frankeren
als briefkaart

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

frankeren
als briefkaart

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

New Brain on-line

uitgave van de
NewBrain-gebruikersgroep

8

oktober 1986

newbrain on-line is een officiële uitgave van de nederlandse newbrain-gebruikersgroep. de bladen zijn licht gelijmd zodat ze eenvoudig los te maken en daarna in een ringband op te bergen zijn. de artikelen kunnen dan bij voorbeeld op onderwerp geordend worden. de perforatie is afgestemd op de ringband met hcc-opdruk, die bij de hcc te koop is.

het is voor de redactie nu eenmaal onmogelijk om alle ingezonden artikelen en programma's op orginaliteit te controleren. de aansprakelijkheid voor de ingezonden stukken ligt derhalve dan ook bij de inzender.

voor informatie omtrent commerciële advertenties of advertenties van leden, kunt u contact opnemen met de redactie.
door middel van de aanmeldingskaart achterop het omslag kunt u zich verzekeren van de toezending van nieuwe nummers van on-line. reeds verschenen nummers kunt u bestellen door overmaking van ƒ 10,- per nummer op girorekening 2505800 ten name van hcc newbrain-gebruikersgroep te utrecht

kopij

newbrain on-line 9 zal verschijnen op de newbraindag op 2 mei 1987. sluitingsdatum voor de kopij is 20 maart 1987. maar stuur uw kopij zo spoedig mogelijk op: het werkt veel prettiger, als niet alles op het laatst gehaast gedaan moet worden. ons adres is: newbrain-gebruikersgroep, postbus 4494, 1009 AL amsterdam.

het is voor de redactie verreweg het makkelijkst de kopij aangeleverd te krijgen op cassette of diskette. dat bespaart de fouten bij het overtypen. vermeld bij een diskette wel het formaat. de tekst mag op alle gangbare manieren zijn opgeslagen: texy-, wordstar-, wp- en alle mogelijke ascii-bestanden zijn bruikbaar. de cassette of diskette wordt natuurlijk teruggezonden.
zeker als in een artikel een programma-listing van enige omvang voorkomt, is een cassette of diskette onontbeerlijk.
dit alles mag u natuurlijk niet beletten om kopij in te sturen. de redactie ontvangt liever kopij, waar ze wat meer werk mee heeft, dan helemaal geen kopij

Memmo fecit

New Brain on-line

TEN GELEIDE

voor u liggen de eerste resultaten van het onderzoek naar de werking van de expanded newbrain. dat probleem wordt nu systematisch aangepakt. van wim luijts hand zijn artikelen over het toevoegen van een device driver op de expanded newbrain, over het poken in het scherm en over de zogenaamde s-p-nummers. de beschrijving van de system page (expanded en unexpanded) is in vertaling opgenomen. in de volgende on-lines wordt dit onderwerp zeker vervolgd. we zien ook uit naar henk bliks ontleding van het paged memory operating system

ook voor de bezitters van een unexpanded machine wordt er weer veel behandeld: top, printzone, tokens, basicode, machinetaal en cp/m. de softwarebibliotheek is spectaculair uitgebreid: zie pagina 5 - 20. een ledenlijst is in het volgende nummer van on-line aan de beurt. de prijsvraag is verdwenen door gebrek aan belangstelling

vele artikelen zijn door wim luijt geschreven. daar zijn we hem zeer dankbaar voor; maar we hebben toch het gevoel, dat de leden van de gebruikersgroep hem daar niet alleen voor mogen laten opdraaien. voor on-line 9 hebt u ruim de tijd om ook iets te schrijven: tot 20 maart 1987. en als er al snel veel kopij binnenkomt, kan er tussentijds een extra nummer van on-line verschijnen

de redactie

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

voorzitter	m s vreedenburg	(02159) 11068
secretaris	menno stevens	(020) 924137
penningmeester	jan de vries	(030) 511243 /19.00 - 20.00 uur/

postrekening 2505800 tnv hcc newbrain-gebruikersgroep, utrecht

de newbrain-gebruikersgroep is een onderdeel van de

HCC HOBBY COMPUTER CLUB

inschrijvingsnummer kvk leiden: V445230

landelijke newbraindagen

coördinatie: m s vreedenburg, (02159) 11068

in 1987 zijn de newbraindagen op de zaterdagen 2 mei en 3 oktober in technische school 'de bron' utrecht. de leden van de gebruikersgroep ontvangen tevoren een uitnodiging met het programma

newbrain on-line

redactie: menno stevens, (020) 924137

sluitingdatum voor de kopij voor on-line 9 is 20 maart 1987

zie de binnenkant van het omslag voor het inzenden van kopij

reeds verschenen nummers zijn te bestellen door overmaking van ƒ 10,- per nummer op girorekening 2505800 ten name van hcc newbrain-gebruikersgroep te utrecht

coördinatoren van de werkgroepen

basicode: pieter van dijk, (01751) 12367

hardware: rob maris, (055) 424485

onderwijs: maarten floor, (02963) 4374

proton: jan wubben, (010) 4557698

cp/m-vraagbaak

henk blik, (02159) 42689

softwarebibliotheken

bibliothecaris: wim luijt, (01100) 28197 /wo-za, ook overdag/

onderwijsbibliothecaris: maarten floor, (02963) 4374

uitlevering op cassette: guus von morgen, (02158) 3381

uitlevering op diskette: jos hermans, (013) 552530 /ma-vr, 19.00 - 21.00 h/

zie de bladzijden 6 - 20 voor de nieuwe delen

het bestellen van software

de bestelprocedure is dezelfde voor de drie bibliotheken: neem het bestelformulier achterin deze on-line, en vul daarop in, welke software u wenst te ontvangen (niet alleen de opgesomde delen zijn leverbaar, maar ook alle volumes van de cp/m-bibliotheek van de hcc cp/m-gebruikersgroep!). kruis het gewenste formaat aan: cassette of diskette van 200k, 400k ss, 400k ds of 800k voor de newbrain, of vul het proton-formaat in. stuur de bon samen met een girobetaalkaart of betaal- of eurocheque (vergeet niet het nummer van giro- of betaalpas in te vullen) op naar nevenstaand adres.

het verschuldigde bedrag kan ook worden voldaan door overschrijving op girorekening 2505800 ten name van hcc newbrain-gebruikersgroep te utrecht. ook in dat geval het bestelformulier opsturen. zonder bestelformulier gaat het vast mis

bemiddeling verkoop tweedehands apparatuur

jan de vries, (030) 511243 /19.00 - 20.00 h/

contactpersonen regionale bijeenkomsten

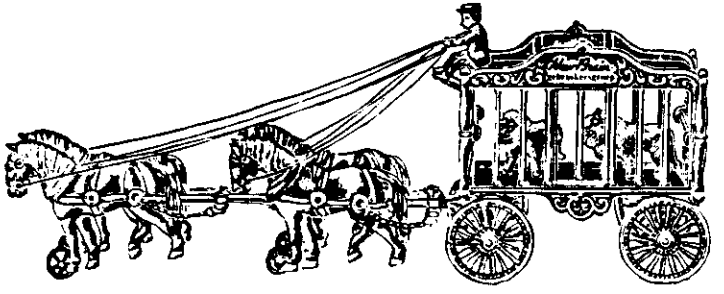
groningen: evert drijver, (050) 778415

deventer: r e van gelden, (05720) 3016

utrecht: m s vreedenburg, (02159) 11068

zeeland: wim luijt, (01100) 28197 /wo-za, ook overdag/

limburg: jos hermans, (013) 552530 /ma-vr, 19.00 - 21.00 h/



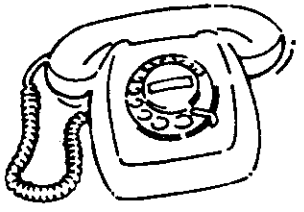
actief

De landelijke bijeenkomsten van de NewBrain-gebruikersgroep zullen in 1987 worden gehouden op 2 mei en 3 oktober.
Verder staan er in de komende maanden verschillende activiteiten op het programma waar de NewBrain-gebruikersgroep ook aanwezig zal zijn:

HCC-dagen: vrijdag 21 en zaterdag 22 november 1986
cp/m-dagen: 7 februari, 9 mei en 24 oktober 1987

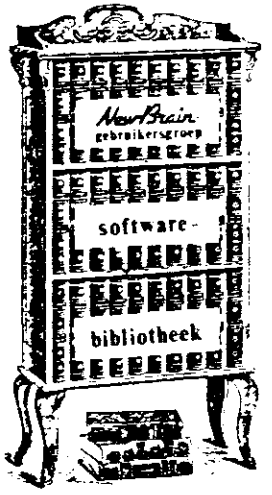
De gebruikersgroep toont op deze dagen zowel NewBrain als Proton. Zoals u kunt zien, is dit een respectabel aantal keren, dat de gebruikersgroep een activiteit organiseert, of aan een presentatie deelneemt. Een probleem waar we in de gebruikersgroep steeds weer mee zitten, is een gebrek aan mensen die meewerken. Meewerken om landelijke dagen tot een succes te maken, of bij andere gelegenheden helpen om de gebruikersgroep als groep te laten optreden. Het is niet mogelijk om alle taken door bestuur met enkele kaderleden te laten uitvoeren. Dit werd een dezer dagen pijnlijk duidelijk bij de blessure die Guus von Morgen opliep aan zijn pols. Hiermee ligt de aanmaak en uitlevering van software op cassette plotseling een tijd stil. Guus beterschap! Ook is er nog een heleboel cp/m-software voor de NewBrain die bekeken moet worden. We zoeken mensen, die daar verstand van hebben: er is een duidelijke ondersteuning nodig voor het cp/m-gebeuren op de NewBrain. Het beheer en uitleveren van software is dan wel leuk werk, maar kost toch tijd. En deze tijd zal toch weer binnen de gebruikersgroep gevonden moeten worden.

Daarom doe ik hierbij een oproep aan al onze leden om, als het maar even kan, ondersteuning te geven aan de gebruikersgroep. Het is echt niet nodig om een volledige taak op de schouders te nemen. Het is naar mijn mening zeer wenselijk om over ondersteuning te kunnen beschikken, als dit nodig is. Dan kan de NewBrain-gebruikersgroep blijven voortgaan met de verwezenlijking van haar doelstelling: de onderlinge ondersteuning van en door de leden. Een briefje of een telefoontje voor aanmelding of wensen en ideeën is altijd welkom.



M. S. Vreedenburg, voorzitter

aanwinsten



het was deze keer helemaal niet moeilijk om een kop te verzinnen voor de rubriek van de softwarebibliotheken: het is niet mis, wat nu erbij gekomen is! alle bibliotheken hebben er nieuwe delen bij (ook de bibliotheek van de hcc cp/m-gebruikersgroep breidt gestaag uit, en al die delen zijn bij ons verkrijgbaar ...), maar echt spectaculair is de uitbreiding van de softwarebibliotheek met maar liefst elf delen met software van de engelse gebruikersgroep nbug. lees daarover ook de inleiding van onze softwarebibliothecaris op pagina 8 en 9. het water loopt je in de mond, als je de catalogus bekijkt

achterin deze on-line vindt u een bestelbiljet, hetzelfde formulier voor alle software, die bij de gebruikersgroep verkrijgbaar is. voor u het gemakkelijkst; en het sluit ook nauw aan bij de organisatie van de uitlevering. op pagina 3 staat beschreven, hoe u kunt bestellen. hier is een overzicht van de prijzen voor de diverse bestelnummers en formaten

PRIJSLIJST			
prijs per deel	cassette	disk 200k	disk 400k ss/ds, 800 k
softwarebibliotheek	ƒ 15,00	ƒ 15,00	ƒ 15,00
onderwijsbibliotheek	ƒ 25,00	ƒ 25,00	ƒ 25,00
cp/m-bibliotheek *)		ƒ 25,00	ƒ 15,00
*) per 10 delen in één bestelling 1 deel gratis			

de onderwijsbibliotheek wordt beheerd door maarten floor, de andere door wim luijt. stuur uw opmerkingen, maar liever nog uw eigen programma's die u ter beschikking stelt, aan hen op naar:

newbrain-gebruikersgroep, posbus 4494, 1009 AL amsterdam

catalogus

van de nieuwe software per 18 oktober 1986

bestelnummer: ASM-1

om het programmeren in z80-code te stimuleren een deel met z80 source files en enkele hulpprogramma's. voorlopig alleen de source files van programma's uit de softwarebibliotheek. de source files staan in het formaat van device 13 en bevatten in plaats van spaties een CHR$ (9) om ze kort te houden

CONT-MON.ASM, TERMINAL.ASM, TERMINA1.ASM, LDCODE.ASM, MCLOADER.ASM, FP-DCONV.ASM, GREXDUMP.ASM, KLOK.ASM, KLOKEXP0.ASM, HIRESDMP.ASM

ASM-EDIT.BAS eenvoudige tekstverwerker, speciaal voor source files

DISASSEM.BAS, -.DAT en -.HAN een z80-disassembler, uitvoerige documentatie ontbreekt echter, engelstalig

LADEN.BAS POKEt onder basic een -.HEX file; v raben

bestelnummer: DIVERS-3

van open #stream, een engelse gebruikersgroep, ontvingen we enkele basic-programma's over lineaire programmering, een methode die gebruikt wordt in de bedrijfskunde. enkele critici, die we de programma's ter beoordeling toezonden, waren zeer enthousiast. hoewel de programma's niet voldoen aan een aantal eisen, die aan programma's voor opname in de software-bibliotheek gesteld worden, zijn ze voorlopig opgenomen in de softwarebi-bliotheek. voor zover we weten bestaan er geen andere programma's op dit gebied voor de newbrain en is de gebruikte techniek zeer moeilijk in basic te programmeren. voor de volledigheid en de juistheid van de programma's kan niet worden ingestaan. evenmin hebben we volledige documentatie, u zult dus zelf het nodige moeten uitzoeken. alles wat we er over hebben, zit in het pakket, bestaande uit: CPA.BAS, CPA-PERT.BAS, CPAPERT.BAS, LINPROG.BAS, LINTRANS.BAS en LPASSIGN.BAS.

verder is dit pakket aangevuld met THTSIM.BAS, een simulatieprogramma. voor verdere bijzonderheden hierover wordt verwezen naar DATABUS van november 1983

CROSSING.BAS, -.DAT en -.HAN eindelijk beschikbaar voor de software-bibliotheek. voor degenen die geen engels kennen, volgen de instructies: eerste vraag: grieks [E] of engels [A]. nederlands was niet toe te voegen. spelregels: je krijgt: 1 punt voor iedere sprong, 20 punten voor het oppakken van een schaap, 25 punten voor het overdragen van een schaap en een extra leven voor iedere 1000 punten. je kunt nooit meer dan 4 levens krijgen. druk iedere keer op toets [A], als je het spel wilt starten; stop en herstart het spel met de toets [S]. naar links met toets [B], naar rechts met [N]

TOOLKIT.BAS, -.DAT en -HAN voegt de volgende basic-opdrachten toe: RENUM hernummert het programma (oude en nieuwe regelnummer van de beginregel en de stapgroote kunnen worden opgegeven; standaard-parameters zijn 1, 10, 10), VARS list de variabelen met hun waarde, PVARS print variabelen op printer, STRIP verwijdert alle REMs en spaties, SEARCH zoekt een variabele of string, TRON trace aan (de regel die uitgevoerd wordt, wordt ook afgebeeld), TROFF trace uit, BRON break aan (GR/VIDEOTEXT stopt ieder programma), BROFF break uit, HELP toolkit-instructies. toetsenbord is stream 5, printer stream 8. TOOLKIT.BAS alleen voor een 32k newbrain

HIRESDMP.BAS screendump-routine voor 32k newbrain; source code in deel ASM-1, zodat je zelf aanpassingen kunt maken; v raben (32k)

SCHUIFRM.BAS spel; goed voorbeeld van device 11 met z80-code; frans van der velde (32k)

BRIDGE.BAS hulpmiddel bij het organiseren van bridgetoernooien, fraai voorbeeld van wat met een eenvoudig programma mogelijk is; wilfried de waele

MCLOADER.BAS snellader voor z80-routines (zie pag. 41); wim luijt

POKESCRN.BAS slaat een grafisch scherm op in een string; zo zet je het gemakkelijk op cassette of disk en kun je het naderhand weer in een flits op het scherm zetten; het geeft ongekende mogelijkheden met left$, right$ en mid$

THTSIM.BAS simulatieprogramma uit databus; frans kalwij

KLOKEXP0.BAS de klok uit on-line 5 nu ook op de 96k (zie pag. 54 sqq.)

PAGEZERO.BAS bekijkt de zero page van de 96k onder basic (96k); cf de lijst op pag. 65 - 72

GBS-TOMB.BAS verrassing in de ROM van de 96k (96k)

EDITOR.OVL aanpassing voor diskdrive van EDITOR.BAS (van bestelnummer DIVERS-1); v raben

NBUG-SOFTWARE

De NewBrain-gebruikersgroep heeft software van NBUG, een van de Engelse gebruikersgroepen, in de softwarebibliotheek opgenomen. Het assortiment van software is hierdoor enorm toegenomen.

De verkoop van deze software zal op vrijwel gelijke wijze als de cp/m-software van de cp/m-gebruikersgroep plaats vinden. Hierbij zullen de volgende regels gelden:

- de delen zijn zowel op schijf als cassette verkrijgbaar; programma's en bestanden waarvoor een NewBrain met diskdrives of cp/m noodzakelijk is, worden niet op cassette geleverd;
- programma's, die geen public domain zijn en uitsluitend aan leden van NBUG mogen worden geleverd, worden door ons niet geleverd (zie onder);

- verder gelden dezelfde regels als voor de andere software.

NBUG heeft tot nu toe 40 delen van 150k tot 200k software uitgegeven. De software is in vier groepen in te delen:

- basic, pascal, cp/m- en Z80-software voor de NewBrain
- algemene cp/m-software
- ROM source code listings
- de inhoud van het tijdschrift NBUG

Voorlopig is alleen de software uit de eerste groep (12 delen) te leveren. Gerald McMullon, ex-Grundy-ingenieur en de voortrekker van NBUG, heeft tegenwoordig een eigen bdrijf en maakt onder andere hard- en software voor NewBrain en Amstrad. Hij heeft een aantal programma's beschikbaar gesteld voor de NBUG-softwarebibliotheek onder de voorwaarde, dat deze alleen aan NBUG-leden wordt geleverd. Helaas kan onze softwarebibliotheek deze programma's dus NIET leveren. NBUG-leden onder de Nederlandse gebruikers kunnen deze delen alleen rechtstreeks van NBUG betrekken. Voor de volledigheid zijn deze programma's wel in de catalogus vermeld (met de aanduiding 'vervallen').

De programma's worden geleverd zoals ze door de gebruikersgroep zijn ontvangen. Met andere woorden ze zijn niet vertaald; er is niet gecontroleerd, of ze werken, noch op welke systemen ze werken. De informatie in de catalogus is ontleend aan de catalogus van NBUG. Alleen is oppervlakkig onderzocht of de software geschikt is voor een NewBrain met cassetterecorder. De gebruikersgroep aanvaardt daarom geen aansprakelijkheid voor de kwaliteit van deze programma's. Enkele delen bevatten programma's, die ook in onze softwarebibliotheek zijn opgenomen. Dit was niet te vermijden, omdat er voortdurend een uitwisseling plaats vindt tussen de verschillende gebruikersgroepen. In een geval (deel 19) zijn deze programma's verwijderd. Deel 19 is gecombineerd met het uitgedunde deel 36 om toch voldoende waar voor je geld te krijgen.

Sommige programma's behoeven nadere informatie uit het tijdschrift NBUG. Deze informatie zal op een fotokopie worden toegevoegd. Andere programma's vereisen het gebruik van hardwaretoevoegingen (meestal het befaamde PIO-board van GFG Microsystems). Inlichtingen hierover zijn telefonisch bij mij te verkrijgen (01100-28197).

Op persoonlijke titel wil ik de volgende programma's onder de aandacht brengen, of omdat ik ze zelf gebruik, of omdat ze me zeer nuttig lijken.

- de random acces files voor pascal (bestelnummer NBUG-17)
- EX.COM (bestelnummer NBUG-17); wie combineert dit met FEXIT?
- SHOW.COM (bestelnumer NBUG-20)
- de tekstverwerkers en tekenprogramma's in NBUG-20 en NBUG-22
- de user defined functions (bestelnummer NBUG-20)
- de forth-achtige interpreter (bestelnummer NBUG-26)
- de grote letters van Bigmess (bestelnummer NBUG-26)
- de aangepaste Small C-compiler (bestelnummer NBUG-33)
- de hulpprogramma's voor pascal (bestelnummer NBUG-34)
- de vluchtsimulator (bestelnummer NBUG-36)
- voor kleine (en grote?) beleggers: bestelnummer NBUG-37
- het morseproject (bestelnummer NBUG-40)
- het RTTY-project (bestelnummer NBUG-40)
- het manipuleren van het grafische scherm met cassette of schijf
- de MUIS (of 'mouse') voor de 96k NewBrain met device 22 en 23 (beide op bestelnummer NBUG-41)

Als er voldoende belangstelling is voor deze software, zullen er meer delen volgen.

Wim Luijt, softwarebibliothecaris

voor de serie NBUG geldt:

* = ook verkrijgbaar op cassette

bestelnummer: NBUG-17

diversen, basic, pascal en cp/m

RANWRITE.PAS (1k) random acces files onder hisoft pascal; mike smith. alleen voor diskdrive. met LOCKCLR.PAS, RANREAD.PAS, RANDOMAC.LIB, RANWRITE.COM (5k), RANREAD.COM (5k), BLOCKCLR.COM (3k) en databestand TESTDATA.DAT (10k)

PRINIT.COM (1k), -.PRN en -.ZSM start printer shinwa cp80 met 4800 baud; mike smith

SETUP.ASM (3k), -.HEX, -.COM en -.PRN download newbrain-tekens naar seikosha gp250-x; david mullin

COPY.ASM (10k), -.COM, -.PRN, -.HEX en -.DOC afzonderlijke bestanden kopiëren met één drive; david mullin

EX.COM (1k) cp/m-exit voor 32k naar basic met 80-koloms scherm

*TOWERS.BAS (2k) grafische demo van nick bane

*WARI.BAS (6k) bordspel van brian lockey

*DATAEDIT.BAS (5k) en *PROMPT.BAS (2k) hulpmiddelen voor gegevensinvoer; d j deane

*COMBAT.BAS (1k) spel van s middleton

*LANDAR.BAS (2k) spel van s middleton

*ZOG.BAS (7k) doolhofspel van dave forth

*DEMO21.BAS (14k) grundy-demonstratie; gfg / n c bane

*SHADOW.BAS (1k) grafische demo van j d bryant

*SHEARFOR.BAS (3k) grafieken voor afschuifkrachten; j d bryant (onderwijs)

*ASTROS.BAS (3k) spel van k gray

*YEARLY.BAS (9k) boekhoudprogramma van hugh thomson

*DUMP.BAS (2k) grafische dump 32k naar facit 4510; geir kjelland

COMMS.BAS (3k) communicatie met mainframe (voor 96k); g cook

*PETROL.BAS (3k) benzinekosten; k gray

*MATRIX.BAS (1k) oplossing van lineaire vergelijkingen; geir kjelland

*DIFFEREN.BAS (1k) vergelijkt twee bestanden

*QUADLINK.BAS (3k) vier-op-een-rij; k gray (32k)

*CONNECT.BAS (3k) vier-op-een-rij; p worland

CIRCLE.4TH (1k) cirkel-opdracht in mpe-forth; ian entecott

*POSITION.BAS (1k) instelroutine voor tape; j ramsay

BYTECOUN.BAS en MINI.BAS zijn vervallen

bestelnummer: NBUG-18

basic en diversen

*CSUM.BAS (1k) checksum voor ROMs; j kelly

*REGISTER.BAS (5k) toont z80-registers op line display (model AD); met *documentatie; m j cox

*FUZZY.BAS (8k) cartoon (grafisch verhaal) van dave forth

CPA.BAS (6k) critical path analysis, met *documentatie; pete hartland

CPAPERT.BAS (6k) CPA en PERT; pete hartland

*WORDSQUA.BAS 'wordsquare', een goed basic-spel van k otter

*BRICK.BAS (2k) het spel 'brickout' van simon murphy, en

*BRICKDAN.BAS (3k) william hempel's verbetering ervan

*SOUND.BAS (2k) geluid via RS-232-uitgang

*FUN.BAS draaiende, grafische demo met 5 *databestanden (voor 96k)

*CARD.BAS (3k) kaartenbak (werkt bijna), met bijbehorend: *INDEX (1k)

*CRIBBAGE.BAS (14k) uitstekend spel van jim moon in z80, met z80-code in apart bestand: *CRIBBAGE.BIN (2k)

*ROMAN.BAS (1k) romeinse cijfers, frank ellis / gfg

nbug volume 19 (bestelnummer: NBUG-36)

pascal programma's (turbo en compas) met interfacing naar de stream-i/o van de 96k onder cp/m. gemaakt door gordon crosse en rob van albada

BITS.PAS (1k) test bits van 1-byte integers

SHOW.COM (10k; compas) vertoont directory, $R/O, $SYS, devices, tekensets, diskparameters, bit map van de drive; zet topbits in bestandsnaam; met printmogelijkheid (voor epson). met overlay SHOW.000 (4k)

WRITEHEX.PAS (1k) hexadecimale uitvoer van source

ABORTSUB.COM (4k) en -.SUB (1k) breekt een submit-handeling af

PAGEZERO.BAS (8k) geheugenplaatsen op de zero page

GBS-TOMB.BAS (1k) een boodschap in de ROM

BAUD12.COM (8k), -.HEX (19k), -.OBJ en -. stelt printer-baudrate op 1200

HIGRAPH.BIN (1k), -.LST, -.BAS (1k) en -.PRN high resolution graphics?

DISKEDIT.PAS (12k), GDEMO.COM (34k), GDEMO2.COM (12k), GDEMO2.PAS (3k), GRAPH.INC (7k), PIO.INC (8k), SYS.INC (3k) en SYSPAGE.PAS (10k) staan ook op bestelnummer NBGG-4. zie daar

CALC.PAS is vervallen

LET OP: nbug-19 staat met nbug-36 op één schijf: bestelnummer NBUG-36

bestelnummer: NBUG-20

diverse programma's in basic

SCAN.COM (1k) cp/m-opdracht TYPE met erase-optie
MENU.BAS (2k) kies basic-programma's van schijf d.m.v. MERGE; t a morris

cassette A

*3DDRAW.BAS (8k) symmetrische driedimensionale voorwerpen tekenen en roteren. met *3DDATA.DAT (5k), *3DINTRO.BAS (1k) en toelichtng *3DPLOT.DOC (4k); simon murphy (onderwijs)
*BIGPRINT.BAS (1k) printen van grote tekens (32k)
*EASYDRAW.BAS (6k) schermtekenprogramma (onvolledig)
*EASYPRNT.BAS (3k) tekstverwerker, met bijbehorend *A4PAGE.EPT (1k), *INSTRUCT.EPT (4k) en *MEMBER.EPT (1k)
*EASY.BAS (4k) tekstverwerker
*EMAX.BAS (6k) tekstverwerker; max hadley
*WORDPROC.BAS (6k) tekstverwerker
*POGO.BAS (10k) basic-interrupter voor een versie van logo turtle graphics; s murphy (onderwijs). *ESTATE, *FLOWER, *TEST, *PATTERN en *POLSPI zijn POGO-programma's (van elk 1k)
*2T.BAS (2k) tweetaktmotor, keith hinton (onderwijs)

cassette B

*SCRNMAP.BAS (1k) indeling schermgeheugen (32k)
*VARDUMP.BAS (2k) dump van basic-variabelen
*ZEROPGMP.BAS (1k) indeling van zero page (32k)
*USERFNSP.TXT (3k) basic DEF FN; *USERFUNC.DOC (16k) is de toelichting; t a morris
*BASMEMAP.BAS (2k) indeling van basic-geheugen (32k)
*DISASSEM.BAS (8k) Z80-disassembler in basic; f goetze
*ERRMSG.BAS (8k) foutmeldingen, basic-subroutine van f goetze
*HEXDUMP7.BAS (4k) dumpt geheugeninhoud in hex en ascii; t a morris
*PG85-B.BAS (17k) uitstekend voetbalpoolprogramma van t dawson. met toelichting *POOLS.DOC (4k) en databestanden *18/8/84 en *TEAMS-B
*FILEPRT1.BAS (7k) kaartsysteem?
*CHARGE!.BAS (6k) lading op een elektron, spel van f goetze
*GROCERY.BAS (4k) voorraadbeheer in de huishouding; clive martyn
*CHECKERS.BAS (5k) een echt goed damspel ... tot de bug; f goetze

LET OP: de programma's zijn verdeeld over twee cassettes: NBUG-20A en NBUG-20B. bij bestellen duidelijk A of B vermelden, anders wordt automatisch NBUG-20A gezonden.
alle programma's staan op één schijf: bestelnummer NBUG-20

voor de serie NBUG geldt: * = ook verkrijgbaar op cassette

bestelnummer: NBUG-26

dit deel bevat een aantal van de de beste programma's die er voor de newbrain te vinden zijn. K-TEXT en FORTH (beide van paul kriwaczek, een brein op de beeb) zijn programma's, die zeer efficient het disksysteem onder basic gebruiken. de snelle fourier-transformatie is ook de moeite waard, als je die kunt gebruiken.

K-TEXT.BAS (1k) tekstverwerker in machinecode (32k + diskdrive), met z80-data K-TEXT.M (4k) en instructies DOCUMENT (13k)
FORTH (1k) forth-interpreter (32k + diskdrive). met de bestanden DBF1, F1, F2, F3, F4, F5, F6 en de forth-dictionary DICTION (7k)
*PRNFORM.BAS (1k) print newbas-z80-sourcecodebestanden
*HFFT (4k) snelle fourier-transformatie van andrew beckett. met *FFTDEMO, *FFTR en *FFTRL
TRACTEXC.C/M (4k) spaanse tekstverwerker in z80 (32k) van j bas diez. met z80-codefile DATOS.C/M (6k) en onderdeel MAIL.BAS (4k)
NLOGO (8k) spaans tekenprogramma in z80 (32k) van jordi ros. met datafile IOCASDAT
LLISTA.BAS (3k) list diverse soorten bestanden naar de printer; manuel peiro mir
*BIGCHR.PRN (4k) tekent grote letters. newbas-bestand (32k), met *BIGCHR.OBJ (1k)
*SONG.PRN (5k) geluidsgolven; marc rouselle. newbas-bestand, met *SONG.BIN (1k)
*SOUNDB.BAS (4k) demonstratie BIGCHR en SONG
*TRANEDIT.BAS (12k) transmissie-editor: gebruikt de newbrain als printerbuffer voor een ql (met editmogelijkheden); j r downie
*MAP.NBG (8k) hi-reskaart van groot-brittannië; dave rickwood
*ASNASC.COR (4k) verbetering van ASN- en ACS-fouten bij graphics in de 1.91-ROM; van bert lauridsen
*COMMS.CHA (3k) softwarematig wisselen van comms- en printer-uitgang; van bert lauridsen
*BIGMESS.BAS (2k) beeldt grote tekens af op het scherm, gebruikt tekenset 4 en geeft 6 regels van 20 tekens. met datafile *BIGMESS.DAT (5k)
BTYPE.COM (2k) cf bestelnummer NBGG-5
WIT.COM (1k) schakelt met ↑ N / O inverse video aan / uit (zwart op wit); anton van de repe
ZWART.COM (1k) als WIT, maar wit op zwart scherm
DPB.COM (6k) na configure, xconfig, etc nu dpb; heb je ooit je configuratie van dat moment willen weten? dit is het antwoord
PRINTLEN en CAPS zijn vervallen

bestelnummer: NBUG-33

david mullin's uitstekende c-compiler met 8080/z80-assembler
door gordon crosse besproken in NBUG 13

deze implementatie van de taal c is ontworpen voor gebruik op cp/m-systemen met beperkte geheugenruimte (32k). met kleine wijzigingen in de functie- bibliotheek is de taal ook geschikt voor 48k-machines. op de 32k newbrain moeten programma, data en stack passen in circa 20k. er is echter nog 32k virtueel geheugen geschikbaar voor de programmeur; dit wordt bijgehouden op schijf en wordt naar behoefte verdeeld over het fysieke geheugen (de gebruiker heeft daar zicht op). de compiler is gebaseerd op die in 'the small-c handbook' van j e hendrix. die was weer gebaseerd op een small c-compiler in de bibliotheek van de cp/m users group. hendrix' compiler is een uitbreiding van die van de cp/m users group, en de newbrain-implementatie bevat bijna alle toevoegingen. voor serieuze gebruikers is hendrix' boek van onschatbare waarde. omdat alle commerciële relocerende assemblers en linkers toch gauw zo'n 48k geheugen nodig hebben om te kunnen werken, wordt de compiler geleverd met een 8080-assembler en -linker (beide door de auteur in c geschreven)
EX.COM, STATC.DAT, CC.COM, AS.COM, LINK.COM, CLIB.OBJ, TOP.OBJ, BOOT.OBJ, STDIO.H, TEST, INSTALL, LETTER, CGUIDE, AGUIDE, LGUIDE, BUGS, COMPAT, CERROR, AERROR, GERROR, CPROG, APROG, LIBRARY, VIRTUAL, HELPBOOT.ASM, HELP.COM, TEST.C, FILE.C, HELP.C, HELP.LNK, READ.ME

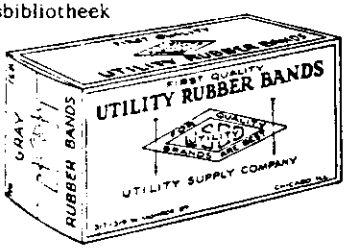
bestelnummer: NBUG-34

gordon crosse's ZCALL-uitbreidingen voor hisoft pascal 80 en turbo pascal onder het paged memory operating system. de source code is goed gedocumenteerd, en geeft inzicht in de werkwijze van het paged system onder cp/m

SYSCON.PAS (1k) systeemconstanten voor paging
SYSFUN.PAS (1k) routines voor adresberekening, die ontbreken in hisoft
SYSGRAPH.PAS (6k) turtle graphics routines
SYSLOAD.PAS (1k) load external file naar RAM
SYSPAGE.COM (6k) en -.PAS (5k) leest data vanaf de zero page, en schermfuncties
SYSPIOS.PAS (4k) zero page in- / uitvoerroutines
SYSTEM.DOC (6k) documentatie voor hisoft-programma's
SYSTEST.COM (14k) en -.PAS test open/close en graphics
SYSTYP.PAS (1k) system TYPE file
SYSVAR.PAS (1k) system VAR file
ZCALLCDE.EXT (1k) en -.ASM load external file naar RAM

FCOPY16.COM (4k) kopieert bestanden met 1 diskdrive op 32k, 16k buffer
FCOPY48.COM (4k) idem dito op 96k, 48k buffer; geschreven in C/80
IV.COM (1k) inverse-video onder cp/m, laat het operating system met rust; ↑O geeft 'low video', ↑N geeft 'high video' - ja het werkt op die manier. werkt met dbase en turbo; maar de cursorbesturing met '22, x, y' kan het niet aan, waardoor regels oplichten afhankelijk van de schermpositie!
SYSLIB.INC (11k) turbo pascal-systeembibliotheek
FPTEST.COM (13k) en -.PAS (1k) testprogramma voor floating-point-berekeningen
GRAPHLIB.INC (8k) met GTEST.COM (14k) en GTEST.PAS (3k) graphics testprogramma
TURTLE.INC (1k) turbo pascal-graphicsbibliotheek
WRITEHEX.INC (1k) ?
ZTEST.COM (11k) en -.PAS (1k)
zcall-tests

voor de serie NBUG geldt:
* = ook verkrijgbaar op cassette



bestelnummer: NBUG-35

gerry cooks programma's en routines om PIOS en ZCALL's van de newbrain te interfacen met FORTRAN (inclusief microsoft REL-files)

CUBE (1k) demo van grachics batch processing, gebruikt TESTD1 (1k)
DEMO1.COM (16k), -.FOR (2k), -.LNK, -.REL eenvoudige demo van batch processing van newbrain-plotopdrachten. die worden omwille van de snelheid ingelezen als getallen, maar zijn zo gebruikersonvriendelijk. het programma kan eenvoudig worden gewijzigd om opdrachten in te lezen als een REAL-variabele (onder FORTRAN alfa-numerieke formaatspec. A4) en te vergelijken met gegeven opdrachtnamen. hiervoor moeten de namen worden ingekort of aangevuld tot precies 4 tekens, bijv. 'CIRC', 'PEN.'. de ingevoerde data kunnen dan worden vergeleken met een geinitialiseerd REAL ARRAY, opgezet in een Fortran-dataregel. nadere opdrachten, bijv. 'GRID' om grafiekpapier te maken, kunnen worden verzorgd door fortran-subroutines, gebruikmakend van bestaande newbrain-opdrachten
DEMOGRPH.COM (20k), -.FOR (4k), -.LNK, -.MAP en -.REL demo, gebruikt CUBE (invoeren met hoofdletters). antwoord '0' op "location of fortran source file 'CUBE'"; geschreven om twee alternatieve schermen te maken, maar ik heb maar voor een ruimte gevonden met parameterstring 'n120'. als ik tijd heb, hoop ik het BIOS te verplaatsen om meer ruimte te vinden

*DEV33DEM.BAS (1k) demo device 33, 96k
overige bestanden op de schijf: GDEV1.MAC (5k), GDEV1.REL (1k), GINIT1.MAC (4k), GINIT1.REL (1k), GRAPH.ASM (7k), GRAPH.COM (10k), PROG.DOC (3k), SETUP.FOR (1k), SETUP.REL (1k)

LET OP: er is geen cassette NBUG-35. DEV33DEM.BAS staat op de cassette met bestelnummer NBUG-37

bestelnummer: NBUG-36

diverse spelletjes

*GUESS.BAS (2k) raad een getal; hans warmdahl
*HANOI.BAS (9k) hanoi-spel van d a mitchell
*IFRFLIGH.BAS (13k) vluchtsimulator; lijkt een goed spel van jean-louis pergod
*KNOW.BAS (4k) spel met meerkeuzevragen
*LADDERS.BAS (9k) slangen en ladders, uitstekend bordspel van ken otter
*LIFE.BAS (9k) levenspel; z80, r seally
*OBSTACLE.BAS (3k) eenvoudig doolhofspel van jean-louis pergod
*SQUATTER.BAS met *SQUATTER.DAT schapen fokken in aussieland; een zeer waardevol spel, lijkt op tycoon, maar er gebeurt nog meer; alle instructies staan in het spel; jean-louis pergod
*WORM.BAS (2k) slangespel met diagonale bewegingen; n c bane
WORMS.COM (12k) compas-versie van het voorgaande; zéér snel
*ZAP.BAS (4k) dood het buitenaardse wezen; data-depotet, w hempel
CROSSING.BAS alleen verkrijgbaar op bestelnummer ONLINE-8
CHASE, DANDEMO.JOY, DEMO.JOY, GAMES.JOY, MAINGAME.JOY, QUEST.BAS, SNAKE, SPACE en ZOMBIE zijn vervallen

LET OP: nbug-36 staat met nbug-19 op één schijf, bestelnummer NBUG-36

bestelnummer: NBUG-37

basic-diversen

BFRENCH.BAS, BGERMAN.BAS, IFRENCH.BAS, IGERMAN.BAS, AFRENCH.BAS, AGERMAN.BAS franse en duitse woordenboeken voor beginners, gevorderden en verder gevorderden (telkens 12 à 13k). de programma's bestaan uit twee reeksen data en een klein programma, dat de vertaling controleert; bedoeld om de kennis op te frissen van degenen die beide talen kennen; carrol fuller
*BENCHMAR.BAS (3k) pcw bench tests
*BIGMESS7.BAS (6k) print grote tekens op 30-regelscherm, met een

datafile *BIGMESS.DAT (5k), en *BIGMAKE.BAS (5k) maakt de datafile; t a morris
*CHESSBOA.BAS (4k) het schaakprobleem van de vier dames; de gebruiker stelt de afmetingen van het bord in; martin cox
*MONTE.BAS (2k) monte carlo; r c worsdell
*NEWTOWER.BAS (5k) tower, nieuwe versie van w hempel
*RAINFALL.BAS (5k) regenvalgrafiek; d a mitchell
*PORTFOLI.BAS (9k) en *SHAREHOL.BAS (11k) 'bijna alle programma's voor beleggers, die ik elders heb gezien, zijn niet te begrijpen voor de leek en de gewone kleine belegger; daarom heb ik ervoor gezorgd, dat alle informatie op eenvoudig te begrijpen wijze is opgenomen. in de loop van het programma wordt uitleg gegeven, maar er is ook telkens aan het begin een korte beschrijving, die gemakkelijk verwijderd kan worden, als ze niet meer nodig is.' e l rowden (met kleine wijzigingen ook geschikt voor cassette)
TRANEDIT.BAS (12k) transmissie-editor; j r downie
PROFILE.BAS (3k) menugestuurd programma dat een leerlingprofiel geeft; j h bronsem, longbenton community high school. met de subprogramma's OBSSKILL.PRF, RECSKILL.PRF, MEASKILL.PRF, PROSKILL.PRF, MANSKILL.PRF en FOLLINST.PRF
CALENDAR.BAS, DAY.BAS, PILOT.BAS en PILOT.PLT zijn vervallen

bestelnummer: NBUG-40

diverse programma's in basic; enkele vereisen extra hardware

*ATV.HAM (14k), *MAP.NBG (8k) en *MAP.BAS (3k) hi-res kaarten van de britse eilanden; dave rickwood
*CHARSET.BAS (3k) generator voor alternatieve tekens voor gebruik op het grafische scherm; pat mcgrane
*BLOWER1.PAS (8k) programma om eproms te blazen met de eprom blower en de pio-kaart van gfg; peter kontopoulos
*HARDWARE.MOR (10k), *SOFT.MOR (6k) en *MORSE.SUB (5k) morse-geluid; h jaremko
*RS232.BAS (4k) van chris korycinski, maar wat het doet?
RTTY.ASM (25k), -.COM (2k), -.DOC (5k) en -.PRN source en programma's voor rtty met newbrain-cp/m; a meynell. NEWS (5k) is een voorbeeld van rtty
*RTTYF21.BAS (1k) en *TY21DU.OBJ (8k) rtty-programma's van jorgen kjldsen, voor de 32k; details bij het ter perse gaan van deze on-line nog niet bekend

bestelnummer: NBUG-41

een serie uitstekende programma's rond het grafische scherm van de newbrain, zowel in basic als z80, van marc rouselle

cassette A

*DOWNLOAD.BAS (1k) laadt een beeld van het default backup device (tape of disk) en zet het op het grafische scherm. z80-code in PIC.BIN (1k). met *CIRCLE.PIC (6k) en de voorbeelden *DMO1.PIC (13k), *DMO2.PIC (14k), *DMO3.PIC (14k), *DMO4.PIC (14k) en *DMO5.PIC (14k); voor details zie PICBIN.SCR

cassette B

de overige voorbeelden *DMO6.PIC (14k), *DMO7.PIC (14k), *DMO8.PIC (14k), *DMO9.PIC (14k)
*GRAF1.PIC (4k)
*SAVEPIC.BAS (1k) voorbeeld om een grafisch scherm te saven op backup, voor 32k en 96k
*MICKEY.BAS (2k), *MOUSE.BAS (7k), *PICMOVER.BIN (1k) en *HIGRAPH.BIN (1k) sluit een muis aan de user port van de EIM aan; in basic, daarom erg traag, snellere z80-routines zijn in ontwikkeling, nadere berichten en updates later in nbug. LET OP: gebruikt devices 22 en 23
*WRAP.BAS (1k) en *WRAP.BIN (1k) deze utility maakt het mogelijk om het grafische scherm te laten scrollen (op, neer, links en rechts) met de cursortoetsen; iedere beweging kan worden gestopt met de stoptoets, andere richting scrollen met een cursortoets, terug naar basic met een andere toets; details in WRAP.SRC, 32k en 96k. gerald mcmullon: 'dit is een heel fascinerend programma, hoewel ik niet weet, waarvoor ik het zou kunnen gebruiken. maar het toont (opnieuw), waartoe de 'oude' newbrain in staat is!'
*GETSOURC.BAS (1k) programma om source listings van newbas-bestanden te maken (.SRC-bestanden)
*PICBIN.SRC (4k) source van PIC.BIN
*WRAPPER.SRC (10k) source van WRAP
*PATCH.SRC (3k) source van z80-routines voor page allocation in POS (96k)
*NEWGRAF.BAS plot grafieken, y = f(x) en parametrisch; een vriendelijk stuk software, slechts 4,1k; erg gemakkelijk in het gebruik; bevat instructies; filip dekeyser

LET OP: de programma's zijn verdeeld over twee cassettes: NBUG-41A en NBUG-41B. bij bestellen duidelijk A of B vermelden, anders wordt automatisch NBUG-41A gezonden.
alle programma's staan op één schijf: bestelnummer NBUG-41

cp/m-bibliotheek

bestelnummer: NBGG-6

dit deel van de softwarebibliotheek bevat hoofdzakelijk basic-programma's die alleen voor een newbrain met diskdrive geschikt zijn. verder zijn er enkele kleine cp/m-programma's aan toegevoegd

LOAD+RUN.BAS uitleg om originele bestanden te verkrijgen
WOORDEN.BAS, WOORDEN.HAN, INFO.BAS, INFO.BRI, INVOEGEN.BAS, INVOEREN.BAS, OPBOUW.BAS, OPBOUWEN.BAS, WOORDEN.IN, WOORDEN.WEG, ALFABET.BAS maak zelf een woordenboek op de newbrain; j woerthuis
INFORMAT.A/D, OM, OMX, D, REM.D, A een vervolg op OPEN #0, "filename"; bouw je eigen opdrachten; patrick vercammen
EDITOR-D.BAS diskversie van EDITOR.BAS (deel DIVERS-1); v raben
SETCON.COM, SETCON.PRN, SETCON.ZSM verander het formaat van drive B: met behulp van de CS...-files van de 800k-systeemdisk (SETCON CS...). lijkt op RESTCON, maar verandert de schijf niet en een reboot is niet nodig
WSTYPE.COM, WSTYPE.PAS type wordstar-files op het scherm
HOME.COM, HOME.ZSM wis het scherm onder cp/m; v raben
LIST.COM, LIST.ZSM stel de printer in onder cp/m; v raben (al eens PIP LST: = CON: <control codes> <NewLine> <Ctrl/Z> geprobeerd?)

onderwijsbibliotheek

bestelnummer: SCHIJF-7

151. OEFME1: programma dat de beginselen van de bewegingsleer behandelt. uitleg van de verschillende bewegingen, welke ook grafisch worden voorgesteld. het programma wordt afgesloten met oefenopgaven

152. OEFME2: vervolg op het programma OEFME1 met uitleg van de te gebruiken symbolen, hetgeen wordt getoetst met meerkeuzevragen. verder wordt de formule S = V x T behandeld met grafieken en vragen

153. OEFME3: uitleg met vragen over de eenparig versnelde en eenparig vertraagde beweging. het programma bevat ook oefenopgaven over de behandelde stof

154. MEPROGR: hiermee kan men berekeningen maken zoals: snelheid uit remspoor, remweg, stopafstand of maximale snelheid bij een bepaald zicht. dit kan men ook voor verschillende situaties in tabelvorm af laten drukken

155. BRVLAK: dit programma geeft uitleg over de krachten die op een luchtband kunnen werken bij het nemen van een bocht en tegelijkertijd remmen

156. 2SMOTOR: grafisch programma, dat de werking van de tweeslagmotor demonsteert. de snelheid van de demonstratie is regelbaar

157. V/T&S/T: programma over het snelheid-tijddiagram en het afstand-tijddiagram. het programma geeft dit ook in grafieken weer

158. 3DGRAF3: het tekenen van grafieken Z = f (x, y) in stereometrische projectie met 9 voorgeprogrammeerde functies die men zelf kan wijzigen

159. 3DPRINT1: dezelfde grafieken als in 3DGRAF3 met de mogelijkheid om deze via de communicatiepoort naar de printer te sturen. werkt niet op elke printer!

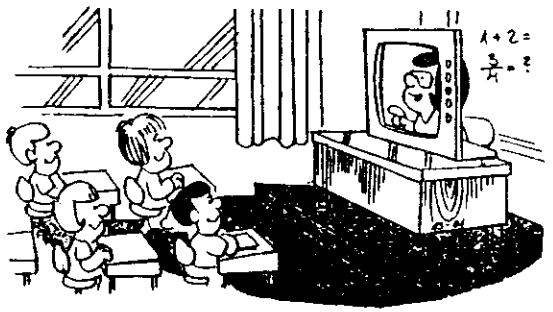
160. GANSBORD: het bekende spel ganzenbord op de newbrain. spel voor twee personen met breuken oplossen om het gegooide aantal punten ook te mogen verplaatsen

161. KLOK: een digitale klok op de newbrain. dit programma is in pascal geschreven. door op T te drukken kan men de tijd instellen en met S stopt het programma

162. NC: het bekende spel 'noughts and crosses' (boter-kaas-en-eieren) in 3d-vorm. ook dit programma is in pascal geschreven

163. PASTRANS: hulpprogramma om op de cassette aangemaakte pascalprogramma's over te brengen naar de disk (in regel 2 de programma-namen invullen)

164. LESLIE: in pascal geschreven programma, waarmee men met de leslie-matrix kan werken. de matrix kan maximaal 12 rijen bevatten



VERBETERING VOOR SSCC

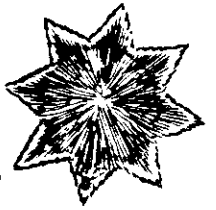
Voor het programma SSCC (On-line 7, pag. 35) is hier een kleine toevoeging, die voorkomt, dat spaties die tussen het einde van een programmaregel en het commentaar staan, meegaan naar het objectprogramma:

```
415  FOR c = e - 1 TO 1 STEP -1 : IF MID$ (a$, c, 1) <> " " THEN 420
417  e = e - 1 : NEXT c
```

De spaties achter '5' blijven zo netjes achter in het source programma in een regel als

```
test      for ~count~ = 1 to 5          ;/ loop initialize
```

Rob Maris



NAUWKEURIGE STERGEGEVENS

Het sterprogramma van de prijsvraag in On-line 7 is, afgezien van de Halley-toevoegingen, te vinden als het programma 'Skyset' in het boek Celestial Basic van E. Burgess.
Wat betreft nauwkeuriger sterrengegevens, beschik ik over de nauwkeurige gegevens (rechte klimming, declinatie en helderheid) van de sterren die ik gebruikte in mijn programma 'Sterrenbeelden' (op bestelnummer ONLINE-7 in de softwarebibliotheek). Ze zijn beschikbaar als sequential file of als DATA-regels, eventueel met programmaatjes om een file aan te maken of DATA-regels uit de file te vormen. De stergegevens zijn overgenomen uit het Guinness' Book of Astronomy. Eventueel zijn er ook de cp/m-volumes uit de cp/m-bibliotheek, maar die gaan waarschijnlijk wel wat ver, evenals de verschillende sterrencatalogi die op de markt zijn.
Terzijde wil ik nog opmerken, dat ik verschillende van de programma's uit Celestial Basic voor de NewBrain heb bewerkt, en het misschien, indien er belangstelling voor bestaat, leuk zou zijn hier iets mee te doen. Vooral omdat ik het idee heb, dat de daarin gebruikte algoritmen niet altijd 100 procent of consequent toegepast zijn.

T. L. van Heummen

FOUT IN SZAP

wie zich het programma SZAP aangeschaft heeft, zal het wel eens overkomen zijn, dat het programma bij een zoekopdracht met de mededeling kwam, dat het einde van het bestand bereikt, terwijl uit de genoemde sectoren blijkt, dat dat niet zo is. de oorzaak is een fout in het programma: waar de inhoud van het DE- en het HL-register vergeleken moet worden, gebeurt dat verkeerd. gelukkig is de oplossing simpel, en kunt u het programma met zichzelf verbeteren: op adres 1240 sqq. moet DA 4B 12 veranderd worden in C2 45 12

menno stevens

MODIFICATIES AAN DIP.COM

Omdat na gebruik van DIP.COM mijn diskdrives (het gaat hierbij om de Teac FD-55F) als een stel varkens stonden te knorren, heb ik de seek-rate aangepast. Deze stond op 3 ms, maar kan veel beter naar 2 ms. De adressen waarop dit moet gebeuren zijn:

52D	5AD	62D	6AD	72D	7AD	82D	8AD
92D	9AD	A2D	AAD	B2D	BAD	C2D	

Slechts '03' wijzigen in '02'.
Indien men op adres 7A9 01 verandert in 00, dan zal zowel Proton-formaat als NewBrain-formaat op drive A: werken.
Met A: is drive A een NewBrain-drive; B: maakt van de A-drive een Proton-drive.

Evert Drijver

CPM64 VAN WIJLEN COMPUTEX

Al lange tijd probeer ik de volgende punten opgelost te krijgen.
1. CPM64.COM brengt me van CPM in CPM64, waar het prettig, zij het wat langzaam werken is. EXIT64.COM brengt me terug in de normale cp/m-omgeving. Tot zover geen problemen. Wil ik nu terug naar basic met EXIT.COM, dan gaat het systeem op tilt.
2. Het is met tot nu toe nog niet gelukt om de printerfunctie onder CPM64 aan de praat te krijgen. Het schijnt mogelijk te moeten zijn met CONFIGUR.

3. De implementatie van CPM64 houdt ook de mogelijkheid in van het gebruik van een RAM-disk. Maar hoe?
Ik zie uit naar eventuele reacties.

T. L. van Heummen
Stevinstraat 133, 2587 ED Den Haag
telefoon (070) 541171

HI-RES ONDER PASCAL OP COMPUTEX-UITBREIDING?

Mijn NewBrain heb ik door Computex in Apeldoorn laten uitbreiden tot 128k, en daardoor ben ik nu in staat onder cp/m met WordStar en Turbo Pascal te werken. Dit laatste gebruik ik graag. Ik heb nu al op vele manieren geprobeerd het hi-res-scherm van de NewBrain en de grafische mogelijkheden onder pascal te gebruiken. Dit is me nog niet gelukt. Wie kan me hierbij helpen?

D. H. Jas
Paalsteenlaan 16, 3761 Lanaken (België)
telefoon (09) 32 11 715909

EXPANSION AANGEVULD

Zijn er nog meer gebruikers die hun Expansion Interface Module (EIM) volledig willen hebben, dat wil zeggen de hardware voor device 22 (analoog-digitaalconverter) en device 23 (geluid) willen toevoegen? De device drivers zijn al in de ROM van de EIM aanwezig, dus de ROMs hoeven niet te worden vervangen.
Bij voldoende belangstelling kan de inbouw binnen een dag gebeuren en zullen de kosten minder zijn (75 tot 100 gulden, exclusief verzendkosten).
Reacties graag aan:

Wim Luijt
Troelstrapad 22, 4463 VM Goes
telefoon (01100) 28197

DRIJVER DRIJFT DRIVES

Teac-diskdrives zijn in Duitsland tamelijk goedkoop verkrijgbaar: zo'n ƒ 450,- voor 5¼" en zo'n ƒ 550,- voor 3½". Als er voldoende mensen

belangstelling hebben, is het de moeite waard naar Hamburg te scheuren. Neem even contact met me op.

Evert Drijver, telefoon (050) 778415

NOGMAALS RESETKNOP

Op bladzijde 77 van NewBrain On-line 7 zijn links van de diodes 4 puntjes getekend. De diodes zitten aan de twee verkeerde puntjes vastgetekend. Bij model AD is de +5 Volt niet de derde draad van onder (van de verbinding met de toetsenbordprint) maar de vijfde draad.

Rob Maris

vraagjes en weetjes
vragen en weeën

WAARSCHUWING

Helaas kan de softwarebibliotheek nog geen overlays aanbieden, waarmee programma's uit de bibliotheek, die uitsluitend geschikt zijn voor een 32k NewBrain, kunnen worden veranderd in versies die ook op de 96k NewBrain draaien. Er waren er al zes gereed, maar de inhoud van de schijf met deze programma's is door de NewBrain vernietigd. Van de meer dan 50 bestanden op deze schijf waren er nog maar 2 (twee) terug te vinden.
Zolang de juiste oorzaak hiervan niet duidelijk is, wil ik iedereen, die met twee diskdrives van verschillend formaat werkt, waarschuwen.
Wat is er gebeurd? Voor de softwarebibliotheek ontvang ik vaak schijven van 200k en 400k. Zelf werk ik met 800k en het eerste, wat ik meestal doe, is de bestanden van de ontvangen schijven overzetten naar een 800k-schijf. Bij dit overzetten onder cp/m zijn drie programma's betrokken: FAST.COM, een commercieel overzetprogramma en DIP.COM (in de softwarebibliotheek op bestelnummer NBGG-5). De 800k-schijf (bestemming) zit in drive A: en de 200k- of 400k-schijf (bron) zit in drive B:.

Na het overzetten blijkt nu, dat oude bestanden
op de bestemmingsschijf zijn overschreven.

Voor ingewijden: bij het overzetten zijn in de directory voor de nieuwe bestanden groepsnummers gebruikt, die bij bestaande oude bestanden horen. Het gevolg is, dat de sectoren met de oude bestanden worden overschreven door de nieuwe bestanden.
Eerst dacht ik aan een bedieningsfout, maar na onderzoek bleek, dat het herhaaldelijk optrad. Werd een ander omschakelingsprogramma (200K.COM) gebruikt, dan trad de fout niet op.

Het voorlopige resultaat van dit onderzoek is, dat de oorzaak hoogstwaarschijnlijk moet worden gezocht in DIP.COM. Een mogelijke fout in het BIOS of een bedieningsfout wil ik echter ook niet uitsluiten.
Zolang de oorzaak niet is opgehelderd, wil ik als dringend advies geven om bij het overzetten van bestanden op schijven van verschillend formaat uitsluitend naar LEGE schijven over te zetten, zodat geen bestaande bestanden kunnen worden vernietigd. Wil men deze bestanden aan een al beschreven schijf toevoegen, dan kunnen ze vanaf de schijf met het gelijke formaat worden overgezet.

Wim Luijt

wat is wat

In de artikelen in deze On-Line wordt een aantal termen en afkortingen gebruikt, die niet voor iedereen vanzelfsprekend zullen zijn.

32k of 32k NewBrain: een NewBrain, model A of AD, met of zonder disk controller (ook wel unexpanded NewBrain genoemd)

96k of 96k NewBrain: een NewBrain, model A of AD met EIM, met of zonder disk controller (ook wel expanded NewBrain genoemd)

128k of 128K NewBrain: een NewBrain, model A of AD, met of zonder disk controller, die na ombouw 128k RAM bevat en waarvan de ROM gewijzigd is. Moet onder basic worden beschouwd als een 32k. Technische details over geheugengebruik zijn nog schaarser dan de 128k NewBrain zelf

Disk controller: het kastje (hardware), dat de opslag op schijf regelt. De toevoeging van disk controller plus diskdrive(s) maakt het mogelijk om cp/m te runnen

EIM: expansion interface module, het kastje (hardware) met de 64k extra geheugen, extra device drivers en het POS

OS: operating system, het besturingsprogramma van de NewBrain.Verzorgt ook de in- en uitvoer

POS: paged memory operating system, het operating system van de 96k, dat het mogelijk maakt om met een Z80 meer dan 64k geheugen te kunnen bereiken (adresseren)

Z80: Z80-microprocessor. Z80-machinetaal wordt ook wel met deze term aangeduid (in een Z80-programma bestaan de opdrachten uit getallen)

Zero page: een stuk RAM-geheugen, dat altijd aanwezig is, en waar OS of POS de noodzakelijke systeemvariabelen bewaart

Wim Luijt

room at the top

Om op de NewBrain een Z80-programma te kunnen draaien moet er in het RAM-geheugen een gedeelte zijn, dat ontoegankelijk is voor basic. Dit gereserveerde gebied wordt geschapen door de basic-opdracht RESERVE. De systeemvariabele TOP heeft als waarde het eerste adres, dat niet voor basic beschikbaar is. Volgens de documentatie bewaart de NewBrain op adres 6 en 7 (B4 genaamd) op de zero page het einde van de voor basic beschikbare RAM + 1. Dit adres heeft dezelfde waarde als TOP. Echter, de waarden op de adressen 6 en 7 blijken bij een RESERVE-opdracht niet te veranderen. B4 is dus niet hetzelfde als TOP. De waarde van TOP kunnen we vinden op adres 32 en 33 van de basic-variabelen (B3PRM + 32 en B3PRM + 33).

Wil men bijvoorbeeld een Z80-routine van minder dan 100 bytes gebruiken, dan geeft men onder basic vaak de opdracht 'RESERVE 100' en POKEt daarna de Z80-routine vanaf TOP. Zolang je op je eigen NewBrain werkt, gaat dit goed: je hebt de routine immers op dezelfde NewBrain geassembleerd en eventueel gedebugged. De (absolute) adressen komen overeen met je NewBrain. Wil je echter de routine op een andere NewBrain gebruiken, dan kan het aardig misgaan.

Een van de oorzaken, het verschil tussen een 32k unexpanded NewBrain en een 96k expanded NewBrain, is door Menno Stevens al in On-line 5 behandeld (Transportabel, p. 33). In deze On-line staan nog meer tips hierover. Er is echter op de 32k NewBrain nog een andere oorzaak mogelijk.

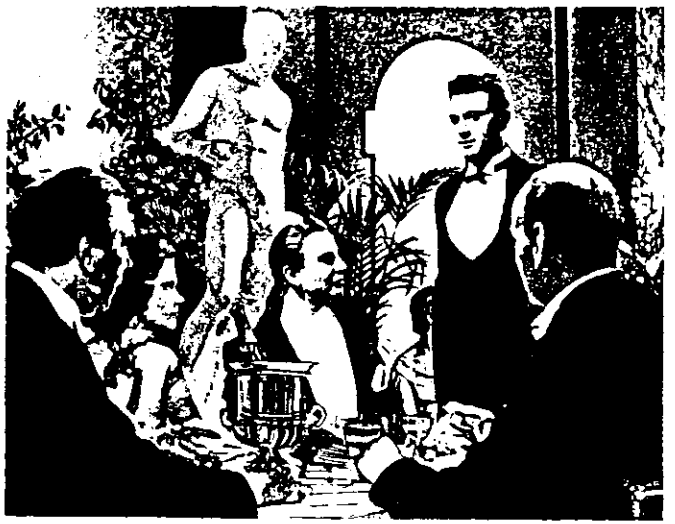
Bij het inschakelen van een 32k NewBrain blijkt TOP niet altijd op hetzelfde adres te liggen. Bij het inschakelen wordt eerst gecontroleerd, of het RAM-geheugen niet defect is, en krijgt TOP de waarde van het hoogste oneven intacte RAM-adres + 1. Ook als alle RAM o.k. is, blijkt nu, dat het

hoogste adres niet bij alle NewBrains gelijk is. Bij de ene NewBrain is TOP gelijk aan 32768, bij een andere is TOP gelijk aan 32766. Heb je een expanded NewBrain en RUN je UNEXPAND.BAS dan is TOP ook gelijk aan 32766. Voer je nu de opdracht uit om gereserveerd geheugen weer voor basic vrij te maken:

```
RESERVE 2 ^ 15 + TOP
```

dan is TOP altijd 32768 geworden.

De TOP-waarde van 32766 is dus kennelijk een of andere bug. Ik heb verschillende versies van het Operating System bekeken, maar er bleek geen relatie te zijn met enig Operating System. Van een hardwaredeskundige hoorde ik dat er in de NewBrain verschillende soorten RAM-chips gebruikt zijn. Mogelijk hangt het hiermee samen, voorlopig houd ik het maar op maandagmorgen-exemplaren.



Joe Lampton (Laurence Harvey) struggling toward the top in *Room at the Top*.

Voor het gebruik van non-relocatable Z80-routines heeft deze bug kwalijke gevolgen, als je ze op de verkeerde wijze POKEt. Stel, dat je de code op 32000 (7D00h) wilt laten beginnen, dan moet je het benodigde RESERVEren met:

```
RESERVE TOP - 32000
```

Met RESERVE 768 en vervolgens POKE TOP, x enz. kan de routine afhankelijk van de NewBrain soms op 32000 en soms op 31998 beginnen. Met de eerste opdracht ben je er zeker van, dat TOP gelijk is aan 32000 en kun je de routine aanroepen met CALL 32000 of CALL TOP. Zelf heb ik dagenlang zitten zoeken, wat er fout was aan een routine, die eerst wel werkte, omdat ik hem op mijn andere NewBrain had gemaakt.

Deze bug kan als gevolg hebben, dat Z80-programma's, die op een andere NewBrain zijn gemaakt, niet goed werken. Je kunt dit zelf oplossen door de waarde van TOP aan te passen en het dan opnieuw te proberen met de volgende opdrachten:

TOP van 32768 naar 32766:
RESERVE 2

TOP van 32766 naar 32768:
RESERVE 2 ^ 15 + TOP

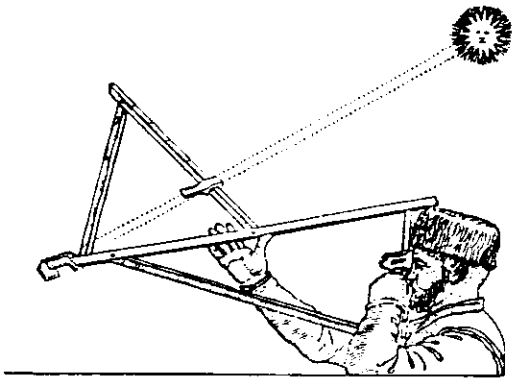
De softwarebibliotheek van de gebruikersgroep is gecontroleerd op het voorkomen van deze bug. Er zijn programma's aanwezig, waarin een niet correcte RESERVE-opdracht staat. Deze programma's werken echter met beide waarden van TOP.

Wim Luijt

de verwarring rond TOP wordt ook weerspiegeld door de programma's UNPAGE.BAS en UNEXPAND.BAS uit de softwarebibliotheek (op deel HULP-1): het eerste heeft een TOP van 32768 als resultaat, het tweede een van 32766. wie dit niet zint, kan het gelukkig gemakkelijk wijzigen door een getal in een DATA-regel te veranderen:

in UNPAGE.BAS kan TOP op 32766 gezet worden door in regel 4970
. . . 210080 . . . te veranderen in . . . 21FE7F . . .
in UNEXPAND.BAS wordt TOP verhoogd tot 32768 door
. . . 33, 254, 127 . . . te vervangen door . . . 33, 0, 128, . . .

printzone



Op de pagina's 35 en 36 van het handboek wordt de lengte van de printzone op het scherm (device 0) behandeld. Deze zou volgens het boek 10 bedragen.

De NewBrain bewaart deze lengte als systeemvariabele PZLEN op adres 29 (1Dh) op de zero page. Je kunt hem dus opvragen met PEEK (29). De waarde van PZLEN blijkt bij het opvragen afhankelijk te zijn van het aantal tekens op een regel op het scherm. Bij 40 tekens per regel is PZLEN 10, bij 80 tekens per regel is hij 17. Omdat PZLEN in RAM op de zero page staat, kun je de lengte van de printzone op het scherm veranderen met POKE 29, x. Let er wel op, dat iedere keer, dat er een nieuw scherm wordt geopend (OPEN #x, 0, y), PZLEN weer de standaardwaarde krijgt.

PZLEN heeft geen invloed op de printzone van device 8 (printer interface). Als de NewBrain een CHR$ (9) naar de printer zendt, wordt dit teken vervangen door een aantal spaties (maximaal 8), totdat het volgende teken in de volgende printzone komt. Deze lengte kan niet gewijzigd worden, omdat hij in de ROM vastgelegd is. Dat is jammer, want het zou slechts enkele bytes meer gekost hebben om de printzone op de printer gelijk te maken aan PZLEN.

Om bij het printen op een printer een gelijke printzone als op het scherm te krijgen, moet het scherm aan de printer worden aangepast. Hiervoor zijn 2 mogelijkheden:

1. POKE 29, 8 en gebruik device 8 (PRINTER);

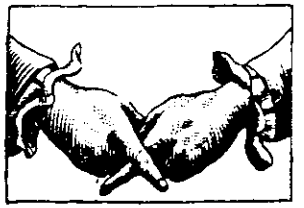
2. POKE 29, x, waarin x de standaardwaarde is voor de printzonelengte van de printer, en gebruik device 9 (COMMS). Zorg er dan wel voor, dat er na iedere CR [CHR$ (13)] een LF [CHR$ (10)] naar de printer gestuurd wordt. Dit gaat het eenvoudigst door een nieuwe device driver toe te voegen. Als hier belangstelling voor is, kan ik die maken. Schriftelijke reacties met opgave van de standaard printzonelengte van de gebruikte printer en de waarde van PEEK (42752) naar ondergetekende.

Voor de devices 9, 17 en 21 kent de NewBrain geen printzones. Bij printerdevice 16 is bepalend voor het gedrag van de NewBrain, welke regellengte is opgegeven in de parameterstring bij opening van de stream. Met "L0" (oneindige regellengte) reageert device 16 net als device 9; is de regellengte beperkt, dan geldt wat hierboven over device 8 gezegd is (waar de regellengte altijd beperkt is).

Wim Luijt

tokens

TOKENS, LISTSAVE, NO-SORT EN NO-EDIT

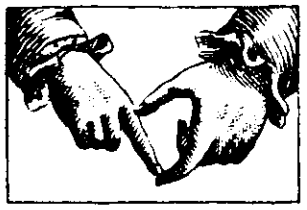


Naast een idee uit NBUG 12 en de behoefte aan een eenvoudige editor voor een assembler, is de aanleiding om dit artikel te schrijven de vraag van A. de Graaf, die, als hij met een programma uit het boek 'The NewBrain Files' van Philip Crookes een listing op het scherm wilde zetten, daar steeds 'rare tekens', grafische symbolen en Griekse letters in te zien kreeg.

Een basic-programma wordt in de NewBrain in twee gedeelten opgeslagen. Van een regel, bij voorbeeld

10 FOR i = 1 TO 255 : CLOSE #i : NEXT i

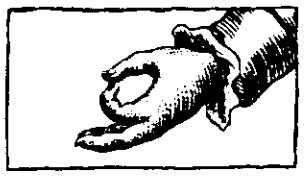
komt het getal 10 in de 'line number table' (LNT) en de rest in de 'source code area'. De nummers in de LNT met de bijbehorende twee wijzers (pointers) staan keurig in volgorde, en iedere opdracht (statement) krijgt een afzonderlijke vermelding in de LNT. Bovenstaand voorbeeld heeft dus drie vermeldingen in de LNT. Bij de uitvoering van het programma worden de opdrachten uitgevoerd in deze volgorde. Opdrachten worden door regeleinde of dubbelepunten gescheiden; in het voorbeeld zijn de opdrachten 'FOR i = 1 TO 255', 'CLOSE #i' en 'NEXT i'. Tijdens het programmeren wordt het 'tekst'-deel van de regel in de source code area in de volgorde van invoer van de regels gezet. De regels komen slechts één keer voor in de source code area, ongeacht het aantal opdrachten in de regel, en kunnen door elkaar staan. Om geheugenruimte te sparen, staan de sleutelwoorden (keywords) niet volledig in de source code area, maar zijn



ze vervangen door een 'token' (symbool), dat wil zeggen ze worden door één byte met een ASCII-waarde tussen 128 en 187 weergegeven.

Als we het natellen, kent NewBrain-basic 88 sleutelwoorden. Sommige sleutelwoorden hebben dezelfde betekenis, bij voorbeeld PRINT en ?, RETURN en RET, GOTO en GO TO. Ze hebben wel ieder hun eigen token, wat het aantal sleutelwoorden al op 92 brengt. Verder mag in 'IF . . . THEN . . .' het THEN achterwege blijven, als er direct een sleutelwoord volgt, bij voorbeeld IF . . . PUT . . . In de source code area komt er voor PUT een onzichtbaar (invisible) THEN, dat wil zeggen er wordt een sleutelwoord toegevoegd, dat tijdens LIST niet wordt afgebeeld. Er zijn dus 93 sleutelwoorden, die door 60 getallen worden weergegeven. Hoe kan dit?

De sleutelwoorden zijn in twee groepen ingedeeld, primaire en secundaire. Primaire sleutelwoorden staan aan het begin van een opdracht of na een THEN, en kunnen worden gevolgd door een of meer secundaire sleutelwoorden. Primaire sleutelwoorden hebben geen logische betekenis, als ze op de plaats van een secundair sleutelwoord voorkomen, en omgekeerd. 'LET a NEXT 16' is geen logische opdracht, terwijl 'LET a = 16' wel logisch is (NEXT en = hebben hetzelfde token, 140). In de regel in de source code area ([128] a [140] 16) weet de basic-interpreter dus, dat 'LET a = 16' de opdracht is. Ik ken drie schijnbare uitzonderingen hierop, namelijk



b = . . .
IF . . . GOTO 510
ON x GOSUB . . . of ON x GOTO . . .

De eerste opdracht mag gewijzigd worden in 'LET b = . . .'. In het tweede en derde voorbeeld staan twee primaire sleutelwoorden achter elkaar. Om enig idee te krijgen van de werkwijze van de 'entokeniser' (het deel van de interpreter, dat ingetypte sleutelwoorden in tokens omzet) kun je met de volgende opdrachten de source code area zichtbaar maken (let op de spaties!).

```
DEFFN(x)=PEEK(x)+256*PEEK(x+1):FORi=FN(FN(4)+14)TOFN(FN(4)+15)-1:
PUTi:PRINTPEEK(i):NEXTi: a = 10: IF a GOTO 10: IF a TO 10:
ONERRORGOTO10
```

Hieruit zijn de volgende conclusies te trekken:

- als het eerste byte in een opdracht kleiner dan 128 is, neemt de compiler een LET aan, zonder dat dit in de source code area staat (een onzichtbaar LET bestaat niet) *)
- ook als het eerste byte na een THEN een letter is, wordt deze aanname gemaakt
- is het eerste byte na een THEN een cijfer, dan vertaalt de compiler dit als GOTO
- als de entokeniser een secundair sleutelwoord verwacht, maar er een primair wordt ingetypt, dan voegt hij een onzichtbaar THEN toe

Tijdens een LIST-opdracht worden de regels met opdrachten aan de hand van de LNT gesorteerd op het scherm gezet, terwijl de sleutelwoorden in hun volledige vorm worden weergegeven. De source code area wordt niet gesorteerd. Tijdens LIST worden de sleutelwoorden in hoofdletters weergegeven, ook als ze in kleine letters zijn ingetypt, omdat de tokens als uitgangspunt voor de 'vertaling' dienen. Voer een programma met kleine letters in, en je krijgt een aardige controle of de sleutelwoorden correct zijn ingetypt.

Het SAVEn van een basic-programma. Als we een basic-programma op cassette of disk willen bewaren, dan opent de SAVE-opdracht een bestand op cassette of disk, zet opeenvolgend op de cassette of disk:

- een CHR$ (13)
- het eerste getal uit de LNT
- een spatie: CHR$ (32), alleen als de regel niet met een spatie begint
- de bijbehorende regel zoals die in source code area staat (met tokens!)
- een CHR$ (13)
- het tweede getal uit de LNT

  enzovoort tot
- het laatste getal uit de LNT
- een spatie: CHR$ (32)
- de bijbehorende regel zoals die in source code area staat
- een CHR$ (13)
- een CHR$ (4) = End-Of-File
- een CHR$ (13)

*) Het programma BTYPE.COM op bestelnummer NBGG-5 houdt hier geen rekening mee, waardoor a = 16 wordt 'geLIST' als a NEXT 16, terwijl LET a = 16 wel juist afgedrukt wordt.

overzicht van sleutelwoorden en tokens

byte	primair	secundair	byte	primair	secundair
128	LET	NOT	158	POKE	ASC
129	LIST	+	159	CALL	LEN
130	NEW	-	160	CONTINUE	VAL
131	PRINT	*	161	DEF	NUM
132	RUN	/	162	CLEAR	ASN
133	GOTO	↑	163	MERGE	ACS
134	GO TO	&	164	VERIFY	PEEK
135	GOSUB	AND	165	GET	ERRNO
136	GO SUB	OR	166	RESUME	INSTR
137	IF	<>	167	DELETE	ERRLIN
138	INPUT	<=	168	PUT	TOP
139	FOR	>=	169	LINPUT	FREE
140	NEXT	=	170	REPORT	CHR$
141	RETURN	>	171	CONT	STR$
142	REM	<	172	RESERVE	RIGHT$
143	END	RND	173		FILE$
144	STOP	PI	174		LEFT$
145	DIM	POS	175		MID$
146	ON	ABS	176		BASE
147	OPTION	ATN	177		IN#
148	SAVE	COS	178		OUT#
149	LOAD	EXP	179		#
150	RANDOMIZE	INT	180		THEN
151	OPEN	LOG	181		onzichtbaar THEN
152	CLOSE	SGN	182		TO
153	?	SIN	183		STEP
154	RET	SQR	184		FN
155	RESTORE	TAN	185		TAB
156	READ	TRUE	186		ERROR
157	DATA	FALSE	187		BREAK

De sleutelwoorden cpm (32k met diskcontroller), exit (96k met diskcontroller) en savf (beide met diskcontroller) hebben geen token.

en sluit het bestand. GeSAVEde basic-programma's staan dus in de verkorte, geTOKENde vorm op cassette of disk. Bovendien is het gesorteerd op regelnummer. Ook tijdens SAVE wordt source code area niet gesorteerd.

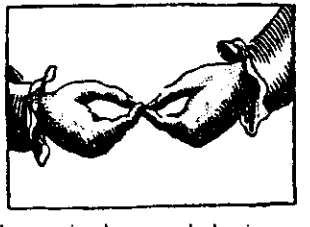
LISTSAVE

We kunnen een programma ook SAVEn met de volledige sleutelwoorden. Dit moet als volgt: OPENOUT een stream met als bestandsnaam de naam van het basic-programma, LIST naar deze stream, PRINT naar deze stream een CHR$ (4) en CLOSE de stream. Bijvoorbeeld:

```
OPEN OUT #1, 1, "KEYWORD.ASC" : REM de extension .ASC is hier
    gebruikt om het bestand te onderscheiden van het normaal geSAVEde
    programma KEYWORD.BAS; gebruikers van een diskdrive mogen ook
    device 12 of 13 kiezen
LIST #1
PRINT #1, CHR$ (4)  of  PUT #1, 4, 13 : REM deze stap kan achterwege
    blijven, zie onder
CLOSE #1
```

Een op deze wijze geSAVEd programma kan verder normaal geLOAD worden, alleen duurt het langer. Zelfs als het in het formaat van device 13 (met een CHR$ (10) na iedere CHR$ (13)) op disk of cassette staat, trekt LOAD zich hier niets van aan. Alleen als vergeten is om het programma met CHR$ (4) + CHR$ (13) af te sluiten, krijgen we de foutmelding ERROR 133 of ERROR 151. Het programma bevindt zich dan al op de goede plaats in het geheugen en kan verder normaal worden geRUNd of aangevuld met nieuwe regels.

Wil je een listing van een basic-programma in een tekst opnemen, dan kan dit door het geLISTSAVEde programma met een tekstverwerker in de tekst op te nemen.
De Basicode2-vertaalprogramma's voor de NewBrain maken ook gebruik van deze geLISTSAVEde programma's.

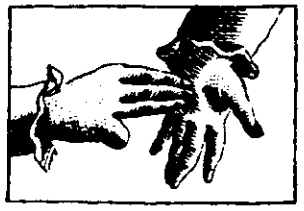


NO-SORT

Het idee in NBUG was om een databestand, waarin de records beginnen met een uniek getal, als basic-programma in te voeren en te SAVEn.

Voordelen zijn de enorme sorteersnelheid op het eerste getal en de mogelijkheid om het bestand onder cp/m met TYPE te kunnen lezen en te printen (als het tenminste in het formaat van device 13 geLISTSAVEd is). In het bestand kunnen echter alleen hoofdletters worden gebruikt. Een ander nadeel is echter, dat sortering op andere kenmerken niet mogelijk is. Door de onzichtbare THENs hoef je niet bang te zijn, dat de tekst bij vertaling naar letters gekke effecten gaat geven. 10 ANDY STOPMAN EXPOSITION ROAD 13 SINGAPORE wordt dus niet: 10 GOSUBY PIMAN LOADOSITIABS ROAD 13 ?GAPGO SUBE.

Kleine letters kunnen worden toegepast door achter het regelnummer met Graphics/N een REM toe te voegen. Een record wordt hierdoor een byte groter, maar zal geen onzichtbare THENs bevatten.

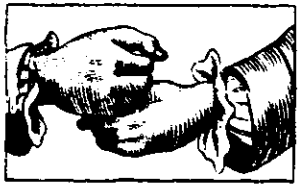


NO-EDIT

Deze truuk kan worden toegepast als editor voor een assembler, waarbij de source code regelnummers moet hebben. Type de source code als basic-programma in. Type na ieder regelnummer Graphics/N en type de regel verder in. Tabulatie kan worden verkregen met Escape gevolgd door Control/Escape. LISTSAVE het programma (de source code) als boven omschreven en assembleer.

Voorwaarde is dat de REM (code 142) door de assembler wordt genegeerd. Is dat niet het geval, wat mij het meest waarschijnlijke lijkt, dan wordt het iets moeilijker en minder fraai. Code 142 kan dan direct achter de commentaar-puntkomma worden gezet. De volgende OP-codes zijn secundaire sleutelwoorden en worden daarom verkeerd geLIST: AND, OR (en XOR) worden GOSUB en GO SUB. Deze moeten dus worden voorafgegaan door een label met een primair sleutelwoord, bij voorbeeld LET1, LET2, en-zovoort. Naar deze labels kan worden verwezen. Verder is het raadzaam om in labels geen sleutelwoorden op te nemen.

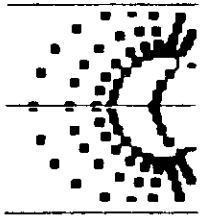
Andere toepassingen voor deze truuk moet je zelf maar bedenken, ik breng alleen maar het idee over.



Wim Luijt

basicode

GEBRUIKERSTIPS



Enige tijd geleden stootte ik op een verzameling programma's in Basicode-2 in mijn cassettotheek, waarvoor ik eigenlijk nog helemaal geen tijd had gehad deze uitgebreid te gaan zitten bekijken. Zoals u allen weet, is daarvoor nogal wat nodig voor de NewBrain, en wellicht was dat dan ook een van de redenen om de programma's te laten, waar ze waren.
Het bloed kruipt echter, waar het niet gaan kan, en omdat ik na vele jaren weer wat tijd voor mijn hobby kreeg, besloot ik de programmatuur dan maar weer eens te gaan bekijken. Natuurlijk, ik had de programma's wellicht ook wel kant en klaar via de Basicode-gebruikersgroep kunnen betrekken, maar ik ben nu eenmaal nieuwsgierig van aard en wilde zelf de nodige routine opdoen in het overzetten van Basicode-programma's naar NewBrain-formaat.
Bij het doorbladeren van de On-line-versies vond ik eigenlijk geen artikel, dat inging op de problemen, die het inlezen van de programma's met zich meebrengt. Natuurlijk, het veranderen van de CT-waarde in het lees-programma vond ik voor mij een nuttige tip, want ik houd niet zo van solderen. Ik koos dus als CT-waarde 50, hetgeen goed werkt, zowel op mijn Rubycom-cassetterecorder als op het exemplaar van het merk Sanyo.

Ter zake nu. Ik ontdekte, dat ik het leesprogramma op twee manieren kon toepassen, hetgeen voortkomt uit het feit, dat ik momenteel over twee NewBrains kan beschikken, waarvan een met schijfgeheugen:

1. Na het inlezen van een Basicode-programma kon ik het wegschrijven naar cassette en het vervolgens weer laden. Na het laden moest ik dan de subroutines nog MERGEn. Vervolgens schreef ik het aldus verkregen programma weg naar cassette. Ik laat het 'debuggen' hier verder buiten beschouwing.

2. Ik wilde eigenlijk mijn schijfgeheugen beter benutten en dacht, dat het wellicht ook minder omslachtig zou kunnen. Ik las daarom eerst een Basicode-programma in en schreef het weg naar cassette. Vervolgens deed ik die cassette in de recorder van het systeem met schijfgeheugen en typte:

OPEN #2, 2 (recorder aan uitgang TAPE 2), en drukte de play-toets in. De recorder begon te lopen en stopte na de titelmelding op het scherm. Het daarop volgende commando was dan: LOAD #2, waarna de recorder weer begon te draaien en de NewBrain het programma verder kon inladen. Ik behoefde nu niet de subroutines te MERGEn om het programma naar cassette weg te kunnen schrijven, maar wel om het, na controle en debugging op schijf te kunnen zetten. Maar eerst sloot ik af met: CLOSE #2. Vervolgens typte ik: MERGE "bcosubr.bco" (de door mij gebruikte naam voor de subroutines op schijf) en na enkele seconden was ik in staat om:

a. het programma weg te schrijven naar schijf, als het foutloos was,
b. het programma te gaan debuggen, een situatie die meer voorkwam dan mij lief was.

Voordat ik het over dit debuggen zal hebben, eerst nog een opmerking over een door mijn gemaakte en gelukkig herstelde vergissing.
Ik had door middel van OPEN #2, 2 een programma ingeladen, maar werd afgeleid door de telefoon. Ik dacht na het telefoongesprek, dat ik al LOAD #2 ingetypt had en ging ijverig verder met: MERGE "bcosubr.bco". Toen ik dat had gedaan, RUNde ik het programma en zag, dat ik alleen de subroutines had ingeladen. Wat nu? Ik had geen zin alles opnieuw te doen en dacht even na. Ik dacht, 'misschien kan ik met het MERGE-commando de zaak alsnog rechtzetten'. Daarbij moest ik niet uit het oog verliezen, dat ik met een systeem met schijfgeheugen werkte, dus typte ik: MERGE #2. En ja hoor, tot mijn grote vreugde slaagde deze opzet en verkreeg ik toch het gewenste Basicode-programma. Ik hoop, dat deze MERGE-tip velen tot nut kan zijn.

Laat ik nu het debuggen maar gaan beschrijven, waarbij ik hoop, dat meerdere gebruikers dit zullen doen. Ik zal in dit geval een vijftal problemen de revue laten passeren.

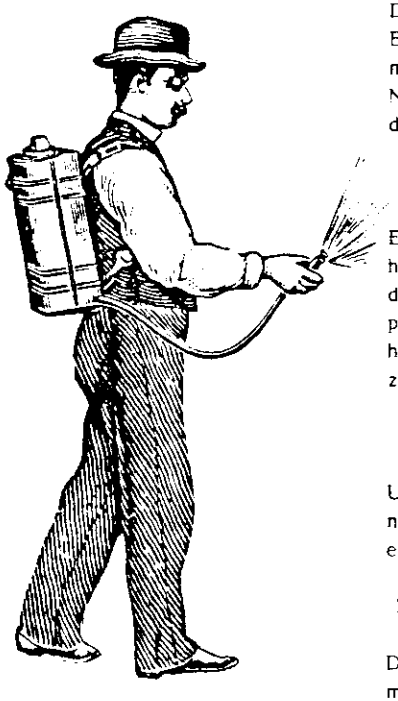
ERROR 29. Het inlezen van Basicode-programma's wordt nogal eens gekenmerkt door de mededeling: checksum klopt niet. Mij is gebleken, dat ERROR 29 dan af en toe kan optreden, dus het feit, dat er ergens een GOTO-statement staat, maar de regel ontbreekt, waarnaartoe gesprongen moet worden. Veelal gaat het dan om een weggevallen cijfer, bij voorbeeld GOTO 1580 is bij het inlezen overgekomen als GOTO 158. Een GOTO-fout is meestal gemakkelijk te herstellen, want het gebeurt zelden, dat er hele regels uit een programma zijn weggevallen bij het inlezen.

ERROR 31. Deze fout komt wel zeer frequent voor. Wat ik het geval? Veel programma's, die op andere merken microcomputers zijn geschreven,

missen de puntkomma. Zo is het bij voorbeeld mogelijk te vinden: PRINT "Hallo, " A$. U ziet het al, voor de NewBrain leidt dit tot ERROR 31, want het moet zijn: PRINT "Hallo, "; A$. Die puntkomma is de boosdoener.
Het is me opgevallen, dat bij het gebruik van de TAB-functie ditzelfde feit een rol speelt. Ook hier staat veelal geen puntkomma op de daarvoor bestemde plaats, bij voorbeeld PRINT TAB (12) "Hallo!" in plaats van PRINT TAB (12); "Hallo!".

ERROR 45. Hierbij gaat het om een NEXT zonder toevoeging van de bijbehorende letter uit het FOR-statement. Zo kan er bij voorbeeld staan:

```
10 FOR i = 1 TO 10 : NEXT
```



De NewBrain meldt zich dan met ERROR 45. Deze fout is dan gemakkelijk te herstellen door de i aan NEXT toe te voegen. In dit geval zou de programmaregel luiden:

```
10 FOR i = 1 TO 10 : NEXT i
```

ERROR 53. U moet eens opletten, hoe vaak het voorkomt, dat er geen dubbelepunt gezet wordt, wanneer de programmeur een REM-statement heeft gebruikt. Er staat dan vaak zoiets als:

```
10 INPUT n$ REM *** naam ***
```

U ziet wel, dat de dubbelepunt tussen n$ en REM ontbreekt, dus die moet u er dan maar weer tussen zetten:

```
10 INPUT n$ : REM *** naam ***
```

Doet u dit niet, dan blijft u zitten met ERROR 53!

ERROR 57. Deze fout wil menig programma verknallen. Hij is ook niet meteen te herkennen, omdat je er gewoon niet bij stil staat, te meer niet, wanneer je met vele merken microcomputers te maken hebt. In ieder geval

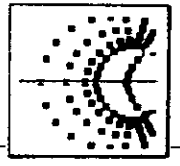
gaat het hierbij om een voor de NewBrain eigenzinnig gebruik van het RND-statement. Ik heb een en ander als volgt opgelost; het werkte dus. Veel programma's hebben achter de letters RND een cijfer tussen haakjes staan. De NewBrain accepteert die niet, want deze werkt anders en kent bovendien het echte RANDOMIZE. Wanneer u nu bij voorbeeld ziet:

```
50  a = INT (1 + 10 * RND (1))
```

dan behoeft u niets anders te doen dan het getal tussen haakjes achter RND weg te halen en uiteraard ook de haakjes, zodat de regel zou worden:

```
50  a = INT (1 + 10 * RND)
```

Er zijn ongetwijfeld nog veel meer van dergelijke problemen. Ik hoop dan ook, dat dit artikeltje een aanzet is tot meer publicaties op dit gebied. Veel succes alvast met het debuggen.



drs Eef Tobé

Commentaar van de softwarebibliothecaris

Basicode2-programma's kunnen tijdens de eerste vertaalstap direct naar disk worden geSAVEd. Ook de subroutines kunnen vanaf disk worden geMERGEd. Het orginele programma, zoals in On-line 2 (pag. 50), doet dit automatisch, als er op een systeem met diskdrives wordt vertaald. Als er een checksum error optreedt, kan het programma meestal met een NewBrain-tekstverwerker (een die ook ASCII 128 - 255 kent) zeer snel verbeterd worden.
De overige genoemde fouten zijn te wijten aan programma's, die niet aan het Basicode-protocol voldoen. De enige uitzondering is misschien ERROR 53, omdat het protocol niet heeft gedacht aan regels met meerdere opdrachten en stelt, dat er in een REM geen : mag voorkomen. Sommige programmeurs kunnen dit hebben geïnterpreteerd als: voor een REM mag geen : voorkomen. Verder is het een goed gebruik om REMs in een afzonderlijke programma regel te zetten.
In een volgende On-line zal nader op het Basicode-vertaalprogramma worden ingegaan, en, als er meer bijzonderheden bekend zijn, op Basicode-3.

Wim Luijt

ldcode

COMMENTAAR OP 'LDCODE'
VAN FRANS VAN DER VELDEN (ON-LINE 7, PAG. 59)

De beschreven methode om een machinetaalprogramma te laden lijkt erg interressant. Toch vroeg ik mij af, of er een wezenlijke tijdwinst mee bereikt wordt. Op de 'klassieke' wijze, dat wil zeggen de Z80-routine staat als DATA in een basic-programma, moet:

1 het programma inclusief de 'Z80-bytes' worden geLOAD. Per byte Z80-instructie zijn gemiddeld 3 bytes of meer nodig (cijfer 1, cijfer 2, cijfer 3, de komma in een DATA-regel en soms 2 spaties zonder functie);
2 een string met een decimaal getal worden gelezen (READ) en worden omgezet in een floating-pointgetal;
3 dit floating-pointgetal worden omgezet in een 16 bits integer; en
4 acht bits moeten als één Z80-byte worden gePOKEt.

Mijn indruk is, dat vooral de stappen 2 tot en met 4 erg traag zijn. Met LDCODE worden alleen de stappen 2 en 4 versneld, terwijl de data in één regel moeten staan. Dit laatste beperkt de Z80-routine tot circa 5000 bytes op een 96k NewBrain en op een 32k NewBrain tot aanzienlijk minder, ervan uitgaande, dat een 'Z80-byte' vier bytes basic kost en de langste programmaregel, die op een 96k kan worden ingevoerd, 80 x 255 = 20400 bytes lang is.

Een andere veel gebruikte methode is om de bytes in een aparte datafile te zetten en deze door een klein basic-programma (een zogenaamde bootstrap = laarzetrekker, een lus in een laars die dient om de laars aan te trekken)

te laden en te POKEn. De bovengenoemde tweede en derde stappen vervallen dan. Een voordeel van deze methode is, dat het bestand kleiner is (maximaal 75 % minder) is, en dat daardoor de tijd nodig om te laden korter is. Een ander voordeel kan zijn, dat er minder geheugen wordt gebruikt: de er hoeven geen DATA-regels te worden opgeslagen in het geheugen. Een extreem voorbeeld van deze manier van laden is het programma NEWMON.BAS uit de softwarebibliotheek, dat niet op de 'klassieke' manier kan worden geladen.

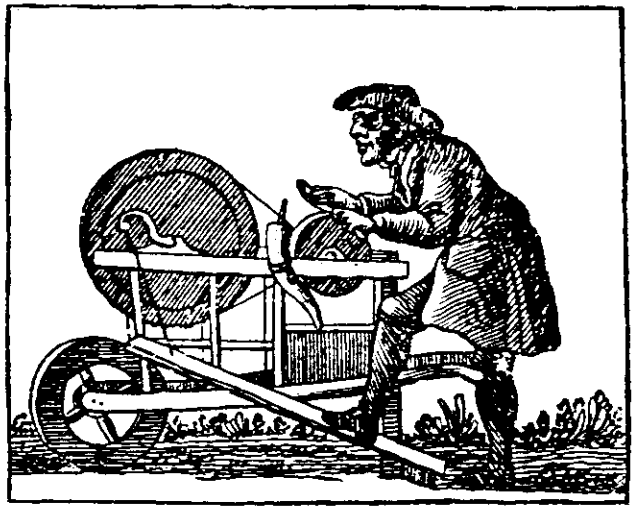
Om een antwoord op mijn vraag te vinden heb ik de volgende tests gedaan: het genoemde programma MONITOR van J. Braga (641 bytes) duurt, inclusief laden, met de klassieke methode als basic-programma 48 seconden en op de tweede manier slechts 30 seconden (vanaf cassette naar een 32k NewBrain). Met diskdrive duurt het 21 resp. 15 seconden. De methode van Van der Velden kon alleen met cassette worden getest *) en duurde 41 seconden. Alle tijden zijn inclusief het laden van het benodigde basic-programma. Door het programma met *20 te SAVEn kon nog enige tijdwinst worden behaald: 36 seconden. Met deze methode wordt dus minder tijdwinst behaald dan met de bootstrap-methode onder basic. Misschien is er met langere machinecodeprogramma's wel enige tijdwinst te behalen.

Een veel grotere winst kan worden behaald door in de bootstrap een machinecoderoutine op te nemen **).

```
; snel-loader voor Z80-routines, copyright W. Luijt, 1986
; vrij voor niet-commercieel gebruik
; relocatable
;
; LET OP! De te laden Z80-routine MOET op TOP beginnen!
;
        ORG 121             ;iedere ORG toegestaan
        LD L, (IY + 20h)    ;na (basic-) CALL: IY = B3PRM
        LD H, (IY + 21h)    ;HL bevat TOP
        LD D, 0             ;port 0
        LD E, 1             ;stream 1
```

*) De methode werkte niet, als ik mijn Expanded NewBrain met UNEXPAND.BAS liet krimpen tot 32k. Een test na het laden vanaf disk was dus niet mogelijk. Verder blijkt hieruit, dat een gekrompen NewBrain NIET identiek is aan een 32k NewBrain.

**) Het kan nog korter, als het aantal bytes van de Z80-routine bekend is: 11 bytes, zie TOOLKIT.BAS, CROSSING.BAS en DISASSEM.BAS.



```
NEXT:   RST 20
        DEFB 31             ;input
        RET C               ;error veroorzaakt door end of file
        LD (HL), A          ;'POKE'
        INC HL              ;volgende adres
        JR NEXT             ;volgende byte
        END
```

De basic-bootstrap ziet er als volgt uit:

```
10 FOR i = 1 TO 255 : CLOSE #i : NEXT i
20 RESERVE TOP - a : REM a = ORIGIN Z80-routine
21 b = 121 : IF PEEK (51) > 127 THEN     REM op 32k en 96k NewBrain
   b = PEEK (168) + 256 * PEEK (169) :   op zeropage; 'RESERVE'
   POKE 169, INT ((b + 17) / 256) :      op 96k
   POKE 168, b + 17 - 256 * PEEK (169)
30 FOR i = 0 TO 16 : READ c : POKE b + i, c : NEXT i : DATA 253, 110,
   32, 253, 102, 33, 22, 0, 30, 1, 231, 49, 216, 119, 35, 24, 249
31 REM regel 30: snelloader
40 ON ERROR GOTO 70
50 OPEN #1, "Z80rout" : REM Z80rout = naam Z80-routine;
                        bij cassette eventueel *20: toevoegen
60 CALL b
70 RESUME 80
```

```
 80 IF ERRNO = 133 OR ERRNO = 151 THEN 100
 90 REPORT
100 CLOSE #1 : END
```

Met deze bootstrap duurt het laden van MONITOR vanaf cassette slechts 25 seconden.

Deze bootstrap zal iets trager zijn dan de methode die door J. de Koning in On-line 3 (pag. 48) is beschreven. Ik kon het verschil niet meten. Het voordeel is, dat het bestand met de Z80-code niet achterste-voor hoeft te staan. Bovendien is deze methode ook geschikt voor de 96k NewBrain.

Hoe maak je van een Z80-routine in een basic-programma een datafile? Dit gaat het beste door het basic-programma te laden, de volgende regel toe te voegen vóór de POKE-loop:

```
OPEN OUT #1, "Z80rout"
```

en de POKE x, y te vervangen door:

```
PUT #1, y
```

Achter de NEXT i van de POKE-loop voeg je CLOSE #1 : END toe. Daarna geef je RUN.

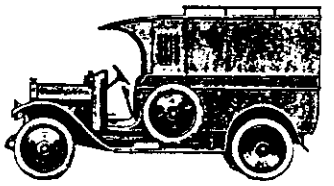
Conclusie: het laden van een Z80-routine gaat het snelste als de Z80-instructies in een afzonderlijke datafile staan en het POKEn in basic wordt vervangen door de beschreven machinetaalroutine. Het laden met de snelste methode duurt bij een Z80-routine van 641 bytes slechts circa 50 % van de tijd van de langzaamste methode. Bij langere routines zal de winst nog groter kunnen zijn, terwijl er geen geheugen nodeloos gebruikt wordt.

Wim Luijt

verkrijgbaar op cassette
zie pagina 7

trans-portabel

WANNEER
IS EEN MACHINETAALROUTINE
VOOR EEN 32K NEWBRAIN
OOK TE GEBRUIKEN OP EEN
EXPANDED NEWBRAIN?



Informatie over het operating system van de 32k (unexpanded) NewBrain door de voormalige fabrikant (Grundy) was erg schaars. De beschikbare informatie was bovendien erg theoretisch. Informatie over de 96k (expanded) NewBrain is nog zeldzamer.

De enige beschikbare informatie over de 96k staat in: 'NewBrain paged memory operating system - issue 0.1' en 'System page description - issue 0.0', twee stencils, die de meeste (?) bezitters van een EIM bij aankoop hebben ontvangen. Het eerste is erg theoretisch en bevat weinig interressants voor de doorsnee gebruiker. Het tweede bevat wel nuttige informatie en is daarom vertaald opgenomen in deze On-line (pag. 65).

John Braga heeft in zijn boek 'The NewBrain dissected' aangegeven, hoe bij de 32k de theorie in praktijk kan worden omgezet. Hij gebruikt hiervoor meestal programma's en voorbeelden in Z80-machinecode. Toen hij zijn boek schreef, bestond de Expansion Interface Module (EIM) nog maar net. Er is nog sprake geweest, dat Braga een soortgelijk boek over de 96k zou schrijven. Gezien het geringe aantal EIM's, dat in Engeland verkocht is, is de uitgave van zo'n boek niet haalbaar.
'The NewBrain Files' van Philip Crookes bevat een schat aan informatie over de 32k en 96k NewBrain. De inhoud is te omschrijven als een samenvatting van wat her en der verspreid bekend was over de beide NewBrains. Het vermeldt echter weinig over de werking van de 96k.

In On-line zijn in de eerste nummers ook artikelen verschenen over de 96k. Voor de doorsnee basic-gebruiker, die van een 32k op een 96k NewBrain

overgaat, bevatten deze artikelen te weinig informatie. Een uitzondering hierop vormen de artikelen van Menno Stevens (On-line 5, pag. 33) en Henk Blik (ib., pag. 87). In de bladen van de buitenlandse gebruikersgroepen staat ook weinig over de 96k, in totaal heb ik drie artikelen met nieuwe informatie kunnen vinden.

De volgende informatie is gebaseerd op wat ik uit deze informatiebronnen heb verzameld, aangevuld met speurwerk en een aantal ervaringen. Veel punten zijn voor mij ook nog onduidelijk.

Bij mijn werk voor de softwarebibliotheek kom ik vaak programma's tegen die slechts op een van de NewBrains kunnen draaien. Om ze voor 32k en 96k geschikt te maken, hoef ik soms slechts kleine wijzigingen aan te brengen. Meestal betreft dit basic-programma's, die variabelen van de zero page gebruiken.

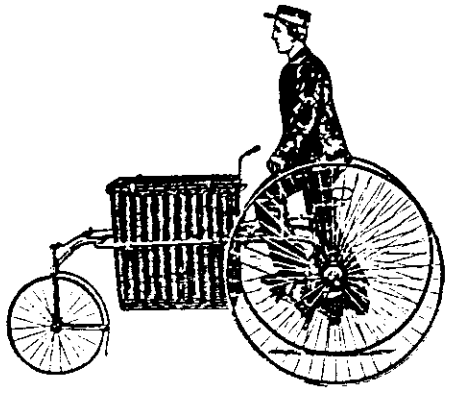
Er zijn (waren) echter vijf onderwerpen, die het functioneren van een 32k-programma op de 96k kunnen verhinderen:

1. Z80-programma's in het algemeen;
2. Z80-programma's die gebruik maken van absolute adressen (vooral op de zero page);
3. Z80-programma's die door voor mij onbekende oorzaak niet werken of crashen. Ik vermoed dat dit meestal te maken heeft met het gebruik van system calls (RST 20h) of van het grafische scherm (device 11);
4. basic-programma's, die in het schermgeheugen POKEn of Z80-routines die het schermgeheugen gebruiken;
5. programma's die extra (zelf gecreëerde) devices toevoegen of bestaande device drivers wijzigen

Enkele van deze problemen zijn opgelost.

Z80-machinecodeprogramma's voor een 32k kunnen dikwijls onveranderd op een 96k draaien. Aan welke voorwaarden dan moet zijn voldaan, is mij nog duister. Gewoon proberen, of het gaat, dat wil zeggen het programma voor 32k op dezelfde wijze RUNnen. Wel rekening houden met het volgende.

Een basic-gebruiker heeft op de 96k slechts direct toegang tot het geheugen van 0000h tot en met BFFFh (0 - 49151). Met een RESERVE TOP - x worden 16k van de 96k (van adres 8000h tot BFFFh, 32768 - 49151) niet effectief gebruikt. Bovendien is de zero page op de 96k veel groter (8k) dan op de 32k (256 bytes = ¼k), zodat er paradoxaal voor basic op de 96k minder geheugen beschikbaar is. Het schermgeheugen valt in tegenstelling tot de



32k NewBrain gelukkig buiten het geheugen voor basic, waardoor er bij een standaard scherm (24 x 40), voor basic circa 1½k meer beschikbaar is. Netto is er na het RESERVEren met hetzelfde adres voor TOP op een 96k dus ruim 6k minder geheugen beschikbaar voor het basic-programma, inclusief variabelen, arrays en tables. Een voorbeeld van een programma met een Z80-routine dat op deze wijze ook op de 96k werkt, is het oorspronkelijke Basicode-leesprogramma, nadat de REMarks verwijderd zijn. Latere toevoegingen hadden meer ruimte voor het basic-gedeelte van het programma nodig, waardoor het niet meer op de 96k of 32k met disk controller werkte. Het gebruik van het programma op de 96k heeft als voordeel, dat er langere Basicode-programma's (+ 8k) vertaald kunnen worden.

Wil je toch gebruik maken van het extra geheugen van de 96k, dan zal je de routine moeten 'relocaten', of gebruikmakend van de source code, de Z80-routine op een hoger adres moeten assembleren, tenzij de routine relocatable is, dat wil zeggen niet van absolute adressen gebruik maakt. Dit laatste is meestal van buiten niet te zien.
NEWMON.BAS (in deel HULP-2 van de softwarebibliotheek) is tot nu toe de enige beschikbare 'relocater', dat wil zeggen een programma om een Z80-routine naar een andere plaats in het geheugen over te brengen en gelijktijdig alle absolute adressen die binnen de routine gebruikt worden, aan te passen. Helaas werkt NEWMON (nog?) niet op de 96k, zodat hij niet gebruikt kan worden om de Z80-routines voor de 32k aan de 96k aan te passen. Blijft dus over om de source code opnieuw te assembleren met een hoger beginadres (ORG).

Een waarschuwing is hier misschien wel op zijn plaats. De 96k plaatst de device drivers voor de in- en uitvoer in slot 4, dat wil zeggen van adres 8000h tot 9FFFh. Onder basic hoef je daar geen rekening mee te houden; de NewBrain regelt dit allemaal voor je. Wat de gevolgen zijn voor een Z80-routine op deze plaats, is mij niet duidelijk. Tot nu toe is het me niet gelukt om een Z80-routine tussen 8000h en 9FFFh te laten werken. Of dit komt door de Z80-routine of door het PIOS, weet ik niet, maar ik denk, dat dit stuk niet zonder meer gebruikt mag worden voor een Z80-routine.

De 32k en 96k bewaren op hun zero page variabelen en adressen die voor het operating system nodig zijn. Deze variabelen en adressen kunnen ook in een Z80-routine worden toegepast. Het formaat van de zero page op de 32k is anders dan het formaat van de 96k. Bovendien kan de inhoud of de betekenis van bepaalde variabelen en adressen anders zijn. Worden deze variabelen en adressen toegepast, dan moet de Z80-routine worden aangepast.

HET PAGED MEMORY OPERATING SYSTEM IN EEN NOTEDOP

De Z80-processor in de NewBrain kent 64k (= 65536) adressen. In een 32k NewBrain zijn, afhankelijk van de aanwezigheid van een disk controller of een ROM-box 56k tot 64k geheugenplaatsen als hardware aanwezig. De Z80 kan dus alle geheugen plaatsen aanwijzen. De 96k NewBrain heeft 128k (= 132072) geheugenplaatsen. De Z80 kan dus niet alle geheugenplaatsen aanwijzen. We zouden dus niets aan de extra geheugenplaatsen hebben als er niet een truc op toegepast wordt.
De 64k adressen van de Z80 zijn in 8 gelijke porties van 8k verdeeld en ieder portie heeft een nummer gekregen. Zo'n portie wordt een slot genoemd. Er zijn dus 8 slots, genummerd van 0 tot en met 7. Slot 0 beslaat de adressen 0000h - 1FFFh, slot 1 de adressen 2000h - 3FFFh enzovoort. Slots zijn softwarematig bepaald.

De 128k (hardware-) geheugenplaatsen zijn nu ook in porties van 8k verdeeld. Zo'n portie wordt een pagina (page) genoemd. Iedere pagina heeft een nummer gekregen om hem te kunnen onderscheiden van de andere pagina's. De nummering van de pagina's wordt verklaard in 'NewBrain paged memory operating system - Issue 0.1' en ook in deze On-line (bladzijde 65).

Het Paged Operating System (POS) zorgt er nu softwarematig voor, dat de Z80 met een slot naar een andere pagina kan wijzen. Zo kan slot 1 (adressen 2000h - 3FFFh) op een bepaald ogenblik wijzen naar pagina 96. We zeggen dan, dat pagina 96 zich in slot 1 bevindt. Even later kan slot 1 naar pagina 136 wijzen en bevindt pagina 136 zich dus in slot 1.

Deze aanwijzing gebeurt dus softwarematig, wat betekent, dat we ze zelf kunnen sturen in een programma. Vanuit basic kan dit bijvoorbeeld door een gecombineerd slot-pagenummer te maken en dit in het geheugen op de juiste plaats (S0 tot en met S7) te bewaren.

Routines die bestaande device drivers wijzigen of nieuwe toevoegen zullen niet werken. Hoe deze moeten worden aangepast is elders in deze On-line beschreven (op bladzijde 51).

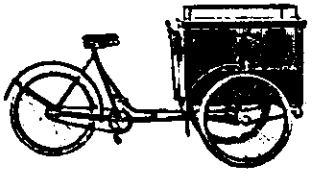
Basic-programma's en Z80-routines, die rechtstreeks adressen in het videogeheugen wijzigen, moeten ook worden aangepast. Die aanpassing is in deze On-line op bladzijde 60 te vinden.

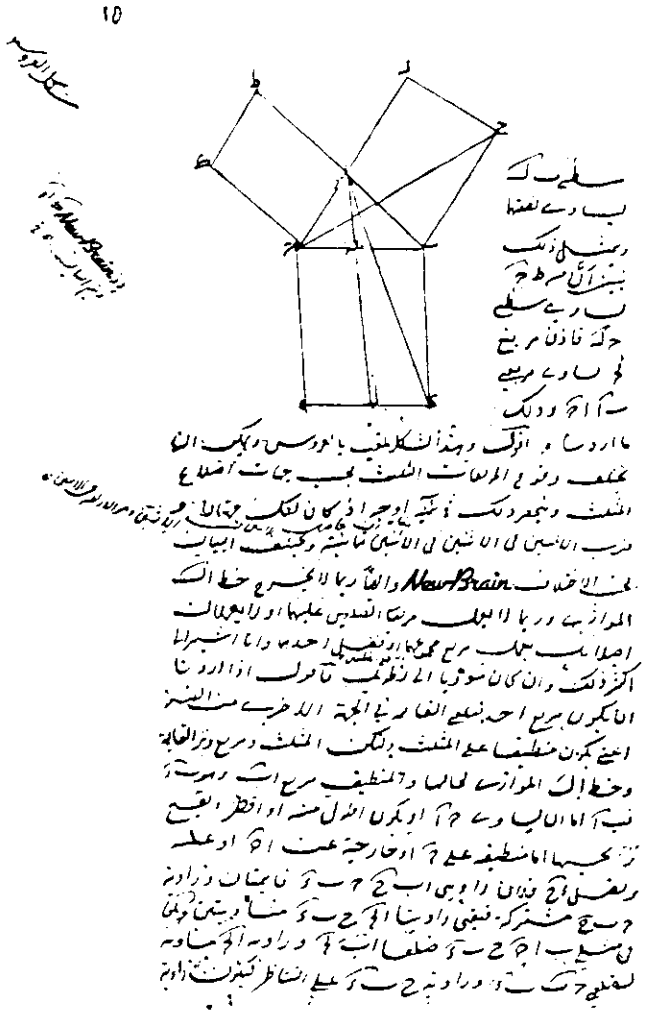
Samenvattend kan worden gesteld, dat Z80-routines voor een 32k zonder wijziging op een 96k kunnen draaien, waarbij de routine op dezelfde adressen in het geheugen gePOKEd moet worden. Met deze werkwijze blijft er op de 96k netto minder geheugen voor basic over dan er op de 32k beschikbaar zou zijn geweest. Een routine in slot 4 (van 8000h tot 9FFFh) plaatsen wordt afgeraden. De routine moet aan een aantal voorwaarden voldoen, waarvan de meeste mij onbekend zijn. Een van de voorwaarden is, dat als de routine van de inhoud van de zero page gebruik maakt, hij soms moet worden aangepast.
Routines die bestaande device drivers wijzigen of nieuwe toevoegen, moeten worden aangepast.
Behalve slot 4 mogen er afhankelijk van de routine waarschijnlijk ook andere slots niet gebruikt worden. Informatie over het gebruik van de slots door het operating system is onvolledig. Een voorlopige inventarisatie over het gebruik van slots en pagina's staat in deze On-line op bladzijde 73.
Verplaatsing van een Z80-routine naar een hoger adres, om het extra geheugen van de 96k te benutten wordt afgeraden:

1. de routine moet opnieuw geassembleerd worden, wat zeer moeilijk is zonder source code;
2. slot 4 mag waarschijnlijk niet zonder aanpassingen van de routine worden gebruikt;

Programma's met Z80-routines die voor een 32k NewBrain geschreven zijn, en die niet zonder wijzigingen op een 96k kunnen RUNnen, kan men het beste op een 96k RUNnen door er eerst met UNEXPAND.BAS of UNPAGE.BAS een pseudo-32k van te maken.

Wim Luijt





drivers

HET TOEVOEGEN EN VERANDEREN VAN EEN DEVICE DRIVER
ZOWEL BIJ DE 32K ALS DE 96K NEWBRAIN

Het toevoegen van een nieuw device aan een 32k NewBrain is door Cor Boer beschreven in On-line 5, pag. 71. Om het tweede deel van dit artikel (over het toevoegen van een device driver aan een 96k NewBrain) te kunnen begrijpen, of om zelf een nieuwe device driver te kunnen toevoegen, wordt hier de te volgen werkwijze stap voor stap behandeld. Aangeraden wordt om ook het artikel van Cor Boer te raadplegen. In dit artikel zal ik een aantal stappen die Cor Boer door middel van Z80 uitvoert in basic beschrijven, zodat het ook voor Z80-analfabeten (of aZ80en?) te begrijpen is.

WAT IS EEN DEVICE DRIVER?

Als er in- of uitvoer plaats vindt naar devices, gebeurt dit door middel van een zogenaamde device driver: een aantal Z80-opdrachten, die normaal in ROM staan. Ieder device kent vijf routines:

	wordt gebruikt door de basic opdracht(en):
OPEN IN	OPEN of OPENIN
OPEN OUT	OPENOUT
INPUT	GET, INPUT, LINPUT, LOAD, MERGE, VERIFY
OUTPUT	PUT, PRINT, SAVE, LIST
CLOSE	CLOSE

Device 0, 3 en 4 kennen ook nog de routine MOVE die in basic alleen in de opdrachten voor de screeneditor (PUT 1 t/m 31) wordt gebruikt.

De device drivers vormen geen onderdeel van het basic-programma. Als een van de bovenstaande basic-opdrachten in een programma voorkomt, wordt de device driver via het operating system aangeroepen. Nadat een nieuwe device driver met een basic-programma geladen is (zie onder), blijft deze driver aanwezig tot de NewBrain wordt uitgeschakeld. Andere basic-

programma's kunnen dus ook het extra device gebruiken.
N. B. Device drivers moeten in Z80-code worden geschreven!

In ROM staat op adres A058h (= 40948 = PEEK (88) + 256 * PEEK (89)) en verder de DEVICE TABLE met:

a. aantal devices (1 byte, er zijn dus 255 devices mogelijk)
b. per device het startadres van de device driver

Een device driver bestaat uit een tabel en vijf (of zes) routines, waarbij sommige routines voor meer dan een driver kunnen functioneren. De tabel bestaat uit een getal (aantal routines - 1) en de relatieve beginadressen van de vijf routines ten opzichte van de tabel.

Hoe installeren we een andere device driver op de 32k? Aangezien alles in ROM staat kunnen we de routines zelf niet veranderen. Wat we wel kunnen doen is de device table van ROM naar RAM kopiëren en adres 88 en 89 zodanig veranderen, dat ze naar de device table in RAM wijzen. RUN het volgende programma en alles gaat nu nog als vanouds.

```
 10  FOR i = 1 TO 255 : CLOSE #i : NEXT i
 20  dt = PEEK (88) + 256 * PEEK (89) :   REM dt = adres device table
 30  da = PEEK (dt) :                     REM da = aantal devices
 40  RESERVE 2 * da + 1 :                 REM 2 bytes per device
 50  FOR i = 0 TO 2 * da :      REM TOP = adres nieuwe device table
 60  POKE TOP + i, PEEK (dt + i) :        REM kopieer oud naar nieuw
 70  NEXT i
 80  POKE 89, INT (TOP / 256) :           REM verander pointer naar
 90  POKE 88, TOP - 256 * PEEK (89) :     REM nieuwe device table
100  END
```

Nu de nieuwe device table in RAM staat, kunnen we hem wijzigen en de pointers naar RAM laten wijzen. In deze RAM kan de nieuwe of gewijzigde device driver worden geplaatst. Een voorbeeld hiervan is de klok van Cor Boer. Er zullen hier twee voorbeelden worden behandeld: het toevoegen van een nieuw device, en het veranderen van een oud device.

EEN NIEUW DEVICE TOEVOEGEN

1. RESERVE het juiste aantal bytes
2. verander in bovenstaand programma: 40 RESERVE 2 * da + 3
3. voer bovenstaand programma uit tot en met regel 90

4. POKE het startadres van de nieuwe driver op TOP + 2 * da + 1 en TOP + 2 * da + 2
5. POKE de extra device driver vanaf het startadres
6. de nieuwe device driver is nu geïnstalleerd en je kunt het basic-programma, dat het nieuwe device gebruikt LOADen en RUNnen.

Je kunt de nieuwe device driver ook het rangnummer van een bestaand device geven, bij voorbeeld de klok van Cor Boer als device 10, maar het bestaande device wordt dan ontoegankelijk, terwijl je maar twee bytes RAM bespaart.

EEN BESTAAND DEVICE VERANDEREN

Een device totaal veranderen gebeurt, zoals hierboven beschreven is, behalve stap 4, welke wordt:

4. POKE het startadres van de nieuwe driver op TOP + 2 * da + x en TOP + 2 * da + x + 1, waarin x = 2 * (rangnummer device) - 1

Het veranderen van een of meer routines van een device is het moeilijkst. In Technical Note 9 is een voorbeeld beschreven. De driver ziet er als volgt uit (stel dat we de invoerroutine van device 2 willen veranderen):

```
DEFB 4
DEFB 5
DEFB 7
DEFB 0FH
DEFB 0AH
DEFB 0CH
JP startadres OPEN IN
JP startadres OPEN OUT
JP startadres OUTPUT
JP startadres CLOSE
INPUT: enz
```

Voor het installeren kan de werkwijze voor het totaal veranderen worden gevolgd. De startadressen van de ongewijzigde routines kunnen uit de tabel van de oude device driver worden gehaald.

HET TOEVOEGEN VAN EEN DEVICE DRIVER AAN EEN 96K NEWBRAIN

Zelf ben ik al bijna een jaar aan het zoeken geweest, hoe aan een 96k NewBrain onder het paged memory operating system (POS) een device driver kan worden toegevoegd. Kort geleden is dit gelukt. De volgende beschrijving is min of meer een verslag van de gevolgde werkwijze, waarin ook mislukkingen zijn opgenomen, omdat deze mislukkingen zeer informatief zijn.

Bij de ontwikkeling zijn drie delen onderscheiden: de wijziging en verbetering van de device driver (de klok van Cor Boer uit On-line 5), ten tweede de bepaling in welke RAM (op welke plaats in het geheugen c.q. welke pagina) de driver komt, en tenslotte wijziging van de device table. Hoewel ik de 3 delen afzonderlijk zal beschrijven, betekent dit niet, dat ze afzonderlijk zijn ontwikkeld.

WIJZIGING EN VERBETERING VAN DE DEVICE DRIVER

Met disassembleren en uit de beschikbare informatie wist ik, dat de opbouw van een device driver op een 96k hetzelfde is als op een 32k (voor de opbouw zie regel 37 - 42 van de source code in On-line 5).
De oorspronkelijke Z80-routine is niet alleen de device driver, maar bevat ook de installatieroutine. Deze laatste kan ook met behulp van basic worden uitgevoerd, waaraan de voorkeur is gegeven. Bovendien is bij het begin van de ontwikkeling de vraagstelling nog niet beantwoord, zodat het installatiegedeelte van de routine kan worden geschrapt. Verwijder de regels 9 tot en met 36 en regel 233.
Het eerste vrije device in de 96k NewBrain is device 24, dus:

```
4: DEV:    EQU 24
```

We weten nog niet, waar de device driver in de RAM moet komen. We gaan daarom proberen om de routine relocatable te maken. De volgende regels moeten worden veranderd:

```
87   JR Z, INITST     (bij assemblage bleek dat het mogelijk was om
89   JR Z, DISON      de JP's te veranderen in JR's)
91   JR Z, DISOFF
```



```
149  JR NZ, QUIT
154  JR NZ, QUIT
```

De routine werd hierdoor nog niet relocatable. Dit zou wel mogelijk zijn, als de volgende pointers en variabelen COUNT, STATUS, POINT, MAXTAB, SEC, MIN, UUR en CLKBUF (regels 221 - 235) samen met de QUIT-routine (regel 184 - 185: 184 ADRES: DEFB 0C3h, 185 DEFW 0000) op de zero page gezet zouden worden en na regel 183 wordt toegevoegd:

```
183  JP ADRES
```

Hiervan is afgezien.

De gebruikte zero-pagevariabelen moeten op de 32k en 96k gelijk zijn en anders worden aangepast. Uit de beschikbare informatie bleek, dat TINT (56 - 58, regel 61) gelijk is, maar dat de omschrijving van TVRAM (5C - 5Dh, regel 191) anders is. Het verschil is echter genegeerd. Over STREG en CLOCKI (regel 5 - 7) had ik geen informatie, dus die heb ik maar ongewijzigd gelaten.

IN WELKE RAM MOET DE DEVICE DRIVER

Ik had de informatie, dat wanneer de device driver gebruikt wordt, de pagina met de device driver zich in slot 4 moet bevinden. Hoe de desbetreffende pagina in slot 4 komt, heb ik aan het POS overgelaten. RESERVEerde ik geheugen in slot 4 en zette ik de device driver in dat slot, dan crashte de zaak of kreeg ik geen klok op het scherm. Bovendien was slot 5 dan niet beschikbaar, waardoor ik 8k voor basic verloor. Wat de oorzaak ven het mislukken is, weet ik niet. Ik ben daarom mijn geluk maar eens aan proberen op de zero page (zie kader). Dat betekende wel, dat als de driver aangeroepen wordt, de zero page (een RAM-pagina) in slot 4 gebracht moet worden. Dit bleek een goede greep te zijn geweest. Het is dus toegestaan om de zero page in een ander slot dan slot 0 te brengen, zonder dat het systeem op tilt gaat en POS zorgt er kennelijk voor dat de zaak na afloop weer in orde komt. In hoevere dit ook geldt voor ander RAM-pagina's, en zelfs voor ROM-pagina's, weet ik niet, maar ik neem aan dat dit voor RAM-pagina's zonder meer is toegestaan. Voor ROM-pagina's geldt dit zeker niet als de Z80-routines worden aangeroepen.

Pogingen achteraf om de device driver in een andere pagina (3, 4 of 5) te zetten zijn allemaal mislukt. De oorzaak hiervan heb ik niet kunnen vinden. Een zeer leuke BUG kwam ik tegen, toen ik de driver op pagina 100 (slot 4)

wilde POKEn. Nadat het basic-programma drie getallen had geREAD, begon de teller voor de DATA (DTINPT) weer van voor af aan, terwijl bij LIST het programma veranderd was! De source code table was nog ongewijzigd.

Op de zero pages van de 32k en 96k NewBrain is een geheugengebied, dat niet door OS of basic wordt gebruikt. Op deze plaats kan een Z80-routine worden gezet, wat als voordeel heeft, dat er geen geheugen voor basic verloren gaat.

Op de 32k is dit vanaf 79h (121) t/m FFh (255) en kan de routine dus maximaal 135 bytes lang zijn. Er hoeft met RESERVE geen geheugen te worden gereserveerd.

Op de 96k is het iets ingewikkelder. In ESYSPAGE, adres A8 - A9h (168 - 169) staat het eeste vrije adres na SLOTZEROCODE (adres 1702h (1906) tot en met ESYSPAGE - 1). De omschrijving van SLOTZEROCODE luidt:

This area contains various important system routines. If it is required to put in a new one then put it in at (ESYSPAGE) and increment (ESYSPAGE) past it.

Het lijkt dus duidelijk: een Z80-routine kan na SLOTZEROCODE worden geplaatst en ESYSPAGE moet worden aangepast. Als de 96k is aangezet is dit adres 1906h (6406). De Z80-routine moet dus beginnen op dit adres en kan maximaal 1784 bytes lang zijn.
In plaats van de opdracht RESERVE moet ESYSPAGE dus worden aangepast. Hier kan een addertje onder het gras schuilen. ESYSPAGE kan tijdens een programma veranderen. De area na SLOTZEROCODE zou in dat geval kunnen worden overschreven, waarbij de Z80-code verloren gaat. Om eventuele risico's te vermijden adviseer ik om de volgende opdrachten in het basic load programma helemaal vooraan te zetten:

```
FOR i = 1 TO 255 : CLOSE #i : NEXT i : OPEN #0, 0 : a = PEEK (168)
+ 256 * PEEK (169) + x : POKE 169, INT (a / 256) : POKE 168, a - 256
* PEEK (169) : REM x = lengte Z80-routine in bytes
```

Een extra voordeel van een Z80-routine op de zero page bij het paged memory operating system is, dat de routine altijd aanwezig is onder basic.

WIJZIGING VAN DE DEVICE TABLE

De device table in de 96k is anders dan in de 32k. Hij staat in RAM, en bovendien wordt DEVTAB niet gebruikt. De enige informatie over de device table, die ik had, staat in: System page description - issue 0.0 en luidt:

E00 - 1FF PDEVTAB. This is the table of device drivers, which is in stretched format. The first two bytes of each entry give the slot 4 page number of the device driver and the second two bytes give its location.

Nogal cryptisch voor mijn beperkte kennis. Met behulp van SYSPAGE.COM en disassembleren met Z80-DIS.BAS (ook van de 32k NewBrain mét disk-controller, zie kader) en enige informatie uit 'NewBrain paged memory operating system - issue 0.1, 9/4/83' kwam ik tot de volgende uitleg van bovenstaande omschrijving:

- een entry voor een device bestaat uit 4 bytes
- de device table is 4 x 256 bytes lang ;
- de 4 bytes staan niet achter elkaar, maar ieder volgende byte staat 256 adressen verder ('stretched format')
- de eerste 2 bytes vormen het S-P-N (zie pagina 73 van deze On-line)
- de tweede 2 bytes vormen het absolute Z80-adres van de device driver, als de ROM- of RAM-pagina met de device driver zich in slot 4 bevindt. Bij verdere analyse vond ik inderdaad, dat device 0 tot en met 23 en 33 een entry in de device table bezitten.

Als we een device driver gaan toevoegen of wijzigen, hoeven we dus niet eerst de device table van ROM naar RAM over te brengen, maar kunnen deze direct veranderen. Voor een nieuwe of gewijzigde entry moeten we dus ten eerste het S-P-N weten, en vervolgens de routine assembleren

Bij het disassembleren van de ROM in de disk controller bleek, dat de device driver voor device 18 (cp/m-scherm) en device 19 (gebufferd toetsenbord) al in de ROM aanwezig zijn. Dat is niet zo verwonderlijk, omdat ze onder cp/m op een 32k NewBrain met disk controller ook gebruikt worden. Pogingen om met behulp van de aanwezige routines device 19 onder basic ook te kunnen gebruiken zijn (voorlopig?) gestrand, omdat de driver van device 19 ZCALLS bevat die alleen in de ROM van de Expansion Interface Module zitten, en omdat de driver subroutines bevat voor page allocation. Mocht het alsnog lukken, dan laten we het u weten.

beginnend met een ORIGIN tussen 8000 - 9FFFh (32768 -40959). Voor de theorie van de berekening van de S-P-N zie pagina 73. De routine staat op de zero page. Met PEEK (139) en PEEK (140) kunnen we het S-P-N van de zero page vinden, als dit in slot 0 staat:

PEEK (140) & PEEK (139) = 0 & 96d = 00 & 60h = 0000 0000 & 0110 0000b

Het paginanummer van de zero page is dus: 000 0110 0000b. Als we de zero page, een RAM-pagina, in slot 4 willen brengen, dan wordt het S-P-N:

100 & 0 & 0 & 000 0110 0000b = 1000 0000 & 0110 0000b =
80 & 60h = 128 & 96d

De device driver begint op 1906h = 6406d en, als de zero page zich in slot 4 bevindt, op 9906h = 99 & 06h = 153 & 6d.
PDEVTAB begint op E00h = 3584d en de klok wordt device 24. We moeten PDEVTAB dus uitbreiden met de volgende opdrachten:

```
POKE 3584 + 24 + 0 * 256,  96 = POKE 3608,  96
POKE 3584 + 24 + 1 * 256, 128 = POKE 3864, 128
POKE 3584 + 24 + 2 * 256,   6 = POKE 4120,   6
POKE 3584 + 24 + 3 * 256, 153 = POKE 4376, 153
```

Nadat dit allemaal was uitgevoerd, device 24 geopend was met OPEN #24, 24 en PUT #24, 68 (dit laatste is hetzelfde als PUT #24, "D"), verscheen de klok op het scherm. En hij liep zowaar ook nog, al was het gebrekkig. Dit

Op mijn systeemschijf vond ik een programma OBJ.BAS, dat een .OBJ-bestand omzet in een DATA-regel. OBJ.BAS werkt goed, behalve dat het DEFS-code over het hoofd ziet. Mijn assembler (ZASM.COM van SD-systems) geeft namelijk in het .OBJ bestand voor iedere routine het startadres. Na de pseudo-operand DEFS in de source code wordt een nieuw startadres opgegeven, overeenkomstig de lengte van DEFS. OBJ.BAS bepaalt slechts een keer het startadres en plakt dus de verschillende routines achter elkaar. Het euvel is te verhelpen, door niet DEFS n maar door n-maal DEFB in de source code te zetten.

kwam echter door een fout in OBJ.BAS (zie kader). Maar ..., hij stond niet links bovenaan. Bij toeval ontdekte ik, dat hij op een 80-kolomsscherm op een andere plaats stond dan op een 40-kolomsscherm, en dat hij in beide gevallen ongeveer evenveel posities van de linkerbovenhoek af verwijderd was, als er met de EXCESS van de regels rekening wordt gehouden. In een gesprek met Menno Stevens en door bestudering van de source code kwam ik er achter, dat TVRAM bij de 32k een pointer naar het begin van het eigen geheugen van device 0 is, en dat TVRAM bij de 96k een pointer is naar het laatste adres, voordat het scherm begint (FLGSTO). FLGSTO staat op de 32k een variabel adressen na het begin van het eigen geheugen en de oorspronkelijke routine berekent dit braaf. Op de 96k kan deze berekening achterwege blijven, waardoor de regels 192 tot en met 196 kunnen vervallen. Wel moet de volgende regel worden toegevoegd:

```
192     INC HL  ; FLGSTO + 1
```

Het toevoegen van een device driver op de 96k is dus mogelijk, als de driver niet te lang is en op de zero page past. Een bestaande device driver wijzigen is moeilijker, omdat er dan tevens van pagina gewisseld moet worden. Dit wordt op het moment onderzocht. In een volgend nummer van On-line hoop ik hierover te kunnen berichten.

Wim Luijt

Naschrift: door een fout is de routine geassembleerd vanaf 6406. Toch werkt hij correct. Mijn conclusie hieruit is, dat de zero page gelijktijdig in slot 0 en slot 4 kan staan!

verkrijgbaar op cassette
zie pagina 7

poken in het scherm

Het scherm is een afbeelding van een stuk geheugen, het schermgeheugen. Verander je nu de inhoud van een adres in dit schermgeheugen, dan verandert er ook iets op het scherm. Andere computers hebben meestal een vaste plaats voor het schermgeheugen, dat wil zeggen het scherm staat altijd op dezelfde adressen en heeft altijd hetzelfde formaat. Een voordeel hiervan is dat er met grafische tekens redelijk snel op het scherm te 'tekenen' is.

Het schermgeheugen van de NewBrain is anders georganiseerd: het is een van de devices. Een voordeel hiervan is, dat we het scherm naar eigen wensen kunnen aanpassen. Een nadeel is echter, dat het schermgeheugen niet altijd hetzelfde formaat heeft en ook niet op dezelfde plaats in het geheugen staat. Directe schermbewerkingen, zoals POKE en PEEK in basic of LDIR in Z80-code, zijn hierdoor veel moeilijker uit te voeren. In basic merk je hier niet zoveel van, omdat je met de opdrachten PUT en GET van de schermeditor al deze bewerkingen kunt uitvoeren, zonder dat je hoeft te weten, waar het schermgeheugen zich bevindt.

Om het begin van het schermgeheugen van een 32k NewBrain te vinden zou een truc van Rob Maris (zie On-line 4, pag. 74) met een kleine wijziging kunnen worden toegepast: maak het scherm schoon en voer in:

```
PUT 12 : ? PEEK (90) + 256 * PEEK (91) : REM 90 = adres TVCUR
```

Je vindt dan een bepaald adres. Voer de opdracht opnieuw in zonder het scherm te wissen, maar zet de cursor op een andere plaats in de opdracht, voordat je op NewLine drukt. Je vindt dan een ander adres. Aangezien ik al

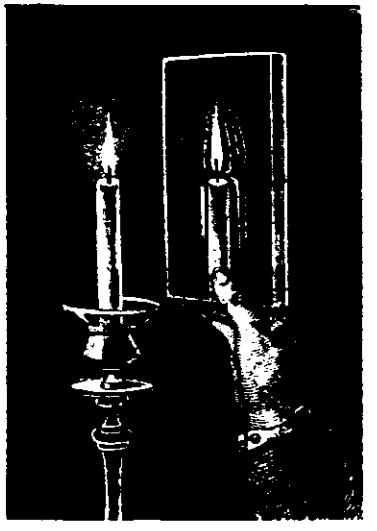
wist, waar het schermgeheugen begon (zie onder), wist ik, dat deze gevonden adressen niet het begin van het schermgeheugen zijn, maar de adressen van de positie van de cursor OP HET MOMENT, dat ik op NewLine drukte. Het begin kan wel bepaald worden door eerst op Home te drukken, dan de opdracht in te typen, weer op Home te drukken en daarna op NewLine.

Op de 32k NewBrain is in basic echter vrij eenvoudig te berekenen, waar het schermgeheugen zich bevindt.

```
1 a = PEEK (86) + 256 * PEEK (87) : REM 86 = adres stream table
2 FOR i = a TO a + 71 STEP 6 : IF PEEK (a) = 0 THEN 4 : REM stream 0
3 NEXT i
4 b = PEEK (a + 4) + 256 * PEEK (a + 5) : REM = startadres eigen
  geheugen stream 0
5 c = (INT (b / 128) + 1) * 128 + 2 : REM c = eerste adres schermgeheugen
  (regel 1)
```

Het begin van de volgende regels staat op c + n * 64 (40 tekens per regel) of op c + n * 128 (80 tekens per regel). Vergeet niet, dat de laatste 24 of 48 adressen van een regel niet worden afgebeeld! De regels 1 - 4 kun je ook vervangen door:

```
4 b = PEEK (92) + 256 * PEEK (93) : REM 92 = adres TVRAM
```



Je kunt dan ook de schermgeheugens van alle streams, die device 0 gebruiken, vinden, bij voorbeeld #1, 0, 1.

We kunnen de 32k NewBrain met de volgende opdrachten in een programma ook zo programmeren, dat het schermgeheugen een vaste plaats heeft. Daarna kan er dan naar behoefte in het schermgeheugen worden gePOKEt.

```
FOR i = 1 TO 255 : CLOSE #i
: NEXT i : OPEN #0, 0
```

Het eerste afgebeelde adres (= c) is dan 642, terwijl het eigen geheugen

(= b) begint op 616.

De meeste toepassingen hiervan in de softwarebibliotheek betreffen spelletjes, waarbij slechts met één device (het scherm) gewerkt wordt. Op deze wijze zijn programma's voor computers met een vast schermgeheugen ook eenvoudig in NewBrain-basic om te zetten.

Heb je bovenstaande opdrachten uitgevoerd op een 32k NewBrain, probeer dan eens het volgende. maak het scherm schoon met Shift/Home en voer in

```
POKE 642, 65
```

Links boven verschijnt dan een A in beeld. Vervolgens:

```
POKE 2114, 65
```

Er gebeurt dan niets! Ga nu met de cursor naar de onderkant van het scherm. Verrassing! Waardoor dit komt, weet ik niet precies, maar enige informatie staat in het bovengenoemde artikel van Rob Maris.
Er moet dus nog iets gebeuren, willen we dus naar believen in het scherm kunnen POKEn. Hiervoor heb ik de volgende oplossing:

```
FOR i = 642 TO 2152 STEP 64 : POKE i, 128 : NEXT i
```

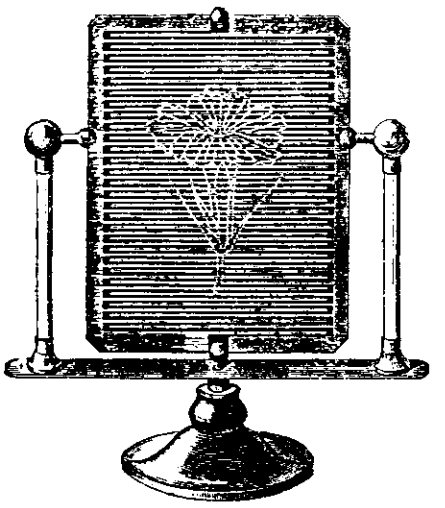
Nu is het gehele scherm 'geactiveerd', hoewel het schermgebeuren mogelijk wat trager is geworden. 'Bedruk' je het gehele scherm met een eigen ontwerp, bijvoorbeeld een doolhof, dan is bovenstaande opdracht niet nodig.

HET SCHERM VAN DE 96K NEWBRAIN

Mijn benadering op de 96k was hetzelfde als boven. Met PEEK (90) en PEEK (91) vond ik als eerste adres 32770. De berekening van het adres met behulp van de stream table ging niet, omdat deze anders is georganiseerd op de 96k en uitvoerige informatie hierover mij onbekend was. Paste ik de verkorte berekening toe met PEEK (92) en PEEK (93) dan vond ik 32769. De gevonden adressen zijn dus bijna gelijk, in tegenstelling tot de 32k (642 resp. 616). De oorzaak hiervan kon ik achterhalen en is als volgt.

Het schermgeheugen bij de 32k is een deel van het zogenaamde eigen geheugen van een stream met als device 0. Dit eigen geheugen van een stream naar device 0 bestaat uit:

- 16 adressen met variabelen en flags
- een variabel aantal adressen om het schermgeheugen te laten beginnen op twee adressen na een adres dat een veelvoud is van 128
- 4 adressen met variabelen
- 1 adres genaamd FLGSTO
- eerste schermadres
- tweede schermadres
  enz
- laatste schermadres
- 2 adressen met 0
- 2 adressen met spatie



Op een 32k wijst TVRAM naar het eerste adres van het eigen geheugen en de 96k wijst TVRAM naar FLGSTO. Waar zich het andere gedeelte van het eigen geheugen op de 96k zich bevindt, weet ik niet. Voor de vraagstelling is het ook niet van belang.

POKEn in het schermgeheugen (bijvoorbeeld POKE 32796, 65) lukte me echter nog niet. Nu stond me er iets van bij, dat het schermgeheugen buiten de 64k adressen van de Z80 vallen. Hoe je dan toch iets op het scherm kan krijgen, was me nog steeds niet duidelijk. Gelukkig kwam ik vrijwel gelijktijdig twee oplossingen tegen, waarmee onder basic het scherm toegankelijk wordt. De ene wordt gebruikt in het programma POKESCRN.BAS (verkrijgbaar bij de softwarebibliotheek onder bestelnummer ONLINE-8), de andere stond in het blad van de Franse gebruikersgroep.

Als je de opdracht POKE 147, 107 geeft, wordt er een zogenaamde videopagina in slot 4 gezet en blijft daar staan. Nu is het wel mogelijk om op een standaard-scherm van 24 regels direct naar het scherm te POKEn. Ook de waarden die in TVCUR staan kloppen nu.

Het enige verschil is dus, dat het startadres, en dus alle andere adressen, een hogere waarde moet hebben. Willen we dus een dergelijk programma zowel geschikt maken voor de 32k als de 96k NewBrain, dan moeten we niet naar een absoluut adres POKEn, maar naar een relatief adres en het programma laten bepalen of het een 32k of 96k NewBrain betreft. Dit kan als volgt:

```
FOR i = 1 TO 255 : CLOSE #i : NEXT i : OPEN #0, 0
a = 642 : IF PEEK (51) > 127 THEN a = 32770 : POKE 147, 107
POKE a + 0, 65: REM A linksboven
```

of:

```
b = 2 * 128 : POKE a + b, 65 : REM A op regel 3 van een 80 kolomsscherm
```

of:

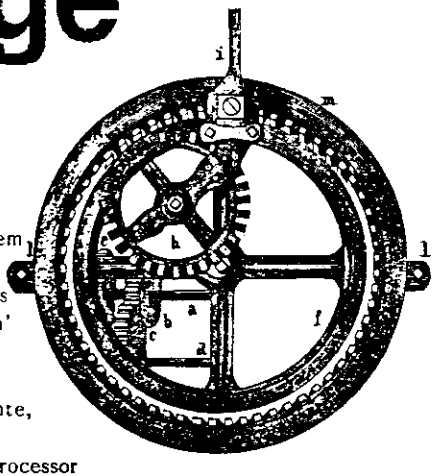
```
POKE a + 2 * 128, 65 : REM idem
```

Bovenstaande geldt voor een scherm ter grootte van het afgebeelde scherm (24 regels). Wat er moet gebeuren met schermen die groter dan 8k zijn, weet ik nog niet.

Wim Luijt

'Als je de opdracht POKE 147,107 geeft, wordt er een zogenaamde video-pagina in slot 4 gezet.'
Een beter te begrijpen omschrijving vind ik persoonlijk, dat met deze opdracht de adressen 8000h - 9FFFh gaan wijzen naar de eerste pagina van de video RAM. Bij het toevoegen van een device driver (zie bladzijde 51) vind ik namelijk, dat verschillende 8k blokken van adressen (slots) naar dezelfde RAM kunnen wijzen, met andere woorden een bepaald stuk RAM (of ROM) kan (of tenminste lijkt) gelijktijdig in verschillende slots voor te komen. Aangezien iedere pagina een stuk hardware is, dat zichzelf of zijn inhoud niet kan kopiëren, kan het zich nooit op meerdere plaatsen tegelijk bevinden.

system page



definities

OS: het non-paged operating system
POS: het paged memory operating system
NP: machines bestuurd door het OS
P: machines waar het POS in werking is
MMS: het 'memory management system'
IOS: in- en uitvoersysteem onder OS
PIOS: in- en uitvoersysteem onder POS
OBJECT: een hoeveelheid geheugenruimte, die door het MMS toegewezen is
COP: de COP-420 in- en uitvoermicroprocessor
PAGINA: een blok van 8k geheugenruimte. alle ROM en RAM is verdeeld in aansluitende pagina's, die elk een eigen uniek PAGINANUMMER hebben. het nummer is 12 bits groot, waardoor er twee bytes voor nodig zijn, waarvan de bits 12 - 15 0 zijn
SLOT: een van de acht gebieden van 8k aan z80-adressen, waaraan een pagina toegekend kan worden
SYSTEM PAGE: de pagina (gewoonlijk in slot 0) met essentiële informatie voor het systeem
PAGINANUMMER VOOR SLOT N: een getal van twee bytes, waar de bits 0 - 11 het paginanummer vormen en 13 tot 15 het slotnummer. bit 12 is 0
LOKATIE: het z80-adres dat gebruikt kan worden om bij een bepaald geheugengebied te komen, als de pagina waar het bij hoort, in het juiste slot geplaatst is
REGIO: een aantal aaneensluitende RAM-pagina's die door de page allocator toegewezen zijn met een soortgelijke bestemming
VIDEOADRES: een lokatie in de 32k video-RAM. bit 15 is meestal 1
UITGEREKTE VORM: de bytes van een element in een tabel liggen 256 bytes uit elkaar. eerst achter elkaar de eerste bytes van alle elementen, daarna alle tweede bytes

PARS

PARS is de naam van het gebied van 0 tot FFh, zowel in OS als POS. het is als volgt ingedeeld

0 DI: z80-instructie disable interrupt (F3h)
1 - 3 TPON: NP bevat de instructie JP PONINT (een adres in de EF-ROM). P bevat JP 0. dit is bestemd voor gebruik van newbrains met de mogelijkheid van power-down. het paged operating system ondersteunt deze mogelijkheid niet, evenmin als de hardware van model A en model AD
4 - 5 B3PRM: NP het adres van het begin van de RAM die beschikbaar is voor basic. het OS houdt dit bij, telkens als de ruimte voor de in- en uitvoerbuffer groter of kleiner wordt. P ter beschikking van het gebruikersprogramma; basic zet het in het begin op 2000h
6 - 7 B4: NP het adres van het einde van de RAM die voor basic beschikbaar is. P beschikbaar voor het gebruikersprogramma; basic zet het op C000h. wordt niet gewijzigd, maar alleen gelezen bij het starten van het gebruikersprogramma (cf het artikel 'room at the top' op pagina 26)
8 - A (8 - 10) TRST8: mag door elke module tijdelijk gebruikt worden (maths pack, cassettedriver). 8 bevat de opdracht JP (C3h) en mag niet veranderd worden
B (11) DEV0: het nummer van het default console device. mag door het gebruikersprogramma gebruikt worden
C (12) EXCESS: gebruikt in het operating system voor videodevices. de waarde 24 moet niet veranderd worden
D (13) TV0: klok voor het knipperen van de cursor; gebruikt door de routine voor de beeldinterrupts
E (14) TV2: statusvlag van schermuitlezing. NP als bit 7 een 1 is, wordt het scherm uitgelezen, bit 6 = cursor in beeld, bit 5 = cursor is onderstreping (ascii 95), niet het blokje (ascii 127), bit 4 = video alleen het volgende beeld uitgeschakeld, bits 3 - 0 ongebruikt. P bit 7 = cursor in beeld, bit 6 = cursor is onderstreping (geen blokje), bit 5 = video alleen het volgende beeld uitgeschakeld, bits 4 - 0 schakelen de video uit (momenteel wordt bit 0 gebruikt door de video-devicedrivers en bit 1 door de video memory allocator; de andere zijn ongebruikt)
F (15) TV1: bevat het teken op de plaats van de cursor
10 - 12 (16 - 18) TEXM: restart voor gebruik door het gebruikersprogramma (basic gebruikt het). de inhoud van 10 (C3h) moet niet veranderd worden
13 (19) DEV2: het nummer van het default opslagdevice (12 voor disk, 1 voor cassette 1; basic gebruikt het bij voorbeeld bij: LOAD "progr")
14 - 15 (20 - 21) SAVE2: NP voor tijdelijke opslag (van ix-register) door IOS. P ongebruikt
16 - 17 (22 - 23) SAVE3: NP idem (iy-register). P ongebruikt

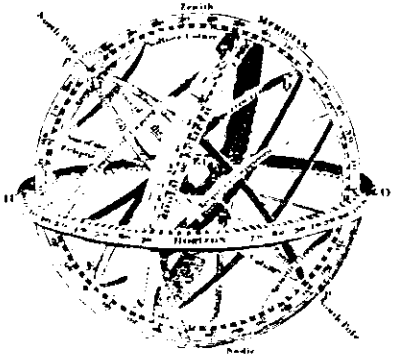
18 - 1A (24 - 26) TEXMNC: restart voor gebruik door het gebruikersprogramma (basic gebruikt het). de inhoud van 18 (C3h) moet niet veranderd worden
1B (27) PLLEN: regellengte van het device van stream 0 (is 0 als het device niet met vaste regellengte werkt)
1C (28) PHPOS: plaats van de printkop op de regel van het device van stream 0 (meest linkse positie is 1). onder basic op te vragen met de systeemvariabele POS. wordt alleen gebruikt, als PLLEN niet 0 is
1D (29) PZLEN: printzonelengte van het device van stream 0. wordt gebruikt om tabulatorstops vast te stellen, maar alleen, als PLLEN niet 0 is (cf het artikel 'printzone' op pagina 29)
1E (30) SAVE1: NP werkruimte voor operating system. P ongebruikt
20 - 22 (32 - 34) TRST32: restartlokatie voor system call
23 (35) BRKBUF: als het byte niet nul is, is de STOP-toets (en * bij het lezen van cassette) buiten werking gesteld
24 (36) ENREGMAP: het laatste byte dat naar het enable register 1 verstuurd is (cf hardwarebeschrijving)
25 (37) IOPUC: teller van het aantal in gebruik zijnde devices, die IOPOWER gebruiken
26 - 27 (38 - 39) UP: lokatie van het gebruikersprogramma. NP normaliter basic. P adres voor slot 6 van het programma met het main menu
28 - 2A (40 - 42) TRST40: restartlokatie voor system call (alleen als carryvlag = 0). 28 is de opdracht JP (C3h) en moet niet veranderd worden
2B (43) KBMODE: wordt gebruikt door KLOOK bij het decoderen van de signalen van het toetsenbord voor de streams naar de devices 0, 3 en 4 (die hebben allemaal dezelfde keyboard mode)
2C - 2D (44 - 45) GSPR: NP adres van de routine 'make space' voor het gebruikersprogramma. P ongebruikt
2E - 2F (46 - 47) RSPR: NP adres van de routine 'return space' voor het gebruikersprogramma. P ongebruikt
30 - 32 (48 - 50) TRST48: restart gereserveerd voor de routines die de interrupts afhandelen. 30 bevat JP (C3h)
33 (51) BUFLG: als bit 7 een 1 is, geeft dit aan, dat het paged operating system in werking is. de andere bits worden niet gebruikt, behalve bit 0 bij NP, dat een 1 is, als het IOS het videobeeld uitschakelt om buffers te kunnen verplaatsen
34 - 35 (52 - 53) DPSP: lokatie van de parameterstring van het default consoledevice

36 - 37 (54 - 55) DPSL: lengte van de parameterstring van het default consoledevice
38 - 3A (56 - 58) TINT: adres van de routine die de interrupts van interrupt mode 1 afhandelt. 38 moet C3h blijven

3B (59) COPCTL: bevestigingsbyte dat direct na de volgende COP-interrupt naar de COP gestuurd wordt
3C (60) COPST: COP-status. bit 7 is ongebruikt, bit 6 is een 1 als er een toetsaanslag wacht in COPBUFF die nog niet ingelezen is, bit 5 als geen opdracht wacht om naar de COP gestuurd te worden, bit 4 als er een cassettefout opgetreden is, bit 3 als het lezen van cassette met * afgebroken kan worden, bit 2 geeft de status van de stoptoets, bit 1 geeft een cassettestop aan en een lege batterij, bit 0 is 0
3D (61) COPBUFF: bevat de gegevens die van toetsenbord of cassette komen en cassettefoutnummers
3E - 4D (62 - 77) OUTBUFF tekst die in het line display uitgelezen wordt
4E - 4F (78 - 79) TIMBUFF niet gebruikt
50 - 51 (80 - 81) CHKSUM teller voor het controlegetal bij cassette
52 - 55 (82 - 85) CLOCK/CLACK: teller van de gewone COP-interrupts. wordt gebruikt voor het vaststellen van de randomgetallen. het laagste byte staat voorop
56 - 57 (86 - 87) STRTAB: NP adres van de tabel van geopende streams. P ongebruikt

58 - 59 (88 - 89) DEVTAB: NP adres van de tabel van device drivers. P ongebruikt
5A - 5B (90 - 91) TVCUR: NP adres van de cursorpositie. P video-adres van de cursorpositie
5C - 5D (92 - 93) TVRAM: NP adres van de IOS-buffer van het scherm dat naar de monitor uitgelezen wordt. P videoadres van het zichtbare deel van het scherm dat naar de monitor uitgelezen wordt
5E - 5F (94 - 95) OTHER1: niet gebruikt, maar wordt wel bijgesteld NP door het operating system als het wijst naar een buffer die verplaatst is, en P door de video memory allocator als het een videoadres is in een videogebied dat verplaatst is
60 - 61 (96 - 97) OTHER2: net als OTHER1
62 - 63 (98 - 99) IOSRAM: NP adres van het begin van de buffers in het IOS. P ongebruikt
64 - 65 (100 - 101) STRTOP: NP adres van het einde van de streamtable van het IOS. P ongebruikt
66 - 68 (102 - 104) TNMI: routine die de non-maskable interrupts afhandelt. 66 bevat C3h
69 - 6B (105 - 107) ICKLM: beeldinterruptteller. daardoor is het een nauwkeurige 50 Hz-klok. het byte met de grootste waarde staat voorop
6C (108) TBRP: default baudrate-parameter voor verzending. de baudrate is te berekenen door 19200 door de waarde van dit byte te delen. wordt soms door een aantal device drivers gebruikt
6D (109) RBRP: default baudrate-parameter voor ontvangst. net als TBRP
6E - 70 (110 - 112) IOPON: routine die de teller van IOPOWER-gebruikers

een verhoogt
71 - 73 (113 - 115) IOPOFF: routine die de teller van IOPOWER-gebruikers een verlaagt
74 - 75 (116 - 117) DEFNF: default formaat voor de uitvoer van getallen. de inhoud is een formaat zoals de OUT-routine in het maths pack nodig heeft
76 (118) STR11: streamnummer van de default stream voor high resolution graphics
77 - 78 (119 - 120) CHRROM: NP adres (P lokatie voor slot 1) van de tabel waarin de vorm van de letters vastgelegd is. wordt gebruikt door de routines die tekst afbeelden op een grafisch scherm



de volgende adressen worden alleen door het paged operating system gebruikt. in unexpanded machines is 79 - FF (121 - 255) dus vrij te gebruiken

79 - 7A (121 - 122) DISCBUFFER: lokatie van het disc command block, voor het slot, waarin het geacht wordt te staan
7B - 7D (123 - 125) INTSERVICE: routine, die de niet-COP-interrupts afhandelt (in de alternatieve slots)
7E - 80 (126 - 128) GETOB: routine die het opgegeven MMS-object opzoekt en in de slots 1 en 2 zet
81 - 83 (129 - 131) IGETOB: idem met index in het object
84 - 86 (132 - 134) GETVIDEO: routine die het opgegeven videogebied opzoekt en in de slots 4 - 7 zet
87 - 88 (135 - 136) PARAMLEN: tijdelijke werkruimte voor PIOS
89 - 8A (137 - 138) IOSTEMP: tijdelijke werkruimte voor PIOS
8B - 9A (139 - 154) S0, S1, S2, S3, S4, S5, S6 en S7: Sn bevat het paginanummer voor slot n van de pagina die op dat moment in slot n zit
9B - 9E (155 - 158) ENDOFOBJECTS: het MMS gebruikt dit om het echte adres van het einde van het laatste object te onthouden, relatief ten

opzichte van het begin van de regio
9F - A0 (159 - 160) VIDEOTOP: paginanummer van het einde van de videopagina's in RAM
A1 - A2 (161 - 162) VIDEOBASE: paginanummer van de eerste videopagina
A3 - A4 (163 - 164) AVIDEOTOP: adres van het einde van het laatste gebruikte videogebied, relatief ten opzichte van het begin van het videogebied
A5 - A6 (165 - 166) UPPAGE: paginanummer voor slot 6 van de pagina met het programma van het main menu
A7 (167) TV3: bevat het byte dat naar het 'tv control register' verstuurd moet worden voor de schermuitlezing (zie hardwarebeschrijving)
A8 - A9 (168 - 169) ESYSPAGE: lokatie van het eerste lege byte op de system page
AA - AC (170 -172) DISC: routine die schrijfbewerkingen start
AD (173) OLDOS: geeft het versie nummer van het non-paged operating system aan. 0 = 1.4, 1 = 1.9, 2 = 1.91, 3 = 1.9 (frans), 4 = 2.0
AE - AF (174 - 175) CHRPAGE: paginanummer voor slot 1 van de pagina met de de tekendefinities voor gebruik door de routines die tekst op een grafisch scherm afbeelden
B0 - B1 (176 - 177) INABROM: paginanummer van de interne AB-ROM voor slot 5
B2 - B3 (178 - 179) INCDROM: paginanummer van de interne CD-ROM voor slot 6
B4 - B5 (180 - 181) INEFROM: paginanummer van de interne EF-ROM voor slot 7
B6 (182) ENREG2MAP
B7 (183) INTFLGS
B8 - B9 (184 - 185) BILL
BA (186) TV4: nummer van het videogebied dat op dat moment uitgelezen wordt
BB - BC (187 - 188) PUSER: aantal benodigde pagina's in regio 0
BD (189) A16COUNT: teller van het gebruik van de alternatieve slots
BE - C0 (190 - 192) ENTERLIBRARY: routine die toegang verschaft tot externe library-routines
C1 - C3 (193 - 195) INTBUFFER
C4 - C6 (196 - 198) GETBUFFER
C7 - D6 (199 - 214) BOXTAB
D7 - D9 (215 - 217) SERVECOPINT: routine die COP-interrupts afhandelt
DA - DB (218 - 219) LOWBUFFERPAGE
DC - DD (220 - 221) HIGHBUFFERPAGE
DE (222) BUFMARGIN
DF - E0 (223 - 224) V2INTBUFFER
E1 - E3 (225 - 227) DISCSTATUS

de vast toegekende geheugenruimte loopt bij unexpanded machines van 100h tot 267h (256 - 615); bij expanded van 100h tot 1FFFFh (256 - 8191). de indeling is als volgt

100 - 19F (256 - 415) FPRAM: werkruimte voor de rekenmodule (maths pack)
1A0 - 4FF (416 - 1279) STKRAM: in gebruik als hardware stack; bij unexpanded machines is het laatste byte voor de stack 267h (615). de rest van deze lijst geldt dus alleen voor machines met het paged operating system
500 - 5FF (1280 - 1535) MOPARS: uitbreiding voor de PARS (0h - FFh). in het begin op nul gesteld
600 - 6FF (1536 - 1791) STATUSTABLE: een byte voor elke stream. byte n in de tabel hoort bij stream n en wordt bijgehouden door de device driver, die bij die stream hoort. het is 0, als de stream gesloten is, of als de device driver geen gegevens verstrekt (en natuurlijk ook als toevallig alle bits 0 zijn). bit 0 (IEST) = invoerfout (het volgende verzoek om invoer van de stream heeft een foutmelding tot gevolg); bit 1 (OEST) = uitvoerfout (bij uitvoer naar de stream volgt er een foutmelding); bit 2 (NRIST) is een 1 als er geen gegevens op invoer staan te wachten; bit 3 (NROST), als de stream nog geen uitvoer kan accepteren; bit 4 (EOFST) geeft bestandseinde aan (bij een poging tot invoer van de stream volgt de foutmelding, dat bestandseinde bereikt is). de bits 5 - 7 zijn nog niet in gebruik. sommige device drivers verstrekken niet alle gegevens
700 - AFF (1792 - 2815) TABLEZERO: deze tabel is de basis van de gegevens-structuur van het MMS en bevat de echte adressen van secundaire tabellen. zie de documentatie van het MMS voor details
B00 - CFF (2816 - 3327) STREAMTABLE: tabel in uitgerekte vorm van alle streams. de twee bytes die bij een stream horen (voor stream #0 B00 en C00, voor stream #1 B01 en C01), geven het nummer van het bufferobject. het eerste byte van het object bevat het devicenummer, het tweede het poortnummer en de rest is de eigen geheugenruimte van de stream. als het objectnummer 0 is, geeft dat aan, dat de stream niet open is
D00 - DFF (3328 - 3583) PARAMAREA: dit gebied wordt door PIOS en FINDLIBRARY gebruikt als werkruimte
E00 - 11FF (3584 - 4607) PDEVTAB: de tabel van device drivers. in uitgerekte vorm. de eerste twee bytes die bij een driver horen, geven telkens het paginanummer voor slot 4 van de device driver en de tweede de lokatie ervan
1200 - 15FF (4608 - 5631) ZPTABLE: de tabel met de systeemroutines. ook in uitgerekte vorm. de eerste twee bytes die bij een element horen, geven telkens het paginanummer voor slot 7 van de device driver en de tweede de

lokatie ervan
1600 - 167F (5632 - 5759) VIDTABLE: de lijst van de videoadressen en de lengtes van de 32 mogelijke videogebieden. de eerste twee bytes geven telkens de videoadressen en de volgende twee de lengte. het aantal bytes, dat een videogebied beslaat, is altijd een veelvoud van 128
1680 - 169F (5760 - 5791) ENTRYAVAILABLETABLE: dit is een hulptabel voor het MMS in de vorm van een 'bit map'. de boel is zo ingericht, dat de bits 0 van de bytes gebruikt worden voor de nummers 0 tot 31, de bits 1 voor 32 tot 63 enzovoort. als een bit 1 is, betekent dat, dat de secundaire MMS-tabel, waarmee het correspondeert, bestaat en dat daar nog ruimte is. zie de documentatie van het MMS
16A0 - 1701 (5792 - 5889) REGIONTABLE: voor elke regio zijn er zes bytes beschikbaar: de eerste twee bevatten het nummer van de eerste pagina van de regio (het einde van de regio blijkt uit het eerste paginanummer van de volgende regio), de volgende twee het paginanummer (voor slot 4) van de region manager (als het hoogste bit 0 is, betekent dat, dat er geen is), en de laatste twee zijn de lokatie van de region manager. de laatste twee bytes van REGIONTABLE (REGTOP, 1700 en 1701) zijn een loze vermelding om het einde van regio 15 aan te geven. regio 0 is het gebruikersprogramma, regio 1 is het DISCIO-systeem, het MMS gebruikt regio 8. regio 255 (de videopagina's) staat niet in de tabel
1702 - ESYSPAGE-1 (5890 - in het begin 6406) SLOTZEROCODE: in deze ruimte staan diverse belangrijke systeemroutines (bij voorbeeld GETOB vanaf 17AC, IGETOB vanaf 17E4, GETVIDEO vanaf 182F, ENTERLIB vanaf 176E). als er een nieuwe aan toegevoegd wordt, moet die beginnen op het adres ESYSPAGE (welk dat is, staat in A8 - A9), en moet in ESYSPAGE genoteerd worden, waar nu de vrije ruimte op de zero page begint.

(bewerking: menno stevens
met dank aan wim luijt)

S-P-N

S-P-N EN ANDERE GETALLEN VOOR DE 96K NEWBRAIN

Bij het werken met het paged Memory operating system op de 96k NewBrain is het noodzakelijk om het gecombineerde slot-pagina-nummer (S-P-N) te kunnen berekenen. Een S-P-N bestaat uit 2 bytes, of 16 bits. We kunnen dit 16-bitsgetal verdelen in drie stukken:

- het eerste stuk van 3 bits is het slot-nummer (bits 15 - 13)
- het volgende bit heeft geen naam en is altijd 0 (bit 12)
- het derde stuk is 1 bit, heeft geen naam, is voor een RAM-pagina 0 en voor een ROM-pagina 1 (bit 11)
- het vierde stuk is 11 bits: het paginanummer (bits 10 tot 0)

Als we deze vier stukken in bovenstaande volgorde achter elkaar zetten, krijgen we twee getallen, die we zowel binair, hexadecimaal als decimaal kunnen schrijven. Het duidelijkste is dit te demonstreren aan de volgende voorbeelden.

(lees verder op pagina 76)

slot	adres (hex)	adres (dec)	zeropage-adres S0 t/m S7
0	0000-1fff	0- 8191	139 (8bH)
1	2000-3fff	8192-16383	141 (8dH)
2	4000-5fff	16384-24575	143 (8fH)
3	6000-7fff	24576-32767	145 (91H)
4	8000-9fff	32768-40959	147 (93H)
5	a000-bfff	40960-49151	149 (95H)
6	c000-dfff	49152-57343	151 (97H)
7	e000-ffff	57344-65535	153 (99H)

TOELICHTING BIJ DE TABEL OP PAGINA 74 - 75

In elk vakje staan onder elkaar: RAM of ROM; welke software het is; welke hardware het is; en het paginanummer. nb = newbrain; dc = disk controller; eim = expansion interface module. Over de plaats van pagina 124 met de driver voor i/o naar een schermdevice is de informatie niet eensluidend: volgens Paris-Micro in slot 3, volgens SYSPAGE.COM in slot 4.

slot 0	slot 1	slot 2	slot 3	slot 4	slot 5	slot 6	slot 7	
ram zero page eim 96 (60H)	ram basic eim 97 (61H)	ram basic eim 98 (62H)	ram basic eim 99 (63H)	ram basic eim 100 (64H)	ram basic eim 101 (65H)	rom basic nb ?	rom zcalls nb of eim 120 (78H) nb 123-125 (7b-7dH) eim	basic RUN
ram zero page eim 96 (60H)	ram basic eim 97 (61H)	ram basic eim 98 (62H)	rom driver eim 124 (7cH)	ram/rom video/basic nb/eim 104 (68H)/ 100 (64H)	ram/rom video/basic nb/eim 105 (69H)/ 101 (65H)	ram/rom video/basic nb/eim 106 (6aH)/ ?	ram video nb 107 (6bH) ?	basic-i/o naar device 0, 3, 4, 11 of 33
			afhankelijk van de lengte van het scherm					
ram zero page eim 96 (60H)	ram basic eim 97 (63H)	ram video nb 107 (6bH)	ram video nb 106 (6aH) 107 (6bH)	ram video nb 105 (69H) 106 (6aH) 107 (6bH)	ram video nb 104 (68H) 105 (69H) 106 (6aH) 107 (6bH)	ram video nb 106 (6aH) 106 (6aH) 106 (6aH) ? (rom nb)	ram video nb 107 (6bH) 107 (6bH) 107 (6bH) 107 (6bH)	basic i/o naar device 0, 3, 4, 11 of 33 met POKEn op 50 t/m 57
		afhankelijk van de lengte van het scherm						
ram zero page eim 96 (60H)	ram basic eim 97 (63H)	ram basic eim 98 (62H)	ram basic eim 99 (63H)	rom driver dc 112 (70H)	ram basic eim 101 (65H)	rom basic nb ?	rom intercept? eim ?	basic-i/o naar device 12 - 15, 18 - 19

ram zero page eim 96 (60H)	ram basic eim 97 (63H)	ram basic eim 98 (62H)	ram basic eim 99 (63H)	rom driver eim 125 (7dH)	ram basic eim 101 (65H)	rom basic nb ?	rom intercept? eim ?	basic-i/o naar device 5 - 10
ram zero page eim 96 (60H)	ram basic eim 102 (66H)? device buffer	ram basic eim 103 (67H)?	ram basic eim 99 (63H)	rom driver eim 125 (7dH)	ram basic eim 101 (65H)	rom basic nb ?	rom intercept? eim ?	basic-i/o naar device 1, 2, 16 - 17, 20 - 23
ram zero page eim 96 (60H)	ram basic eim 97 (61H)	ram basic eim 98 (62H)	ram basic eim 99 (63H)	ram basic eim 100 (64H)	ram basic eim 101 (65H)	rom basic nb ?	rom ? nb of eim ?	basic gedurende zcall
ram 0-pag|tpa eim 97 (61H)	ram tpa eim 104 (68H)	ram tpa eim 103 (67H)	ram tpa eim 102 (66H)	ram tpa eim 101 (65H)	ram tpa eim 100(64H)	rom tpa eim 99 (63H)	rom bdos-pseudobios eim 98 (62H)	cp/m uitvoering routine in tpa
ram zero page eim 96 (60H)	ram tpa eim 104 (68H)	ram tpa eim 103 (67H)	rom driver eim 124 (7cH)	ram video nb 104 (68H)	ram video nb 105 (69H)	ram video nb 106 (6aH)	ram video nb 107 (6bH)	cp/m conout of edit
ram zero page eim 96 (60H)	ram eim 104 (68H) buffer	ram eim 103 (67H)	ram tpa eim 102 (66H)	rom driver eim 125 (7dH)	ram tpa eim 100(64H)	rom tpa eim 99 (63H)	rom intercept? eim ?	cp/m list

voorbeeld 1
gevraagd: het S-P-N voor ROM-pagina 124 in slot 3
berekening: 3d1 = 3h = 011b, 124d = 7Ch = 0111 1100b

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	slot			0	type	pagina										
binair	0	1	1	0	1	0	0	0	0	1	1	1	1	1	0	0
hexadecimaal	6				8				7				C			
decimaal	104								124							

voorbeeld 2
gevraagd: het S-P-N voor RAM-page 107 in slot 5
berekening: 5d = 5h = 101b, 107d = 6Bh = 0110 1011b

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	slot			0	type	pagina										
binair	1	0	1	0	0	0	0	0	0	1	1	0	1	0	1	1
hexadecimaal	A				0				6				B			
decimaal	160								107							

Met behulp van de beschikbare informatie heb ik de bijgaande tabel samengesteld. Ze bevat de volgende informatie:

- S0 tot en met S7
- een aantal paginanummers
- in welk slot welke pagina's zijn bij verschillende acties
- waar de hardware zich bevindt
- of het ROM of RAM betreft

WAARSCHUWING: de tabel is naar beste weten samengesteld. Hij is niet volledig en ik acht het, gezien mijn gebrekkige kennis op dit gebied, niet uitgesloten, dat er fouten inzitten. Verbeteringen zijn daarom zeer welkom.

Wim Luijt

cp/m

na zijn inleiding in de vorige on-line (nr 7, pagina 67) vervolgt henk blik de artikelenserie over cp/m met een uitleg over het basic disk operating system (BDOS)

Welke BDOS-functies er allemaal zijn, ga ik hier niet uitleggen; dat is wel terug te vinden in de cp/m manuals (waarom niet geleverd bij de NewBrain disk controller?) of in andere boeken *).
Het BDOS zorgt voor het manipuleren van files op schijf. Om files te kunnen vinden op schijf houdt het BDOS een administratie bij in de

*) Jan Wilmink, CP/M Operating System (Kluwer)
Andy Johnson-Laird, The Programmer's CP/M Handbook (McGraw-Hill)
J. W. Meerman e.a., Introductie in CP/M (verkrijgbaar via de HCCBestelservice)

directory tracks van de schijf. Voor iedere file is er minstens een entry; voor grotere files zijn er twee of meerdere entries.

Een entry is als volgt opgebouwd:

u	f i l e n a m e	e x t	ex	x1	x2	rc	disk allocation map
0	1	8	9 11	12	13	14	15 16 17 30 31

u is het user number voor de file (0 - 31) of e5h, als de file gewist is.
Filename is de naam van de file en ext is de type-aanduiding voor de file.
ex = extend nummer. Telkens als er 128 sectoren gebruikt zijn, wordt het extendnummer een verhoogd. De eerste extend heeft altijd nummer 0. Als een file meerdere directory entries heeft, worden de extends doorgeteld. De volgende entry heeft altijd een hoger extendnummer.
x1 en x2 zijn gereserveerd, er staat 0 in.
rc = record count, dit is het aantal records (logische of cp/m-sectoren van 128 byte) van de laatste extend die gebruikt zijn.
In de disk allocation map staan de blocknummers die gebruikt zijn voor de file. Deze blocknummers zijn of 1 byte groot, dan staan er 16 blocknummers (NewBrain) of ze zijn twee bytes groot, dan staan er dus 8 blocknummers met het meest significante byte als laatse (CB-80).

Een file bestaat uit een aantal cp/m-sectoren die voor de BDOS-administratie zijn gebundeld in blocks. Het aantal sectoren per block wordt bepaald door het Disk Parameter Block (DPB), dat zich in het BIOS bevindt. Ik geef hier het voorbeeld van een DPB voor een NewBrain met 800k-drives.

DPB:

```
        dw  28h         ;logische sectoren per track
        db  05h         ;block shift
        db  1fh         ;block mask
        db  03h         ;extend mask
        dw  c4h         ;maximale blocknummer (= aantal blocks - 1)
        dw  7fh         ;aantal directory entries - 1
        db  1000000b    ;bit map: gebruikte blocks voor
        db  0000000b    ;         directory
        db  20h         ;aantal bytes in directory check buffer
        db  02h         ;aantal gereserverede tracks
```

Deze waarden worden door het BDOS gebruikt om de blockgrootte te bepalen, het aantal blocks, en op welke track block 0 begint.

sectors per block = block mask + 1 = 2 ^ block mask
voor de extend mask geldt:

```
if (aantal blocks < 256)
    extend mask = (sectors per block) / 8 - 1
else  { aantal blocks >= 256) }
    extend mask = (sectors per block) / 16 - 1
```

De combinatie van blocks van 1kb (8 logische sectoren) en meer dan 255 blocks is dus onmogelijk, de extend mask zou dan immers negatief zijn. Gezien de samenhang van deze waardes moeten deze waardes op elkaar afgestemd zijn, anders raakt het BDOS de kluts kwijt.
Verder geldt, dat block 0 begint na de reserved tracks, waaruit volgt, dat de directory in block 0 (en eventueel verder) ligt. Een waarde van 0 in de disk allocation map voor een file geeft aan dat deze locatie niet in gebruik is. Gewoonlijk is de disk allocation map aaneensluitend gevuld maar bij random access files kunnen er gaten in vallen. We laten random access files hier even buiten beschouwing en gaan uit van gewone files. Bij deze files geeft het eerste blocknummer dat 0 is, aan, dat het einde van de file is bereikt. Alle aangegeven blocks worden helemaal gebruikt behalve het laatste block in de allocation map. Het aantal sectoren die hiervan gebruikt worden wordt gegeven door:

sectors (last block) = rc modulo (sectors per block)

Met deze informatie kunnen we dus ook bepalen, hoe groot een file precies is.

Voor het berekenen van de plaats van een block op de schijf geldt:

track = reserved tracks + int (blocknummer * (sectors per block) / (sectors per track))
sector = blocknummer * (sector per block) modulo (sectors per track)

Het BDOS gebruikt tijdens file access een zogenaamd File Control Block (FCB) om bij te houden, waar het bezig is met de file. Dit FCB ziet er als volgt uit:

d	filename	ext	ex	x1	x2	rc	disk alloc map	nr	r1	r2	ro
0	1 8	9 11	12	13	14	15	16 31	32	33	34	35

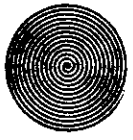
Een FCB moet worden gemaakt door het programma dat een file wil gaan gebruiken. Het FCB vertoont sterke overeenkomsten met de directory entries, maar er zijn een aantal verschillen:

d = drive nummer (0 = default, 1 = A, 2 = B, . . .)
nr = volgende sector in block om te lezen of te schrijven
r1, r2 = random record te lezen of te schrijven
ro = random overflow, wordt gebruikt om aan te geven dat een file 8 Mbytes groot is (maximum)

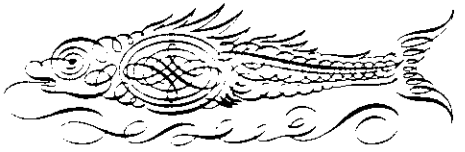
De gebruiker moet bij het begin van de file de velden ex, x1, x2, rc, disk allocation map, nr en ro op nul zetten, het BDOS zal zelf de relevante informatie uit de directory in het FCB invullen. Bij het random lezen en schrijven kan de gebruiker in de velden r1 en r2 aangeven welke record hij wil lezen of schrijven.
Het veld nr wordt door het BDOS gebruikt om aan te geven, wat de volgende sector is, die moet worden gelezen of geschreven. Wat in x1 en x2 staat is niet gedocumenteerd.

Tot zover het verhaal over het functioneren van het BDOS, in een volgende aflevering zal ik iets vertellen over het BIOS.

H. J. Blik



wp(demo?)



INSERT EN CHANGE GEKOPPELD

bij het programma wp is er voor full screen tekstredactie de keuze tussen de opdrachten c(hange characters) en i(nsert/remove characters). maar er ontbreekt een handige insert-toets, waardoor je van overschrijven naar tussenvoegen kunt overschakelen. nu moet je van te voren kiezen, of er alleen letters veranderd worden, of dat er ook tussen moeten

wie schetst mijn verbazing (tekening insturen naar de redactie), toen ik bij het bekijken van het programma ontdekte, dat de opdrachten change en insert goeddeels door hetzelfde programmagedeelte uitgevoerd worden: na de mededeling op het scherm, welke opdracht gekozen is, wordt er een byte gezet overeenkomstig die keuze, en dan is het programma verder gelijk. in beide gevallen wordt na elke toetsaanslag in dat ene byte gekeken, of insert of change de bedoeling was, en op grond daarvan gaat het programma verder

op die manier moet het zeer wel mogelijk zijn om zonder naar de command mode terug te gaan van het een naar het ander over te schakelen. wat toegevoegd moet worden, is een kleine routine, die het signaal van de insert-toets onderschept en dan het beslissende byte (21c1) verandert. dat is wel te maken, maar de ruimte om het neer te zetten moet er ook zijn. nu heb ik die gelukkig gevonden

als de schijf vol is, geeft het programma de mededeling 'sorry but your disk is full'. nu heb ik een hekel aan zulke gevoelige computers en ben ik zeer tevreden met de eenvoudige mededeling 'disk is full'. dat levert dan precies de vijftien bytes op, die voor de onderschepping van insert nodig zijn

dan is hier de routine, die in plaats van de tekst in het programma gezet moet worden, te beginnen bij adres 3618:

```
3618   fe 11       cp control-q ;of, als de insert-toets een andere
                                ;waarde afgeeft, die hier invullen
361a   c2 64 0f    jp nz, f64   ;geen insert, dan verder met programma
361d   21 c1 21    ld hl, 21c1  ;het byte, dat over insert of change beslist
3620   3e 01       ld a, 1
3622   96          sub (hl)     ;1 min het byte geeft de andere waarde
3623   77          ld (hl), a   ;berg de nieuwe waarde op
3624   c3 64 0f    jp f64       ;verder met het programma
```

als dit in het programma gezet is, zijn we er nog niet: het nieuwe addres voor de mededeling, dat de schijf vol is, moet vastgelegd worden: op adres 360b de 18 in 27 veranderen.
en het programma moet zich elke keer gewillig laten onderscheppen. daartoe verandere men op adres cc7 en volgende de sprongopdract fa 64 0f in fa 18 36

als dat klaar is, functioneert de insert-toets als een zogenaamde toggle switch: in insert heeft het aanslaan van de insert-toets als gevolg de overgang naar change, en in change de overgang naar insert. het is me een beetje te ingewikkeld om ook de omschrijving van de opdracht, die onderin het scherm staat, te veranderen. ik vind het al mooi, dat ik niet telkens uit de editor moet om er met de andere opdracht weer in te gaan

menno stevens

OOK REGELNUMMERS DOOR DE COMMS-UITGANG

in het vorige nummer van on-line (pag. 73) heb ik gezegd, dat met een simpele ingreep wp de boel naar de comms-poort stuurt in plaats van naar de printerpoort. bij de opdracht list file functioneert het uitstekend, maar bij de opdracht type lines resulteert het in de potsierlijke situatie, dat de regels zelf naar de comms-poort gaan, maar de regelnummers nog steeds naar de printerpoort. dat is te verbeteren door ook op adres 252c 05 te veranderen in 04. dan is het volgens mij echt goed ...

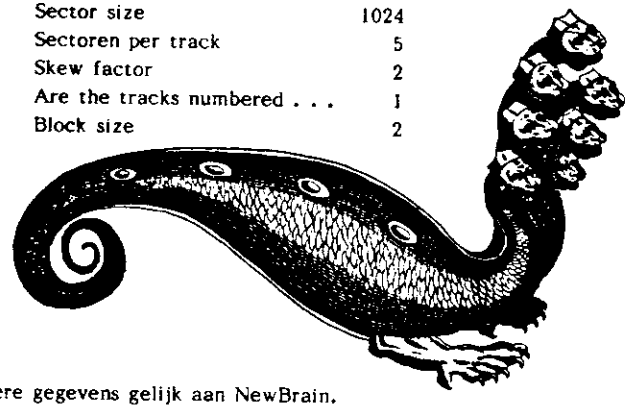
formaten

MEERDERE FORMATEN OP EEN NEWBRAIN

Wanneer men op de NewBrain ook regelmatig Proton-schijven gebruikt, kan het handig zijn om twee FDD's op Proton-formaat te hebben. Maar daarbij wil men de NewBrain-formaten niet missen. Men configureert dan vier drives, A: en B: als NewBrain-drive, C: en D: als Proton-drive. Bij C: physical drive number 0 opgeven, en bij D: een 1.

Configuratiegegevens voor Proton-drives:

Sector size	1024
Sectoren per track	5
Skew factor	2
Are the tracks numbered . . .	1
Block size	2



Alle andere gegevens gelijk aan NewBrain.
Voor de seek-rate kan men het beste 2 ms kiezen (TEAC FD-55F).

Wanneer men met deze configuratie het systeem opstart, zal de computer 10 tot 15 seconden wachten: 'zoeken' naar de derde en vierde drive, die er niet zijn. Gedurende deze tijd probeert hij de koppen van C: en D: op track 0 te zetten. Hij geeft daartoe pulsen aan de drive, totdat hij van de drive een signaal ontvangt op de 'track 0'-lijn, wat hij nu niet krijgt.
Dit is heel simpel te imiteren: een diode van de 'track 0'-lijn (26) naar de

'DS 2'-lijn (14) en een diode van de 'track 0'-lijn (26) naar de 'DS 3'-lijn (6) (kathode aan DS); en een condensator van circa 10 nF tussen 'DS 2' en massa, en tussen 'DS 3' en massa.
Nu krijgt de controller zijn 'track 0'-indicatie en gaat direct door met het opstarten; sneller zelfs dan met vier disk-drives, want de koppen hoeven niet meer naar track 0 gezet te worden.
Het mooiste is om deze vier onderdelen op een flatcable-connector te solderen, en deze op de flatcable te 'prikken'.

Dit gaat natuurlijk ook op voor andere formaten voor C: en D:. Ook kan dit gebruikt worden om met slechts een drive te werken, wanneer de systeemtracks aangeven dat er twee drives zijn, als men de configuratie om een of andere reden niet wil wijzigen. In dit laatste geval moet men dan ook een diode naar 'DS-1' leggen.

Eventueel kan ik een blokje leveren, dat men zo op de flatcable kan prikken.

Evert Drijver
telefoon (050) 778415

inhoud on-line 8

2 lijst van medewerkers
4 actief /m s vreedenburg/
5 aanwinsten van de softwarebibliotheken
nbug-software
cp/m-bibliotheek
onderwijsbibliotheek
21 vraagjes en weetjes - vragen en weeën
verbetering voor sscc /rob maris/
nauwkeurige stergegevens /t l van heummen/
fout in szap /menno stevens/
modificaties aan dip.com /evert drijver/
cpm64 van wijlen computex /t l van heummen/
hi-res onder pascal op computex-uitbreiding? /d h jas/
expansion aangevuld /wim luijt/
drijver drijft drives /evert drijver/
nogmaals resetknop /rob maris/
waarschuwing /wim luijt/
25 wat is wat /wim lujt/
26 room at the top /wim luijt/
29 printzone /wim luijt/
31 tokens, listsave, no-sort en no-edit /wim luijt/
37 basicode: gebruikerstips /drs eef tobé/
41 commentaar op 'ldcode' van frans van der velden /wim luijt/
45 wanneer is een machinetaalroutine voor een 32k newbrain ook te gebruiken op een expanded newbrain? /wim luijt/
51 het toevoegen en veranderen van een device driver zowel bij de 32k als de 96k newbrain /wim luijt/
60 poken in het scherm /wim luijt/
65 system page
73 s-p-n en andere getallen voor de 96k newbrain /wim luijt/
77 cp/m (aflevering 2) /h j blik/
81 wp(demo?): insert en change gekoppeld /menno stevens/
83 meerdere formaten op een newbrain /evert drijver/

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

softwarebibliotheek

BESTELFORMULIER

naam

adres

postcode en woonplaats

telefoon

hcc-lidmaatschapsnummer

bestelt bij deze de aan ommezijde aangegeven software

___ bestelnummers à f 15,00 = f ___

___ bestelnummers à f 25,00 = f ___

___ bestelnummers à f 10,00 = f ___

TOTAAL f ___

bovenstaand totaalbedrag

☐ is overgemaakt d d ___

op postrek 2505800 tnv hcc newbrain-gebruikersgroep utrecht

afkomstig van bank-/postrek nr ___

tnv ___ te ___

☐ wordt voldaan met bijgesloten girobetaalkaart, betaal- of eurocheque

nummer giropas of betaalpas ___

handtekening

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

softwarebibliotheek

formaat aankruisen:

- ☐ cassette
- ☐ diskette 200k
- ☐ diskette 400k ss
- ☐ diskette 400k ds
- ☐ diskette 800k
- ☐ _____

- __ ASM-1
- __ DIVERS-1
- __ DIVERS-2
- __ DIVERS-3
- __ EDUC-1
- __ EDUC-2
- __ FIN-1
- __ GRAFI-1
- __ HULP-1
- __ HULP-2
- __ MATH-1
- __ NBFILES
- __ ONLINE-5
- __ ONLINE-6
- __ ONLINE-7
- __ ONLINE-8
- __ SPEL-1
- __ SPEL-2
- __ SPEL-3
- __ SPEL-4
- __ SPEL-5
- __ SPEL-6
- __ VIDITEL-1

- __ NBUG-17
- __ NBUG-18
- __ NBUG-20 *)
- __ NBUG-20A **)
- __ NBUG-20B **)
- __ NBUG-26
- __ NBUG-33 *)
- __ NBUG-34 *)
- __ NBUG-35 *)
- __ NBUG-36
- __ NBUG-37
- __ NBUG-40
- __ NBUG-41 *)
- __ NBUG-41A **)
- __ NBUG-41B **)

onderwijsbibliotheek

- __ SCHIJF-1
- __ SCHIJF-2
- __ SCHIJF-3
- __ SCHIJF-4
- __ SCHIJF-5
- __ SCHIJF-6
- __ SCHIJF-7

cp/m-bibliotheek *)

- __ NBGG-1
- __ NBGG-2
- __ NBGG-3
- __ NBGG-4
- __ NBGG-5
- __ NBGG-6

__ catalogus-diskette (ƒ 10,-)

andere volumes uit de cp/m-bibliotheek

- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____
- __ _____

*) alléén verkrijgbaar op diskette
**) alléén verkrijgbaar op cassette

PRIJSLIJST

prijs per deel	cassette	disk 200k	disk 400k ss/ds, 800 k
softwarebibliotheek	ƒ 15,00	ƒ 15,00	ƒ 15,00
onderwijsbibliotheek	ƒ 25,00	ƒ 25,00	ƒ 25,00
cp/m-bibliotheek *)		ƒ 25,00	ƒ 15,00

*) per 10 delen in één bestelling 1 deel gratis

naam

straat

postcode en woonplaats

telefoon

computerconfiguratie

○ geeft zich op als lid van de HCC en de NewBrain-gebruikersgroep

○ is al lid van de HCC, lidmaatschapsnummer
en geeft zich op als lid van de NewBrain-gebruikersgroep

○ geeft zich op voor regelmatige toezending van NEWBRAIN ON-LINE
beëindiging van dit abonnement geschiedt door schriftelijke opzegging

datum handtekening

AANMELDINGSKAART

NewBrain-
gebruikersgroep

naam

straat

postcode en woonplaats

telefoon

computerconfiguratie

○ geeft zich op als lid van de HCC en de NewBrain-gebruikersgroep

○ is al lid van de HCC, lidmaatschapsnummer
en geeft zich op als lid van de NewBrain-gebruikersgroep

○ geeft zich op voor regelmatige toezending van NEWBRAIN ON-LINE
beëindiging van dit abonnement geschiedt door schriftelijke opzegging

datum handtekening

AANMELDINGSKAART