand formulier in te vullen en de kontributie voor het lopend jaar te voldoen. U krijgt dan de in dat jaar reeds verschenen nummers van DE 6502 KENNER thuisgestuurd.

Wilt U het voorgaande jaargang ook ontvangen ? Geef dat dan aan op het aanmeldingsformulier.

Vrienden of kennissen ? Vraag even een nieuw aanmeldingsformulier aan.

* In 1986 verschijnt DE 6502 KENNER, evenals in 1984 en 1985, zesmaal, t.w. in februari, april, juni, augustus, oktober en december !

A A N M E L D I N G

Ondergetekende wil graag lid worden m.i.v. het jaar ☐ 19 ..

Ondergetekende wenst ook de edities van DE 6502 KENNER van het voorgaande jaar (Hfl. 45,= extra storten) ☐ 19 ..

Naam en voornamen
Adres
Postcode en plaats
Telefoon
Systeem

Handtekening

BUITENLAND: indien betaling anders dan met
Eurocheque: Hfl. 7,50 extra transfers !!!

Ik heb een bedrag van f. 45,-- overgemaakt op postrekening 3757649
t.n.v. KIM Gebruikersclub, Krimpen a.d. IJssel.

DE 6502 KENNERS

WIE WE ZIJN

Een vereniging van 6502 gebruikers. De eerste 6502 machine was de KIM. Onze vereniging is de oudste, geheel zelfstandige gebruikersclub, opgericht op 29 januari 1977. We hebben een geheel ander karakter dan veel andere verenigingen, die zich veelal richten op een bepaald systeem i.p.v. op een microprocessor.

In onze statuten staat onze doelstelling omschreven:

"HET BEVORDEREN VAN DE KENNISUITWISSELING TUSSEN DE GEBRUIKERS OVER DE TOEPASSING EN EIGENSCHAPPEN VAN DE KIM EN DERGELIJKE 6502 SYSTEMEN."

Bij de oprichting alleen de KIM. Nu zijn het er nogal wat:

PET, CBM, COMMODORE-64, APPLE, BASIS-108, PEARCOM, KIM, SYM, JUNIOR, AIM-65, ACORN, BBC, OSI, CHE-1, STARLIGHT, CV-777, ESTATE III, PALLAS, ATARI, PROTON-COMPUTERS, VIC-20, FOR-MOSA, SBC 65/68, NCS 6502, KEMPAC.

Onze leden zijn woonachtig in Nederland, België, Duitsland, Frankrijk, Spanje, Portugal, Amerika. Het illustreert het unieke karakter van onze vereniging. Onze leden zijn geïnteresseerd in het programmeren in machinetaal, Basic, de eigen club-COMAL, de assembler/disassembler/teksteditor "Micro-ADE" waarvan de club de copyrights heeft, de assembler van Carl Moser, FORTH, FOCAL en wat dies meer zij.

HOE WE ZIJN

We staan ingeschreven in het verenigingsregister bij de Kamer van Koophandel te Alkmaar onder nr. V624305. Het bestuur wordt gekozen door de leden op de algemene ledenvergadering die tenminste eenmaal per jaar wordt gehouden. Het bestuur is verantwoording schuldig aan de algemene ledenvergadering. Alle leden hebben stemrecht.

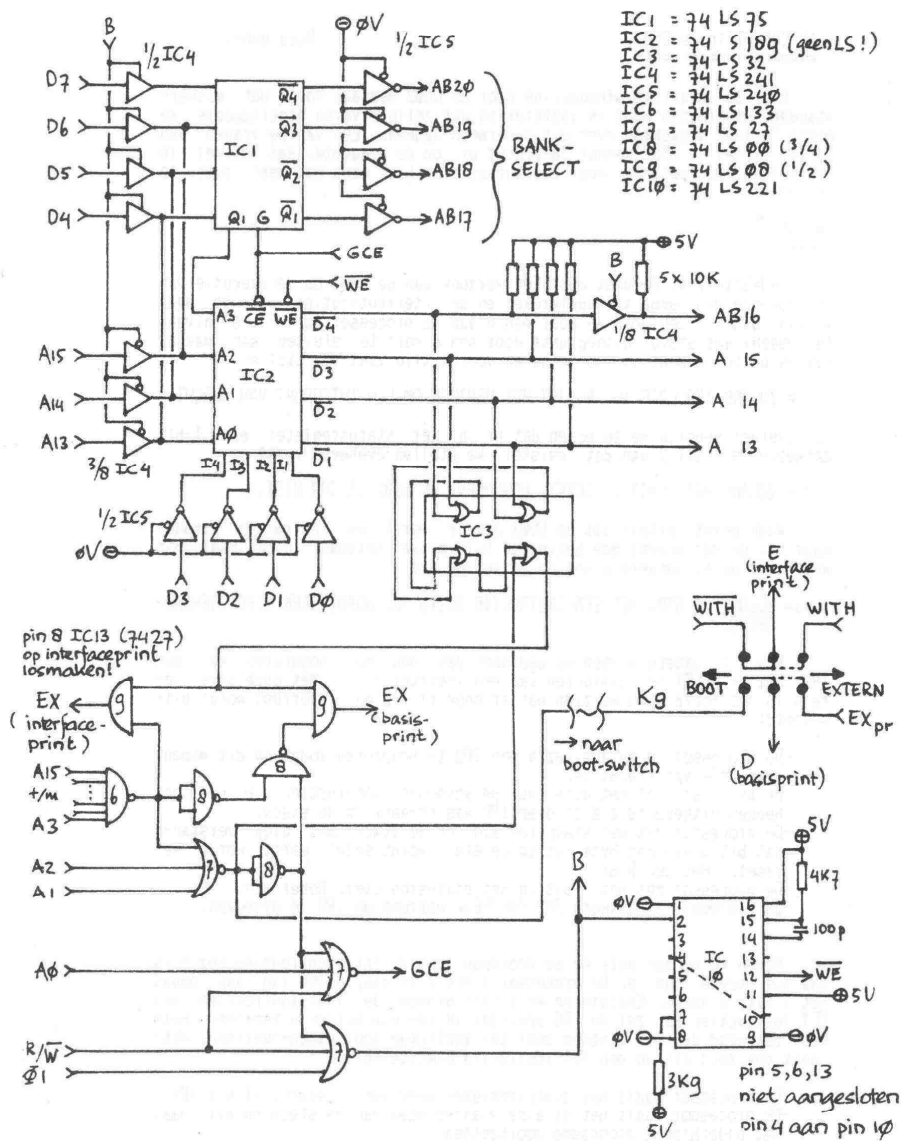
WAT WE DOEN

Onze doelstelling werd al genoemd. Er zijn twee hoofdactiviteiten: de landelijke bijeenkomsten en het verenigingsblad. Daarnaast kunnen de leden tegen geringe vergoeding beschikken over de software die door de leden is ingebracht, of op andere wijze door de vereniging is verworven. Ook passen we bestaande software aan nieuwe 6502 systemen aan in samenwerking met de bonafide leverancier. De vereniging laat zich niet in met software diefstal, ook al is de Nederlandse wetgeving hier op z'n zachtst gezegd "ruim".

Dit aanmeldingsformulier in open enveloppe
en gefrankeerd als drukwerk verzenden aan:

Redactie DE 6502 KENNER
p/a Willem L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpen a.d. IJssel.
Tel.: 01807 - 19881

Zie voor betaling lidmaatschap ommezijde !



BOUW EENS EEN MILJONOR - J.H.VERNIMMEN

26-2-1985

Aan de tekentafel:
 Fridus Jonkman

INTERRUPT in de 6502

Ruud Uphoff

Over de interrupt afhandeling door de 6502 bestaan nogal wat misverstanden. Regelmatig moet ik vaststellen dat zelfs ervaren programmeurs de meest "wilde" gedachten over dit onderwerp hebben. Een van de vragen van Marcel Visser in 6502-kenner 38 bracht mij op de gedachte, dat het wel in het belang van meerdere leden zou kunnen zijn hier eens wat meer over te vertellen.

A. IRQ

IRQ=Interrupt ReQuest dus: Een verzoek aan de 6502 om de executie van het lopende programma te onderbreken en de interruptroutine uit te voeren. Dit verzoek doen we door pen 4 van de processor aan TTL-0 niveau te leggen, wat grover uitgedrukt: door pen 4 kort te sluiten aan massa. Het is nu het moment om voor eens en voor altijd vast te stellen:

=1= ZOLANG IRQ LAAG IS, BLIJFT HET VERZOEK OM EEN INTERRUPT VAN KRACHT.=

.....Verder behoren we te weten dat er in het statusregister een I-bit aanwezig is. (bit 2 van dat register) We stellen eveneens vast:

=2= ZOLANG HET I-BIT IS GEZET, HONOREERT DE 6502 DE IRQ NIET.=

Vaak wordt verteld dat de 6502 altijd eerst de instructie afmaakt waar hij op dat moment mee bezig is. Dat is niet helemaal juist, maar het komt voor de buitenwereld wel op hetzelfde neer:

=3= ZOLANG DE 6502 MET EEN INSTRUCTIE BEZIG IS, WORDT GEEN IRQ GEHONOREERD.=

Bij dit laatste moeten we bedenken dat ook het honoreren van een IRQ, NMI of RESET het uitvoeren van een instructie is. Met deze drie regels in ons hoofd gaan we zien wat er gebeurt als een interrupt wordt uitgevoerd:

- De processor is nu dus bezig een IRQ te honoreren zodat op dit moment regel =3= van kracht is.
- De processor zet het adres van de volgende instructie, die hij zou hebben uitgevoerd als er geen IRQ was gedaan, op de stack.
- De processor zet het statusregister op de stack met dien verstande dat bit 4 van het byte dat op de stack wordt gezet, eerst wordt gereset. (Het zg. B-bit)
- De processor zet het I-bit in het statusregister. Regel =2= !
- De processor executeert JMP (\$FFFE), waarmee de IRQ is afgedaan.

Nemen we nu aan dat, nu de processor met de interruptroutine bezig is IRQ nog steeds laag is. De processor trekt zich daar niets van aan omdat het I-bit is gezet. Executeren we echter binnen de interruptroutine een CLI instructie, dan zal de IRQ onmiddellijk opnieuw worden uitgevoerd hetgeen betekend dat het systeem snel zal vastlopen door stack-overloop. Het-saat ook fout als we een RTI instructie executeren:

- De processor haalt het statusregister weer van de stack. (I-bit =0)
- De processor haalt het originele adres weer van de stack en wil daar het onderbroken programma voortzetten

De IRQ wordt dus onmiddellijk na RTI opnieuw uitgevoerd ! De processor komt niet meer aan het hoofdprogramma toe zolang IRQ laag is. Uit het voorgaande kunnen we dus leren:

- a. Om een interrupt uitgevoerd te krijgen moeten we IRQ LAAG HOUDEN TOT-DAT DE IRQ WORDT GEHONOREERD.
- b. We moeten IRQ HOOG MAKEN VOORDAT EEN RTI OF CLI WORDT UITGEVOERD. Om in dit laatste nog even volledig te zijn: Voordat het I-bit, door welke oorzaak dan ook, wordt gereset.

Uit deze twee laatste conclusies kunnen we opmaken dat het een haast onmogelijke opgave is iets zinnigs met de IRQ-aansluiting van de processor te doen aangezien de processor hardwarematig geen enkele informatie terug stuurt over het al of niet honoreren van een IRQ, terwijl we softwarematig geen toegang tot pen 4 van de chip hebben. Daarom moet in een 6502-systeem in een correcte IRQ afhandeling worden voorzien. Dat maakt het tevens mogelijk meerdere IRQ's door elkaar af te handelen. We zullen tot slot van ons IRQ-verhaal eens zien hoe dat bijvoorbeeld in een COMMODORE-64 is geregeld:

In de COMMODORE-64 wordt een toetsenbord toegepast, dat softwarematig wordt afgetast via een matrix die wordt gevormd door twee poorten van een en hetzelfde CIA. (Deze CIA lijkt erg veel op het 6522-VIA. Beide namen zijn lekker commercieel: Versatile Interface Adapter en Complex Interface Adapter) Deze aftasting gebeurt 60 keer per seconde in een interrupt welke wordt opgewekt door een continue lopende timer in dat CIA. Net als in de 6522 wordt bij nul-doorgang van de timer een interrupt-vlag gezet in het IFR (Interrupt Flag Register). Het bijbehorende ICR (Interrupt Control Register) is zodanig geprogrammeerd dat deze IRQ doorgeven wordt aan de 6502 (eigenlijk 6510). Eenmaal gezet houdt deze vlag in het IFR dus pen 4 van de processor laag, totdat deze vlag wordt gereset, hetgeen gebeurt door simpelweg het IFR te lezen. De IRQ blijft dus van kracht tot hij door de processor wordt gehonoreerd. In de IRQ-routine zelf komt een LDA naar het IFR voor waardoor de processor via het CIA zijn eigen IRQ lijn dus weer hoog zet. Daarmee is alles correct verlopen.

In andere machines zoals de JUNIOR kunnen we desgewenst dergelijke methoden toepassen. In de APPLE-II ontbreekt daarvoor de vereiste hardware die dan ook, in voorkomend geval, deel zal moeten uitmaken van een kaart in een van de sloten.

Er is ook nog een andere methode om een IRQ aan de processor te doen. Daarbij is de benaming IRQ wat misplaatst want het is een BEVEL aan de processor maar geen I-bit iets tegen vermag: De BRK-instructie. Ook over deze instructie zijn de misverstanden buitengewoon groot, hetgeen nog eens in de hand wordt gewerkt doordat alle literatuur BRK als een IMPLIED-instructie van 1 byte af doet. BRK is echter de enige IMPLIED die NIET uit 1 byte bestaat maar uit 2 ! Dat tweede byte is de operand die door de processor WEL wordt GELEZEN maar waarmee de processor NIETS DOET !

De BRK instructie heeft tot gevolg dat eveneens de IRQ routine wordt uitgevoerd. Met een verschil: En nu moet U zeer letterlijk nemen wat ik hier zeg: Van het op de stack veilig te stellen statusregister wordt NIET zoals bij een hardware IRQ bit 4 gereset, maar gezet. Hier ligt dan ook een ander misverstand waarmee we moeten afreken:

ER BESTAAT GEEN B-BIT IN HET STATUSREGISTER. BIT 4 IS ONGEBRUIKT NET ALS BIT 5. BEIDE BITS ZIJN ONGEDEFINIEERD (maar meestal 1)

Als we praten over het B-bit bedoelen we bit 4 van het byte dat door de 6502 als status op de stack wordt gezet bij het honoreren van een IRQ. Hieraan is in de IRQ-routine te zien wat de interrupt veroorzaakte:

0 = Hardware (pen 4 laag)
1 = Software (BRK instructie)

B. NMI

De Non Maskable Interrupt heeft, zoals in de benaming besloten ligt, geen relatie met het I-bit in het statusregister. Wel wordt met uitvoeren gewacht tot de processor de lopende instructie heeft uitgevoerd. NMI heeft een eigen vector op \$FFFA maar behoudens de laatste operatie: JMP (\$FFFA) is er geen softwarematig verschil met IRQ. Dus:

NMI wordt ALTIJD uitgevoerd.
NMI heeft een eigen vector.
Honoreren van NMI houdt eveneens in dat het I-bit wordt gezet, zodat zonder RTI of CLI een IRQ kan worden gehonoreerd. (WEL een NMI !)

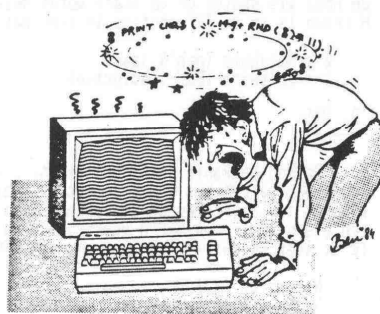
Een NMI wordt niet via pen 4 maar via pen 6 van de chip gedaan. Daarbij is er hardwarematig nog een groot verschil: Pen 6 is flank-gevoelig, hetgeen wil zeggen dat het geen verschil maakt of NMI nu laag of hoog is. NMI ontstaat door het laag WORDEN van pen 6 (negatief flank gevoelig). Dat betekent dat we NMI via een drukknopje aan pen 6 wel kunnen vergeten, want de kontaktender die daarbij ontstaat levert een hele serie elkaar interruperende NMI's op of helemaal geeneen omdat door een overgangs weerstand in het kontakt pen 6 relatief langzaam laag wordt. Daarom dient ook het aanbieden van een NMI aan de processor beheerst te verlopen, op een soortgelijke methode als we hierboven bij IRQ als voorbeeld gaven.

Hex/Ascii-dumps zijn populair in computertijdschriften. In de eerste plaats omdat het voor software-ontwikkelaars erg handige routines zijn, en in de tweede plaats omdat er ook veel van geleerd kan worden. Heel dikwijls verschijnen er routines in machinetaal, maar deze keer kregen we een inzending in Basic. Het bijzondere in deze Basic is dat er met labels gewerkt wordt.

```
*****
*
*                               HEX/ASCII-DUMP VOOR DE PROTON PC-2 COMPUTER
*
*****

10 rem *** proton hexdump ***
20 rem by Simon Voortman
30 rem  Beatrixweg 28
40 rem  3253 BB Ouddorp
50 rem  The Netherlands
60 rem need no '?' by input
70 rem print chr$(26) =CLS
80 rem hex$ = print hex. (e.g. 0A)
90 rem whex$ = print hex. (e.g.000A)
100 rem *** Start here ***
110 #START:print chr$(26);
120 input"Start address";ST$
130 if len(ST$)>4 then goto #ERROR
140 ST=val("&"&ST$)
145 if ST>&FFFF then goto #ERROR
150 input"End address";EN$
160 if len(EN$)>4 then goto #ERROR
170 EN=val("&"&EN$)
175 if EN>&FFFF then goto #ERROR
180 rem *** Dump routine ***
190 print chr$(26);"Hexdump &";
200 print ST$;" -&";EN$
210 print:print
220 repeat
230 for LOOP=0 to 120 step 8
240 print whex$(ST+LOOP);
250 for BYTE=0 to 7
260 D=peek(ST+LOOP+BYTE)
270 poke &FF0+BYTE,D
280 print hex$(D);
290 next
300 print " ";
310 for CHAR=0 to 7
320 AS=peek(&FF0+CHAR)
330 if AS>&7E then gosub #HIGH
340 if AS<&1F then AS=asc(".")
350 print chr$(AS);
360 next
370 print
380 next
390 ST=ST+&80
400 until ST=>EN
410 end
420 #ERROR
430 print"Error: type mismatch"

440 for X=0 to 2000:next X
450 goto #START
460 #HIGH
470 if AS<&A4 then goto #OOR
480 if AS>&D7 then goto #OOR
490 if AS=&A4 then AS=asc("+"):return
500 if AS=&A5 then AS=asc("-"):return
510 if AS=&A6 then AS=asc("*"):return
520 if AS=&A7 then AS=asc("/"):return
530 if AS=&A8 then AS=asc("^"):return
540 if AS=&A9 then AS=asc(">"):return
550 if AS=&AC then AS=asc("="):return
560 if AS=&AD then AS=asc("<"):return
570 if AS=&CA then AS=asc("\"):return
580 if AS=&CB then AS=asc("#"):return
590 if AS=&D3 then AS=asc("@"):return
600 if AS=&D6 then AS=asc("["):return
610 if AS=&D7 then AS=asc("]"):return
620 #OOR:rem Out Of Range
630 AS=asc("."):return
```



Fred Behringer, München, Duitsland

VIC-20: Recovering BASIC Programs when All Seems Lost

Suppose you are developing a BASIC program. The way I do it is as follows:

(1) Load program, (2) modify program, (3) test, (4) reset (see below),
(5) reload, (6) modify definitely, (7) save. Clearly, it is but a question
of time when it first occurs to me to inadvertently hit the reset button
prior to trying to save the program. The VIC then writes an empty header on
the tape with the result that the program copy on the tape gets destroyed.
(A similar situation arises when NEW was typed in.) All seems lost. However,
everyone knows that the whole program is still contained in the VIC's
memory with the mere exception that the first link address has gone.

So what we need is what some BASIC extensions call an UNNEW routine, which
can be typed in afterwards when the BASIC program seems to have gone for
ever. The following recovering program should be typed in line by line,
every line being closed by <RETURN>.

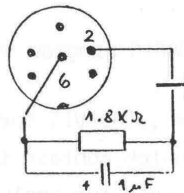
```
POKE46,PEEK(56)-1:POKEPEEK(43)+256*PEEK(44)+1,1:CLR:SYS50483
63996 I=43
63997 A=PEEK(I):B=PEEK(I+1):I=A+256*B
63998 IFPEEK(I+1)THEN63997
63999 A=I+2:POKE45,255ANDA:POKE46,A/256:CLR
RUN63996
63996
63997
63998
63999
```

Routine for recovering a BASIC program apparently lost

If the reset button was pressed, the VIC operating system charges the BASIC
memory locations No.2 and 3, which contain the link address to the next BASIC
line, with the value zero and sets the begin-of-variable pointer to the
fourth BASIC address. If we kept the contents of 45/46, the variables of the
utility program shown above would destroy the first line of the program to be
recovered. So we apply POKE46... (see above) in order to set location No.46
to an address sufficiently high so that no damage to the program to be
recovered will occur. It is PEEK(56)-1 which lowers the top of memory by one
page. Of course, any other number which leaves enough space for the recovery
utility will do as well. Next, POKEPEEK(43)... (see above) charges the BASIC
memory location No.3 with a value different from zero. This is to show

the BASIC line chaining routine 50483 that a BASIC program is present. Then, the above utility 63996-63999 should be added to the BASIC program now already recovered. By RUN63996, the utility program searches the program to be recovered for the three consecutive zeros indicating the end of the program to be recovered. The end-of-program (or rather begin-of-variables) pointer 45/46 is then set to the address of the first memory location following the three zeros. The remaining pointers are appropriately adjusted by CLR. Finally, the BASIC lines No. 63996-63999 of the utility program shown above are cleared again.

What remains to be discussed is the problem of how to get a reset switch, the VIC not having one built-in. The easiest way to get one is certainly by taking an ordinary six-pin plug which fits into the VIC's serial output connector taking advantage of the fact that contrary to what is said in Commodore's handbooks, pin No.6 is not "not connected", but rather connected instead to the CPU 6502's reset pin. So connect pins 6 and 2 of the plug fitting into the serial output connector the way is shown in the schematic diagram below. By using carefully chosen components, there should be enough room for adding a resistor and a capacitor which prevent the system from suffering a continuous reset if the reset button should be impatiently kept pressed for longer than but a moment's time.



Reset circuitry avoiding continuous reset, viewed from the six-pin plug's interior as corresponding to a view on the VIC's serial output connector.

Directe geheugenadressering vanuit Pascal.

Het komt wel eens voor dat je vanuit een Pascal programma de inhoud van een of andere byte in het geheugen wil wijzigen, bijvoorbeeld voor het bedienen van een bepaalde 'soft switch'. Helaas, (of hoera?) biedt Pascal hiervoor geen rechtstreekse ondersteuning. Een eerste oplossing bestaat er dan in dat je een routine schrijft in assembler, deze als external declareert in je Pascal programma en na compilatie m.b.v. de linker toevoegt aan de object code. Een hele operatie uiteindelijk, zodat je al vlug gaat uitkijken naar een alternatief. Dit wordt ons geboden door een combinatie van het pointer-mechanisme en de variant-record van Pascal.

Een pointer variabele is in feite niets anders dan een geheugenlocatie om het adres van een andere variabele in op te slaan. De beperkingen hierbij zijn dat die andere variabele uitsluitend een dynamisch gecreëerde variabele kan zijn en dat je als programmeur (althans in priciepe) geen enkele invloed kan uitoefenen op de waarde van een pointer variabele. Laten we een en ander even illustreren aan de hand van een eenvoudig voorbeeld.

Beschouw de volgende Pascal declaraties:

```
type pointer = ^integer;
```

```
var p:pointer;
```

Met deze declaraties wordt er 'at compile time' alleen maar geheugenruimte gereserveerd voor de variabele p. De procedure oproep new(p) in het programma heeft nu echter als gevolg dat er 'at run time' (dynamisch) geheugenruimte wordt gereserveerd om een integer waarde te stockeren, en dat het adres van deze ruimte wordt toegekend aan de variabele p. Er wordt dus a.h.w. tijdens de uitvoering van het programma op een dynamische manier een integer variabele gedeclareerd. Deze integer variabele wordt dan aangeduidt als p[^], zodat je ze bijvoorbeeld kan initialiseren m.b.v. het statement p[^]:=10. Dit pointer mechanisme wordt vooral gebruikt voor toepassingen waarbij de benodigde geheugenruimte niet op voorhand kan vastgelegd worden.

Pointer variabelen bevatten dus 'adressen' en dit is de eerste sleutel tot de oplossing van ons probleem. Wat ons nu nog rest is een manier vinden om de waarde van een pointer variabele op een of andere manier (tegen de goede zeden van Pascal in) te manipuleren. Hiervoor gebruiken we de zgn. 'variant-record'.

Variant records zijn records die verschillende vormen kunnen aannemen alnaargelang de informatie die ze bevatten. Dit kan het best verduidelijkt worden aan de hand van een voorbeeld. Neem aan dat we informatie wensen bij te houden over personen, maar dat deze informatie afhankelijk is van

het geslacht van de persoon. We willen in elk geval de naam van de persoon bijhouden, maar voor de mannen wensen we ook hun beroep te kennen, terwijl we voor de vrouwen geïnteresseerd zijn in het feit of ze al of niet gehuwd zijn.

Dit probleem kan opgelost worden d.m.v. de volgende declaratie:

```
type sex = (male,female);
      alfa = packed array[1..50] of char;
      person = record
        name:alfa;
        case s:sex of
          female:(married:boolean);
          male: (profession:alfa)
        end;
```

```
var p:person;
```

Het veld s van dit record noemt men het 'tag-field'; hierin kan je informatie opbergen omtrent de variant van het record. Bijvoorbeeld:

```
voor een man:  p.s:=male;
                p.name:='Janssens';
                p.profession:='dokter';
```

```
voor een vrouw: p.s:=female;
                p.name:='Peeters';
                p.married:=false;
```

Wil je nu de inhoud van zo'n record variabele uitschrijven dan kan je dit bijvoorbeeld als volgt doen:

```
writeln('naam = ',p.name);
if p.s = male
  then writeln('beroep = ',p.profession)
  else if p.married
    then writeln('gehuwd')
    else writeln('ongehuwd');
```

Hoe kunnen we nu dit concept gaan gebruiken om ons oorspronkelijk probleem op te lossen. Welnu we zullen gebruik maken van het feit dat de variant record in Pascal op een erg onveilige manier geïmplementeerd is geworden. De informatie in het tag-field is er uitsluitend voor de programmeur (hij hoeft dit zelfs niet te gebruiken) en wordt door Pascal zelf niet gebruikt. In het vorige voorbeeld mag men rustig schrijven write(p.profession) ook al is de waarde van p.s gelijk aan female. Het resultaat zal dan in het algemeen zijn dat er iets onleesbaars wordt uitgeschreven. Hierin ligt de tweede sleutel die we zochten verborgen. Beschouw maar even de volgende declaraties:

```
type byte      = 0..255;
word           = packed array[0..1] of byte;
address        = ^word;
variant = record
    case tag:boolean of
        true  : (adr:address);
        false : (int:integer)
    end;
```

```
var rec:variant;
```

Herinner u dat we op zoek waren naar een manier om de waarde van een pointer variabele te manipuleren. Welnu, wanneer de compiler ruimte reserveert voor de recordlocatie rec dan zal hij voor wat betreft het variant gedeelte van de record zoveel ruimte voorbehouden als nodig is om de grootste variant te bevatten. Het is dus niet zo (gelukkig maar) dat voor beide varianten aparte geheugenruimte wordt gereserveerd. In ons geval is de benodigde geheugenruimte voor beide varianten gelijk. Een integer neemt evenveel plaats in beslag als een adres (nl. 16 bits). Het effect dat ons nu interesseert is het feit dat de benamingen rec.adr en rec.int precies dezelfde fysische geheugenruimte aanduiden. Merk op dat rec.int een integer variabele is terwijl rec.adr een pointer variabele is. De aandachtige lezer heeft nu reeds begrepen dat ons probleem opgelost is. Een integer variabele kunnen we een waarde toekennen, maar als die integer variabele precies dezelfde geheugenruimte beslaat als een pointer variabele dan heeft deze laatste meteen ook een waarde gekregen!

Laten we bijvoorbeeld even aangeven hoe we de geheugenlocatie met adres 12417 de waarde 123 kunnen geven.

```
rec.int:=12417;
rec.adr^0:=123;
```

De eerste toekenning initialiseert de pointer variabele rec.adr met het gewenste adres 12417. De twee opeenvolgende bytes op dit adres worden nu beschouwd als een variabele van het type word, die wordt aangeduidt door rec.adr. Elke component van deze array neemt 1 byte in beslag, zodanig dat rec.adr^0 de gewenste byte aanduidt. (Deze constructie hebben we gekozen omdat voor elke variabele minstens 2 bytes aan geheugenruimte worden gereserveerd).

In de bijhorende listing heb ik van deze principies gebruik gemaakt om twee pascal routines te schrijven die de peek en poke instructies van Basic simuleren.

```
type byte      = 0..255;
word           = packed array(0..1) of byte;
address       = ^word;
variant = record
    case tag:boolean of
        true   : (adr:address);
        false  : (int:integer);
    end;
procedure poke(a:integer; v:byte);

var rec:variant;
begin
    rec.int:=a;
    rec.adr^:=v;
end;
{ de byte op het adres a krijgt de waarde v }

function peek(a:integer):byte;
var rec:variant;
begin
    rec.int:=a;
    peek:=rec.adr^;
end;
{ het resultaat van de functie is de inhoud van de byte op
  adres a }
```

Slotbemerking.

Ik wil hier onderstrepen dat het op een dergelijke manier gebruik maken van een hogere programmeertaal in geen geval een voorbeeld is van een goede programmeergewoonte. Ik ben zeker de laatste om het gebruik van soortgelijk kunst- en vliegwerk aan te moedigen, tenzij het niet of moeilijk anders kan. Gebruik het dan echter op een verantwoorde manier. Daarom zou ik aan raden de hiervoor beschreven techniek uitsluitend te gebruiken via de routines peek en poke. Als je drie maand later je programma bekijkt en je vindt de instructies `rec.int:=833; rec.adr^:=15` dan is de kans groot dat je flink moet nadenken voor je weet wat daarvan ook alweer de bedoeling was. Met de oproep `poke(833,15)` is deze kans een heel stuk kleiner, en zelfs nihil als je een klein beetje commentaar toevoegt aan je programma.

DE 6502 KENNER

BASICODE SWITCH-UNIT

1. FUNCTIE

Deze schakeling met bijbehorende software, zorgt ervoor dat er automatisch omgeschakeld wordt van juniorload/save naar basicodeload/save, zodat men met twee (tulp)pluggen (1xLOAD + 1xSAVE) kan volstaan.

2. HARDWARE

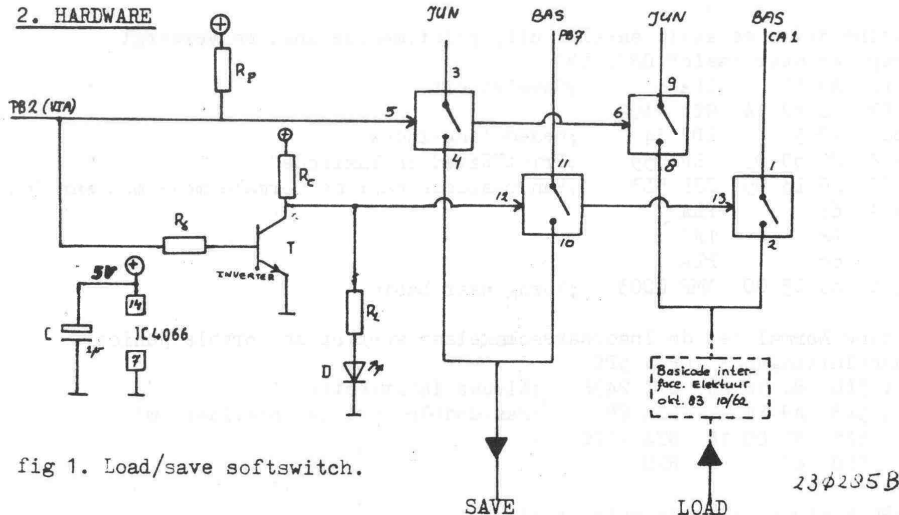


fig 1. Load/save softswitch.

Componenten

IC = 4066
T = BC 547
D = LED
Rp=Rc = 1 k
Rb = 10 k
R1 $\geq$ 330 Ω
C = 1 μ F

BEDRADING

5=6 Normal switch
12=13 Special switch
4=10 Save out
2=8 Load in
3 Normal save
11 Basicodesave
9 Normal load
1 Basicodeload

Via de PB2-poortuitgang van de VIA wordt de schakelaar bedient. Indien voor basicode gekozen is, licht de LED op. De basicodeinterface krijkt zowel het basicodesignaal als het normale juniorsignaal dat uit de cassette-recorder komt op. Bij basicodesave zit de poortuitgang (CA1) regelrecht aan de cassetterecorder. (althans via de 4066-schakelaar)
Van de elektuur basicodeinterfaceprint kunnen dus R6, R7 en C3 vervallen.

3. SOFTWARE

Om deze schakelaar te bedienen, en om de standaard load/saveleds ook bij basicode te schakelen, moet het basicodevertaalprogramma en de basicode subroutines enkele wijzigingen ondergaan.

3.1 Wijzigingen in het basicodevertaalprogramma

- 1.) 40D A9 33 LDA 33 ;Load-led aan. (was A9 73)
- 2.) 23D 20 B4 05 JSR 5B4 ;Saveled aan subroutine. (was 8D FB 03)
- 3.) 325 4C BD 05 JMP 5BD ;Saveled uit + exit. (was 68 A8 68)
- 4.) 4BA 20 D0 05 JSR 5D0 ;Normaal. (was 8C 58 24)

3.2 TOEVOEGINGEN AAN HET VERTAALPROGRAMMA

*Subroutine Sledon schakelt de saveled aan * ORG. 5B4
 SLEDON 5B4 8D FB 03 STA 3FB ;Vernietig het programma niet (clear destroyflag)
 5B7 A9 47 LDA 47 ;Normale I/o + saveled aan
 5B9 8D 82 1A STA 1A82
 5BC 60 RTS

****Routine Sexit schakelt saveled uit, print mededeling, en verzorgt de terugkeer naar basic* ORG. 5BD
 SEXIT 5BD A9 67 LDA 67 ;Saveled uit
 5BF 8D 82 1A STA 1A82
 5C2 A0 34 LDY 34 ;Mededeling-index
 5C4 20 59 05 JSR 559 ;Print"Saved in basicode"
 5C7 20 D3 05 JSR 5D3 ;Van basicode naar de normale mode mbt sav/load.
 5CA 68 PLA
 5CB A8 TAY
 5CC 68 PLA
 5CD 4C 03 00 JMP 0003 ;Terug naar basic

Subroutine Normal zet de load/saveschakelaar weer op de normale junior tape aansluitingen. ORG. 5D0
 NORMAL : 5D0 8C 58 24 STY 2458 ;Nieuwe inputvector
 NORMAL1: 5D3 A9 FF LDA FF ;Basicodeled uit (=schakelaar om)
 5D5 81 00 18 STA 1800
 5D8 60 RTS

Vanaf 59C komt de volgende data te staan:

59c 0A 0D 'Saved in basicode' 0A 0D 03
 59c 0A 0D 53 41 56 45 44 20 49 4E 20 42 41 53 49 2D 43 4F 44 45 0A 0D 03 03

Zet het programma nu op de band: S01,200,5D9 (CR)

3.3 TOEVOEGINGEN AAN DE BASICODESUBROUTINES

900 PRINT:PRINT"BASI-LOAD":POKE 6146,255:POKE 6144,0
 901 POKE 8256,0:POKE 8257,4:X=USR(X) (POKE 574,0:POKE575,226:X=USR(X))
 904 END
 905 PRINT:PRINT"BASI-SAVE":POKE 6146,255:POKE 6144,0
 906 POKE 8256,0:POKE 8257,2:X=USR(X):LIST 1000-
 907 (POKE 574,0:POKE 575,224:X=USR(X):LIST 1000-)
 908 END

Door RUN 900 in te tikken probeert de junior een basicodeprogramma te laden. Indien RUN 905 wordt ingetikt wordt het programma dat zich in het geheugen bevindt vanaf regel 1000 gesaved in basicodeformaat.

Regel 900 (voor laden) en regel 905 (voor save) zet de 4066-CMos-schakelaar in de basicode stand. De rest is conform de elektuur beschrijving.

NB: Gebruikers van Scred (Screen-editor) dienen nog de volgende drie bytes te wijzigen: 4C2 EA EA EA (was 20 34 13).

Voor de Dos-junior moeten uiteraard de adressen aangepast worden.

E. de Bruine
 Oosterland.

VOORSTEL: Wijzig subroutine 200 in:

200 GET IN\$:RETURN.

Dit is in het algemeen handiger, dan onmiddellijke terugkeer met een lege IN\$.

DE 6502 KENNER

PROGRAMMA: CASSETTE INLEGVEL PLOTTEN.

```
10 REM ***PLOTTEN VAN CASSETTE INLEGVEL
  ***OP CBM64 EN CBM1520 PLOTTER**
20 :
30 :
40 REM *****
50 REM ** PROGRAMMA VAN FER WEBER *****
  ** GEBR.WIENERSTRAAT 139 *****
60 REM ** 5913 XS VENLO *****
  *****
70 :
80 :
90 :
100 REM ***INLEIDING***
110 :
120 REM ***SCHERMKLEUREN INSTELLEN***
  00 POKE53280,6:POKE53281,6:POKE646,13
  09 REM ***SCHERM SCHOON EN VERGRENDELEN
    ***HOOFDLETTERS***
  10 PRINTCHR$(147)CHR$(142)CHR$(8)
  19 REM ***UITLEG NAAR SCHERM***
  20 PRINT" * * * C A S S E T T E [4SPC] I N
    L E G * * * "
  30 PRINT"[3SPC]DIT PROGRAMMA PLOT EEN V
    ERKLEUREN[3SPC]";
  40 PRINT"INLEGVEL DAT, NA AFKNIPPEN EN
    OUWEN[4SPC]";
  50 PRINT"LANGS DE LIJNEN, PAST IN EEN C
    SSETTE-[2SPC]";
  60 PRINT"DOOSJE. DE RUG BEVAT CASSETTEN
    AM EN[4SPC]";
  70 PRINT"NUMMER, TERWIJL OP DE VOORKANT
    MAX. 23[2SPC]";
  80 PRINT"PROGRAMMANAMEN KUNNEN WORDEN V
    RMELD.[3SPC]"
  90 PRINT"[3SPC]PLOTTER STARTKLAAR ? ! ?
    ! ? ! ? [3SPC]":PRINT:PRINT
  99 REM ***INTOETSEN VAN GEGEVENS***
  00 DIMA$(23)
  10 PRINT"TITEL VAN CASSETTE:"
  20 PRINT"[2SPC][REV][19SPC][OFF][21CL]"

30 INPUT$
40 IFLEN($)>19THENPRINT"TE LANG (19 TE
  ENS MAX.)!":T$="":GOTO210
50 PRINT"NUMMER VAN CASSETTE:"
70 INPUT$
80 IFLEN($)>3THENPRINT"NIET HOGER DAN
  99!":N$="":GOTO250
90 PRINT:PRINT
00 PRINT"VOORBEELD: 01. HI-RES COPY....
  005[5SPC]"
10 FORI=1TO23
20 PRINT"NO.[3SPC]NAAM PROGRAMMA[3SPC]T
  LLER[3SPC]<"I">:"
30 PRINT"[2SPC][REV][3SPC][OFF] [REV][1
  SPC][OFF] [REV][3SPC][OFF][26CL]";
```

```
340 INPUTA$(I)
350 IFLEN(A$(I))>24THENPRINT"TE LANG (MA
  X. 24 TEKENS)!":A$(I)="":GOTO320
360 NEXTI
399 REM ***VRAAG DEVICE NUMBER PLOTTER*
400 PRINTCHR$(147)
410 PRINT"DEVICE NUMBER PLOTTER:[3SPC]6[
  3CL]";
420 INPUTDU$:IFDU$<>"4"ANDDU$<>"6"THENDU
  $="":PRINT:PRINT"6 OF 4 !!":GOTO410
430 DU=VAL(DU$)
499 REM ***SCHERM SCHOON EN MEDEDELING*
500 PRINTCHR$(147)"PLOTTERING...."
999 REM ***OPEN BENODIGDE FILES***
1000 OPEN4,DU:OPEN1,DU,1:OPEN2,DU,2:OPEN
  44,DU,4
1099 REM ***KIES KLEUR PEN***
1100 KL=1:PRINT#2,KL
1199 REM ***MAAK PAPIER URIJHANGEND***
1200 PRINT#1,"M",0,-550:PRINT#1,"H"
1299 REM ***KNIP-VOUW-KADERLIJNEN***
1300 PRINT#1,"D",145, 0
1310 PRINT#1,"M",145, -50
1320 PRINT#1,"D", 80, -50
1330 PRINT#1,"M", 80, 0
1340 PRINT#1,"D", 80,-500
1350 PRINT#1,"M", 80,-450
1360 PRINT#1,"D",145,-450
1370 PRINT#1,"M",145,-500
1380 PRINT#1,"D", 0,-500
1390 A=145:B=-500:C=479:D=0
1400 FORI=1TO3
1410 PRINT#2,KL
1420 PRINT#1,"M",A,B
1430 PRINT#1,"D",A,D
1440 PRINT#1,"D",C,D
1450 PRINT#1,"D",C,B
1460 PRINT#1,"D",A,B
1470 A=A+5:B=B+5:C=C-5:D=D-5:KL=KL+1
1480 NEXTI
1599 REM ***RUG CBM64 EN NUMMER***
1600 A=98:B=-20:KL=3
1610 PRINT#2,KL
1620 FORI=1TO3
1630 PRINT#1,"M",A,B
1640 PRINT#4,"CBM";
1650 A=A-1:B=B-1
1660 NEXTI
1700 A=105:B=-45
1710 FORI=1TO3
1720 PRINT#1,"M",A,B
1730 PRINT#4,"64";
1740 A=A-1:B=B-1
1750 NEXTI
1800 A=110:B=-470
1810 FORI=1TO3
1820 PRINT#1,"M",A,B
1830 PRINT#4,"C";
```

DE 6502 KENNER

```

1840 A=A-1:B=B-1
1850 NEXTI
1899 REM ***VUL NUMMER AAN EN PLOT***
1900 A=98:B=-495:KL=2
1910 PRINT#2,KL
1920 IFLEN(N$)=3THEN1950
1930 IFLEN(N$)=2THENN$="0"+N$:GOTO1950
1940 IFLEN(N$)=1THENN$="00"+N$
1950 FORI=1TO3
1960 PRINT#1,"M",A,B
1970 PRINT#4,N$;
1980 A=A-1:B=B-1
1990 NEXTI
1999 REM ***PROGRAMMANAMEN***
2000 A=0:B=-35:KL=2
2010 PRINT#2,KL
2020 PRINT#1,"M",A,B
2030 FORI=1TO23
2040 PRINT#4,"[14SPC]";A$(I)
2050 NEXTI
2199 REM ***NU NOG CASSETTETITEL***
2200 A=0:B=410:KL=0
2210 PRINT#2,KL
2220 PRINT#1,"M",A,B
2230 PRINT#4,1
2240 FORI=1TOLEN(T$)
2250 PRINT#4,"[9SPC]";MID$(T$,I,1)
2260 NEXTI
2399 REM ***KLAAR EN AFSLUITEN***
2400 PRINT#4,0

```

```

2410 CLOSE4:CLOSE1:CLOSE2:CLOSE44
2999 REM ***NOG EEN KEER?***
3000 PRINTCHR$(147)"NOG EEN INLEGVEL (J/
N)";
3010 GETNO$:IFNO$=""THEN3010
3020 IFNO$="J"THENRUN
3099 REM ***TERUG NAAR BASIC***
3100 PRINTCHR$(9)CHR$(147)"COMPUTER";:EN
D

```

READY.

WAT TUSSEN VIERKANTE HAKEN STAAT ZIJN
 'VERTAALDE' GRAFISCHE STUURTEKENS; B.U.
 [21CL] = 21 MAAL CURSOR LEFT
 [REV] = REVERSE ON
 [OFF] = REVERSE OFF
 [9SPC] = 9 MAAL SPACE

*Beste Willem,
 ik vond het toch tijd worden nog eens wat van
 me te laten horen!*

*Groeten,
 Jhr.*

VIC-20 TIP

=====

Fred Behringer, Munchen, Duitsland

Het verbinden van programma's met de VIC-20.

De in de editie van april (red: DE 6502 KENNER nr. 37) door M. v.d. Velde gepu-
 bliceerde oplossing voor het verbinden van programma's voor de Commodore 64
 werkt ook met de VIC-20, omdat de RAM-plaatsen 43 t/m 46 dezelfde bedoeling heb-
 ben. Maar het gaat nog wel wat korter. Hier is mijn oplossing:

- | | |
|----------------------------|--|
| 1. LOAD | Laadt het eerste programma |
| 2. POKE254,PEEK(43) | Noteren van het eerste getal |
| 3. POKE255,PEEK(44) | Noteren van het tweede getal |
| 4. X=(PEEK(45)-2)AND255 | Eind min twee low byte |
| 5. POKE43,X | in 43. |
| 6. POKE44,PEEK(46)+(X)253) | High byte in 44. |
| 7. LOAD | Laadt het tweede programma |
| 8. POKE43,PEEK(254) | Terughalen van het eerste getal |
| 9. POKE44,PEEK(255) | Terughalen van het tweede getal |
| 10. LIST | U ziet nu een listing van de twee programma's. |
- De punten 4 t/m 7 kunnen herhaald worden, indien
 nog meer programma's verbonden moeten worden.
 Tenslotte de punten 8 t/m 9 niet vergeten!

JUNIOR IN APPLE

Frans Verberkt
Hillekensacker 12 - 10
6546 KB NIJMEGEN
080 - 779555

Aangezien er in DE 6502 KENNER nogal veel programma's voor de JUNIOR comoter afgedrukt worden en ik deze programma's overneem in een APPLE, heb ik de routines RECCHA en PRCHA herschreven voor de APPLE computer. Ook worden enkele voorbeelden van veel gebruikte en/of nuttige routines getoond. De sorongen aan het begin van het programma zijn voor Uw gemak bedoeld, zodat er niet allerlei adressen opnieuw aangepast hoeven worden als u later deze programma's aanpast of verandert. Verder verzoek ik auteurs om in programma's niet zoveel geheugenplaatsen in page zero te gebruiken, omdat APPLE hiervan druk gebruik maakt. Als u geheugenplaatsen nodig hebt, doe dat dan op de manier zoals in dit programma met de geheugenplaatsen SAVEA, SAVEX en SAVEY is gebeurd.

Gebruik page zero alleen dus voor (IND),Y of (IND,X)

Indien u PRCHA en RECCHA gebruikt, zijn de geheugenplaatsen \$20 t/m \$4F taboe. In de meeste gevallen mag u wel \$18 t/m \$1F gebruiken. Voor andere specifieke JUNIOR routines als bijvoorbeeld INPAR en PRBYT verwijs ik u naar JUNIOR COMPUTER boek 4 waarvan u ze simoel overneemt, eventueel met wijziging van geheugenplaatsen en laat ze verwijzen naar deze PRCHA en/of RECCHA. Wellicht is dit programma ook toepasbaar voor andere soorten computers door de adressen aan te passen van de APPLE routines. Probeer u dit eens en stuur uw reactie aan de redaktie van DE 6502 KENNER.

```

0930 :PAGE ZERO
0940
0950 HOR      .DE $24      :positieteller
0960 VERT     .DE $25      :reelnummer
0970
0980
0990 :APPLE ROUTINES
1000
1010 RECHTS   .DE $FBF4
1020 OP       .DE $FC1A
1030
1040 CLSCRN   .DE $FC58
1050 KEYIN    .DE $FDOC
1060 PRCHAR   .DE $FDED
1070
1080
1090 :*****
1100
1110 :SUBROUTINES
1120
1130 :*****
1140
1150
1160          .BA $9000
1170
1180
9000- 4C 09 90 1190 RECCHA   JMP REKAR
9003- 4C 18 90 1200 PRCHA     JMP PRKAR
1210
1220
9006- 1230 SAVEA     .DS 1
9007- 1240 SAVEX     .DS 1
9008- 1250 SAVEY     .DS 1
1260
1270
1280 :SUBROUTINE REKAR (RECCHA)
1290
1300 : registers X en Y
1310 : veranderen niet.
1320 : routine keert terug met
1330 : ingedrukte toets in accu.
1340
1350

```

```

9009- 20 65 90 1360 REKAR JSR SAVEXY :save reo X en Y
900C- 20 0C FD 1370 JSR KEYIN :haal toets op
900F- 29 7F 1380 AND #$7F :APPLE toets = )$80
9011- 20 6F 90 1390 JSR HERXY :herstel reo X en Y
9014- 20 03 90 1400 JSR PRCHA :en druk karakter af
9017- 60 1410 RTS
1420
1430
1440 :SUBROUTINE PRKAR (PRCHA)
1450
1460
1470 : registers X,Y en ACCU
1480 : veranderen niet in deze
1490 : routine.
1500
1510 : carriage return ($0D) is een
1520 : echte CR en geen CRLF
1530 : dus word terug gesprongen
1540 : naar begin regel.
1550
1560 : backspace ($0B) en
1570 : linefeed ($0A)
1580 : worden door APPLE
1590 : behandeld als in JUNIOR
1600
1610
1620 PRKAR JSR SAVE :save registers
1630
1640 PKRE CMP #$09 :cursur rechts?
1650 BNE PKOP
1660 JSR RECHTS
1670 JMP PKEIND
1680
1690 PKOP CMP #$0B :cursur op?
1700 BNE PKCLSC
1710 JSR OP
1720 JMP PKEIND
1730
1740 PKCLSC CMP #$0C :clear screen?
1750 BNE PKCR
1760 JSR CLSCRN
1770 JMP PKEIND
1780
1790 PKCR CMP #$0D :CR?
1800 BNE PKHOME
1810 LDA #$00 :cursur naar
1820 STA *HOR :begin regel
1830 JMP PKEIND
1840
1850 PKHOME CMP #$1C :cursur home?
1860 BNE PKGO
1870 LDA *VERT :is cursor op
1880 BEQ HOMEND :bovenste regel?
1890 JSR OP :nee dan 1 regel hoger
1900 JMP HOMTES :en test weer
1910
1920 HOMEND LDA #$00 :cursur naar
1930 STA *HOR :begin regel
1940 JMP PKEIND
1950
1960 PKGO ORA #$80 :APPLE print kar )$80
1970 JSR PRCHAR
1980 PKEIND JSR HERSTL :haal registers op
1990 RTS
2000
2010
2020 SAVE STA SAVEA
2030
2040 SAVEXY STX SAVEX
2050 STY SAVEY
2060 RTS
2070
2080
2090 HERSTL LDA SAVEA
2100
2110 HERXY LDX SAVEX
2120 LDY SAVEY
2130 RTS
2140

```

```

2150
2160 :enkele voorbeelden
2170
2180
2190 PIEP      LDA #07
9076- A9 07    JSR PRCHA
2200
9078- 20 03 90 RTS
907B- 60      2210
2220
907C- A9 08    LDA #08
907E- 20 03 90 JSR PRCHA
9081- 60      2230
2240
2250
2260
9082- A9 09    LDA #09
9084- 20 03 90 JSR PRCHA
9087- 60      2270
2280
2290
2300
9088- A9 0A    LDA #0A
908A- 20 03 90 JSR PRCHA
908D- 60      2310
2320
2330
2340
908E- A9 0B    LDA #0B
9090- 20 03 90 JSR PRCHA
9093- 60      2350
2360
2370
2380
9094- A9 0C    LDA #0C
9096- 20 03 90 JSR PRCHA
9099- 60      2390
2400
2410
2420 :
2430 : APPLE behoeft geen wachtlus
2440 : JUNIOR wel in clear screen
2450 :
2460
2470
909A- A9 0D    LDA #0D
909C- 20 03 90 JSR PRCHA
909F- 20 88 90 JSR LF
90A2- 60      2480
2490
2500
2510
2520
90A3- A9 1C    LDA #1C
90A5- 20 03 90 JSR PRCHA
90A8- 60      2530
2540
2550
2560
90A9- A9 20    LDA #20
90AB- 20 03 90 JSR PRCHA
90AE- 60      2570
2580
2590
2600
2610
2620 : N.B. Voor reactie spellen
2630
2640 : is APPLE veel te snel
2650
2660 : omdat JUNIOR de karakters
2670
2680 : bit voor bit uitzend
2690
2700 : Maak voor dit soort
2710
2720 : gevallen een wachtlus
2730
2740 : in PRCHA
2750
2760
2770

```

.EN

DESCRIPTION OF COPYING JOURNAL-LAYOUTS

By : Siegfried Losensky, Gross-Umstadt, Germany

Make a photocopy of the corresponding page. The copy must be high contrasted, to get sharp black (=) white borders.
 But make sure, that the backside of original-page does not shine through on copy. Otherwise lower contrast of copier.
 The best results you will have from copiers working with liquid-toner such as Nashua or Toshiba. Copiers with powdered toner are not suitable, because the borders of black will disperse, when you will spray this copies with "Pausblat-Spray", to make it translucent.
 The copy of liquid-toner-copiers you can copy onto photopositive-epoxyvd as usual and described at Elektor layout-page-instruction.

 WIE HEEFT EEN OPLOSSING
 VOOR HET VOLGENDE PROBLEEM:
 MIJN JUNIOR+VDU-KAART WIL
 WEL MET ALLE SOFTWARE SAMEN
 WERKEN, MAAR MET EEN KB-9
 BASIC IS EEN PROGRAMMA NIET
 TE LISTEN. ALS DE EERSTE
 REBEL BIJV. 10 PRINT ... IS
 KOMT ER NA DE OPDRACHT LIST
 OP HET SCHERM: 10 P SYNTAX
 ERROR. KONFLIKTEN OP PAG. 0
 ONTSTAAN ER VOLGENS MIJ
 NIET. WAT ZOU DE OORZAAK
 KUNNEN ZIJN?
 M.J. SPITHORST
 WETH. HUISMANLAAN 51
 9902 LP APPINGEDAM.

 OP APPLE II GEBRUIK IK O.A.
 CPM MET DE Z 80-KAART. CPM
 FORMATEERT 35 TRACKS OP DE
 DISKETTES. ER BESTAAN MOGE-
 LIJKHEDEN OM 40 TRACKS TE
 FORMATEREN VOOR HET APPLE-
 DOS. WEEET IEMAND HOE DAT
 MOET MET CPM?
 JOHN VAN SPRANG
 TULP 71
 2925 EW KRIMPEN A.D. IJSSEL

 TE KOOP:
 JUNIOR COMPUTER + INTERFACE
 KAART + ELEKTERMINAL + MINI
 EPROMKAART (DISASSEMBLER +
 EPRUTL) + 2*16k RAM + RCA -
 KEYBOARD + CASSETTES.
 VRAAGPRIJS FL. 250,-.
 M.LAMEIJ
 PRINS CLAUSLAAN 81
 LOCHEM TEL.: 05730-4427.

 OP DE BIJEENKOMST TE LEIDEN
 OP 21 SEPTEMBER JONGSTLEDEN
 WAREN LEDEN AANWEZIG AFKOM-
 STIG O.A. UIT APPINGEDAM,
 GENT EN ZELFS ONZE VRIEND
 UIT COGOLIN (FRANKRIJK) WAS
 ER WEER. EEN BEWIJS DAT DIE
 BIJEENKOMSTEN DE MOEITE VAN
 EEN LANGE RIT WAARD ZIJN.

 HEEFT U DE LAATSTE TIJD NOG
 BASICODE-PROGRAMMA'S OPGE-
 NOMEN? STUUR ZE DAN OOK OP
 NAAR DE REDAKTIE.
 DANK AAN DE LEDEN DIE POSI-
 TIEF REAGEERDEN OP DE
 OPROEP.

C64JUN The JUNIOR promoting CY (W&J) 290785

```

0010: *****
0020: * R.A.F. BENS *
0030: * TJALKSTRAAT 25 *
0040: * 1784 RX DEN HELDER *
0050: * TEL: 02230-33379 *
0060: *****
0070:
0080: M.b.v. dit programma kunnen Commodore programma's
0090: welke op cassette staan (niet in Turbo) in de Ju-
0100: nior worden ingelezen.
0110: Als alle data correct van de cassette is ingelezen,
0120: blijft het programma in een "loop" staan.
0130: Zodra er een fout tijdens het inlezen wordt gedetec-
0140: teerd, stopt het programma en meldt de computer zich
0150: met JUNIOR.
0160: Ander soort foutmeldingen zijn niet ingebouwd om het
0170: programma overzichtelijk te houden.
0180: Het standaard display wordt gebruikt om visuele
0190: informatie te verkrijgen tijdens het inlezen.
0200: De data, welke van de cassette wordt gelezen, komt in
0210: het geheugen op $2000 en hoger en niet op de adressen
0220: zoals in de Commodore.
0230: Let er wel op, dat de Commodore de programma's 2x op
0240: tape zet, zodat na deze geladen is, het programma 2x
0250: in het geheugen staat. (De Commodore gebruikt dit
0260: voor controle)
0270: Voor het inlezen van het programma moet de cassette-
0280: kop tussen de ident en het programma gepositioneerd
0290: worden, wat m.b.v. een koptelefoon te horen is, daar
0300: de ident header langer is dan de programma header.
0310:
0320: Als hardware gebruik ik de Basicode-interface zoals
0330: beschreven in Elektuur Okt-83.
0340:
0350: 0200 C64JUN ORG $0200
0360:
0370: * USED ZERO PAGE ADDRESSES
0380:
0390: 00 00 CNTL * $0000 COUNT 256 SYNCs
0400: 01 00 CNTH * $0001 COUNT 16x256 SYNCs
0410: 02 00 HLFPTM * $0002 CONTAIN HALF PERIOD TIME
0420: 03 00 MEDIUM * $0003 CONTAIN MEDIUM PERIOD TIME
0430: 04 00 LONG * $0004 CONTAIN LONG PERIOD TIME
0440: FA 00 POINTL * $00FA
0450: FB 00 POINTH * $00FB
0460:
0470: 5F 10 JUNIOR * $105F
0480: 13 12 INCPNT * $1213
0490:

```

```

0500:          * VIA ADDRESSES
0510:
0520:      0C 1B PCR    *    $180C PERIPHERAL CONTROL REGISTER
0530:      0D 1B IFR    *    $180D INTERRUPT FLAG REGISTER
0540:      0E 1B IER    *    $180E INTERRUPT ENABLE REGISTER
0550:
0560:          * PIA ADDRESSES
0570:
0580:      B0 1A PAD     *    $1A80 PORT A DATA REGISTER
0590:      B2 1A PBD     *    $1A82 PORT B DATA REGISTER
0600:      F5 1A CNTB    *    $1AF5 CLK8T, DISABLE TIMER IRQ
0610:      F6 1A CNTC    *    $1AF6 CLK64T, DISABLE TIMER IRQ
0620:
0630: 0200 A9 7F      READ  LDAIM $7F
0640: 0202 8D 0E 1B  STA  IER      SET INTERRUPT DISABLE MODE
0650: 0205 A9 00      LDAIM $00
0660: 0207 A2 20      LDXIM $20
0670: 0209 86 FB      STX  POINTH SET POINTER ON $2000
0680: 020B 85 FA      STA  POINTL
0690: 020D 8D 0C 1B  STA  PCR      SET CA1 NEGATIVE EDGE DETECT
0700: 0210 A9 73      LDAIM $73
0710: 0212 8D 82 1A  STA  PBD
0720: 0215 A9 10      HEADER LDAIM $10
0730: 0217 85 01      STA  CNTH      SET PERIOD COUNTER
0740: 0219 A9 36      LDAIM $36
0750: 021B 20 6A 02  JSR  NOSYNC DISPLAY NOSYNC CHARCTER
0760: 021E 20 76 02  HDR    JSR  PERIOD GET PERIOD TIME FROM TAPE
0770: 0221 C9 22      CMPIM $22
0780: 0223 90 F0      BCC  HEADER PERIOD TIME < 2000 HZ
0790: 0225 C9 4E      CMPIM $4E      NO
0800: 0227 B0 EC      BCS  HEADER PERIOD TIME > 2800 HZ
0810: 0229 E6 00      INC  CNTL      NO
0820: 022B D0 F1      BNE  HDR      256 PERIODS ?
0830: 022D AB      TAY
0840: 022E 20 68 02  JSR  SYNC      DISPLAY SYNC CHARACTER
0850: 0231 98      TYA
0860: 0232 C6 01      DEC  CNTH
0870: 0234 D0 E8      BNE  HDR
0880: 0236 18      CLC
0890: 0237 69 0A      ADCIM $0A      CREATE MEDIUM PULS
0900: 0239 85 03      STA  MEDIUM AND STORE IT
0910: 023B 18      CLC
0920: 023C 69 14      ADCIM $14      CREATE LONG PULS
0930: 023E 85 04      STA  LONG      AND STORE IT
0940: 0240 A9 0A      LDAIM $0A
0950: 0242 85 00      STA  CNTL      FIRST 10 BYTE'S WILL BE DELETED
0960: 0244 20 68 02  WAIT  JSR  SYNC
0970: 0247 20 76 02  JSR  PERIOD GET PERIOD TIME FROM TAPE
0980: 024A C5 04      CMP  LONG      AND WAIT FOR LONG PULS
0990: 024C 30 F6      BMI  WAIT      LONG PULS ?
1000: 024E 20 76 02  LOAD  JSR  PERIOD GET PERIOD TIME FROM TAPE
1010: 0251 20 9B 02  JSR  RDBYTE READ ONE BYTE FROM TAPE
1020: 0254 C6 00      DEC  CNTL
1030: 0256 D0 07      BNE  PAR      NOT 10 DELETIONS ?
1040: 0258 E6 00      INC  CNTL

```

```

1050: 025A 91 FA      STAIY POINTL AND STORE IT
1060: 025C 20 13 12   JSR   INCPNT POINTER=POINTER+1
1070: 025F 20 76 02   PAR   JSR   PERIOD GET PERIOD TIME FROM TAPE
1080: 0262 C5 04      CMP   LONG   WAIT FOR LONG PULS AND
1090: 0264 30 F9      BMI   PAR    THROW PARITY-BIT AWAY
1100: 0266 10 E6      BPL   LOAD
1110:
1120: 0268 A9 69      SYNC   LDAIM $69
1130: 026A 8D 80 1A   NOSYNC STA   PAD
1140: 026D AD 82 1A   LDA   PBD
1150: 0270 49 02      EORIM $02      CHANGE DISPLAYS
1160: 0272 8D 82 1A   STA   PBD
1170: 0275 60      RTS
1180:
1190: 0276 20 79 02   PERIOD JSR   HLFPER
1200: 0279 A9 02      HLFPER LDAIM $02
1210: 027B 2C 0D 18   HLF   BIT   IFR
1220: 027E F0 FB      BEQ   HLF   NO ACTIVE EDGE ON CA1
1230: 0280 8D 0D 18   STA   IFR   CLEAR CA1 FLAG
1240: 0283 A9 01      LDAIM $01
1250: 0285 4D 0C 18   EOR   PCR
1260: 0288 8D 0C 18   STA   PCR   OPPOSITE ACTIVE CA1 EDGE DETECT
1270: 028B A9 FF      LDAIM $FF
1280: 028D AA      TAX
1290: 028E 4D F6 1A   EOR   CNTC  GET ELAPSED TIME IN ACCU
1300: 0291 8E F5 1A   STX   CNTB  RESET TIMER
1310: 0294 AA      TAX
1320: 0295 18      CLC
1330: 0296 65 02      ADC   HLFPTM FULL PERIOD TIME IN ACCU
1340: 0298 86 02      STX   HLFPTM SAVE LAST HALF PERIOD TIME
1350: 029A 60      RTS
1360:
1370: 029B A9 55      RDBYTE LDAIM $55
1380: 029D 20 6A 02   JSR   NOSYNC DISPLAY LOAD CHARACTER
1390: 02A0 A0 08      LDYIM $08   SET BIT COUNTER
1400: 02A2 48      RB    PHA    SAVE ACCU
1410: 02A3 20 76 02   JSR   PERIOD GET PERIOD TIME FROM TAPE
1420: 02A6 C5 03      CMP   MEDIUM
1430: 02A8 30 15      BMI   SRT   IF SHORT PULS THEN BRANCH
1440: 02AA C5 04      CMP   LONG   MEDIUM OR LONG PULS
1450: 02AC B0 07      BCS   FAULT  IF LONG PULS THEN FAULT
1460: 02AE 20 76 02   MED   JSR   PERIOD MEDIUM PULS
1470: 02B1 C5 03      CMP   MEDIUM
1480: 02B3 30 03      BMI   ONE   IF MEDIUM PULS THEN C=1
1490: 02B5 4C 5F 10   FAULT JMP   JUNIOR
1500: 02B8 38      ONE   SEC
1510: 02B9 68      SHIFT PLA    RESTORE ACCU
1520: 02BA 6A      RORA   ROTATE ACCU WITH CARRY
1530: 02BB 88      DEY    BIT COUNTER NOT ZERO
1540: 02BC D0 E4      BNE   RB    THEN BRANCH
1550: 02BE 60      RTS    ELSE EXIT
1560: 02BF 20 76 02   SRT   JSR   PERIOD GET PERIOD TIME FROM TAPE
1570: 02C2 C5 04      CMP   LONG
1580: 02C4 B0 EF      BCS   FAULT  IF LONG PULS THEN FAULT
1590: 02C6 90 F1      BCC   SHIFT  MEDIUM PULS THEN C=0

```

AIM-65 Basic

Tokenized Microsoft Basic Keywords and addresses

W.L. van Pelt

Analogous to the publication for Commodore-64 by A. Mueller,

DE 6502 KENNER, December 1983, pages 5-8.

COMMANDS		
KEYWORDS CORRESPONDING TO B090		
ADDRESSES CORRESPONDING TO B00A		
THE ADDRESSES OF ROUTINES FOR COMMANDS ARE THE ADDRESSES		
MINUS 1, BECAUSE THE ROUTINES ARE INVOKED THROUGH RTS.		

KEYWORD	TOKEN	ADDR-1			
END	80	B65D	WAIT	92	C56B
FOR	81	B55B	LOAD	93	E847
NEXT	82	BAFF	SAVE	94	B69E
DATA	83	B766	VERIFY	-	-
INPUT#	-	-	DEF	95	COF0
INPUT	84	B9BB	POKE	96	C562
DIM	85	BDD9	PRINT#	-	-
READ	86	B9EF	PRINT	97	B8A8
LET	87	B813	CONT	98	B684
GOTO	88	B713	LIST	99	B4BB
RUN	89	B6EB	CLR	9A	B480
IF	8A	B796	CMD	-	-
RESTORE	8B	B630	SYS	-	-
GOSUB	8C	B6F6	OPEN	-	-
RETURN	8D	B740	CLOSE	-	-
REM	8E	B7A9	GET	9B	B9AC
STOP	8F	B65B	NEW	9C	B464
ON	90	B7B9			
NULL	91	BF86			

MISCELLANEOUS KEYWORDS		
KEYWORDS CORRESPONDING TO B100		
KEYWORD	TOKEN	
TAB(	9D	
TO	9E	
FN	9F	
SPC(	A0	
THEN	A1	
NOT	A2	
STEP	A3	

DE 6502 KENNER

AIM-65 Basic

Tokenized Microsoft Basic keywords and addresses

DYADIC OPERATORS

KEYWORDS CORRESPONDING TO B117

PRIORITIES AND ADDRESSES CORRESPONDING TO B072

THE ADDRESSES OF ROUTINES FOR DYADIC OPERATORS ARE THE ADDRESS MINUS 1. BECAUSE THE ROUTINES ARE INVOKED THROUGH A RTS INSTRUCTION.

KEYWORD	TOKEN	ADDR-1	PRTY	
+	A4	C5A8	79	addition
-	A5	C591	79	subtraction
*	A6	C769	7B	multiplication
/	A7	C850	7B	division
^	A8	CC7E	7F	exponentiation
AND	A9	BD41	50	logical AND
OR	AA	BD3E	46	logical OR
monadic "-"	AB	CCB7	7D	negation
monadic NOT	AC	BC9B	5A	logical NOT
>=<	AD	BD6E	64	comparison

FUNCTIONS

FUNCTIONS CORRESPONDING TO B148

ADDRESSES CORRESPONDING TO B044

KEYWORD	TOKEN	ADDR			
SGN	AE	C978	TAN	BA	CE22
INT	AF	CA0B	ATN	BB	00BB
ABS	B0	C997	PEEK	BC	C54C
USR	B1	0003	LEN	BD	C4BA
FRE	B2	C0BD	STR\$	BE	C1A3
POS	B3	CODE	VAL	BF	C4EB
SQR	B4	CC75	ASC	C0	C4C9
RND	B5	CD96	CHR\$	C1	C42A
LOG	B6	C729	LEFT\$	C2	C43E
EXP	B7	CCF1	RIGHT\$	C3	C46A
COS	B8	CDD2	MID\$	C4	C475
SIN	B9	CDD9			

Note: 1) Bit 7 in the last character of each keyword is set to determine the end of a keyword

2) Keywords are tokenized by adding x'80' to their relative (hex) position in the table.

AIM-65 BASIC Extension

Tokenized Microsoft Basic Keywords and addresses

Phons Bloemen

Analogous to the publication for Commodore-64 by A. Mueller.

DE 6502 KENNER, December 1983, pages 5-8.

COMMANDS					
EXTRA KEYWORDS CORRESPONDING TO D97A					
ADDRESSES CORRESPONDING TO D9C6					
THE ADDRESSES OF ROUTINES FOR COMMANDS ARE THE ADDRESSES					
MINUS 1. BECAUSE THE ROUTINES ARE INVOKED THROUGH RTS.					
KEYWORD	TOKEN	ADDR-1			
AUT#	--	D300	HEX\$	--	D835
DEC	--	D8C5	TRACE	--	D68B
TAPIN	--	D45C	TRNC	--	D963
SYS	--	D343	OFF	--	D6F4
RAD	--	D921	HELP	--	D757
WRITE	--	D394	FRAC	--	D870
SST	--	D601	TIME	--	D397
FETCH	--	D45D	CLOSE	--	D4A2
DEG	--	D90D	OPEN	--	D354
ROUND	--	D9D5			

The extra keywords are not tokenized. The interpreter stores them not as tokens, but as variables into memory. So look for the word not for the token !

WIJ STRAAN OP VRIJDAG EN ZATERDAG 22/23 NOVEMBER AANSTAANDE MET VEEL ENTHOUSIASME OP DE HCC-DAGEN IN DE JAARBEURS TE UTRECHT. KOM ONZE STANDS BEZOEKEN EN BESONDER HET NIEUWE MONGS/DOS65 SYSTEEM VAN ONZE CLUB. BRENG UW HOBBIEVRIENDEN MEE. ZIJN ZE NOG GEEN LID? BRENG ZE DAN ZEKER MEE.

TE KOOP AANGEBODEN:

ZWAAR METALEN KAST VOOR INBOUW VAN BIJVOORBEELD ELEKTUUR-COMPUTER EN TOETSENBORD. OP DE KAST KAN EEN MONITOR WORDEN GEPLAATST.

TWEE INBOUW FRONTLOADER CASSETTEDECKS TYPE TOURING 108, BEKEND DOOR RADIO SERVICE TWENTHE. GEEN ONBOUW EN VOEDING AANWEZIG DUS. GELIEFD OBJECT VOOR ZELFBOUWERS. ARTIKEL UIT RADIO BULLETIN AANWEZIG.

DIVERSE RAMKAARTEN VOOR DE ELEKTUUR JUNIOR-COMPUTER, VOORZIEN VAN COMPONENTEN.

ZWAAR TERMINALKAST MET INGEBOUWD SCHERM EN TOETSENBORD, ALSNEDE (ONTKOPPELD) VIDEO-BOARD (16 REBELS), GADEN.

WILT U MEER INFORMATIE OVER PRIJZEN EN DERGELIJKE, SCHRIJF DAN EVEN EEN BRIEFJE AAN

WILLEN L. VAN PELT, JACOB JORDAENSSTRAAT 15, 2923 CX KRIMPEN AAN DEN IJSSSEL.

OCTOPUS/SAMSUNG5:

INMIDDELS IS BEKEND GEWORDEN DAT. INDIEN MEN DE HIER GE-NOEMDE COMPUTER GAAT BOUWEN. MEN OOK GRAAG OVER DE SYSTEEM-SOFTWARE WIL BESCHIKKEN. STUUR EEN OP UW NAAM GESTELDE RE-KENING VAN EEN GEAUTORISEERDE DEALER NAAR DE REDAKTIE. IN-DIEN HIERUIT BLIJKT DAT U DE RECHTWATIGE EIGENAAR BENT VAN DE SYSTEEMSOFTWARE VOOR DE OHIO-DOSCOMPUTER, DAN KAN DE RE-DAKTIE U VERDER HELPEN AAN SYSTEEMSOFTWARE VOOR UW COMPUTER. UITERHAARD TEGEN BETALING. DE PRIJZEN MOETEN NOG WORDEN VASTGESTELD, EVENALS HET PAKKET. TOT HET PAKKET ZUL-LEN GEEN ONDERDELEN BEHOEDEN WELKE NORMAAL ONDER DE WETTE-LIJKE BESCHERMING VALLEN, ZOALS HOGERE PROGRAMMEERTALEN.

LEDEN MET EEN APPLE-COMPUTER DIE PRO-DOS WILLEN GAAN GE-BRUIKEN SLAGEN ER NIET ALLEN ALTIJD IN HUN PRO-DOS DISKETTE OP TE BOOTEN, WANNEER EEN NIET-ORIGINELE FLOPPY-CONTROLLER GEBRUIKT WORDT. DIT KAN VERDOORZAAKT WORDEN DOOR HET FEIT DAT DE EPROM IN DE FLOPPY-DRIVE KAART (FDC) NIET GEPROGRAM-MEERD IS MET 00 OP DE HOOGSTE BYTES IN HET ADRESBEREIK \$300 T/M \$3FF. IS UW FDC BEREIKBAAR IN SLOT 6, DAN MOETEN DUS DE ADRESSEN \$C6FB T/M \$C6FF MET 00 GEVULD WORDEN. BOVENDIEN CONTROLEERT PRO-DOS BIJ OPBOOTEN OF DE NAAM APPLE II ER IS.

```

36 39 MLIST
SCR # 36
0 \ ***** PATCHES FOR PDS-65 FORTH V1.0 *****
1 \
2 \ THESE PATCHES ARE TO BE USED WITH A SENIOR COMPUTER
3 \ AND A FORTH INTERPRETER/COMPILER FROM
4 \ PROTON OR MR FRANSSEN.
5 \
6 \ THE INTERPRETER/COMPILER BECOMES PATCHED WHEN YOU
7 \ LOAD THESE PATCHES BY THE COMMAND
8 \ SCR# LOAD
9 \
10 \ WHEN FORTH IS PATCHED, YOU CAN WRITE IT TO DISK.
11 \
12 \ THE PROGRAM CHECKS FOR THE RIGH VERSION OF FORTH.
13 \ IF FORTH IS ALREADY PATCHED BY THIS ROUTINE OR IF YOU
14 \ TRY TO PATCH A WRONG VERSION, YOU GET THE MESSAGE:
15 \ "WRONG DATA."

SCR # 37
0 \ ***** PATCHES ON PDS-65 FORTH V1.0 *****
1 HEX \ PATCHES FOR 1. 40 TRACKS/SIDE
2 \ 2. 2 DOUBLE SIDED DRIVES WITH
3 \ THE SAME NUMBERING AS IN TE
4 \ PDS-MONITOR
5 : CONVERT \ CONVERTS DRIVE NR. 1.2.3.4 TO 0.2.1.3
6 DUP 1 ( OVER 4 ) OR
7 IF DROP CR ." ILLEGAL DRIVE NUMBER." 6 TO ERR ABORT
8 ELSE
9 2 /MOD SWAP 0= IF 1+ ENDIF
10 ENDIF :
11 : DUNIT CONVERT DUNIT : \ NEW DEFINITION OF DUNIT
12 : FORMAT CONVERT FORMAT : \ NEW DEFINITION OF FORMAT
13 : INIT1 INIT 1 DUNIT : \ NEW INIT ROUTINE TO MAKE SURE THAT
14 --) \ THE SYSTEM IS WORKING ON DRIVE 1
15 \ AFTER A COLD START.

SCR # 38
0 \ ***** PATCHES ON PDS-65 FORTH V1.0 *****
1 \ PATCHES FOR 40 TRACKS/SIDE, 720 BLOCKS, 90 SCREENS
2 : PATCH
3 \ 40 TRACKS IN R/W I.S.O. 35:
4 3F2A DUP CR ." PATCHING: " H. DUP
5 B@ 23 = IF 28 SWAP B! ELSE ." WRONG DATA." ENDIF
6 \ 40 TRACKS IN FORMAT:
7 3BC9 DUP CR ." PATCHING: " H. DUP
8 B@ 22 = IF 27 SWAP B! ELSE ." WRONG DATA." ENDIF
9 \ 720 BLOCKS I.S.O. 630:
10 41A1 DUP CR ." PATCHING: " H. DUP
11 @ 0276 = IF 02CF SWAP ! ELSE ." WRONG DATA." ENDIF
12 \ 90 SCREENS IN INDEX I.S.O. 78:
13 450C DUP CR ." PATCHING: " H. DUP
14 B@ 4E = IF 5A SWAP B! ELSE ." WRONG DATA." ENDIF
15 --)

SCR # 39
0 \ ***** PATCHES ON PDS-65 FORTH V1.0 *****
1 \ PATCH FOR 2 DOUBLE SIDED DRIVES.
2 \ REPLACE 0 DUNIT BY 0 DROP IN R/W:
3 418C DUP CR ." PATCHING: " H. DUP
4 @ 401C = IF 2039 SWAP ! ELSE ." WRONG DATA." ENDIF
5 \ REPLACE INIT BY INIT1 IN COLD.
6 39A7 DUP CR ." PATCHING: " H. DUP
7 @ 3FDD = IF INIT1 SWAP ! ELSE ." WRONG DATA." ENDIF
8 :
9 PATCH CR ." PDS-65 FORTH PATCHED." DECIMAL
10 FORGET PATCH HERE TO FENCE COLD
11 \
12 \ AUTHOR: GERT VAN OPBROEK (GEVOP)
13 \ HOOGLANDEN 20
14 \ 9801 LB ZUIDHORN 05940-5627
15 :S

```

```

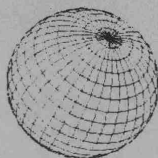
4 LIST
SCR # 4
0 ( LIST zonder regelnummers          LLIST  LLISTS      )
1 EDITOR DEFINITIONS HEX ( voeg de woorden toe aan de editor )
2 ( n LLIST list scherm n zonder nummers )
3 : LLIST ( n ---- )
4   FORTH ( zoek in FORTH vocabulary )
5   SCR ! ( bewaar het scherm nummer )
6   10 0 DO ( lus voor zestien regels )
7     CR I SCR @ ( regelnummer en schermnummer )
8     .LINE ( druk van scherm n1 regel n2 af )
9   LOOP EDITOR : ( terug naar EDITOR vocabulary )
10 ( n1 n2 LLISTS list scherm n1 t/m scherm n2 zonder nummers )
11 : LLISTS ( n1 n2 ---- )
12   1+ SWAP ( stack voorbereiding voor DO )
13   DO FORTH I ( schermnummer )
14   EDITOR LLIST ( list het scherm zonder nummers )
15   LOOP CR : :S
OK
5 LIST
SCR # 5
0 ( printer hulpoes. constanten. ?ON-LINE . ?OFF-LINE )
1 FORTH DEFINITIONS HEX
2 1820 CONSTANT STATUS ( adres inputpoort printer status )
3 5F CONSTANT ON-LINE ( waarde van statuspoort )
4 3F CONSTANT OFF-LINE ( waarde van statuspoort )
5
6 : ?ON-LINE ( wacht totdat de printer on-line staat )
7   BEGIN
8   STATUS C@ ON-LINE =
9   UNTIL :
10
11 : ?OFF-LINE ( wacht totdat de printer off-line staat )
12   BEGIN
13   STATUS C@ OFF-LINE =
14   UNTIL :
15 --)
OK
6 LIST
SCR # 6
0 ( afdrukken zonder nummers          BRIEVEN )
1 EDITOR DEFINITIONS DECIMAL
2 ( BRIEVEN druk scherm n1 t/m n2 af zonder nummers. Wacht op
3   een toets en begin opnieuw als deze ongelijk is aan NULL )
4 : BRIEVEN ( n1 n2 ---- )
5   CR ." Zet de printer aan !" CR
6   ?ON-LINE ( wacht op on-line van printer )
7   BEGIN
8   OVER OVER ( n1 n2 ---) n1 n2 n1 n2 )
9   LLISTS ( druk scherm n1 t/m scherm n2 af )
10  KEY ( wacht op een toetsaanslag )
11  0= UNTIL ( begin opnieuw als ascii < > $00 )
12  DROP DROP ( werk de stapel bij )
13  ?OFF-LINE ( wacht op off-line van printer )
14  CR :
15 ;S frans bakx huissteden 1112 6605 hd wijchen
OK

```

40 grafische programma's
voor de APPLE II, IIE en IIC

Leer programma's
met Apple II grafische
in APPLEBASIC

Manual Sutter



ACADEMIC SERVICE

40 GRAFISCHE PROGRAMMA'S VOOR DE APPLE II, IIE en IIC

M. Sutter

f 29,50 130 p. isbn 90 6233 154 8

Op het hoge resolutiescherm van de Apple (280 bij 192 beeldpunten) kunnen met vrij eenvoudige programma's schitterende tekeningen gemaakt worden. Dit boek geeft hiervan een 40-tal voorbeelden. De programma's met uitleg omvatten onder meer het tekenen van wiskundige functies, driedimensionaal tekenen, educatieve toepassingen en LOGO-simulaties.

DE 6502 KENNER

FATE 65 MANUAL ENGLISH VERSION

PAGE: 0001

Editorial Office DE 6502 KENNER
c/o Jacob Jordaensstraat 15
2923 CK Krimpen aan den IJssel
The Netherlands

```

FFFFFFFFFFFF      AAAA      TTTTTTTTTTTT      EEEEEEEEEEEE
FFFFFFFFFFFF      AAAAAA      TTTTTTTTTTTT      EEEEEEEEEEEE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FFFFFFFFFF      AAAAAAAAAA      TT              EEEEEEEE
FFFFFFFFFF      AAAAAAAAAA      TT              EEEEEEEE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE
FF              AA      AA      TT              EE

```

F ormat lister
A ssembler
T ool-utilities
E ditor

```

M  M      A      N  N      U  U      A      L
MM MM      A A      NN N      U  U      A A      L
M  M  M      A A      N N N      U  U      A A      L
M  M  M      AAAAA      N NN      U  U      AAAAA      L
M  M  M      A A      N  N      U  U      A A      L
M  M  M      A A      N  N      U  U      A A      L
M  M  M      A A      N  N      UUU      A A      LLLL

```

Published with permission
of Proton Electronics. Naarden
Author: Rob Banen
Transl: Jaap de Hood

JUNI 1984