All Elektuur Junior articles 1982

1-58 EPROMmer (P.R. Boldt)

3-47 Junior spreekt Basic, adapt KIM KB-9 Basic to Junior

4-46 mini-EPROM-kaart

4-50 dynamische RAM-kaart

5-39 Software-uitpluizer (disassembler)

5-62 PSS Prive Software Service (EPROMmer)

5-67 Mini-teller met microprocesssor

7-59 Junior-vektoren ophalen (R. Mattysek)

7-46 single-cycle voor junior-computer (E.Kytzia)

10-59 Van 6502 naar 6809

11-58 Floppy-disk interface voor junior en andere 6502-computers deel 1

12-26 floppy-disk interface deel 2 (Ohio Scientific DOS)

```
0010: 0200                    ORG   $0200
0020:
0030:                  DATE :  10-7-'81
0040:
0050:
0060:                  PAGE ZERO DATA BUFFERS :
0070:
0080: 0200             SAL     *     $0000  DATA BLOCK START ADDRESS
0090: 0200             SAH     *     $0001
0100: 0200             EAL     *     $0002  DATA BLOCK END ADDRESS + 1
0110: 0200             EAH     *     $0003
0120: 0200             MOVL    *     $0004  EPROM PROGRAM START ADDRESS
0130: 0200             MOVH    *     $0005
0140:
0150: 0200             INH     *     $00F9  DISPLAY BUFFER   ( DATA )
0160: 0200             POINTL  *     $00FA     "        "    ( ADDRESS L )
0170: 0200             POINTH  *     $00FB     "        "    ( ADDRESS H )
0180:
0190:                  EXTERNAL SUBROUTINES :
0200:
0210: 0200             GETBYT  *     $1D6F
0220: 0200             SCAND   *     $1D88
0230:
0240:                  JUNIOR MONITOR START :
0250:
0260: 0200             RESET   *     $1C1D
0270:
0280:
0290:                  PROGRAM START ADDRESS     : $0200   ( PROG )
0300:                  DUPLICATE START ADDRESS   : $0222   ( DUPL )
0310:                  VERIFY START ADDRESS      : $0233   ( VERIFY )
0320:
0330:
0340:                  ****************
0350:                  EPROM-PROGRAMMER
0360:                  ****************
0370:
0380: 0200 20 55 02  PROG    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0390: 0203 A0 00     PRGR    LDYIM $00     CLEAR Y-REGISTER
0400: 0205 B1 FA             LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H) AND
0410: 0207 85 F9             STAZ  INH     STORE THIS IN DISPLAY BUFFER INH
0420: 0209 20 6F 1D          JSR   GETBYT  READ TWO HEXKEYS AND STORE THEIR VALUE IN THE
0430:                                      ACCUMULATOR. RETURN WITH N=1 IF
0440:                                      ONLY HEXKEYS WHERE DEPRESSED.
0450:                                      IF A COMMAND KEY WAS DEPRESSED,
0460:                                      RETURN WITH N=0
0470: 020C 10 07             BPL   PR      COMMAND KEY DEPRESSED?
0480: 020E A0 00             LDYIM $00     CLEAR Y-REGISTER
0490: 0210 91 FA             STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0500:                                      THE EPROM MEMORY LOCATION
0510:                                      SPECIFIED BY POINTL(H)
0520: 0212 4C 03 02          JMP   PRGR
0530: 0215 C9 12     PR      CMPIM $12
0540: 0217 D0 35             BNE   PRGR    +KEY?
0550: 0219 E6 FA             INCZ  POINTL  INCREMENT ADDRESS BY ONE
0560: 021B D0 E6             BNE   PRGR
0570: 021D E6 FB             INCZ  POINTH
0580: 021F 4C 03 02          JMP   PRGR
ID=02
0010: 0222 20 55 02  DUPL    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0020: 0225 A0 00     DU      LDYIM $00
0030: 0227 B1 00             LDAIY SAL     GET DATA SPECIFIED BY SAL(H)
0040: 0229 91 FA             STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0050:                                      THE EPROM MEMORY LOCATION
0060:                                      SPECIFIED BY POINTL(H)
0070: 022B 20 5E 02          JSR   INCMNT  INCREMENT SAL(H) AND POINTL(H) BY ONE
0080: 022E D0 F5             BNE   DU      NOT LAST ADDRESS
0090: 0230 4C 1D 1C          JMP   RESET   RETURN TO JUNIOR MONITOR
0100:
0110: 0233 20 55 02  VERIFY  JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0120: 0236 A0 00     VER     LDYIM $00
0130: 0238 B1 FA             LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H)
```

P.R. Boldt

EPROMmer

Ditmaal presenteren we een EPROMmer, oftewel EPROM-programmer, voor de huiscomputers van Elektuur, de SC/MP en de junior. De afmetingen zijn bescheiden gehouden: alles zit op een enkele print van euro-formaat. Desalniettemin biedt deze programmer de mogelijkheid tot het programmeren van de 2716 en de 2732. De EPROM kan trouwens niet alleen geprogrammeerd, maar ook uitgelezen worden.

Misschien lijkt het een beetje veel van het goede. Zo kort na de publikatie van een EPROM-programmer alwéér een EPROM-programmer? Nou, tussen die twee zit toch wel een groot verschil. De vorige was geschikt voor gebruik bij "Elektuur-vreemde" processors, terwijl deze hier speciaal ontworpen is voor de SC/MP en de 6502. Met deze programmer kan men zowel de 2716 als de 2732 programmeren èn lezen. De euro-kaart waarop de schakeling is ondergebracht, is voorzien van een standaard-konnektor die zo op de bus van de SC/MP en de uitbreidingskonnektor van de junior past. Genoeg redenen om dit ontwerp zo snel mogelijk te publiceren voor alle Elektuur-computer-bezitters, dachten we.

Op welke manier zo'n 2716 (en ook de 2732) geprogrammeerd kan worden, is uitvoerig beschreven in het oktober-nummer. Daarin werd al verteld, dat met name de 2716 erg goedkoop is en dit soort EPROM's bovendien gemakkelijk te programmeren is. We zullen nog even in het kort herhalen hoe dat in zijn werk gaat. Op pen 21 van de 2716 (pen 20 van de 2732) wordt een programmeerspanning van 25 V aangesloten. Een programmeerpuls met een duur van 50 ms op de $\overline{CE}$-ingang is dan voldoende om de gegevens op de data-ingangen te programmeren op het aangegeven adres.

Als we toch al over een computer beschikken (waarvoor we EPROM's willen programmeren), dan ligt het natuurlijk voor de hand deze ook te gebruiken bij het programmeren van de EPROM. Er is eigenlijk maar een punt dat daartegen spreekt: bij de junior computer is het niet mogelijk de processor stop te zetten in een schrijfcyclus, omdat een adres met de bijbehorende data minstens 50 ms lang op de EPROM aanwezig moet zijn (bij de SC/MP geeft dat geen problemen). Voor de junior is het dus nodig enkele buffers toe te voegen waarin de gegevens tussentijds kunnen worden opgeslagen. Omdat de print toch met die buffers er op werd ontworpen, hebben we die voor de toepassing met de SC/MP maar laten zitten; met behulp van enkele draadbrugjes kan men de kaart nu instellen op de gebruikte processor.

Verder is de schakeling zo opgezet dat alles voor de programmering via hardware wordt opgewekt.

De schakeling

Voordat we de programmering gaan bekijken, is het eerst nodig om te weten hoe de programmer zelf in elkaar zit. Het schema van de schakeling is te zien in figuur 1.

Zoals al is verteld, zijn er enkele buffers nodig waarin tussentijds de data en adressen kunnen worden opgeslagen. Dit zijn IC1, IC2, IC3 en IC4. Afhankelijk van de te programmeren EPROM moeten er 11 (voor de 2716) of 12 adresbits (voor de 2732) worden opgeslagen in de buffers. De twaalfde adreslijn wordt verbonden met de EPROM door middel van de schakelaar S1 (keuzeschakelaar 2716/2732).

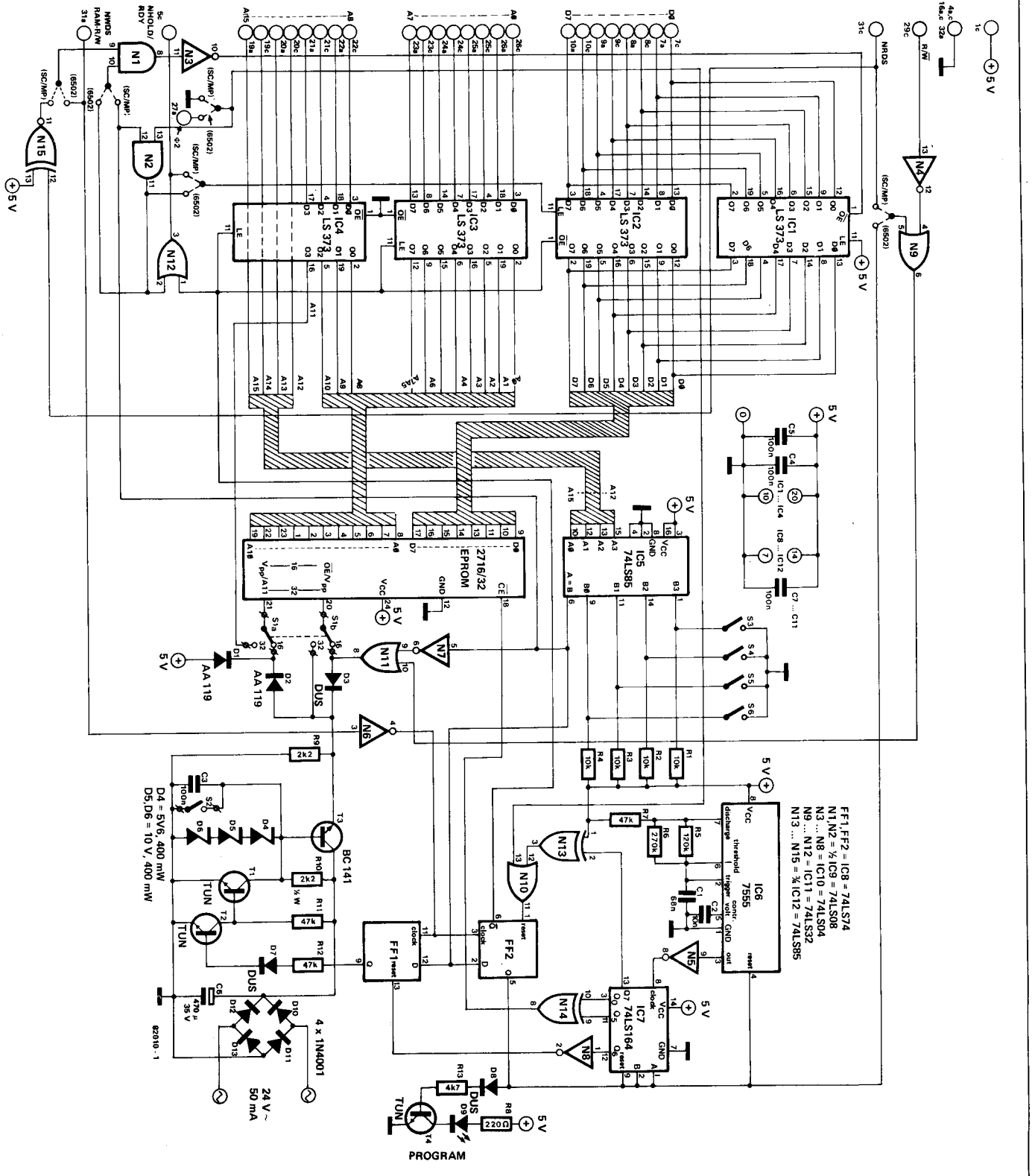
De data-lijnen DØ . . . D7 zijn aangesloten op twee latches, die parallel geschakeld zijn (IC1 en IC2). Daarbij zijn de ingangen van IC1 verbonden met de uitgangen van IC2 en de uitgangen van IC1 verbonden met de ingangen van IC2. Op deze manier is het mogelijk de data in de EPROM ook weer via de computer te kunnen uitlezen. Verder zijn nog enkele verbindingen met de computer nodig, die eveneens aan de linkerzijde in het schema getekend zijn.

Verder moet worden voorzien in een adresdekodering. Deze is opgezet rond een vier bits vergelijker, IC5. Aan de ene kant is dit IC verbonden met de adreslijnen A12 . . . A15, terwijl men aan de andere kant een adresbereik kan instellen door middel van de schakelaars S3 . . . S6. Een gesloten schakelaar betekent een logische nul, waarbij S3 korrespondeert met adreslijn A12, S4 met A13, S5 met A14 en S6 met A15. Zo'n bereik omvat steeds vier kbyte. Dat lijkt misschien erg veel, maar dat is noodzakelijk als we bedenken dat de schakeling de mogelijkheid biedt tot het programmeren en lezen van een 4 kbyte-EPROM (2732). Bij het programmeren van een 2716 heeft dit tot gevolg dat die EPROM zowel vanaf XØØØ als vanaf X8ØØ kan worden geadresseerd.

IC6 is een astabiele multivibrator met start/stop-mogelijkheid. De dimensionering van de frekwentiebepalende komponenten is zodanig dat de periodeduur 10 ms bedraagt. Via een inverter is de uitgang van de multivibrator verbonden met het schuifregister IC7. De reset-ingang van IC6 ligt aan de Q-uitgang van FF2. De clock-ingang van deze flipflop (en ook die van FF1) wordt voorzien van pulsen via N6, welke aangesloten is op pen 31a van de konnektor. Dit betekent dat de flipflops een clock-puls krijgen als de processor een schrijfsignaal geeft.

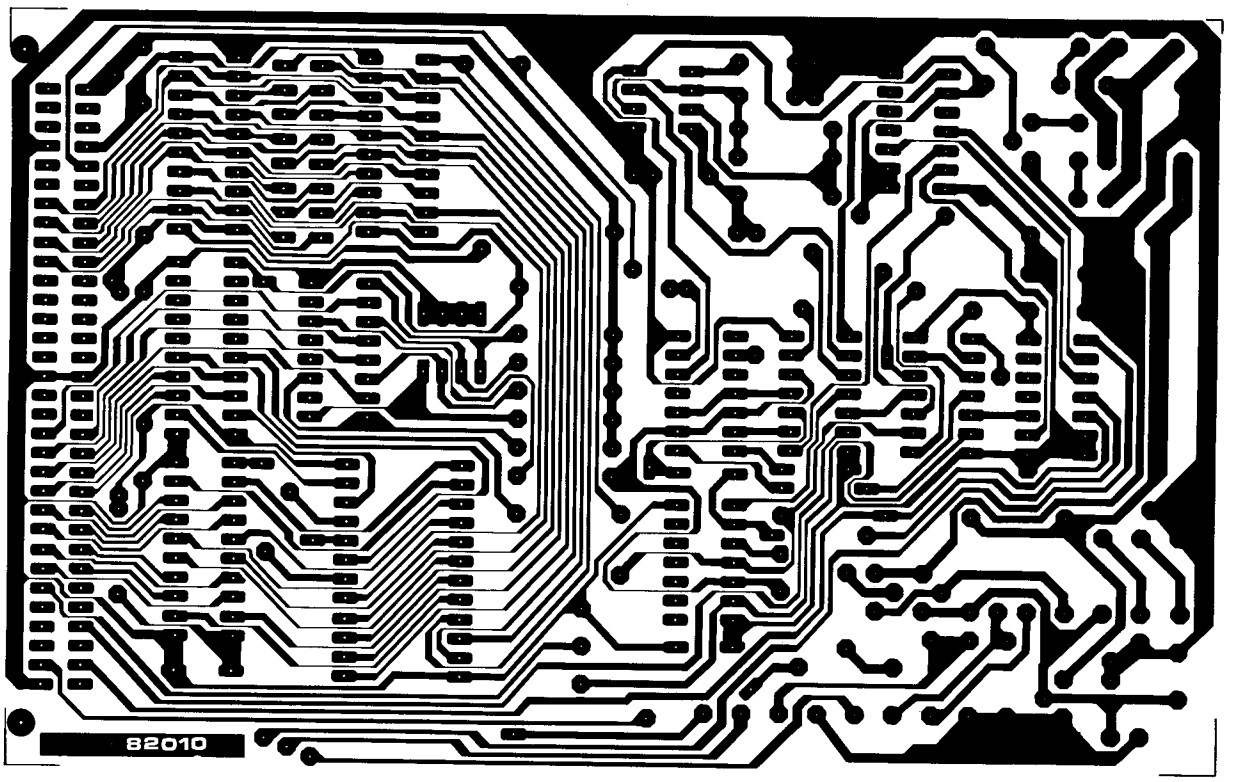
Op het moment dat het ingestelde adresbereik herkend wordt, zal de uitgang A = B van IC5 "1" worden. Dit nivo verschijnt direkt daarna (nadat de flipflops een schrijf-puls hebben gekregen van de processor in het geval dat men iets wil programmeren in de EPROM) aan de Q-uitgang van FF2 zodat de multivibrator IC6 gestart wordt. Gelijktijdig met het omschakelen van FF2 schakelt ook FF1 om, zodat een spanning van 25 V op de programmeerpen van de EPROM komt te staan. Voor die spanning van 25 V is een kleine voeding ontworpen, bestaande uit

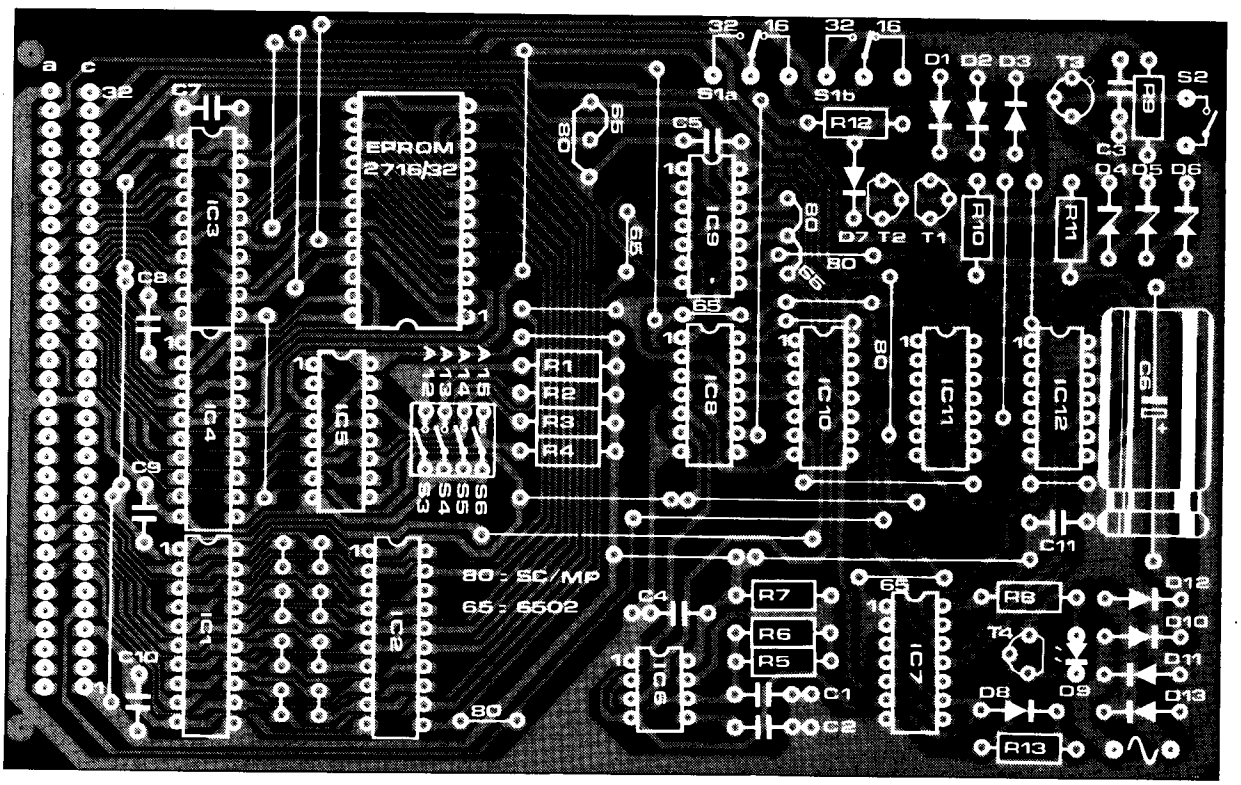
1



Figuur 1. Het schema van de EPROMmer. In het midden van het schema is de EPROM te zien, met daarboven de adresdekoder. Links daarvan staan de buffers voor de data en adressen en helemaal rechts is de logika te zien die zorgt voor de volgorde en tijdsduur van de verschillende programmeerpulsen. Rechts onderaan staat de voeding die de 25 V-programmeerspanning levert.

2





Figuur 2. De koperzijde en komponentenopstelling van de print voor de EPROM-programmer.

Onderdelenlijst

Weerstanden:
R1 . . . R4 = 10 k
R5,R6 = 220 k
R7,R11,R12 = 47 k
R8 = 220 Ω
R9 = 2k2
R10 = 2k2/½ W
R13 = 4k7

Kondensatoren:
C1 = 68 n
C2 = 10 n
C3,C4,C5 = 100 n
C6 = 470 µ/35 V
C7 . . . C11 = 100 n

Halfgeleiders:
D1,D2 = AA 119
D3,D7,D8 = DUS
D4 = zenerdiode 5V6/400 mW
D5,D6 = zenerdiode 10 V/400 mW
D9 = LED
D10 . . . D13 = 1N4001
T1,T2,T4 = TUN
T3 = BC 141
IC1 . . . IC4 = 74LS373
IC5 = 74LS85
IC6 = 7555
IC7 = 74LS164
IC8 = 74LS74
IC9 = 74LS08
IC10 = 74LS04
IC11 = 74LS32
IC12 = 74LS86

Diversen:
S1 = dubbelpolige wisselschakelaar
S2 = enkelpolige schakelaar
S3 . . . S6 = 4 miniatuur schakelaars in DIL-behuizing
24-pens IC-voet (evt. met hendel)
64-polige printkonnektor volgens DIN 41612, male, haaks

T1, T2 en T3 met bijbehorende onderdelen. Een bruggelijkrichter met afvlakelko zijn ook aanwezig zodat men alleen nog maar een wisselspanning van 24 V hoeft aan te sluiten. De Q-uitgang van FF1 kan nu de voeding in en uit schakelen. Met behulp van schakelaar S2 kan de schakeling op programmeren of lezen worden ingesteld. Als deze schakelaar wordt gesloten, is de programmeerspanning namelijk afgeschakeld. In de andere stand van de schakelaar kan men dan programmeren en lezen.

Maar nu weer even terug naar waar we gebleven waren. De programmeerspanning was ingeschakeld en de timer IC6 gestart. In het ritme van de pulsen die de timer levert wordt IC7 nu volgeschreven met "enen". 10 ms nadat de 25 V-spanning is ingeschakeld, wordt uitgang Q0 logisch een en daardoor $\overline{CE}$ van de EPROM ook "1". Door de verknoping van Q0 en Q5 met de EXOR N14 krijgt de $\overline{CE}$-ingang een "1"-nivo gedurende 50 ms. 10 ms daarna wordt uitgang Q6 "1" en wordt zo via FF1 de 25 V-spanning weer afgeschakeld. Gedurende die hele tijd zijn de data en adressen op de ingangen van de latches vastgehouden. Aan de uitgang van FF2 is via transistor T4 een LED aangesloten die oplicht als de EPROM geprogrammeerd wordt. Bij het lezen doet de hele hardware van de schakeling niet mee, omdat de processor dan geen schrijf-signaal geeft. Bij het lezen worden de uitgangen van IC2 in de tri-state-toestand gezet en IC1 vrijgegeven, wat ongeveer er op neer komt dat de door de EPROM geleverde data praktisch rechtstreeks verbonden is met de datalijnen van de processor. De schakelaar S1 dient voor het instellen van de soort te programmeren (of te lezen) EPROM. In de getekende stand van S1 (2716) ligt de programmeerspanning aan pen 21, terwijl de $\overline{OE}$-ingang verbonden is met N11. Als men de schakelaar omzet (2732), is adreslijn A11 verbonden met pen 21 en staat de programmeerspanning op pen 20.

Met behulp van enkele poorten (N1 . . . N4, N9, N11, N12 en N15) worden, afhankelijk van de toegepaste processor, uit enige kontrolesignalen nog een paar stuursignalen voor de schakeling afgeleid.

Eén eurokaart

In figuur 2 is de print voor de EPROMmer afgedrukt. Het volbouwen van deze eurokaart met onderdelen zal voor de doorgewinterde knutselaar, die tenslotte al over de nodige bouwervaring met de SC/MP of de junior beschikt, geen problemen opleveren. Neem wél een goede kwaliteit voetje voor de EPROM, aangezien dit vaker gebruikt zal worden. Heel mooi is hiervoor een voetje waarin de EPROM zonder kontaktdruk kan worden geplaatst en de kontakten daarna kunnen worden gesloten door middel van een hendeltje.

De kaart is vrij universeel van opzet. Alleen moeten de draadbrugjes op de print worden gelegd afhankelijk van het gebruik met de SC/MP of de 6502. Verder lopen alle verbindingen via de standaard-konnektor. Wel moet nog een wisselspanning van 24 V worden aangesloten voor de programmeerspanning.

SC/MP + EPROMmer

Voor bezitters van een SC/MP-systeem is het werken met de EPROMmer bijzonder eenvoudig. Na het instellen van het gewenste kaartadres wordt eerst S2 gesloten, dan wordt een gewiste EPROM in het voetje gestoken, de kaart in de bus gestoken en men kan beginnen met programmeren nadat dan de programmeerspanning is aangesloten en S2 weer geopend. Het verdient aanbeveling direkt na het programmeren S2 weer te sluiten.

Software voor het programmeren heeft men bij de SC/MP niet nodig. ELBUG is hiervoor voldoende. Als men enkele geheugenplaatsen wil programmeren, kan dat door het intikken van MO . . . YYYY, waarbij YYYY het te programmeren adres is. Bij een lege EPROM verschijnt daarna FF op het display. Als we op deze plaats bijvoorbeeld Ø8 willen zetten, wordt gewoon Ø8 ingegeven, en daarmee staat deze data al op het ingegeven adres in de EPROM.

Als men grotere blokken met data wil programmeren in de EPROM, verdient het aanbeveling deze eerst in RAM op te slaan en dan via een block transfer het geheel over te schrijven in de EPROM (BL . . . SSSS, EEEE, BBBB, waarbij S = startadres, E = eindadres en B = beginadres waar het blok moet worden neergeschreven).

Het lezen van de EPROM gaat op dezelfde manier als het lezen van een normaal adres.

Junior + EPROMmer

Bij de junior computer hebben we een programma nodig om te kunnen werken met de EPROMmer. Het hiervoor gegeven programma (zie tabel 1) staat op de geheugenplaatsen Ø2ØØ . . . Ø277. Als dit programma ingetypt is op de desbetreffende plaatsen kan de EPROMmer op de expansion-konnektor worden gestoken. De programmer is dan bedrijfsklaar.

Het zou ook mogelijk zijn het gegeven programma in een EPROM op te slaan zodat dit niet elke keer opnieuw hoeft te worden ingetypt als men iets wil programmeren (let er daarbij wel op dat de absolute sprongadressen binnen het programma aangepast moeten worden). Een mogelijkheid is om de TM-EPROM van de interface-kaart voor de junior hiervoor te gebruiken. Deze is momenteel namelijk nog niet helemaal vol, zodat op de plaatsen ØC8Ø . . . ØFFF overal FF geprogrammeerd staat (in de EPROM zelf zijn dat de plaatsen 48Ø . . . 7FF). Als men nu het programma op ØC8Ø laat beginnen in plaats van op Ø2ØØ (zoals in tabel 1) en de absolute sprongadressen binnen het programma aanpast (alle drie-byte-instrukties die op Ø2 eindigen), kan het programma in de TM-EPROM worden opgeslagen. We geven nu een korte beschrijving van de drie programma-delen PROGRAM, DUPLICATE en VERIFY.

De programmeer-routine

Op de geheugenplaatsen MOVL, MOVH (ØØØ4, ØØØ5) worden het rechter- en linker byte van het adres waar men wil beginnen met programmeren neergezet. Daarna wordt de programmeer-routine gestart (in tabel 1 Ø2ØØ). Op het display verschijnt nu het adres en de data die op dat adres staat (FF als de EPROM leeg is).

Als men bijvoorbeeld A9 wil programmeren, wordt eerst A ingedrukt (het display blijft onveranderd) en vervolgens 9. Het display dooft dan even (gedurende ongeveer 70 ms), waarna het gegeven adres en de data A9 weer op het display verschijnen. Daarmee is A9 op de desbe-

```
0140: 023A D1 00          CMPIY SAL    COMPARE THIS DATA WITH DATA SPECIFIED BY SAL(H)
0150: 023C F0 0F          BEQ   NEXT   DATA EQUAL?
0160: 023E 20 88 1D ANYKEY JSR  SCAND  DISPLAY EPROM ADDRESS AND DATA
0170: 0241 D0 FB          BNE   ANYKEY ANY KEY DEPRESSED?
0180: 0243 20 88 1D       JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
0190: 0246 D0 F6          BNE   ANYKEY ANY KEY DEPRESSED?
0200: 0248 20 88 1D NKEY  JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
0210: 024B F0 FB          BEQ   NKEY   NO KEY DEPRESSED?
0220: 024D 20 5E 02 NEXT  JSR   INCMNT INCREMENT SAL(H) AND POINTL(H) BY ONE
0230: 0250 D0 E4          BNE   VER    NOT LAST ADDRESS?
0240: 0252 4C 1D 1C       JMP   RESET  RETURN TO JUNIOR MONITOR
0250:
0260:                  ***********
0270:                  SUBROUTINES
0280:                  ***********
0290:
0300: 0255 A5 04     TRF  LDAZ  MOVL
0310: 0257 85 FA          STAZ  POINTL TRANSFER MOVL TO POINTL
0320: 0259 A5 05          LDAZ  MOVH
0330: 025B 85 FB          STAZ  POINTH TRANSFER MOVH TO POINTH
0340: 025D 60             RTS
0350:
0360: 025E 20 88 1D INCMNT JSR  SCAND  DISPLAY FOR ABOUT 5MS POINTH, POINTL
0370:                                  AND INH ( = EPROM ADDRESS AND DATA
0380:                                  ON THIS ADDRESS )
0390: 0261 E6 00          INCZ  SAL    INCREMENT SAL(H) BY ONE
0400: 0263 D0 02          BNE   INCDA
0410: 0265 E6 01          INCZ  SAH
0420: 0267 E6 FA     INCDA INCZ  POINTL INCREMENT POINTL(H) BY ONE
0430: 0269 D0 02          BNE   COMP
0440: 026B E6 FB          INCZ  POINTH
0450: 026D A5 01     COMP LDAZ  SAH
0460: 026F C5 03          CMPZ  EAH    COMPARE EAH WITH SAH
0470: 0271 D0 04          BNE   RTRN   EAH NOT EQUAL SAH?
0480: 0273 A5 00          LDAZ  SAL
0490: 0275 C5 02          CMPZ  EAL    COMPARE EAL WITH SAL
0500: 0277 60        RTRN RTS
```

Tabel 1. Het bij de junior computer behorende programma voor de EPROMmer.

treffende geheugenplaats in de EPROM geprogrammeerd.
Als men daarna de plustoets drukt verschijnt het volgende adres en kan men daar ook data programmeren op de hiervoor beschreven manier. Als men de EPROM alleen wil lezen kan dat door steeds op de plustoets te drukken of door terug te keren naar de monitor van de junior computer (reset-toets indrukken).

De dupliceer-routine

Als we een programma, dat ergens in het geheugenbereik staat, willen kopiëren in de EPROM, is dat mogelijk met de DUPLICATE-routine.
Daartoe wordt op de SAL, SAH (0000, 0001) het startadres van het te kopiëren programma gezet. Op EAL, EAH (0002, 0003) komt het eindadres + 1 en op MOVL, MOVH (0004, 0005) het adres van de EPROM waar het eerste byte van het te kopiëren programma-blok moet komen te staan.
Dan kan het dupliceer-programma gestart worden (adres 0222) en het display dooft. Na elke geprogrammeerde geheugenplaats meldt het display zich heel kort terug met het geven van de geprogrammeerde geheugenplaats. Op het display kan men dus zien hoe het programmeren vordert (let wel: het display licht slechts kort en vaag op!).
Als het dupliceren gebeurd is, meldt de junior zich terug door het geven van het laatste adres + 1.

De verify-routine

Met deze routine kan een bepaald programma-blok vergeleken worden met de inhoud van de EPROM. Hiervoor worden weer het startadres, het eindadres + 1 en het doeladres ingegeven, net zoals bij de dupliceer-routine. Daarna wordt het programma gestart op 0233.
Als nu ergens een fout ontdekt wordt, stopt het programma en verschijnt op het display het adres waar de fout staat met de data die op dat moment op die plaats in de EPROM staat. Als men daarna op een willekeurige toets drukt (behalve RES of ST) gaat het programma verder met kontroleren.
Als alles zo gekontroleerd is, meldt de junior zich weer terug met het laatste adres + 1 (we zitten dan weer in de junior-monitor).
Zo, met al de gegeven informatie moet het nu mogelijk zijn met behulp van de EPROMmer vrij snel en betrekkelijk eenvoudig zelf EPROM's te programmeren. Hoe gemakkelijk zo'n EPROMmer is, zult u in de praktijk heel snel merken.

de recorder gehaald. Het is namelijk zeer verstandig om straks de junior-Basic op een andere band te schrijven en de ongewijzigde KB-9 te bewaren.

- Wijzig met behulp van PM de inhoud van 31 geheugenplaatsen; zie hiervoor het eerste deel van de tabel. Kontroleer eerst aan de hand van de tabel of de oude data op de aangegeven adressen klopt. Is dat niet het geval, dan heeft u geheid een verkeerde Basic te pakken!
- Zet een verse cassette in de recorder, begin bij het begin, zet de teller op nul en start de recorder in de opnamestand. Toets na ca. 10 sekonden:

SB1, 2000, 4261 (CR)

Het op de band schrijven van de junior-Basic duurt net zo lang als het van de band lezen van KB-9, dus een paar minuten. Dat komt omdat het geheugenbereik ongewijzigd is: de adressen $2000 tot en met $4260. Het programmanummer is nu B1.

- Zodra de junior-Basic op de band staat volgt de terugmelding "READY". Laat de band nog gedurende een tiental sekonden doorlopen en druk dan de pauzetoets in.
- Op 18 geheugenplaatsen op pagina 1 A (PIA-RAM) moeten zes instrukties worden gezet die te maken hebben met het lezen van, respektievelijk schrijven op de csssetteband, dus met de Basic-kommando's LOAD en SAVE. Het gaat om de geheugenplaatsen $1A00 tot en met $1A11; voor details: zie de tweede helft van de tabel. Deze data gaat, voorzien van een ID = B2, de band op:
- druk de pauzetoets opnieuw in. Toets:

SB2, 1A00, 1A12 (CR).

- Na de terugmelding "READY" kan de cassette-recorder stopgezet worden. Nu moet men de in het geheugen staande junior-Basic beproeven. Bijvoorbeeld door te kijken of het vraag-en-antwoordspelletje na het starten van de Basic (startadres $4065) goed verloopt. Verder is het geen gek idee om een proefprogramma aan de junior-computer op te geven. De goede werking van de cassette-kommando's kun je het beste testen door een Basic-programma op de band te schrijven (SAVE), het programmageheugen te wissen (NEW) en vervolgens het Basic-programma opnieuw in te lezen (LOAD). Nadat de junior-Basic die in het geheugen staat (dus de ingelezen en gewijzigde KB-9) goed is bevonden kan men op dezelfde manier de junior-cassette beproeven. Schakel daartoe de junior-computer uit en na een tijdje weer in en lees de programma's B1 en B2 van de band in.

De maaltijd

Niets weerhoudt u ervan om nu met de junior-Basic te gaan werken. Het is wèl nodig dat u de bij de cassette geleverde schriftelijke dokumentatie (in het Amerikaans!) grondig bestudeert; met name de Basic-gebruiksaanwijzing van Microsoft ("Introduction", "Dictionary" en "Usage Notes"). Helaas is de letterlijke leesbaarheid van de dokumentatie niet al te best (kopieermachine met een slechte signaal/ruisverhouding). Inhoudelijk is het allemaal wat summier en onoverzichtelijk, maar er valt mee te werken. En daar gaat het om. De software-dokumentatie is nihil, er worden alleen een paar adressen opgegeven.

Als aanvulling op de dokumentatie bij de cassette zijn de volgende opmerkingen van belang:

1) Na het inlezen:
   RST 1 0 0 0 GO RES (RUB OUT)
   GB1(CR)
   READY (pauzetoets indrukken)
   GB2(CR) (pauzetoets opnieuw indrukken)
   READY

kan de junior-Basic worden gestart. Het startadres voor de koude start is $4065:

4065(SP)R

Het opstarten moet via PM gebeuren en niet via de standaard-monitor omdat dan de I/O niet goed is gedefinieerd. Trouwens: PM heb je tòch nodig voor het inlezen.

2) De junior-Basic legt beslag op een aantal geheugenplaatsen in pagina nul, te weten $0000 t/m $00DC en $00FF. Er wordt dus één geheugenplaats (MODE) van de standaard-monitor gebruikt, maar dat kan geen kwaad, want deze is alleen nodig voor het opstarten van PM.
3) Het startadres voor een warme start is $0000. Bij de KIM is de warme start nodig om naar de Basic terug te keren na het schrijven op of lezen van de band van een Basic-programma. Bij de junior-Basic gaat dat anders; zie punt 9. De warme start kan bij de junior-Basic worden gebruikt om van PM naar Basic terug te keren. De sprong van Basic naar PM kan het gevolg zijn van een NMI (indrukken toest ST) of van het indrukken van de BRK-toets op het ASCII-toetsenbord. De BRK-sprongvektor is gericht op het label LABJUN ($105F) van PM. Na het afdrukken van de tekst "JUNIOR" volgt de sprong naar het centrale label RESALL van PM (zie boek 4, hoofdstuk 14). In het geval van een NMI wordt RESALL bereikt na het doorlopen van de STEP-voorroutine ($14CF).
4) De toets ST kan worden gebruikt om tussentijds met behulp van PM allerlei geheugenplaatsen te bekijken, bijvoorbeeld die in pagina nul (zie punt 2). Via een warme start komt men terug in Basic.
5) Indien men een Basic-programma uitvoert (RUN) en men de Elekterminal gebruikt (maximaal 16 regels), en indien verder de output van het Basic-programma meer dan 16 regels omvat, kan men het volgende doen:
   RUN (CR)
   BRK (terwijl 16-de regel wordt afgedrukt)
   bekijk resultaat
   (SP) R de computer drukt af:
   OK Start het programma opnieuw:
   RUN (CR)
   druk 16-de en de daarop volgende 14 regels af
   enzovoorts.
6) Na de koude start van de junior-Basic wordt ondermeer uw "TERMINAL WIDTH" gevraagd. Bij gebruik van de Elekterminal geeft men 64 (CR) op.
7) Op het ASCII-toetsenbord zit geen toets "↑" of "^", die nodig is bij machtsverheffen: A↑4 komt overeen met A^4. Het gaat om de toets die bij indrukken de ASCII-kode $5E genereert. Deze toets kan worden gemaakt door een bestaande toets op te offeren. Het ene kontakt wordt verbonden met rij x7 en het andere kontakt met kolom y9 van de toetsenmatrix (de pennen 32 en 22 van IC1). Er komen twee toetsen in aanmerking: de toets "PAGE↑", bovenste rij, uiterst rechts, of de toets "ESC", tweede rij van boven, uiterst links. De laatste toets biedt de meest elegante oplossing omdat de funktie ESC behouden blijft (een kwestie van shiften). Verbreek daartoe de twee aansluitingen van de toets "ESC" op de leidingen x5 en y10 (de leidingen zèlf niet onderbreken!) en maak twee draadverbindingen met de punten 22 en 32 van IC1. Voor meer bijzonderheden: zie het artikel over het ASCII-toetsenbord in Elektuur, november 1978; boek 3 of het SC/MP-boek, deel 2.
8) Voor het opnieuw koud starten van de junior-Basic tijdens een computersessie moet deze opnieuw van de band worden ingelezen (uitsluitend file B1). Dit komt omdat, afhankelijk van de behoefte aan goniometrische funkties, een min of meer groot deel van datablok B1 wordt opgeofferd en als eerste deel van het Basic-werkgeheugen dienst doet. Een en ander is bekend na het vraag-en-antwoordspelletje, direkt na de koude start. Dus als u alsnog wel of juist geen prijs stelt op drie of vier goniometrische funkties, moet de wijziging worden opgegeven via een hernieuwde koude start — na het opnieuw inlezen van B1.

N.B. Van file B1 ($2000 t/m $4260) zijn:
de plaatsen $4041 t/m $4260 aan het werkgeheugen toegevoegd indien prijs wordt gesteld op alle 4 goniometrische funkties (toets Y),
de plaatsen $3F1F t/m $4260 aan het werkgeheugen toegevoegd indien geen prijs wordt gesteld op de goniometrische funkties (toets N);
de plaatsen $3FD3 t/m $4260 aan het werkgeheugen toegevoegd indien de funktie ATN vervalt (toets A).
De eerste geheugenplaats wordt met 00 geladen (BOF: Begin Of File). Bij gebruik van 16K RAM omvat het werkgeheugen in het geval:
Y de geheugenplaatsen $4042 t/m $5FFF (8126 bytes),
N de geheugenplaatsen $3F20 t/m $5FFF (8416 bytes),
A de geheugenplaatsen $3FD4 t/m $5FFF (8236 bytes).

9) Het schrijven op en lezen van de band van Basic-programma's (SAVE

eindelijk!

Junior spreekt Basic

"Ja, met X uit Y. Wanneer komt er nu eindelijk eens een Basic voor de junior-computer?"
"Nu!"
"Wordt-ie tiny, gewoon of extended?"
"Wat dacht u van een Basic met dezelfde mogelijkheden als de Apple-Basic?"
"Nou als dàt zou kunnen! Maar dat moet een hele klus zijn."
"Valt wel mee. Ooit van Microsoft gehoord?"
"Ja, die handelt in hoofdletters C met zo'n cirkeltje erom."
"Die hebben indertijd een prima Basic voor de KIM ontwikkeld. En die is aan te passen aan de junior-computer"
"Ja maar . . . màg dat zo maar?"
"Ja, dit gebeurt met medeweten en medewerking van de Europese vertegenwoordiger van de firma die KIM-Basic verkoopt."
"Nou prima! Bedankt alvast."
"Graag gedaan."

Kwa populariteit is Basic het Engels van de programmeertalen. Dat betekent niet automatisch dat deze programmeertaal het beste is op het punt "grammatica" en "spelling". Want Pascal en Comal gooien ook hoge ogen. Populariteit is onafhankelijk van kwaliteit: Dostojevski is "beter" dan Asterix, maar niet populairder.

Bij de junior-computer heeft het aksent altijd gelegen op het gebruik ervan via "zijn moedertaal": de machinetaal. De aandacht die tot nu toe aan hogere programmeertalen is gewijd, is net zo groot als de aandacht die in de dokumentatie van de Apple-computer – ook een 6502-systeem – aan machinetaal is gewijd. Zonder ook maar iets af te dingen op de merites van Basic, Pascal en Comal onderstrepen wij het belang van machinetaal; niet alleen in tijdkritische toepassingen maar net zo goed vanwege de edukatieve kant van de zaak.
Goed, men wil Basic. Ja, wat doe je dan als Elektuur? Het zelf ontwikkelen van kilo's systeemsoftware kost kilo's (lees: duizenden) man-uren. Het 200%-entousiasme van sommige lieden in de redaktie om de klus aan te pakken moge er dan zijn: die man-uren niet. En al zouden ze er wèl zijn – de uiteindelijke kostprijs voor u, lieve lezers, zou niet lager zijn dan die voor de alternatieve oplossing.
Dus we doen de was de deur uit. Dat wil zeggen. We maken een bestaande, voor het doel geschikte Basic geschikt voor de junior-computer. Er is een prima kandidaat: de KB-9-Basic van Microsoft/Johnson Computer. Dat is een negencijferige 8K-Basic op cassette, voor de KIM-computer. Slechts 31 van de ruim acht duizend geheugenplaatsen hoeven kwa inhoud te worden gewijzigd; verder moet een handvol instrukties worden toegevoegd. De aanpassing kost u een kwartiertje werk; dat is weinig ten opzichte van de jaren, gedurende welke u plezier kunt beleven aan een prima Basic èn aan de junior-computer.

De ingrediënten

Wat is er allemaal nodig om op de junior-computer met Basic te kunnen werken? Eerst maar de hardware. U heeft een uitgebreide junior-computer mèt buskaartgeheugen nodig. Hoe u dat voor elkaar krijgt is in boek 3 beschreven. Verder heeft u 16K RAM nodig, die gedekodeerd is op het adresbereik $2000 tot en met $5FFF, dus op de buskaart is aangesloten. Hiervoor kunt u twee RAM/EPROM-kaarten nemen met op elke kaart 8K RAM, òf de 16K dynamische RAM-kaart, die in het volgende nummer zal worden gepubliceerd. Hoewel daar op verscheidene plaatsen in boek 3 op is gewezen, doen we het hier nog maar eens dunnetjes over, want het is belangrijk genoeg: bij aangesloten buskaartgeheugen moeten ook de drie vektoren via de buskaart worden binnengehaald (adressen $FFFA . . . $FFFF). In aanhangsel 3 van boek 3 zijn twee mogelijkheden beschreven om die vektoren binnen te halen zonder dat daarvoor een dure RAM/EPROM-kaart nodig is. Een elegant alternatief biedt de mini-EPROM-kaart die de volgende maand kompleet met een echt Elektuur-printje zal worden gepubliceerd.
Ja, en dan de software. Voor het goed funktioneren van de junior-Basic is de aanwezigheid van zowel TM als PM vereist; TM (ESS 506) vanwege de cassette-subroutines DUMP ($Ø9DF) en RDTAPE ($ØBØ2) en PM (ESS 507 of ESS 507N; PME is hier niet van belang) vanwege het opstarten van de junior-Basic en de seriële I/O-subroutines RECCHA ($12AE) en PRCHA ($1334), en de subroutine RESTTY ($14BC). En verder heeft u een cassette met de KB-9-Basic nodig (*niet* KB-6 of KB-8!!), inklusief de onontbeerlijke bijgeleverde "paperware".
En verder? Een cassette-recorder, een ASCII-toetsenbord, een printer of videodisplay, een goed humeur en kennis van programmeren in Basic. En wat dat laatste betreft: als men alle boeken over Basic op elkaar zou leggen zou de stapel zijn omgevallen lang voordat de top is bereikt. Kortom: keuze genoeg. Maar neem het ons niet kwalijk als wij een beetje voor eigen parochie preken en u de Elektuur-Basic-kursus aanbevelen; zie Elektuur, maart t/m juni 1979 of het SC/MP-boek, deel 2.

Het recept

Hoe maak je van een KB-9-Basic een junior-Basic? We gaan ervan uit dat u beschikt over de cassette met KB-9 en dat de eerder omschreven junior-hardware aanwezig is.

- Schakel de junior-computer in en start PM; plaats de KB-9-cassette in de recorder:

RST 1 Ø Ø Ø GO RES
G1 (CR)

- Start de recorder (weergave) vanaf het begin van de band; KB-9 heeft een programmanummer (ID) Ø1. Het inlezen duurt enige minuten. Uitgelezen? Dan volgt de terugmelding "READY". De KB-9-cassette wordt uit

en LOAD) gaat met de junior-Basic handiger dan met de KIM-Basic, dankzij het feit dat de daarvoor benodigde software in de vorm van subroutines is gegoten. De prijs die men hiervoor betaalt is de tweede file: B2. Nadat SAVE (CR) is ingetoetst wordt het Basic-programma op de band geschreven (met ID = FE) en volgt na enige tijd de terugmelding OK, voorafgegaan door een regel spatie. Nadat LOAD (CR) is ingetoetst wordt een Basic-programma van de band gelezen (met ID = FF, dus zorg ervoor dat het gewenste Basic-programma vóór staat!). Na enige tijd volgt de terugmelding LOADED. Hierop volgt geen melding OK en er wordt ook niet op het begin van een nieuwe regel begonnen. Met andere woorden: Indien men het zojuist ingelezen programma wil bekijken vertoont het TV-scherm de tekst LOADEDLIST. Nou ja, daar valt mee te leven.

Nog vragen?

Om onnodige vragen te voorkomen hier alvast een paar antwoorden.

Ten eerste. Elektuur kan niet ingaan op verzoeken (in het kader van de EKS) voor de levering van een kopie van de bij de cassette geleverde schriftelijke dokumentatie van Microsoft en Johnson Computer. Waarom? Zie het colofon (Auteurswet). Wat u niet mag, mogen wij ook niet!

Ten tweede. Voor een flink aantal dollars kan men een volledige listing (source listing) van KB-9 "krijgen". Dus vraag het òns niet. Een "EDS" (Elektuur Disassembleer Service) behoort *niet* tot het Elektuur-servicepakket.

Ten derde. De junior-Basic is afgeleid van de KIM-Basic. Een goede gelegenheid om nog eens te wijzen op de "KIM Gebruikersclub Nederland" (Voorburgpad 10, 6843 EM Arnhem; op werkdagen van 19.00 tot 20.00: 085 - 816935). In het clubtijdschrift, "De 6502-kenner" (ex-"KIM-kenner") hebben de nodige artikelen over KB-9 gestaan. Veel van de beschreven aanpassingen ("patches") van KB-9 zijn, soms met opnieuw aanpassingen, van toepassing op de junior-Basic.

Ten vierde. Hoe kom ik aan een KB-9? De Europese vertegenwoordiger van Johnson Computer is:

MSB-Verlag
R. Nedela
Mangoldstraße 10
D-7778 Markdorf
West-Duitsland
tel. 07544 - 3058

Het is de bedoeling dat de Basic-cassette *kompleet met de voor de junior-computer noodzakelijke aanpassingen en toevoegingen* wordt geleverd! In dat geval hoeft u de wijzigingen volgens de tabel niet meer aan te brengen. Volgens onze informatie kost de cassette DM 288,—, inklusief 13% Mehrwertsteuer (BTW). Op het tijdstip van schrijven van dit artikel (begin januari 1982) is het nog niet bekend hoe het zit met de landelijke wederverkoop-adressen en met de prijs in guldens, francs en ponden. Ook is de Johnson-junior-Basic nog niet door ons getest.

Ten vijfde. Ja! (Op de vraag of alle junior- en Basic-fans nu tevreden zijn). ■

Van KB-9 naar junior-Basic.
(gebaseerd op KB-9, cassette 4065 © 1977 by Microsoft Co; versie V1.1)

I. Interpreter

a) ID = B1 in plaats van Ø1.
b)
1. Op adres $2457 data AE in plaats van 5A;
2. Op adres $2458 data 12 in plaats van 1E;
3. Op adres $26DD data 8Ø in plaats van 4Ø;
4. Op adres $26DE data 1A in plaats van 17;
5. Op adres $2746 data 79 in plaats van F9;
6. Op adres $2747 data 1A in plaats van 17;
7. Op adres $274D data 7Ø in plaats van F5;
8. Op adres $274E data 1A in plaats van 17;
9. Op adres $275Ø data 71 in plaats van F6;
10. Op adres $2751 data 1A in plaats van 17;
11. Op adres $2757 data 72 In plaats van F7;
12. Op adres $2758 data 1A in plaats van 17;
13. Op adres $275A data 73 in plaats van F8;
14. Op adres $275B data 1A in plaats van 17;
15. Op adres $275E data 1A in plaats van 18;
16. Op adres $2791 data 7Ø in plaats van F5;
17. Op adres $2792 data 1A in plaats van 17;
18. Op adres $2794 data 71 in plaats van F6;
19. Op adres $2795 data 1A in plaats van 17;
20. Op adres $2799 data 79 in plaats van F9;
21. Op adres $279A data 1A in plaats van 17;
22. Op adres $27A4 data Ø9 in plaats van 73;
23. Op adres $27A5 data 1A in plaats van 18;
24. Op adres $27B9 data FA in plaats van ED;
25. Op adres $27BA data ØØ in plaats van 17;
26. Op adres $27BC data FB in plaats van EE;
27. Op adres $27BD data ØØ in plaats van 17;
28. Op adres $2A52 data 34 in plaats van AØ;
29. Op adres $2A53 data 13 in plaats van 1E;
30. Op adres $2AE6 data AE in plaats van 5A;
31. Op adres $2AE7 data 12 in plaats van 1E.

II. Extra instrukties op pagina 1A

a) ID = B2.
b)
1. Op adres $1AØØ data 2Ø;
2. Op adres $1AØ1 data DF;
3. Op adres $1AØ2 data Ø9;
4. Op adres $1AØ3 data 2Ø;
5. Op adres $1AØ4 data BC;
6. Op adres $1AØ5 data 14;
7. Op adres $1AØ6 data 4C;
8. Op adres $1AØ7 data 48;
9. Op adres $1AØ8 data 23;
10. Op adres $1AØ9 data 2Ø;
11. Op adres $1AØA data Ø2;
12. Op adres $1AØB data ØB;
13. Op adres $1AØC data 2Ø;
14. Op adres $1AØD data BC;
15. Op adres $1AØE data 14;
16. Op adres $1AØF data 4C;
17. Op adres $1A1Ø data A6;
18. Op adres $1A11 data 27.

impulsdiagram in figuur 6 wordt een en ander nog wat duidelijker. Bij elke positieve flank van Φ1 wordt de inhoud van de teller met één verhoogd. De buffers N11 . . . N17 zijn uitgeschakeld, aangezien zij gestuurd worden door de uitgang van N2 (geïnverteerd t.o.v. de uitgang van N3). Via de buffers N20 . . . N26 staat dan een adres op de adresingangen van de RAM's. Met behulp van MMV1 en MMV2 is een vertraging opgebouwd, die er voor zorgt dat korte tijd na de opgaande flank van de clock een negatieve puls op de $\overline{RAS}$-ingangen van de RAM's wordt gegeven. We hebben in het begin van het artikel al gezien dat dit voldoende is om een hele rij te verversen. Bij elke positieve clock-puls wordt op die manier een rij ververst, zodat de teller na 128 clock-periodes alle tellerstanden heeft doorlopen en daarmee ook alle rijen (dat waren er ook 128) zijn ververst. Een hele refresh-cyclus duurt dus steeds 128 μs (bij een clock-frekwentie van 1 MHz). De tijden van MMV1 en MMV2 zijn zo gekozen dat de refresh-tijd gegarandeerd midden in het positieve clock-gedeelte valt.

Het in- en uitlezen van een adres gebeurt altijd in het negatieve gedeelte van de Φ1-clock. Hierbij is nog enige "timing-logika" nodig, die nauwkeurig gedimensioneerd moet zijn om zeker te stellen dat de verschillende flanken in de juiste volgorde bij de RAM's (en de multiplexers) komen. Dit gebeurt met drie "pulsvertragers", bestaande uit N4 . . . N10, R1 . . . R3 en C3 . . . C5. In het schema is duidelijk zichtbaar welke onderdelen bij elkaar horen.

Als er in het adresbereik van de RAM's een adres wordt aangeboden, zal dat tot gevolg hebben dat de uitgang van de NAND N29 door de stand van de adresdekoder logisch nul wordt. Via N31, waarop ook Φ1 is aangesloten, gaat het clock-signaal dan naar N7 en N9. De neergaande flank van de clock wordt vertraagd door de kombinatie R1/C3 en via enkele poorten naar de $\overline{RAS}$-ingangen gevoerd (zie ook figuur 6). Dat betekent dat de eerste zeven adres-bits zijn ingelezen in de RAM's. Daarna moeten de multiplexers worden omgeschakeld, wat gebeurt door de neergaande $\overline{RAS}$-flank nog eens te vertragen door middel van R3 en C5. Als dan de volgende zeven adresbits op de RAM-ingangen staan kan een neergaande flank op de $\overline{CAS}$-ingangen worden gegeven. Deze flank wordt ook afgeleid van de neergaande clock-flank via de vertraging R2/C4. De $\overline{WE}$-ingangen zijn rechtstreeks verbonden met de desbetreffende aansluiting op de konnektor.

Daarmee zijn alle belangrijke signalen behandeld. Er zijn nog enkele poorten en een flipflop die nog niet ter sprake zijn gekomen (onder andere links boven in het schema), maar die dienen alleen maar voor de aanpassing van diverse signalen, zodat de RAM-kaart ook met andere processoren dan de 6502 gebruikt kan worden.

Onderdelenlijst

Weerstanden:
R1 . . . R3 = 270 Ω
R4,R5 = 2k2
R6 . . . R8 = 390 Ω

Kondensatoren:
C1 = 33 p
C2 = 100 p
C3 = 68 p
C4 = 470 p
C5 = 120 p
C6 . . . C21 = 1 μ/16 V tantaal

Halfgeleiders:
IC1,IC2 = 74LS14
IC3 = 74LS221
IC4,IC5,IC20 = 74LS244
IC6 = 74LS393
IC7 = 74LS08
IC8 = 74LS32
IC9,IC10 = 74LS157
IC11 = 74LS154
IC12 . . . IC19 = 4116 (250 ns)
IC21 = 74LS15
IC22 = 74LS74 (zie tekst)

Diversen:
1 64-polige konnektor volgens DIN 41612, male

Tabel 1

6502	Z-80	8085
1-1'	1-1'	1-1'
A-B	2-2'	2-2'
C-D	J2	3-3'
E-F	J3	4-4'
G-H	J4	5-5'
J8	J5	J1
J9	J6	J2
	J9	J4
IC22 vervalt		J6
	IC22 vervalt	J9
		J10

Tabel 1. Hier is aangegeven welke draadbruggen op de print moeten worden gelegd voor gebruik bij een bepaalde processor.

Tabel 2

Uitgang IC5	adres 4 Kbyte-blok	A15	A14	A13	A12
Ø	ØØØØ . . . ØFFF	Ø	Ø	Ø	Ø
1	1ØØØ . . . 1FFF	Ø	Ø	Ø	1
2	2ØØØ . . . 2FFF	Ø	Ø	1	Ø
3	3ØØØ . . . 3FFF	Ø	Ø	1	1
4	4ØØØ . . . 4FFF	Ø	1	Ø	Ø
5	5ØØØ . . . 5FFF	Ø	1	Ø	1
6	6ØØØ . . . 6FFF	Ø	1	1	Ø
7	7ØØØ . . . 7FFF	Ø	1	1	1
8	8ØØØ . . . 8FFF	1	Ø	Ø	Ø
9	9ØØØ . . . 9FFF	1	Ø	Ø	1
A	AØØØ . . . AFFF	1	Ø	1	Ø
B	BØØØ . . . BFFF	1	Ø	1	1
C	CØØØ . . . CFFF	1	1	Ø	Ø
D	DØØØ . . . DFFF	1	1	Ø	1
E	EØØØ . . . EFFF	1	1	1	Ø
F	FØØØ . . . FFFF	1	1	1	1

Tabel 2. Het adresbereik dat kan worden ingesteld door het leggen van verbindingen tussen de uitgangen van IC5 en de punten V, W, X en Y. Elke verbinding betekent een blok van 4 Kbyte, er moeten dus beslist vier verbindingen worden gelegd als men de hele 16 K wil gebruiken.

De opbouw

In figuur 7 is de print voor de dynamische RAM-kaart afgebeeld. Het is natuurlijk zaak bij een ontwerp waarbij de timing bijzonder kritisch is, ook heel zorgvuldig te werk te gaan bij de opbouw van de schakeling. Het is ook noodzakelijk dat men zich houdt aan de voorgeschreven komponenten; vooral aan de weerstand- en kondensatorwaarden mag niets veranderd worden. Als men gebruik maakt van de verkrijgbare print zal de bouw verder weinig problemen geven.

De aanwezigheid van verschillende draadbruggen op de print is afhankelijk van de gebruikte processor. In tabel 1 is af te lezen welke draadbruggen voor welke processor moeten worden gelegd. Bij de 6502 en de Z80 moet IC22 vervallen, aangezien flipflop FF1 daarbij niet nodig is. Bij gebruik van de 8085 moet men met enkele punten rekening houden. In afwijking tot de Z80 levert de 8085 geen resfresh-signaal. Dat wordt in dit geval gemaakt uit SØ, $\overline{S1}$ en INTA (deze geven de opcode-fetch aan). In de tijd die de processor nodig heeft om de kode te detekteren wordt de RAM niet gebruikt en kan er worden gerefreshed. Ten tweede heeft de 8085 een databus waarop ook adressen beschikbaar zijn (gemultiplext). De dynamische RAM-kaart is echter alleen geschikt voor een niet-gemultiplexte bus. Bij de 8085 moet de databus dus elders in het systeem gedemultiplext zijn.

De verbindingen tussen de punten V, W, X, Y en de uitgangen van IC11 bepalen het gekozen adresbereik. Elke uitgang van het IC vertegenwoordigt een adresbereik van 4 Kbyte. In tabel 2 is de indeling gegeven. Aangezien het dynamische geheugen 16 Kbyte groot is, worden dus vier uitgangen van IC11 verbonden met de punten V, W, X en Y. Men kan het geheugen op die manier op bijna elke willekeurige plaats leggen in 4 K-blokken. Het enige waar goed op moet worden gelet, is het feit dat niet twee maal dezelfde kode voor A12 en A13 mag worden genomen (zie ook de laatste kolom van tabel 2). Dat is noodzakelijk omdat die twee adreslijnen ook al zijn aangesloten op de adresingangen van de RAM's. De vier draadbruggen moeten dus altijd zo worden gelegd, dat de volgende kombinaties van A13 en A12 bij elkaar zijn genomen:

A13	A12
Ø	Ø
Ø	1
1	Ø
1	1

Uit de tabel kan gemakkelijk worden afgeleid welke kombinaties zijn toegestaan. Een goede kombinatie is bijvoorbeeld de blokken 8ØØØ, 9ØØØ, AØØØ en BØØØ, terwijl de kombinatie ØØØØ, 4ØØØ, 8ØØØ en CØØØ beslist niet is toegestaan, omdat hierbij voor alle blokken A13-A12 ØØ is.

Zoals gebruikelijk bevelen wij ook hier

16 Kbyte in 8 IC's

dynamische RAM-kaart

De prijzen van dynamische geheugens zijn tegenwoordig zo laag, dat het ondanks de hierbij benodigde extra stuur-elektronica zeker de moeite waard is deze te gebruiken in plaats van de gewoonlijk toegepaste statische geheugens. Met slechts acht geheugen-IC's is het dan mogelijk 16 Kbyte op een eurokaart onder te brengen, waarbij nog een zee van ruimte overblijft voor de stuurlogika. Zo'n dynamische RAM-kaart heeft trouwens nog meer voordelen, namelijk een laag stroomverbruik en een korte toegangstijd. Nadelen zijn er eigenlijk niet (meer)!

Bij de computersystemen die heden ten dage door de hobbyisten worden gebruikt, wordt voor de opslag van gegevens voornamelijk gebruik gemaakt van twee soorten vluchtige geheugens: statische en dynamische. Bij de statische geheugens gaat het om RAM's, waarbij de data die er in wordt gezet daar zo lang in blijft staan als de voedingsspanning aanwezig is. Bij een dynamisch geheugen is dat wat moeilijker, daarbij moet alle data regelmatig vernieuwd worden om te voorkomen dat deze verloren gaat. De flink gedaalde prijzen van de dynamische RAM's zijn er de oorzaak van dat het zo langzamerhand aantrekkelijker wordt hieraan de voorkeur te geven boven hun statische broertjes.

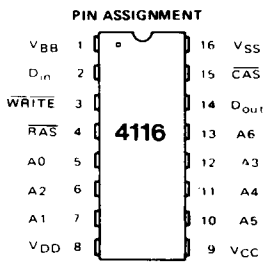
Dynamisch tegen statisch

Tot nu toe werd, vooral bij zelfbouw, voornamelijk gebruik gemaakt van statische RAM's. De redenen daarvoor liggen (of beter gezegd: lagen) eigenlijk ook voor de hand. Een statisch geheugen is namelijk heel eenvoudig in het gebruik. Alles zit in het IC, dus buiten de RAM's is er eigenlijk niets nodig om ze als geheugen te gebruiken. Ook timing-problemen zijn er nauwelijks, als je maar RAM's neemt die snel genoeg zijn voor de gewenste toepassing.
Een geheugencel van een statische RAM bestaat uit een soort set-reset-flipflop. Zo'n enkele flipflop bestaat in werkelijkheid uit minimaal 5 à 6 transistoren, wat betekent dat de fysieke opbouw van een RAM nogal omvangrijk is.
Bij een dynamische RAM wordt geen gebruik gemaakt van flipflops, maar van kapaciteiten. Elke cel bestaat uit een kondensator met een FET-schakelaar, waarbij de spanning over de kondensator bepaalt of er een nul of een één in de cel is opgeslagen. Als we deze opbouw vergelijken met die van een statische RAM, dan zal het duidelijk zijn dat een geheugencel van een dynamische RAM veel minder plaats inneemt dan een geheugenplaats van een statische RAM. Dat zou betekenen dat men zo een veel grotere geheugenkapaciteit kan onderbrengen op een zelfde chip-oppervlakte. Helaas zijn een kondensator en een FET niet voldoende om een goed werkende cel te verkrijgen, en hierbij komen we dan aan de nadelen van de dynamische RAM. Elke kapaciteit heeft een kleine lekstroom, wat tot gevolg heeft dat de spanning over de kondensator langzaam afneemt. Om te voorkomen dat de in de kondensatoren opgeslagen gegevens daardoor verloren gaan, is het nodig dat de spanningsnivo's van die kondensatoren periodiek vernieuwd of "ververst" (van het Engelse refresh) worden. En daarvoor is natuurlijk weer extra stuur-elektronica benodigd plus een heel nauwkeurige "timing" om alles goed en toch snel te laten werken.

Dat is echter niet het enige probleem. Doordat men zo ontzettend veel geheugenplaatsen op een chip kan onderbrengen, wordt de adressering ook minder eenvoudig. De fabrikanten van dynamische RAM's hebben dit proberen op te lossen door het toepassen van een gemultiplexte adresbus. Ook daarmee moet men dus rekening houden bij het gebruik van dit soort geheugens.

De prijzen van de huidige dynamische RAM's zijn in verhouding (per geheugenplaats) tot hun statische konkurrenten zo laag geworden, dat de kosten voor de extra stuur-elektronica in de totale prijs niet meer als nadeel hoeven te worden beschouwd. De stroomopname van een dynamisch systeem is ook veel lager dan een vergelijkbaar groot statisch systeem, hoewel we daar voor alle eerlijkheid bij moeten vermelden dat voor deze dynamische RAM-kaart drie voedingsspan-

1



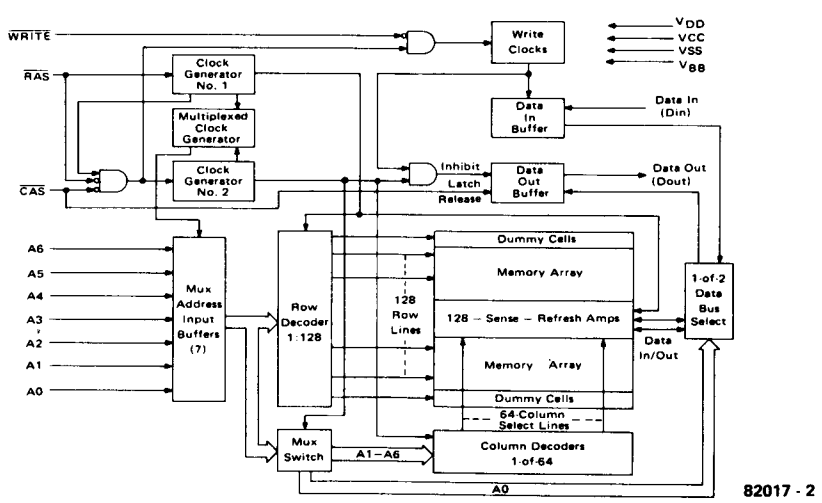
PIN NAMES

A0–A6	Address Inputs
$\overline{CAS}$	Column Address Strobe
D_{in}	Data In
D_{out}	Data Out
$\overline{RAS}$	Row Address Strobe
$\overline{WRITE}$	Read/Write Input
V_{BB}	Power (-5 V)
V_{CC}	Power (+5 V)
V_{DD}	Power (+12 V)
V_{SS}	Ground

82017 - 1

Figuur 1. De "pinning" van het dynamische RAM-IC 4116. Opmerkelijk is het feit dat er drie voedingsspanningen nodig zijn. De stroomopname is daarentegen vrij klein.

2



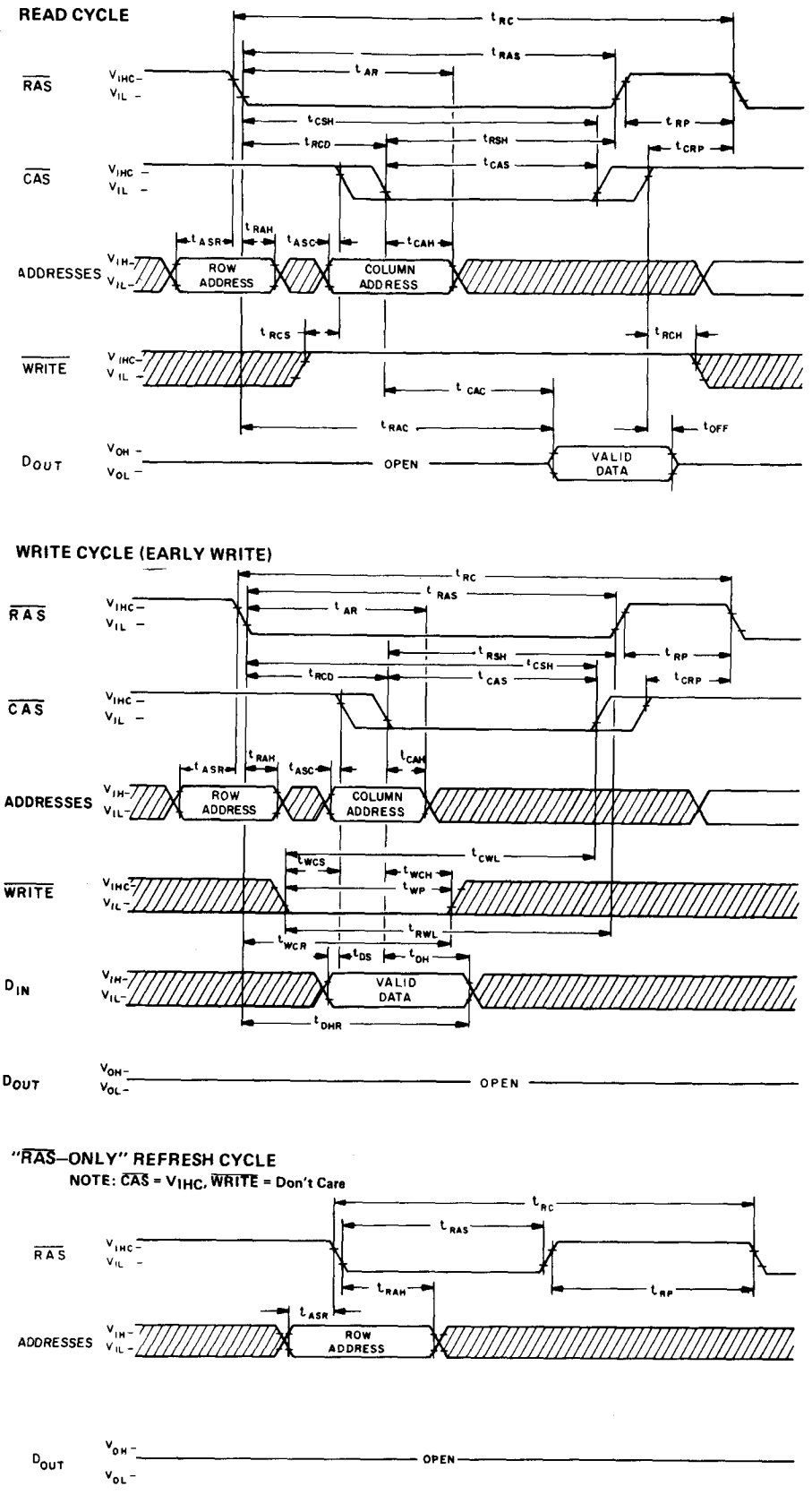
Figuur 2. Het blokschema van de 4116. Het geheugen is verdeeld in 128 rijen (rows) en 128 kolommen (columns). In het midden van de rijen zitten de sense amplifiers.

ningen nodig zijn.
Alles bij elkaar genomen kunnen we toch gerust stellen dat het op dit moment de moeite waard is in plaats van statische geheugens dynamische geheugens toe te passen.

Opbouw en werking van een dynamische RAM

In dit ontwerp is uitgegaan van de goedkope en goed verkrijgbare 4116, die door verschillende fabrikanten geleverd wordt. Dit is een 16384 x 1 bit dynamische RAM, zodat men met 8 IC's een 8 bit breed 16 K-geheugen kan opzetten. Het IC is vrij snel; het is verkrijgbaar met toegangs(access)-tijden van 150 . . . 300 ns, wat wordt aangegeven door een cijfer achter het typenummer. De chip is ondergebracht in een "normale" 16-pens IC-behuizing. En dat brengt ons gelijk bij de interne opbouw, want gewoonlijk zou je voor 16 Kbit zeker niet genoeg hebben aan 16 aansluitpennen.
Het geheugen van de 4116 is opgezet als een matrix van 128 kolommen en 128 rijen (128 x 128 = 16384). Voor de adressering van een bit zijn 14 adres-bits nodig (2^{14} = 16384), 7 voor het aanspreken van een kolom en 7 voor het aanspreken van een rij. Een geïntegreerde kolom-dekoder en rij-dekoder vertalen elk 7-bits adres naar één kolom en één rij. Zoals uit de aansluitgegevens van de 4116 uit figuur 1 blijkt, zijn er slechts 7 adres-ingangen. Dat is mogelijk doordat op de chip nog een multiplexer is ondergebracht, waardoor op de 7 ingangen eerst het rij-adres kan worden aangeboden en daarna het kolom-adres. Om aan het IC duidelijk te maken om welke adressen het gaat, zijn de ingangen $\overline{RAS}$ en $\overline{CAS}$ toegevoegd. Een negatieve puls op de $\overline{RAS}$-ingang (Row Address Strobe) zorgt er voor dat de informatie op de adres-ingangen wordt ingelezen als rij-adres en een negatieve puls op de $\overline{CAS}$-ingang (Column Address Strobe) doet de gegevens inlezen als kolom-adres. Aangezien het hier om een één bit breed geheugen gaat, zijn er slechts één data-ingang en één data-uitgang (D_{in} en D_{out}). Door middel van het aangeboden logische nivo op de $\overline{WRITE}$-ingang wordt bepaald of een bit geschreven danwel gelezen wordt. Met de voedingsaansluitingen V_{DD}, V_{CC}, V_{BB} en V_{SS} (respektievelijk +12 V, +5 V, –5 V en nul) zijn we dan alle IC-aansluitingen langs geweest.
Nu nog even terug naar de interne opbouw van het IC. Zoals het blokschema in figuur 2 laat zien, zijn er nog verschillende blokken die we niet besproken hebben. Voor de principiële werking zijn deze echter niet belangrijk. Interessant is wel nog om te weten dat in het midden van de 128 rijen 128 "lees"-versterkers zijn opgenomen (sense amplifiers), die enerzijds tot taak hebben de ladingen van de kondensatoren weer op peil te brengen tijdens een refresh-cyclus en anderzijds dienen voor het in-

3



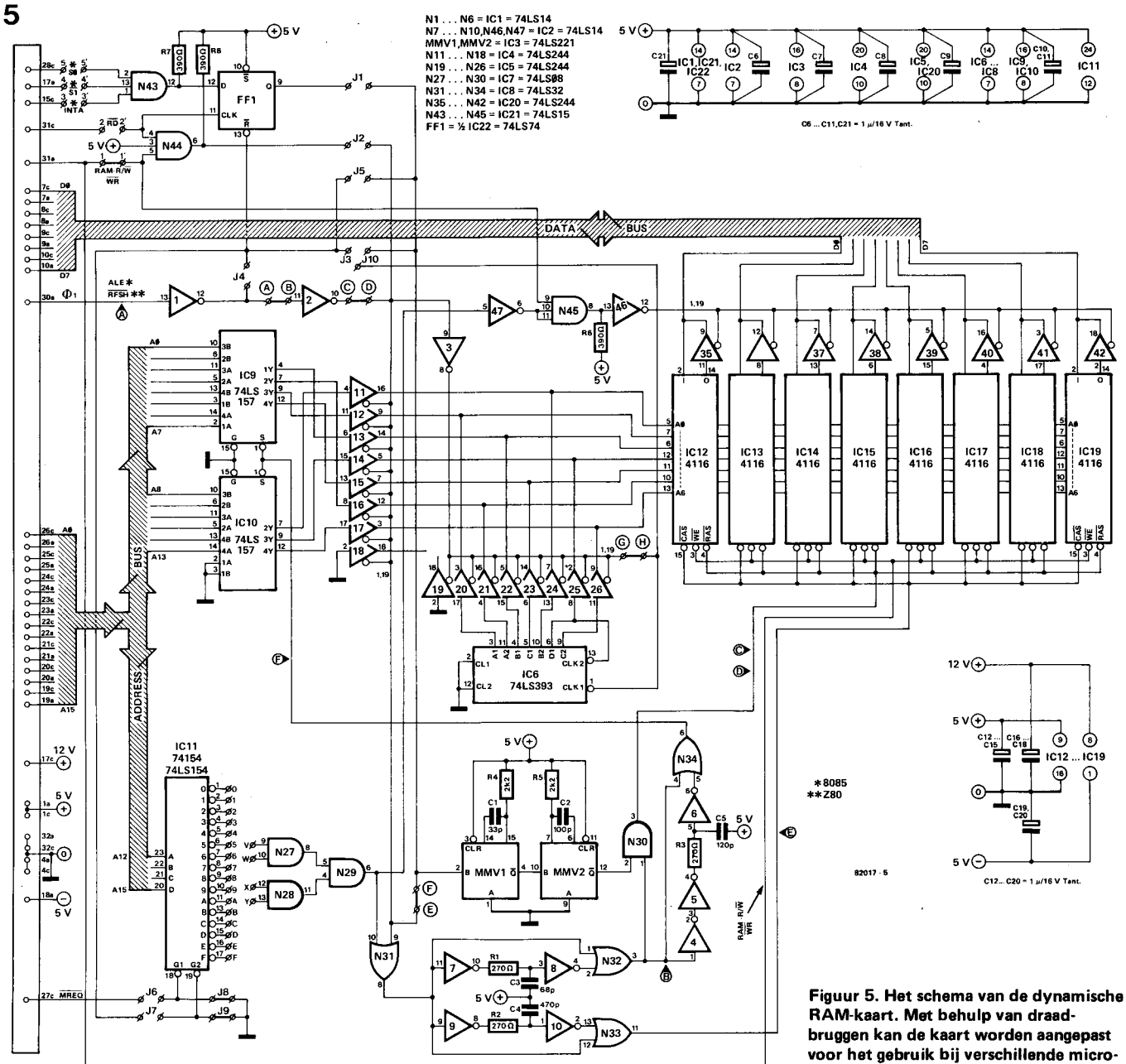
Figuur 3. De tijdvolgordediagrammen voor het lezen, schrijven en verversen (refresh) van de 4116. Tijden zijn hierbij niet aangegeven, omdat deze afhankelijk zijn van de snelheid van het toegepaste type. De tijden liggen wel alle in het nanosekondenbereik.

4

Figuur 4. Dit tekeningetje laat de stroomopname zien bij verschillende RAS- en CAS-signalen. Hieruit blijkt dat het gemiddelde stroomverbruik vrij klein is, maar dat er wel korte pieken optreden van zo'n 100 mA. Bij het ontwerp moet daar goed rekening mee worden gehouden.

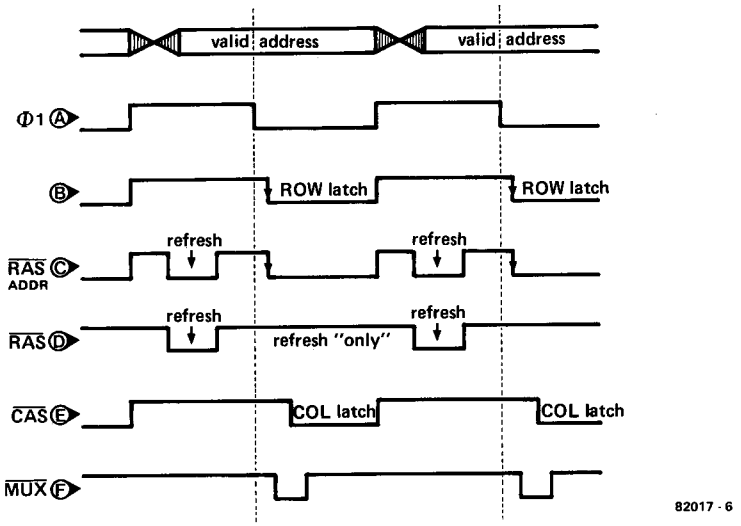
en uitlezen van de geheugencellen. Zo'n sense amplifier is een soort flipflop waarbij op elke ingang de helft van een kolom is aangesloten. Als dan een rijadres wordt toegevoerd aan het IC wordt de data van alle kapaciteiten van die rij op een destruktieve manier uitgelezen, doordat elke kondensator ontlaadt via de ingang van de sense amplifier. De optredende spanningsverandering wordt door de flipflop "versterkt" en teruggevoerd naar de kolomlijn, zodat het originele (versterkte) logische nivo weer in de kondensator wordt teruggeschreven. De sense amplifier bevat nu dezelfde data als de uitgelezen (en meteen weer opnieuw ingelezen) kondensator. Deze metode zorgt er voor dat bij het "aanspreken" van een rij meteen alle in de kondensatoren opgeslagen logische nivo's van die rij ververst worden. Om een idee te krijgen van de kapaciteit van zo'n geheu-

5



Figuur 5. Het schema van de dynamische RAM-kaart. Met behulp van draadbruggen kan de kaart worden aangepast voor het gebruik bij verschillende microprocessoren.

6



Figuur 6. Dit impulsdiagram laat nog eens goed het verloop van de belangrijkste signalen in het schema zien. De letteraanduidingen zijn ook in het schema bij de korresponderende lijnen aangegeven.

gen-kondensator: bij een 4116 is dat ongeveer 0,04 pF!
Na deze beschrijving zal het zo langzamerhand min of meer duidelijk zijn in welke volgorde de verschillende signalen moeten worden aangeboden. Bij het uitlezen van een geheugenplaats wordt eerst een 7-bits adres op de adresingangen gezet. Dan wordt een impuls gegeven op de $\overline{RAS}$-ingang. Het rij-adres moet gedurende een bepaalde tijd aanwezig zijn. Daarna kan het 7-bits kolomadres worden aangeboden, waarna een impuls op de $\overline{CAS}$-ingang wordt gegeven. Ook het kolom-adres moet minstens gedurende een voorgeschreven tijd aanwezig zijn. Via een interne uitgangsbuffer verschijnt dan het logische nivo van het gekozen adresbit aan de datauitgang. Tijdens deze procedure moet op de $\overline{WRITE}$-ingang een logische één staan.
Bij het schrijven gebeurt (bijna) hetzelfde. Alleen wordt nu van te voren een logisch nivo aangeboden op de dataingang en wordt op de $\overline{WRITE}$-ingang een logische nul gezet. Het tijdvolgordediagram in figuur 3 maakt het een en ander nog eens grafisch duidelijk.

Refresh: het noodzakelijke verversen

Zoals we al hebben beschreven, heeft het gebruik van kondensatoren als geheugenelementen niet alleen voordelen, maar ook een groot nadeel. De lading van zo'n kondensator lekt namelijk langzaam maar zeker weg, wat tot gevolg heeft dat het in een kondensator opgeslagen logische nivo na enige tijd verloren gaat. Vóór dat verloren gaan moet het betreffende nivo dus vernieuwd of ververst worden. Bij de 4116 moet dat minstens éénmaal in de 2 ms plaats vinden. En dat is nog een respektabele tijd, rekening houdend met een celkapaciteit van 0,04 pF.
Dat verversen is gelukkig betrekkelijk eenvoudig, wat vooral te danken is aan de opbouw van het IC, waarbij midden tussen de cellen de leesversterkers zijn geplaatst. Bij de bespreking van het IC hebben we al gezien wat de taak is van de sense amplifiers: voornamelijk het versterken van de logische nivo's in de geheugencellen. Als het rij-adres wordt ingelezen door het geven van een $\overline{RAS}$-puls, wordt een hele rij van 128 bits in de sense amplifiers gelezen. Daarbij worden de logische nivo's versterkt en weer teruggeschreven in die 128 rijkapaciteiten. Dat betekent dus dat na het geven van een rij-adres en een $\overline{RAS}$-puls 128 bits "gerefreshed" zijn. Als we er voor zorgen dat deze procedure (het aanbieden van een rij-adres en het geven van een $\overline{RAS}$-puls) voor alle rijen wordt doorlopen binnen 2 ms, dan zijn we er van verzekerd dat de data in het IC behouden blijft.
Het is natuurlijk niet nodig voor het refreshen die 2 ms aan te houden, het mag ook korter zijn. Bij de hier beschreven RAM-kaart wordt gebruik gemaakt van de clock van de gebruikte processor. Voor een junior computer (1 MHz) betekent dit dat alle 128 rijen in 128 μs ververst zijn.

De timing

In figuur 3 staan de tijdvolgordediagrammen voor respektievelijk lezen, schrijven en verversen. In deze diagrammen is duidelijk te zien in welke volgorde de diverse signalen moeten worden aangeboden. Daarbij zijn de verschillende tijden aangegeven waarmee bij het werken rekening moet worden gehouden. De waarden voor die tijden zijn hier niet gegeven, omdat die van fabrikant tot fabrikant enigszins verschillen. Ook zijn deze tijden afhankelijk van het toegepaste type. De beschrijving van de volgorde is al gegeven bij de behandeling van het IC.

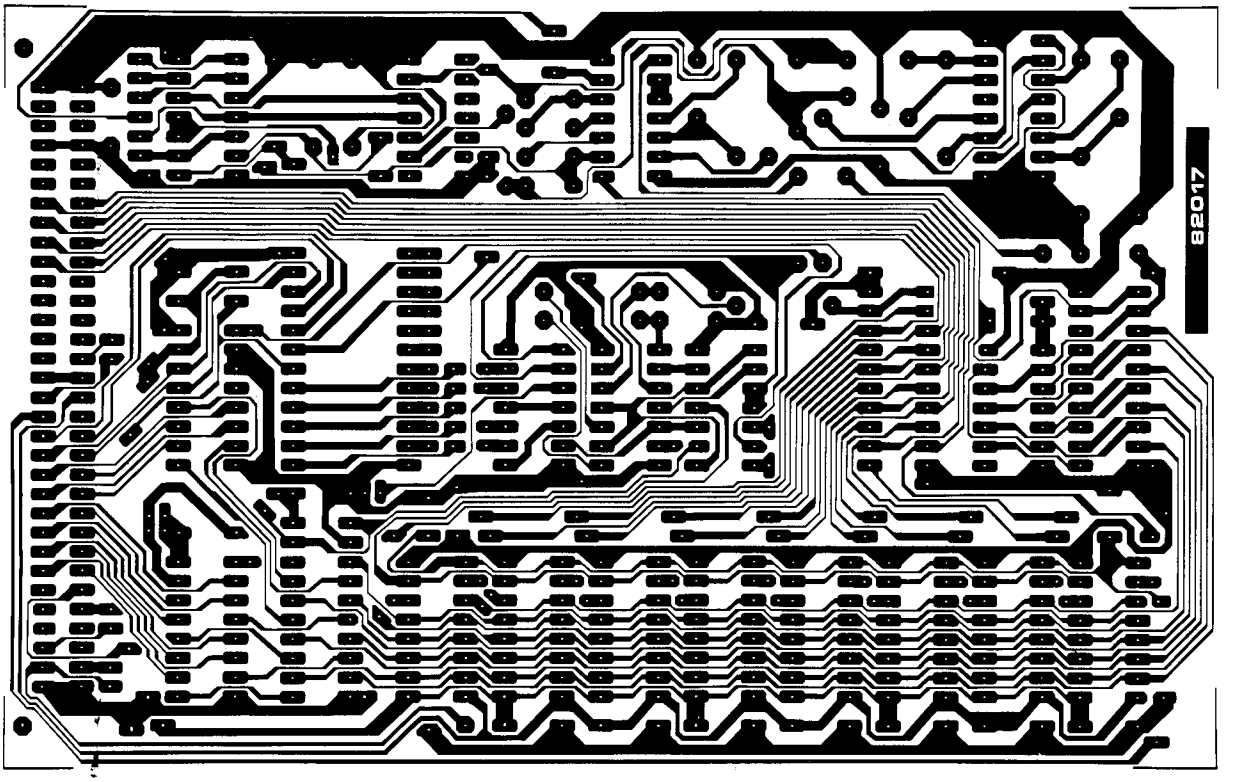
Voeding: rotsvast voor piekstromen

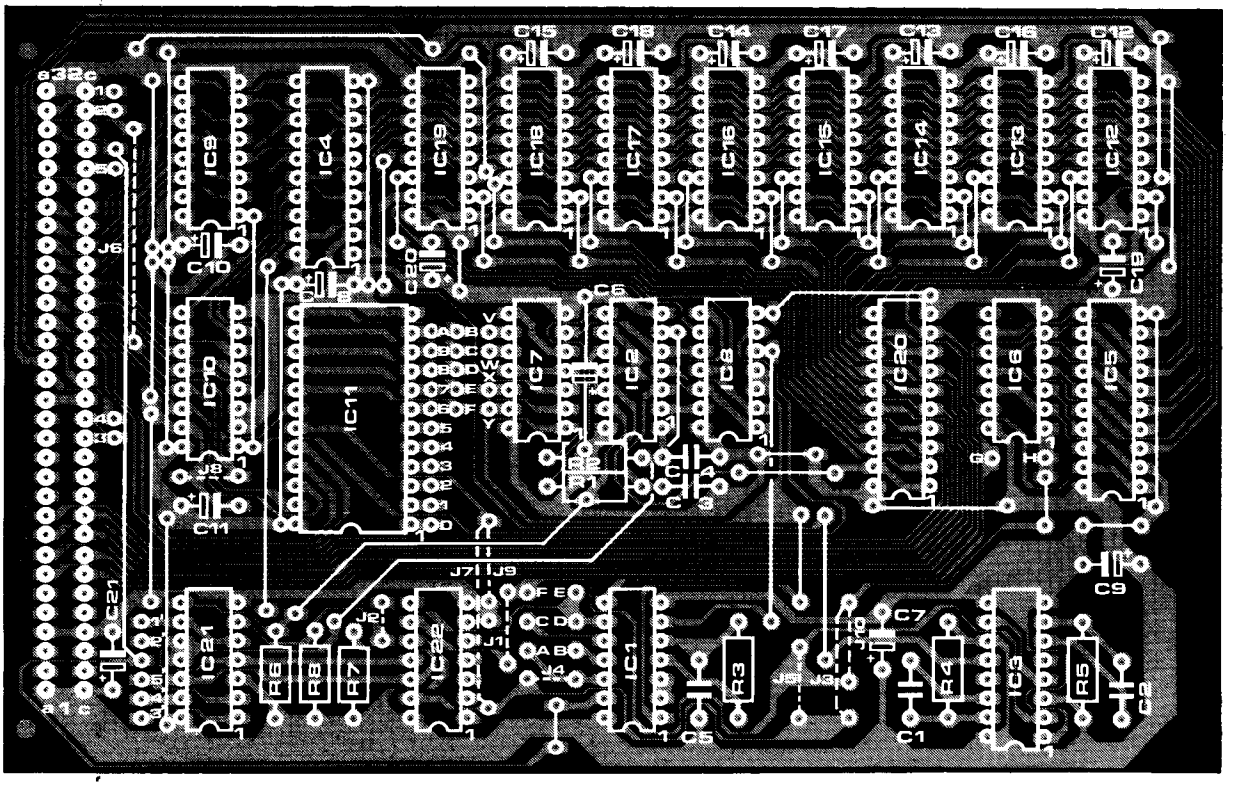
Een bijzonder punt van aandacht verdient het voedingsgedeelte voor de dynamische RAM's. Als je naar de opgegeven cijfers kijkt van de gemiddelde stroomopname voor de drie voedingsspanningen, dan is die vrij laag. Bij een dynamische RAM vindt het grootste stroomverbruik plaats bij het geven van een flank op de $\overline{RAS}$- en/of $\overline{CAS}$-ingang. In figuur 4 is daar een voorbeeld van gegeven. Daaruit blijkt dat rond die flanken naar verhouding erg grote stroompieken voorkomen. Deze pieken kunnen waarden bereiken van zo'n 100 mA (per IC)!
Bij de opzet van een dynamisch RAM-systeem moet met deze stroompieken terdege rekening worden gehouden. Het is hierbij niet nodig de voeding zo groot te dimensioneren dat deze al die piekstromen zonder problemen kan leveren. Voldoende bufferen van de voeding in de nabijheid van de RAM's, met behulp van kondensatoren, kan voldoende zijn als dat op een doeltreffende manier wordt gedaan. Het is echter wel noodzakelijk als men verzekerd wil zijn van een goede werking.

Het schema

In figuur 5 is het schema afgebeeld van een kompleet 16 Kbyte-geheugen.
IC12 . . . IC19 vormen samen het 16 k x 8 k bit dynamisch geheugen. De dataingangen van de IC's zijn rechtstreeks verbonden met de data-aansluitingen op de konnektor (links in de tekening). De data-uitgangen zijn op de data-lijnen aangesloten via tri-state buffers. De adreslijnen AØ . . . A13 gaan naar IC9 en IC10, die elk vier twee-naar-een-multiplexers bevatten. De veertien aangeboden adreslijnen worden hierdoor gemultiplext naar zeven lijnen, die via de tri-state buffers N11 . . . N17 zijn verbonden met de adres-ingangen van de RAM's. De adres-lijnen A12 . . . A15 zijn aangesloten op de adresdekoder IC11, zodat het geheugen op een willekeurig adresbereik kan worden gelegd door het aanbrengen van draadbruggen tussen de uitgangen van IC11 en de poorten N27 en N28.
Voor de regelmatige verversing van de geheugens dient IC6, die als 7-bits teller geschakeld is. De uitgangen van het IC zijn via tri-state buffers (N20 . . . N26) ook op de adres-ingangen van de RAM's aangesloten. In de tijd dat de processor de adresbus niet gebruikt, wordt gerefreshed. Via de poorten N1, N2 en N3 zijn de clock-ingang van de teller IC6 en de stuur-ingangen van de tri-state buffers N20 . . . N26 verbonden met de clock Φ1 van het processorsysteem. Bij elke processor zijn er gedeelten van de clock dat het geheugen zeker niet aangesproken wordt. In het geval van de junior computer is dat het positieve gedeelte van Φ1. De getekende draadbruggen gelden, tussen haakjes, voor de junior computer en in de rest van de schemabeschrijving zullen we dan ook deze situatie beschrijven. Dat positieve deel van Φ1 kan nu worden gebruikt voor het refreshen. Aan de hand van het

7





Figuur 7. Layout en komponentenopstelling van de print voor het dynamische geheugen. Voor de draadbruggen dient men tabel 1 te raadplegen.

bij de bouw IC-voetjes aan, om destruktieve gevolgen van het solderen bij voorbaat uit te sluiten. Gebruik een soldeerbout die geschikt is voor het fijne printwerk en soldeer goed. Een slechte soldeerverbinding is bijzonder moeilijk op te sporen.

Als de kaart wordt gebruikt bij de junior computer zullen de benodigde voedingsspanningen reeds aanwezig zijn. Bij andere processorsystemen zal men met behulp van geïntegreerde spanningsregelaars de ontbrekende voedingsspanningen er nog bij moeten maken. Schema's van dergelijke voedingen zijn er in rijke mate in Elektuur gepubliceerd.

Het testen

Voordat de voedingsspanningen op de kaart worden aangesloten, dient eerst te worden gekontroleerd of alles goed is opgebouwd. Dan kan de kaart op de bus van het processorsysteem worden gestoken.

De volgorde waarin de voedingsspanningen worden aangelegd is niet kritisch, hoewel de fabrikant aanbeveelt de −5 V het eerst aan te sluiten. Dit is echter alleen van belang voor het verkrijgen van een extra veiligheidsmarge ten aanzien van overspanningen op de voeding bij het inschakelen. Gewoonlijk (bij het gebruik van een goede voeding) hoeft men daar geen rekening mee te houden.

Als alles goed is, zal het geheugen direkt na het inschakelen van de voeding goed werken. Die goede werking is van buiten af wel moeilijk waar te nemen, omdat men daarvoor alle geheugenplaatsen zou moeten in- en uitlezen en de resultaten met elkaar vergelijken. Daarom is in tabel 3 nog een programma voor de junior computer gegeven, waarmee de RAM-kaart getest kan worden. Het programma kan ook gebruikt worden om andere RAM-geheugens mee te testen. Nadat men het programma ingegeven heeft, kan op de adresplaatsen ØØØØ (= ADL) en ØØØ1 (= ADH) het beginadres van het te testen geheugenbereik worden gezet en op de plaatsen ØØØ2 (= ADL) en ØØØ3 (= ADH) het eindadres. Het programma wordt daarna gestart op adres ØØØ4, waarna wordt begonnen met het volschrijven van het geheugenbereik met ØØ. Dan gaat het programma kijken of er ook werkelijk ØØ op het eerste adres van het te testen bereik staat. Is dat zo, dan wordt er achtereenvolgens Ø1, Ø2, Ø4, Ø8, 1Ø, 2Ø, 4Ø en 8Ø op dat adres geschreven en meteen daarna weer uitgelezen. Dit betekent dat elk bit van het adres eenmaal logisch één is geweest. Hierna wordt in het adres FF gezet. Dit laatste wordt gedaan om adresseerfouten op te sporen. Als er namelijk ergens een adresseerfout is, wordt deze FF ook nog op een ander adres geschreven. Aan de hand daarvan kan later die adresseerfout worden gekonstateerd als ergens FF wordt uitgelezen.

De hierboven beschreven procedure wordt nu voor elk adres afgewerkt, totdat het programma aan het einde van het te testen bereik komt. Daarna wordt het hele testprogramma nog eens uitgevoerd, waarbij het bereik weer eerst met ØØ wordt volgeschreven en dan van achteren naar voren wordt doorlopen. Ook dat laatste is weer nodig om eventuele adresseerfouten op te sporen.

Als er geen enkele fout wordt gevonden verschijnt na afloop van het programma adres ØØØØ op het display met daarachter het lage adresbyte van het ingegeven startadres. Als het programma een fout ontdekt, wordt het adres op het display getoond waar de fout is ontdekt, met daarachter de (foute) inhoud van dat adres. Als men daarna verder wil gaan met testen vanaf het punt waar een fout was gevonden, dan kan men het programma weer starten op adres ØØØA.

```
0010: 0004                    ORG   $0004
0020:
0030:
0040:                 *** RAM TEST PROGRAM ***
0050:
0060:
0070:                 DEFINITIONS
0080:
0090: 0004            BEG     *     $0000  BEGIN OF MEMORY
0100: 0004            END     *     $0002  END OF MEMORY
0110: 0004            CUR     *     $00E6  CURRENT ADDRESS POINTER
0120: 0004            POINT   *     $00FA  MONITOR'S ADDRESS POINTER
0130: 0004            PATTER  *     $00E5  CURRENT TEST PATTERN
0140: 0004            MONITO  *     $1C1D
0150:
0160:
0170: 0004 20 45 00   RAMTST  JSR   WRZERO FILL WORKSPACE WITH $00
0180: 0007 20 54 00           JSR   CURBEG CUR = BEG
0190:
0200: 000A 20 84 00   TSTA    JSR   WALK   WALKING BIT ROUTINE
0210: 000D D0 2B              BNE   TSTC   BRANCH IF MEMORY CELL IS DEFECT
0220: 000F A9 FF              LDAIM $FF    TEST PATTERN FOR DOUBLE ADDRESSING
0230: 0011 91 E6              STAIY CUR
0240: 0013 20 5D 00           JSR   INCCHK INCREMENT AND CHECK CUR
0250: 0016 B0 F2              BCS   TSTA   TEST FINISHED?
0260: 0018 20 45 00           JSR   WRZERO FILL WORKSPACE WITH $00
0270: 001B A6 02              LDX   END    CHECK FROM BOTTOM TO TOP
0280: 001D 86 E6              STX   CUR
0290: 001F A6 03              LDX   END    +01
0300: 0021 86 E7              STX   CUR    +01
0310:
0320: 0023 20 84 00   TSTB    JSR   WALK
0330: 0026 D0 12              BNE   TSTC   BRANCH IF MEMORY CELL IS DEFECT
0340: 0028 A9 FF              LDAIM $FF    TEST PATTERN FOR DOUBLE ADDRESSING
0350: 002A 91 E6              STAIY CUR
0360: 002C 20 6D 00           JSR   DECCHK DECREMENT AND CHECK CUR
0370: 002F B0 F2              BCS   TSTB
0380: 0031 A9 00              LDAIM $00    DISPLAY "0000 XX" IF
0390: 0033 85 FA              STA   POINT  MEMORY IS O.K.
0400: 0035 85 FB              STA   POINT  +01
0410: 0037 4C 1D 1C           JMP   MONITO
0420:
0430: 003A A5 E6      TSTC    LDA   CUR    DISPLAY THE ADDRESS OF
0440: 003C 85 FA              STA   POINT  THE DEFECT MEMORY CELL
0450: 003E A5 E7              LDA   CUR    +01
0460: 0040 85 FB              STA   POINT  +01
0470: 0042 4C 1D 1C           JMP   MONITO
0480:
0490:
0500:                 SUBROUTINES
0510:
0520:
0530: 0045 20 54 00   WRZERO  JSR   CURBEG FILL THE MEMORY BETWEEN BEG & END
0540: 0048 A0 00              LDYIM $00    WITH $00
0550:
0560: 004A A9 00      WRZ     LDAIM $00
0570: 004C 91 E6              STAIY CUR
0580: 004E 20 5D 00           JSR   INCCHK
0590: 0051 B0 F7              BCS   WRZ
0600: 0053 60                 RTS
0610:
0620: 0054 A6 00      CURBEG  LDX   BEG    CUR = BEG
0630: 0056 86 E6              STX   CUR
0640: 0058 A6 01              LDX   BEG    +01
0650: 005A 86 E7              STX   CUR    +01
0660: 005C 60                 RTS
0670:
0680: 005D E6 E6      INCCHK  INC   CUR    CUR = CUR+01
0690: 005F D0 02              BNE   IA
0700: 0061 E6 E7              INC   CUR    +01
0710:
0720: 0063 38         IA      SEC          C=0 IF CUR >END
0730: 0064 A5 02              LDA   END
0740: 0066 E5 E6              SBC   CUR
0750: 0068 A5 03              LDA   END    +01
0760: 006A E5 E7              SBC   CUR    +01
0770: 006C 60                 RTS
0780:
0790: 006D 38         DECCHK  SEC          CUR = CUR -01
0800: 006E A5 E6              LDA   CUR
0810: 0070 E9 01              SBCIM $01
0820: 0072 85 E6              STA   CUR
0830: 0074 A5 E7              LDA   CUR    +01
0840: 0076 E9 00              SBCIM $00
0850: 0078 85 E7              STA   CUR    +01
0860: 007A 38                 SEC          C=0 IF CUR < BEG
0870: 007B A5 E6              LDA   CUR
0880: 007D E5 00              SBC   BEG
0890: 007F A5 E7              LDA   CUR    +01
0900: 0081 E5 01              SBC   BEG    +01
0910: 0083 60                 RTS
0920:
0930: 0084 A9 01      WALK    LDAIM $01    INIT. PATTERN
0940: 0086 85 E5              STA   PATTER
0950: 0088 A0 00              LDYIM $00
0960: 008A B1 E6              LDAIY CUR    IS STILL $00 IN THE CELL
0970: 008C D0 0F              BNE   WALKB  IF NOT, THEN BRANCH
0980: 008E A2 08              LDXIM $08    WALKING BIT COUNTER
0990:
1000: 0090 A5 E5      WALKA   LDA   PATTER CURR. PATTERN INTO ACCU
1010: 0092 91 E6              STAIY CUR    STORE IT IN MEMORY
1020: 0094 D1 E6              CMPIY CUR    DOES IT MATCH?
1030: 0096 D0 05              BNE   WALKB  IF NOT, THEN BRANCH
1040: 0098 06 E5              ASL   PATTER WALKING BITS!
1050: 009A CA                 DEX
1060: 009B D0 F3              BNE   WALKA
1070:
1080: 009D 60         WALKB   RTS
1090:
1100:
```

Tabel 3. Programma voor het testen van het geheugen. Het begin- en eind-adres van het te testen gedeelte worden op de adressen ØØØØ . . . ØØØ3 gezet. Het programma start op adres ØØØ4.

mini-EPROM-kaart

small is beautiful (sometimes)

Nu eens geen geheugenkaart op Euro-, maar op "Luxemburg"-formaat. Waarom zo'n klein printje, slaat de bezuinigingswoede ook hier toe? Nee hoor, gewoon een elegant en goedkoop alternatief voor junior-uitbreiders die aan met name RAM-geheugenuitbreiding denken (bijvoorbeeld voor de junior-Basic of een grote assembler) en de hoeveelheid "vast" geheugen tot het noodzakelijke minimum willen beperken.

Lekker klein (die figuur 2), vindt u ook niet? En toch nog altijd goed voor 2048 bytes EPROM. Waarom moet dat eigenlijk zo nodig? Wel, in boek 3 heeft u kunnen lezen dat, in het geval van geheugenuitbreiding op de buskaart, de drie vektoren NMI, RES en IRQ uit een op de buskaart en op pagina FF aangesloten EPROM moeten worden opgehaald. In aanhangsel 3 van boek 3 is aangegeven hoe je dat kunt doen met een PROM 82S23. Nu is een 2716 nauwelijks duurder dan een 82S23, àls-ie al duurder is. Een 2716 is gemakkelijker te programmeren; zie hiervoor het januarinummer 1982. En als er verder dan óók nog sprake is van een echte print, in plaats van een doe-het-zelf-toestand op veroboard, en als je met ongeveer dezelfde hoeveelheid hardware toch nog 2K EPROM in huis haalt, nou, waar wachten we dan nog op!?

Het schema van de mini-EPROM-kaart staat in figuur 1. De EPROM, IC2, wordt geselekteerd door IC1. Het geheugenbereik omvat de adressen $F8ØØ t/m $FFFF. Men kan met twee draadbruggen kiezen tussen selektie van de EPROM via pen $\overline{OE}$ of pen $\overline{CE}$. Selektie via de Output Enable levert een iets snellere werking van de EPROM, maar een hoger stand-by-stroomverbruik, terwijl selektie via Chip Enable een besparing van 300% in stroomverbruik oplevert ten koste van een iets tragere werking. U mag zelf kiezen, als u er maar voor zorgt dat de ene enable-pen aan massa ligt en de andere aan de uitgang van IC1.

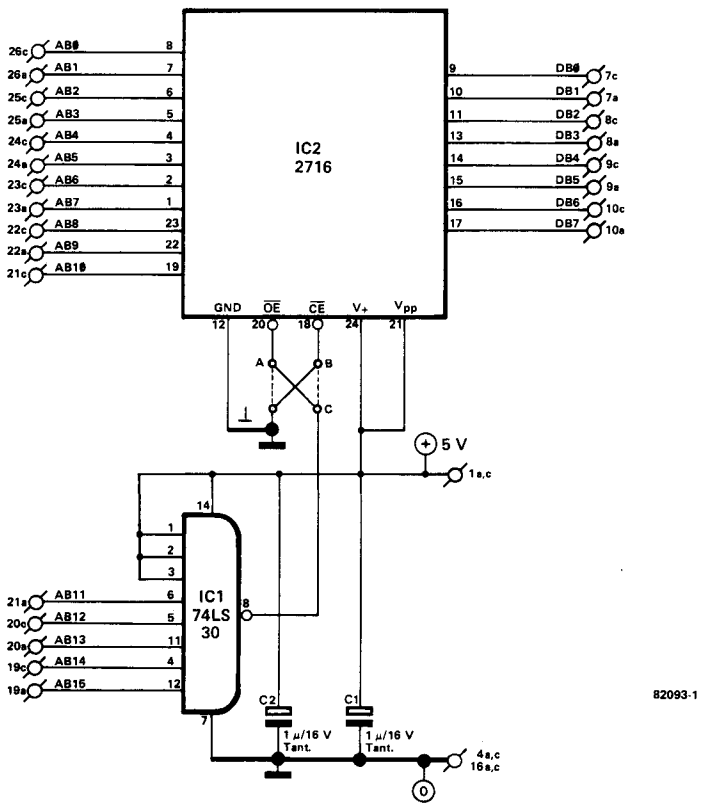
Voor de volledigheid wordt hier nog even de data vermeld, die op de hoogste zes geheugenplaatsen moet worden geprogrammeerd:

adres $FFFA data 2F,
adres $FFFB data 1F,
adres $FFFC data 1D,
adres $FFFD data 1C,
adres $FFFE data 32;
adres $FFFF data 1F,

De overige 2042 geheugenplaatsen staan u vrij ter beschikking.

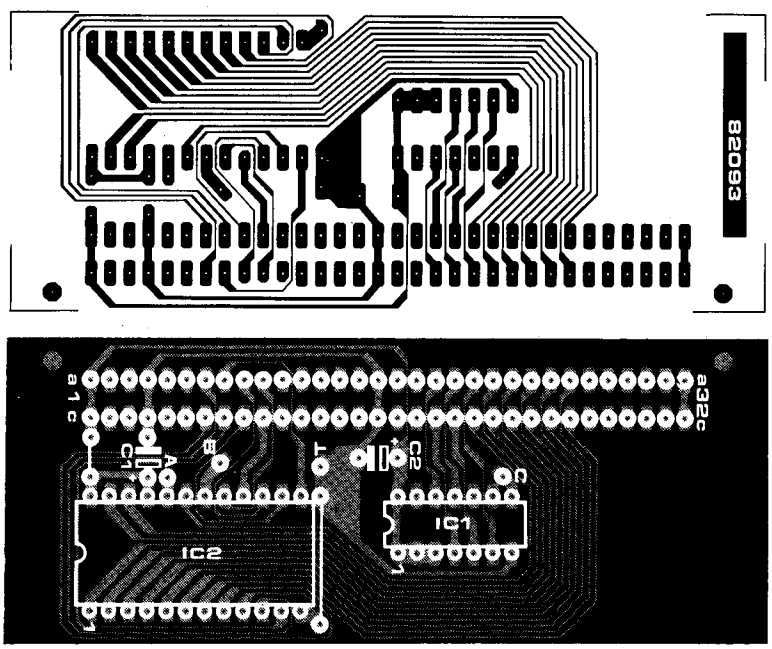
Tja, méér valt er niet over te zeggen. Dus dat doen we dan ook maar niet. ◀

1



Figuur 1. Het schema van de mini-EPROM-kaart.

2



Figuur 2. De mini-EPROM-kaart, vanuit twee onderling tegenovergestelde richtingen gezien.

Onderdelenlijst

Kondensatoren:
C1,C2 = 1 µ/16 V tantaal

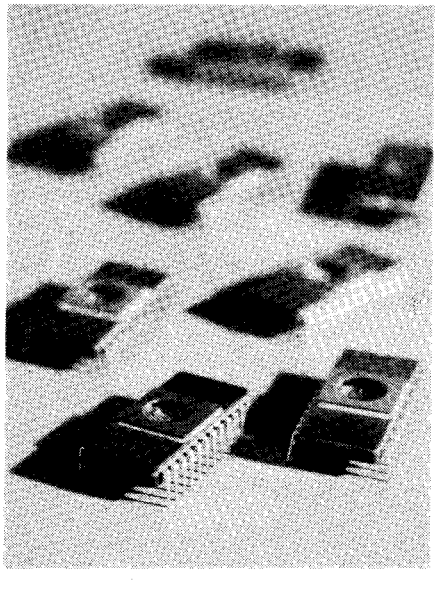
Halfgeleiders:
IC1 = 74LS30
IC2 = 2716

Diversen:
1 print EPS 82093
1 konnektor 64-polig male, a/c

PSS . . . Privé Software Service

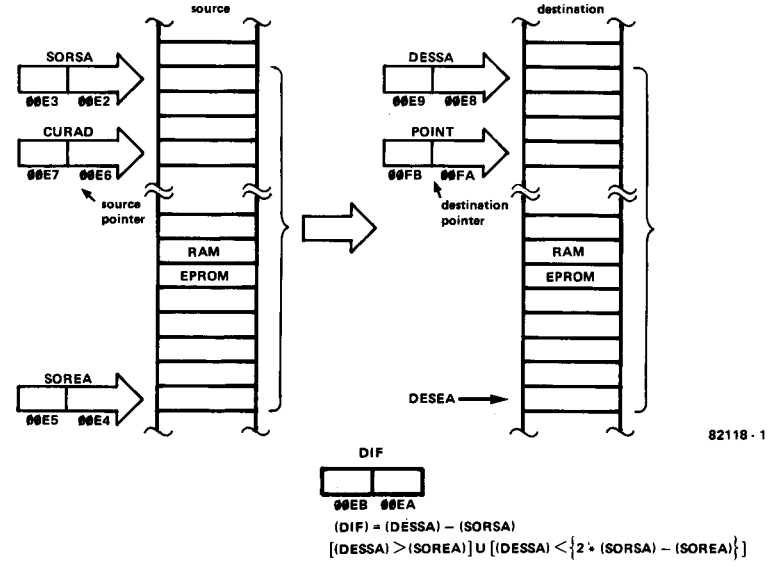
een programma om (EPROM's) te programmeren

In den beginne was daar de EPROMmer (januari j.l.). Toen was er een voor driekwart volle ESS 511-EPROM (april j.l.). Nu is-ie helemaal vol. Aangevuld met handige junior-software voor het programmeren van EPROM's.
Kortom: een stukje ESS dat ESS overbodig maakt. Oftewel: van ESS naar PSS.



Op pagina 5-39 zijn we bij adres $FDD9 van de ESS511-EPROM blijven steken, nu gaan we verder vanaf $FDDA. Dat is het startadres van een systeemprogrammaatje, genaamd EPROM PROGRAMMING UTILITIES (afgekort: EPRUTL). Met dit programma èn de EPROMmer-hardware van januari j.l. kunt u EPROM's programmeren; hetzij met RAM-data, hetzij met EPROM-data (kopiëren, dupliceren). Maar dat is nog niet alles. Vooraf kan worden gekontroleerd of de EPROM wel programmeerbaar is, dus of er sprake is van data FF. En verder kunnen, indien nodig, alle daarvoor in aanmerking komende absolute adressen binnen het te programmeren datablok worden aangepast aan de positie van dat datablok in de EPROM (relocating). Na het programmeren kan worden geverifieerd dat het programmeren goed is verlopen.
Hoe gaat het allemaal in zijn werk? Na het opstarten via PM (startadres $FDDA) wordt de naam van het programma afgedrukt, alsmede een overzicht van geldige toetsen. Daarna is het zinvol om de *parametertoets P* in te drukken. U moet namelijk drie adressen opgeven (zie figuur 1). Allereerst "FIRST, LAST SOURCE ADDRESS:": de adressen SORSA en SOREA die de begrenzing vormen van het te programmeren of te verplaatsen datablok. S.v.p. SOREA groter dan SORSA, anders moet u opnieuw beginnen (procedure: eerste adres-komma-laatste adres-CR). Vervolgens "FIRST DESTINATION ADDRESS:": het beginadres DESSA, bestemming van het eerste adres van het te programmeren of te verplaatsen datablok (procedure: eerste adres-CR).
Een overzicht van de toetsfunkties:
Toets M (Move). Indrukken ervan zorgt ervoor dat het datablok SORSA t/m SOREA wordt gekopieerd of geprogrammeerd (dat laatste indien de EPROMmer is aangesloten en op programmeren is ingesteld — hierover later meer) in het bestemmingsblok DESSA t/m DESEA. Mede in verband met de funktie van toets V mogen de beide blokken elkaar niet overlappen. De inhoud van de drie adrespointers moet aan bepaalde eisen voldoen: zie die toestand met allerlei haakjes in figuur 1.
Na afloop volgt de terugmelding "DATA MOVED".
Toets F (FF-check). Na het indrukken of aantippen van F wordt er gekeken of de geheugenplaatsen DESSA t/m DESSA+n-1, met als n het aantal geheugenplaatsen van het te programmeren datablok, een inhoud FF hebben, dus of er kan worden geprogrammeerd. Van geheugenplaatsen met een van FF afwijkende inhoud wordt het adres en de inhoud afgedrukt. Na het kontroleren van alle n geheugenplaatsen volgt de terugmelding "DATA COMPARED".

1



Figuur 1. Deze geheugenplaatjes horen bij het systeemprogramma EPRUTL. Na het indrukken van toets M, R, V of F doorlopen de pointers CURAD resp. POINT alle posities vanaf SORSA resp. DESSA tot en met SOREA resp. DESEA.

software-uitpluizer

een disassembler voor de junior-computer

Het ontwerpen van software is een avontuur, het ontleden ervan vaak een ontdekkingsreis. Tijdens die reis is een kompas onontbeerlijk: een disassembler-programma, waarmee losse bytes worden verwerkt tot instrukties, inklusief gedetailleerde operand-informatie. De software krijgt onderdak in een EPROM en is in het kader van de ESS beschikbaar.

De redaktie krijgt nogal wat junior-programma's toegezonden. Een gezonde dosis scepsis gebiedt ons om alles uit te proberen. Dus toetsen we de data in (gaat zeer snel met PME) en pluizen het programma uit, met de disassembler. Het resultaat is overigens niet altijd dat van een ontdekkingsreis, maar heeft ook wel eens iets te maken met de werkzaamheden van Dr. J. Zeldenrust, patholoog-anatoom.
Een ander voorbeeld. In het maartnummer is beschreven hoe je van een KIM-Basic een junior-Basic kunt maken.

Zonder disassembler zou het haast onbegonnen werk zijn om deze wijzigingen te realiseren.
Ten derde. Een disassembler bewijst u goede diensten bij de dokumentatie van uw ego-software, eventueel als voorstudie voor dokumentatie met behulp van een grote editor/assembler.

Meer bijzonderheden

De software-uitpluizer zit in een EPROM van het type 2716, die in het kader van ESS 511 van de betreffende data kan worden voorzien. Het gaat om de adressen $F8ØØ . . . $FFFF. De EPROM kunt u op een RAM/EPROM-kaart zetten of op de elders in dit nummer beschreven mini-EPROM-kaart. De geheugenplaatsen $F8ØØ . . . $FDD9 bevatten de software-uitpluizer-software, $FDDA . . . $FFF9 "EPROM PROGRAMMING UTILITIES" (zie het artikel "Privé Software Service" op pagina 5-62) en $FFFA . . . $FFFF, de bekende vektor-data.
Wat doet de software-uitpluizer eigenlijk? Eerst maar het disassembler-gedeelte ervan. De aanwezigheid van tabel 1 maakt een bondige beschrijving mogelijk. Na het opstarten (startadres $FC4E via PM opgeven!) volgt een terugmelding met een opsomming van de relevante funktietoetsen. Met toets D geeft men twee adressen op die de begrenzing vormen van het te disassembleren geheugenbereik (afsluiten met CR). In het voorbeeld van tabel 1 dus van $Ø2ØØ tot en met $Ø22F. Merk op dat het laatste adres moet worden opgegeven; geen toestanden met (laatste adres + 1), zoals je die wel eens tegenkomt.
Na het opgeven van de twee adressen volgt de terugmelding "L,P,SP?". Door het indrukken van toets L kunt u de hele handel ineens disassembleren, met toets P in blokken van 15 instrukties (een vol TV-scherm, waarvan de bovenste regel als laatste vóór het indrukken van P is afgedrukt), en met de spatiebalk SP kunt u het instruktie voor instruktie, dus regel voor regel doen.
Het uitgeplozen "programma" van tabel 1 is bedoeld als representatief voorbeeld van wat er zo al wordt afgedrukt. Dat is in ieder geval het adres met de opcode van de instruktie en verder het byte of de bytes van de instruktie, na een aantal spaties gevolgd door de mnemonics ("instruktie-steno"). Indien van toepassing volgt als laatste de operand-informatie. U ziet dat offsets van voorwaardelijke sprongінstrukties zijn vertaald in het sprongadres.
Aan data die niet wordt herkend als de opcode van een instruktie wordt een mnemonic toegekend die bestaat uit drie "croissants" (het Amerikaanse AT-symbool @). Tevens wordt aan dergelijke data een instruktielengte één toegekend. Merk op dat de data FF niet wordt herkend als label-opcode.
Nu toets R. Druk hem in, en je bent terug in PM. Het gemak dient de mens!
Blijven over de toetsen H en A. Het indrukken van H is hetzelfde als het indrukken van toets M tijdens het verblijf in PM. Kortom: Na het opgeven van twee adressen, afgesloten met CR, wordt een hex dump afgedrukt. Waarom deze doublure? Omdat de daarvoor benodigde software tòch aanwezig moet zijn vanwege toets A. Die A staat voor "ASCII-dump". Wat is dat nou weer? Dat is een hex dump, waarbij de eventueel af te drukken data wordt geïnterpreteerd als de ASCII-kode van een afdrukbaar karakter. Indien de data behoort tot het gebied $2Ø tot en met $7E, wordt het overeenkomende ASCII-karakter afgedrukt. Zoniet, dan worden twee spaties, dus niets afgedrukt. Het hogere nut hiervan? Men kan in de te onderzoeken software snel af te drukken zaken zoals terugmeldingen lokaliseren. Indien u de software-uitpluizer-software met behulp van de software-uitpluizer gaat uitpluizen (hallo, bent u er nog?), dan zult u merken dat er hier en daar rijkelijk is gestrooid met "boodschappen".

En verder nog dit:

Men kan het afdrukken van een dump of listing onderbreken door de BRK-toets in te drukken. De BRK-sprongvektor voert de 6502 naar een centraal punt, waar gewacht wordt op een (nieuwe) door u in te drukken toets.
Bij de opgave van de twee adressen die de begrenzing vormen van de listing of de dump moet het tweede adres hoger zijn dan het eerste. Zoniet, dan moet u die twee adressen opnieuw opgeven. En nu wèl goed graag!
Naast een aantal bekende geheugenplaatsen van de pagina's ØØ en 1A doet de software-uitpluizer een beroep op de geheugenplaatsen $ØØ1Ø . . . $ØØ27. Voor de resterende ESS 511-software komt daar nog $ØØ28 bij. Zorg ervoor dat het uit te pluizen programma deze geheugenplaatsen niet gebruikt.
Wij wensen u veel plezier met deze gereedschappen, die u beter in staat stellen om software-amateur-detektiefje te spelen. Weer eens iets anders dan Tatort.

Tabel 1.

```
FC4E
FC4E A9 R
VALID COMMANDS: A D H L P R SP

D
DISASSEMBLE: 200,22F
L,P,SP ?
L
0200 A9 00        LDA #$00
0202 AD 01 02     LDA $0201
0205 A5 03        LDA $03
0207 A1 04        LDA ($04,X)
0209 B1 05        LDA ($05),Y
020B B5 06        LDA $06,X
020D BD 07 08     LDA $0807,X
0210 B9 09 0A     LDA $0A09,Y
0213 B6 0B        LDX $0B,Y
0215 20 0C 0D     JSR $0D0C
0218 4C 0E 0F     JMP $0F0E
021B 6C 10 11     JMP ($1110)
021E 77           @@@
021F FF           @@@
0220 00           BRK
0221 00           BRK
0222 CA           DEX
0223 C8           INY
0224 E8           INX
0225 0A           ASL  A
0226 F0 12        BEQ $023A
0228 D0 FE        BNE $0228
022A B0 34        BCS $0260
022C 90 EE        BCC $021C
022E FA           @@@
022F 00           BRK
```

Toets R (Relocate). Alle absolute adressen *binnen* het datablok SORSA t/m SOREA worden aangepast aan de situatie, na het verplaatsen c.q. programmeren van dat blok. Bij de vaststelling van het nieuwe adres speelt de inhoud van DIF (zie figuur 1) een grote rol. Na afloop volgt de terugmelding "RELOCATED". Deze toetsfunktie is niet nodig indien er "relocatable" software wordt geprogrammeerd (geen interne JMP's en subroutines), en indien de inhoud van een EPROM wordt gedupliceerd in een andere EPROM. Van RAM naar EPROM? Eerst R, dan M. Van EPROM naar RAM? Eerst M, dan R.

Toets V (Verify). Het oorspronkelijke datablok en de verplaatste versie ervan worden byte voor byte vergeleken. Bij onderling afwijkende geheugeninhouden worden adres en inhoud van de desbetreffende geheugenplaats(en) van het bestemmingsblok afgedrukt. Na afloop volgt de terugmelding "DATA COMPARED".

Toets B (Back). Kun je mee terug naar PM. Gewoon omdat men klaar is of om tussentijds een verplaatst of geprogrammeerd datablok te disassembleren (verifikatie van de uitwerking van R).

Toets ST (ST/NMI) (standaard-toetsenbord). Kun je mee terug van PM naar EPRUTL. Is een soort warme start. Nu volgen niet de initiële blurps, maar de terugmelding:

"XXXX < =AD=<YYYY TO >=ZZZZ"
met XXXX: FIRST SOURCE ADDRESS
YYYY: LAST SOURCE ADDRESS
ZZZZ : FIRST DESTINATION ADDRESS

Trouwens, ST kun je ook tijdens het verblijf in EPRUTL indrukken. Die toets zorgt voor een tussentijds overzicht van de drie adresparameters. Reuze handig als men bedenkt dat sommige operaties tussentijdse wijziging van parameters vereisen en als men wil weten wat men drie TV-schermen geleden ook al weer had opgegeven.

Een handvol tips

1) De EPROMmer moet, net als de ESS511-EPROM, zijn aangesloten op de buskaart. Tijdens het programmeren wordt deze kaart normaal geadresseerd. Dit betekent dat, vlak voor het programmeren, een "FIRST DESTINATION ADDRESS" $2000 of hoger moet worden opgegeven (Waarom? Zie boek 3). Betekent dat nou, dat geen EPROM's kunnen worden geprogrammeerd die in de "memory map" onder $2000 zitten, zoals de standaard-, TM- en PM-EPROM? Nee, foute konklusie. Zie punt 3.

2) Op de EPROMmer moet met behulp van de schakelaars S3 . . . S6 een 4K-adresblok worden gekozen, en wel zodanig dat dit blok niet geheel of gedeeltelijk samenvalt met fysiek aanwezig geheugen. Dit in verband met problemen door dubbeladressering. Eventueel tijdelijk een paar geheugenkaarten uit de buskaart trekken. Ook de laagste twee 4K-blokken zijn verboden (zie punt 1).

3) Het vlak voor het programmeren opgegeven FIRST DESTINATION ADDRESS moet binnen het ingestelde 4K-blok (zie punt 2) liggen. Dit adres hoeft niet noodzakelijkerwijs het uiteindelijke eerste adres te zijn. Waar het om gaat is dat de op de EPROMmer aanwezige EPROM na het indrukken van toets M byte voor byte wordt beschreven. Echter, pas op! Moeten er absolute adressen worden aangepast dan geeft men eerst met P het *echte* FIRST DESTINATION ADDRESS op (vervolgens R, dan via P de EPROMmer-versie van dat eerste adres en tenslotte M).

4) Pas vlak voor het programmeren (toets M) opent men schakelaar S2 van de EPROMmer. Tijdens het programmeren licht LED D9 op. Dat duurt enige tijd (ca. 20 bytes per sekonde, dus reken maar na). Na het doven van D9 en de terugmelding "DATA MOVED" S2 direkt sluiten!

5) EPROM's van het type 2716 of 2732 ervaren het als buitengewoon onprettig als de 25 V-programmeerspanning er al is of er nog is als de 5 V-voedingsspanning er nog niet of niet meer is. Zoja, dan zijn ze voor de chipbegraafplaats geprogrammeerd. De aan de EPROMmer toe te voegen schakeling van figuur 2 voorkomt dat. Een goedkope levensverzekering.

6) Gewoon kijken of een 2716 leeg is? Dat kan. Kies een 4K-blok op de EPROMmer, kies verder een FIRST DESTINATION ADDRESS dat overeenkomt met het eerste adres van dat blok of 2048 geheugenplaatsen verder, en geef een willekeurig 2K-blok op. Druk toets F in en u weet hoe laat het is.

7) EPROM dupliceren? Zet de "master" op een RAM/EPROM-kaart, tenzij die een van de systeem-EPROM's is. Zet de hopelijk nog maagdelijke EPROM op de EPROMmer-kaart. Ga verder te werk volgens de punten 3 en 4, en na enige tijd is de datatransfusie een feit.

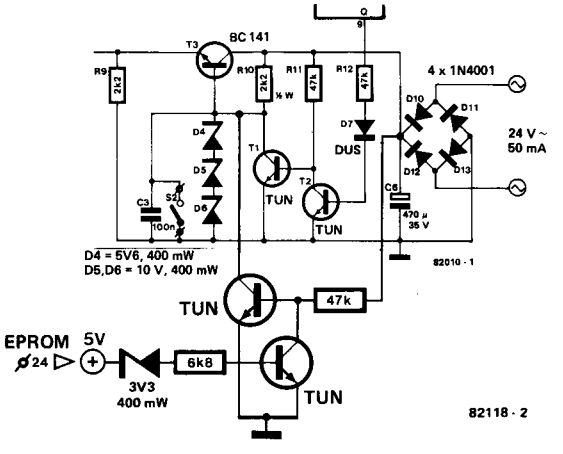
8) RAM-versie van EPROM? Ook dat is mogelijk. Een dergelijke operatie is zinvol als men een cassette-versie van een bepaald systeemprogramma wil hebben of als men de inhoud van een EPROM wil wijzigen. In dit geval moet men eerst de data kopiëren (M) en, indien van toepassing, pas daarna relocaten (R). Met toets V kan men nagaan welke geheugenplaatsen kwa inhoud zijn gewijzigd als gevolg van het indrukken van toets R.

9) Pas bij het gebruik van toets R op voor lookup- of teksttabellen ("strings")! De data . . . 20 41 54 . . . is wellicht de ASCII-kode van " AT", maar kan worden uitgelegd als JSR-$5441! Indien 54 een ADH is binnen het te programmeren datablok (dynamische RAM-kaart: $2000 . . . $5FFF) is de kans groot dat na het indrukken van R op de betreffende geheugenplaats geen 54 meer staat. Het is een goede zaak om vooraf na te gaan waar dit soort tabellen zit en om na het indrukken van R, maar vóór het indrukken van M te kontroleren of de tabellen ongewijzigd zijn. Met behulp van de disassembler kom je daar gauw achter.

10) Handwerk mogelijk, dus byte voor byte? Ja. Met de standaard-monitor is daarvoor een speciaal programma nodig, zie de EPROMmer-software van januari j.l. Met PM is dat niet nodig. Geef de te programmeren EPROM-geheugenplaats op (en wel de EPROMmer-versie daarvan, zie punt 3), druk de spatietoets in, geef de data op en druk toets "." in. Uiteraard moet de EPROMmer op programmeren zijn ingesteld, zie punt 4.

Tot zover de tips. U zult merken dat er met die handvol toetsfunkties verrassend veel uit te voeren is. Veel sukses!

2



Figuur 2. Een aanvulling van de EPROMmer-hardware die voorkomt dat de te programmeren EPROM overlijdt indien S2 is geopend èn indien de voeding voor de EPROMmer eerder wordt ingeschakeld en later uitgeschakeld dan (in dit geval) de voeding van de junior-computer. Een gemeenschappelijke netschakelaar biedt geen 100%-garantie!

leuk en toch simpel

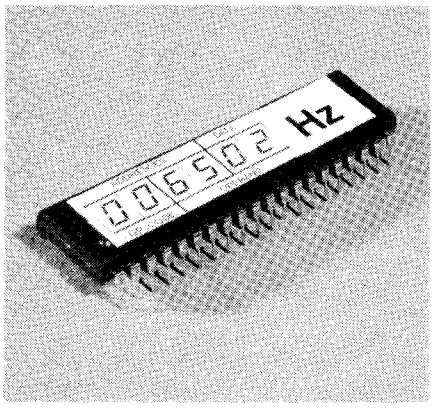
mini-teller met microprocessor

Waar de μP tegenwoordig al niet goed voor is! Je kunt het zo gek niet bedenken. Tellen kan-ie ook al. Bijvoorbeeld tellen hoe vaak iets gedurende een bepaalde tijd optreedt. Dat noemen ze "frekwentie".

G. Sullivan

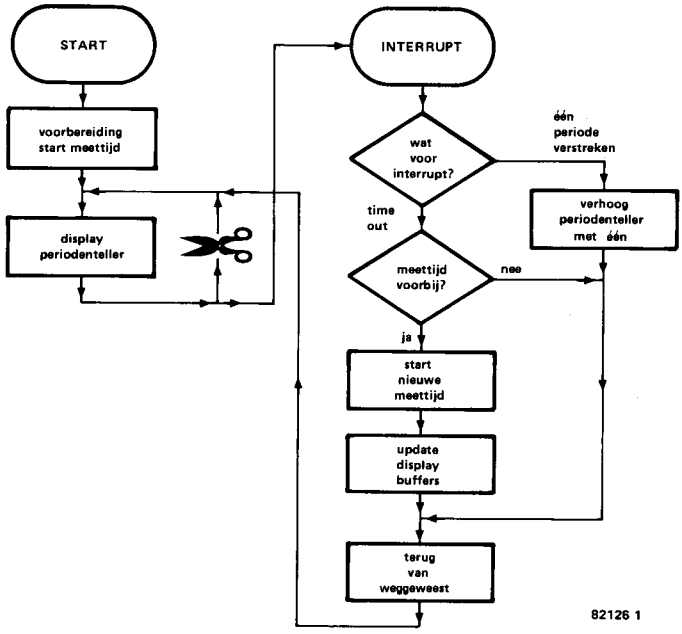
De kreet "frekwentie" is gekoppeld aan periodieke, dus regelmatig terugkerende verschijnselen. Zo kent de maand mei een frekwentie van 1 zonsondergang per etmaal (hetzelfde geldt trouwens voor de andere maanden). En als een wisselspanning 100 x per sekonde op een bepaalde manier van polariteit wisselt, spreken we van een frekwentie van 100 Hz. In de elektronica zijn we met name geïnteresseerd in de frekwentie van een wisselspanning of -stroom.
Hoe kun je de frekwentie meten? Door gedurende één sekonde het aantal polariteitswisselingen te tellen: òf van "plus" naar "min" òf juist omgekeerd.

En hoe doe je dat met een μP? Wel, zorg voor een programma dat aan één stuk door de inhoud van een aantal geheugenplaatsen laat zien. Wat je ziet is niets meer en niets minder dan de laatst gemeten frekwentie. Zorg er verder voor dat het genoemde programma kan worden onderbroken (interrupt) indien er òf een meettijd van 1 sekonde is verstreken, òf is vastgesteld dat de te meten wisselspanning op een bepaalde manier van polariteit is gewisseld. Het programma dat na een interrupt (onderbreking) wordt afgewerkt moet eerst nagaan wat de oorzaak van de onderbreking was. Ging het om een nuldoorgang, dan wordt de periodenteller met één verhoogd. Indien blijkt dat het ging om een verstreken 1 s-meettijd, wordt de inhoud van de geheugenplaatsen van de periodenteller gekopieerd in de displaybuffers; tevens volgt de start van een nieuwe meettijd. Het eind van het liedje is de sprong terug naar het onderbroken hoofdprogramma, waarna een nieuw liedje begint, enzovoorts. De gang van zaken is geschetst in het stroomschema ("het programma van het programma") van figuur 1.
Hoe gaat dat in de praktijk in zijn werk? Bekijk figuur 2. Dat is de hardware die je op de poortkonnektor van de juniorcomputer moet aansluiten om de frekwentie-informatie binnen te halen. Bij een voldoende sterk ingangssignaal zorgt een negatieve nuldoorgang van dat ingangssignaal voor een 1/Ø-overgang op poortlijn PA7. Er wordt in het programma voor gezorgd dat dat tevens gepaard gaat met een IRQ-interrupt.
Het programma vindt u in de tabel. Het startadres is $1AØØ. Door het schrijven van data in de geheugenplaats EDETC



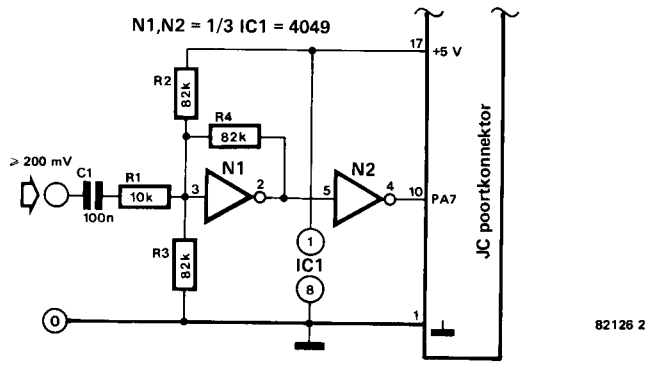

```
$1AØØ A9 ØØ     INITPR LDAIM $ØØ
$1AØ2 85 DØ            STAZ  ACCUL
$1AØ4 85 D1            STAZ  ACCUM
$1AØ6 85 D2            STAZ  ACCUH
$1AØ8 A9 29            LDAIM IRQSRV
$1AØA 8D 7E 1A         STA   IRQL
$1AØD A9 1A            LDAIM IRQSRV/256
$1AØF 8D 7F 1A         STA   IRQH
$1A12 8D E6 1A         STA   EDETC
$1A15 A9 1Ø            LDAIM $1Ø (16₁₀)
$1A17 85 D4            STAZ  TIMEH
$1A19 85 D3            STAZ  COUNT
$1A1B A9 3D            LDAIM $3D (61₁₀)
$1A1D 85 D5            STAZ  TIMEL
$1A1F 8D FF 1A         STA   CNTH
$1A22 58               CLI
$1A23 2Ø 8E 1D LOOP    JSR   SCANDS
$1A26 4C 23 1A         JMP   LOOP
$1A29 48       IRQSRV  PHA
$1A2A 8A               TXA
$1A2B 48               PHA
$1A2C 98               TYA
$1A2D 48               PHA
$1A2E 2C D5 1A         BIT   RDFLAG
$1A31 1Ø 1C            BPL   ADD
$1A33 A5 D5            LDAZ  TIMEL
$1A35 8D FF 1A         STA   CNTH
$1A38 C6 D3            DECZ  COUNT
$1A3A DØ 28            BNE   EXIT
$1A3C A2 Ø2            LDXIM $Ø2
$1A3E AØ ØØ            LDYIM $ØØ
$1A4Ø B5 DØ    STORE   LDAZ  ACCUL,X
$1A42 95 F9            STAZ  INH,X
$1A44 94 DØ            STYZ  ACCUL,X
$1A46 CA               DEX
$1A47 1Ø F7            BPL   STORE
$1A49 A5 D4            LDAZ  TIMEH
$1A4B 85 D3            STAZ  COUNT
$1A4D DØ 15            BNE   EXIT
$1A4F F8       ADD     SED
$1A5Ø 18               CLC
$1A51 A5 DØ            LDAZ  ACCUL
$1A53 69 Ø1            ADCIM $Ø1
$1A55 85 DØ            STAZ  ACCUL
$1A57 A5 D1            LDAZ  ACCUM
$1A59 69 ØØ            ADCIM $ØØ
$1A5B 85 D1            STAZ  ACCUM
$1A5D A5 D2            LDAZ  ACCUH
$1A5F 69 ØØ            ADCIM $ØØ
$1A61 85 D2            STAZ  ACCUH
$1A63 D8               CLD
$1A64 68       EXIT    PLA
$1A65 A8               TAY
$1A66 68               PLA
$1A67 AA               TAX
$1A68 68               PLA
$1A69 4Ø               RTI

ADDITIONAL ZERO PAGE LOCATIONS
ACCUL   $ØØDØ
ACCUM   $ØØD1
ACCUH   $ØØD2
COUNT   $ØØD3
TIMEH   $ØØD4
TIMEL   $ØØD5
```

1



Figuur 1. De aanpak: konstruktieve onderbrekingen (IRQ) voor het meten van de frekwentie . . .

2



Figuur 2. . . . van een wisselspanning die via deze schakeling op de junior-computer wordt aangesloten.

geeft een 1/Ø-overgang op PA7 tot een IRQ aanleiding. Andere voorbereidingen betreffen het specificeren van de IRQ-sprongvektor op het startadres van het interruptprogramma IRQSRV, het starten van de interval-timer (CNTH, dus elke 1024 clock-pulsen een IRQ) en het vastleggen van de inhoud van geheugenplaats COUNT. Daarna wordt de programmalus LOOP permanent doorlopen – in afwachting van een IRQ.

Zodra een IRQ is vastgesteld – van welke aard ook – volgt de afwikkeling van het programma IRQSRV. Na het bewaren op de stack van A, X en Y (spelen een rol tijdens SCANDS) gaan we de N-vlag bekijken. Indien N, dus de timer-vlag nul is kan de IRQ niet het gevolg zijn van een time out. Dus was de IRQ het gevolg van een nivo-wisseling op PA7: er is een nieuwe periode van de wisselspanning verstreken en we gaan naar het label ADD. Het 24-bits BCD-getal (ACCUH, ACCUM, ACCUL) (de periodenteller van figuur 1) wordt met één verhoogd. Na het herstellen van A, X en Y (EXIT) en na de RTI zijn we terug in de LOOP-tredmolen.

Stel, de IRQ was het gevolg van een time out van de interval-timer. Dan wordt de timer opnieuw gestart en de inhoud van COUNT met één verlaagd. Zolang die COUNT nog niet nul is volgt direkt de sprong naar EXIT. Is COUNT daarentegen wèl nul, dan wordt eerst nog het programmadeel STORE doorlopen; de 1 s-meettijd is voorbij en de welbekende displaybuffers POINTH, POINTL en INH krijgen een inhoud die gelijk is aan die van ACCUH, resp. ACCUM, resp. ACCUL.

De praktijk? Sluit de hardware van figuur 2 aan op de poortkonnektor, toets het programma in (nog beter: lees het in van de cassette) en start het (via het standaard-toetsenbord; dit in verband met de voor SCANDS noodzakelijke I/O-definitie). De hoogst meetbare frekwentie bedraagt ca. 10 kHz. Bij lage frekwenties kan men een hogere nauwkeurigheid bereiken door de meettijd te verhogen tot 10 s (laad TIMEH met AØ in plaats van 1Ø; adres $1A16). Uiteraard moet het weergegeven getal dan door tien worden gedeeld om de frekwentie te krijgen.

Wij wensen u mega-plezier met deze nano-gekompliceerde toepassing van de junior-computer. ■

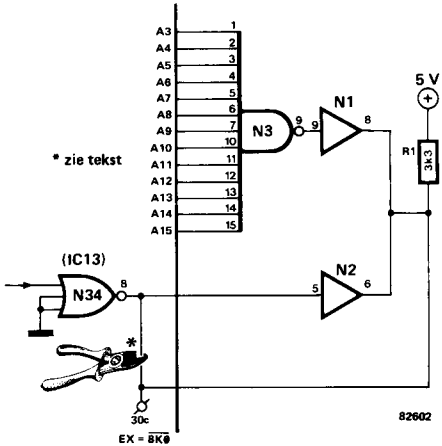
Verder lezen:
Hoofdstuk 6 van junior-computerboek 2

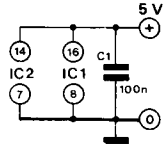
34 Junior-vektoren ophalen

R. Matyssek

zo kan het ook!

In aanhangsel 3 van Junior-computerboek 3 zijn metoden aangegeven om, mèt aangesloten buskaartgeheugen en zonder gebruik te maken van extra EPROM, de systeemvektordata uit de standaard-EPROM te halen. Nevenstaande schakeling is een elegant alternatief voor "oplossing één" uit voornoemd aanhangsel (waarin overigens N102 dient te worden vervangen door een doorverbinding).
Op de interfase-kaart wordt draadbrug R-S en op de standaardkaart D-EX aangebracht. Pen 8 (uitgang N34) van IC13 (op de interface-kaart) buigt men omhoog, zodat de interne doorverbinding met het punt EX-8KØ komt te vervallen. Deze verbinding komt nu tot stand via de open-kollektoruitgangen van de poorten N1 en N2. Omdat IC1 maar liefst 13 ingangen c.q. aangesloten adreslijnen heeft, worden slechts acht geheugenplaatsen opgeofferd: de geheugenplaatsen $FFF8 . . . $FFFF.



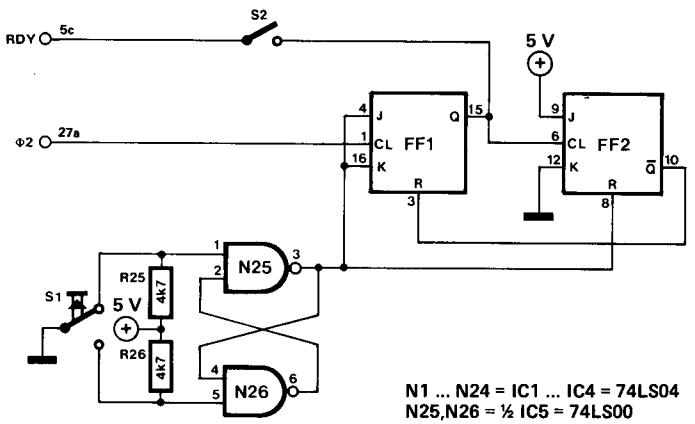


N1,N2 = 1/3 IC2 = 7417
N3 = IC1 = 74LS133

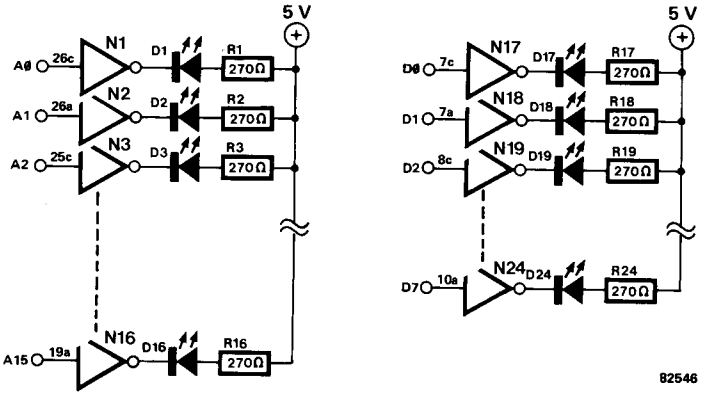
15 single cycle voor Junior-computer

E. Kytzia

logic analyse in slow motion



N1 ... N24 = IC1 ... IC4 = 74LS04
N25,N26 = ½ IC5 = 74LS00
FF1,FF2 = IC6 = 74LS76



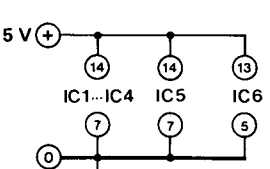
82546

Met deze hulpschakeling kan de Junior-computer in de single-cycle-mode werken. In tegenstelling tot single step wordt bij single cycle niet een hele instruktie, maar een enkele clock-cyclus afgewerkt. Dat is vooral prettig bij de analyse van de logische nivo's op de bus. Hierbij komen ook de nivo-indikators (N1 t/m N24) goed van pas.

Na een reset-kommando bevindt de CPU zich in een gedefinieerde toestand. Dient men nu met S1 clock-pulsen toe, dan begint de CPU met de reset-cyclus. Na 8 clock-pulsen staan de beide reset-vektoren RESL (FFFC) en RESH (FFFD) op de adresbus. Het programma wordt dan vanaf deze adressen doorlopen.

Een gedetailleerde informatie over single-cycle-instruktie biedt het "MCS 6500 microcomputer family hardware manual" van MOS Technology. Ook van Rockwell is dergelijke informatie verkrijgbaar.

Let op: bij een printinstruktie stopt de CPU niet.



– de verbinding tussen pen 2 van IC10 en massa;
– de verbinding tussen pen 3 van IC10 en massa;
– de verbinding tussen pen 2 van IC10 en pen 3 van IC10.

Kontroleer de onderbrekingen met behulp van een ohmmeter!

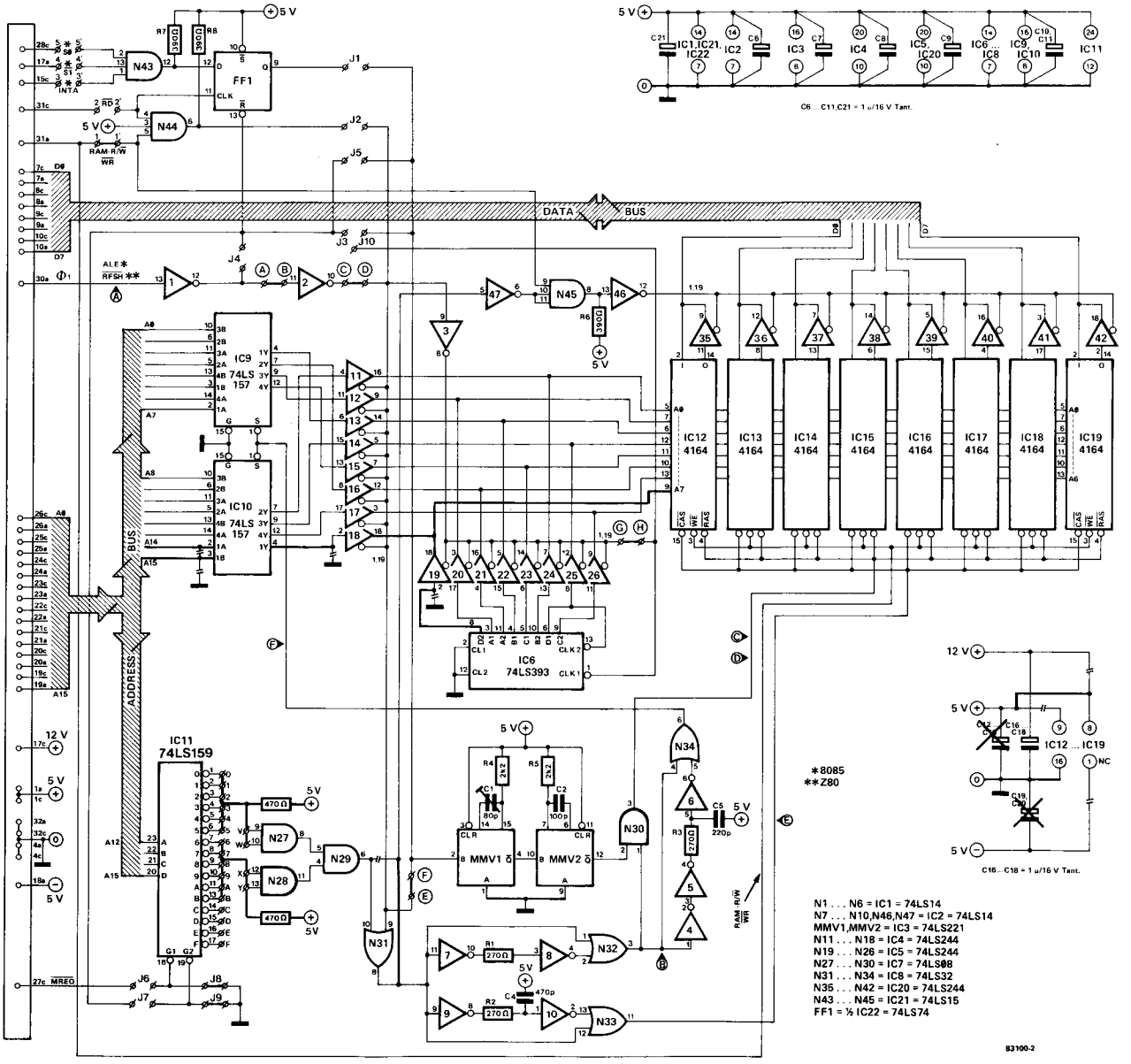
. . . en dan weer opbouwen

Vervolgens gaan we enkele nieuwe verbindingen leggen tussen de volgende punten:

- pen 8 van IC12 . . . IC19 en pen 1a/1c van de konnektor (+5 V);
- pen 6 van IC7 (N29) en pen 10 van van IC8 (N31);
- pen 8 van IC8 (N31) en pen 5 van IC2 (N47);
- pen 8 van IC6 en pen 2 van IC5 (N19);
- pen 4 van IC10 en pen 2 van IC4 (N18);
- pen 2 van IC10 en pen 19c van de konnektor (A14);
- pen 3 van IC10 en pen 19a van de konnektor (A15);
- pen 18 van IC4, pen 18 van IC5 en pen 9 van IC12 . . . IC19 (A7);
- pen 9 en pen 10 van IC7 (V-W);
- pen 12 en pen 13 van IC7 (X-Y);
- Afhankelijk van de gewenste adresdekodering kunnen de uitgangen van de adresdekoder IC1 worden verbonden met de ingangen V/W en X/Y. Twee pull-upweerstanden van 470 Ω moeten nog worden toegevoegd aan de ingangen van N27 en N28, zoals in het schema is aangegeven. Volgens de in het schema getekende verbindingen ligt de kaart in het adresbereik $0000 . . . $BFFF (dat is 48 Kbyte, de

Figuur 2. Het schema van de dynamische RAM-kaart met de wijzigingen. De belangrijkste punten zijn het toevoegen van de adreslijnen A14 en A15 (zodat in totaal 64 K kan worden geadresseerd) en het vervangen van de adresdekoder IC11 door een type met open-kollektor-uitgangen.

2



En, hoe gaat het met de junior computer? Oh prima, luister maar, hij maakt zelfs muziek. En niet zo maar een paar toontjes. Nee, een kompleet deuntje! Bovendien houdt hij precies het juiste tempo en de goeie toonhoogte aan. Hoe ook uw junior computer zijn eigen melodietje leert spelen wordt in dit artikel uit de doeken gedaan, kompleet met het benodigde programma en een leuke melodie.

de muzikale junior

junior computer speelt zijn eigen lied

A. Bricart

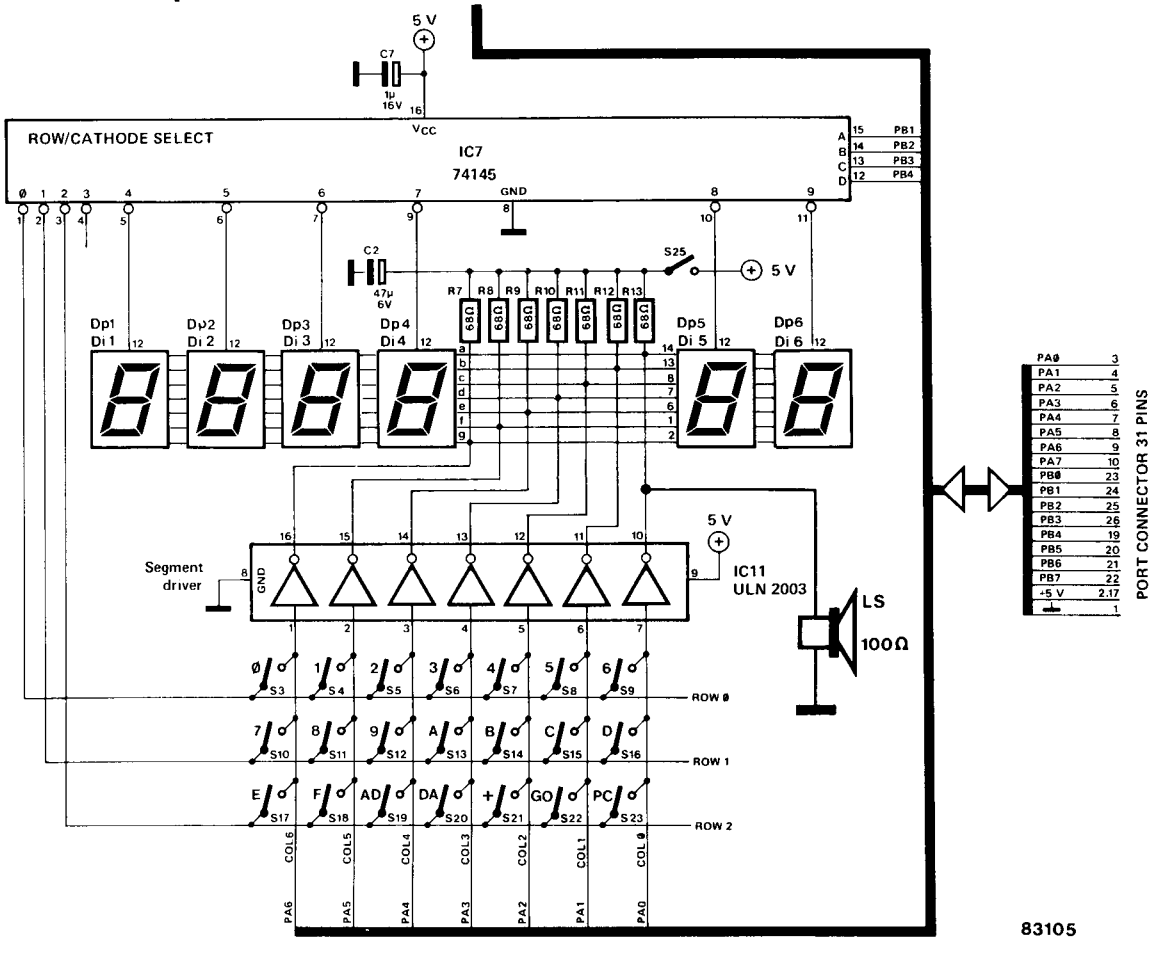
In de muziek wordt heden ten dage veelvuldig gebruik gemaakt van microprocessoren. Nu zullen de meeste lezers wel niet direkt plannen hebben om hun junior computer te gebruiken voor het sturen van een synthesizer, maar het maken van een leuk muziekje met de junior is toch wel een aardige afleiding van al die bits en bytes.

Een zingende uitlezing

Voor het maken van muziek met de junior computer is ditmaal nu eens geen aparte schakeling nodig voor de sturing van het luidsprekertje. Voor deze toepassing kan men gewoon een 100 Ω-luidsprekertje (of een 8 Ω-luidsprekertje in serie met een 100 Ω-weerstand) aansluiten tussen massa en een van de uitgangen van de ULN 2003 op de basisprint van de junior.

Het geluid bestaat uit een reeks blokgolven, waarbij de frekwentie en de duur kan worden bepaald door de processor. Het te spelen melodietje kan worden opgeslagen op pagina $Ø3ØØ van het RAM-bereik. Het melodieprogramma bestaat uit een serie bytes waarbij steeds twee opeenvolgende bytes de gegevens én toon bevatten. De bytes op de even adresplaatsen bevatten de toonhoogte en de bytes op de oneven adressen de toonduur. De toonhoogte hangt af van de periodetijd van de blokgolven en de toonduur wordt bepaald door het aantal geproduceerde blokgolven.
Er kunnen vier verschillende toonduren worden ingesteld: een halve noot, een kwartnoot, een achtste en een zestiende noot.
De frekwentie wordt afgeleid van de 1 MHz-kristalfrekwentie van het processorsysteem.

1



Uitgaande van de A van 440 Hz dient de periodeduur precies 2,28 ms te bedragen. Bij een symmetrische blok is de "1"-tijd dan 1,14 ms. De bij de junior beschikbare vertragingslus (DELAY) geeft een vertraging van 14 μs. Voor een A moet deze lus dus 81 keer worden doorlopen voor het verkrijgen van de halve periodeduur (81 x 14 μs = 1,134 ms). In hexadecimale vorm uitgedrukt bedraagt de "toonhoogte" van de A dan $51.

Eenvoudige opzet

De eenvoud van het programma heeft tot gevolg dat de melodie alleen in pagina $Ø3ØØ kan worden gezet. Dat houdt in dat de lengte van de melodie niet meer dan 127 noten mag zijn. Het tempo wordt bepaald door de inhoud van adres MULT ($ØØØ2). De toonduur hangt af van de inhoud van de oneven adressen, waarbij nog moet worden opgemerkt dat de toonduur samenhangt met de toonhoogte. Tabel 1 geeft de kodes voor toon en toonduur.
Als de processor in een even adres de waarde ØØ tegenkomt, dan ziet hij dat als een pauze. De inhoud van het volgende adres bepaalt in dat geval de lengte van de pauze. Als het oneven adres de data ØØ bevat wordt de melodie op dat punt afgebroken en begint de processor weer vooraan. In tabel 3 is tenslotte een muzikaal voorbeeld gegeven waarbij de junior zich van zijn beste kant laat horen.

Tabel 1

noot	Hz	toon-hoogte	♩	𝅗𝅥	♪	𝅘𝅥𝅯
				toonduur		
E	1318.5	1B			84	42
D#	1244.5	1D		F9	7C	3E
D	1174.6	1E		EB	76	3B
C#	1108.7	2Ø		DE	6F	37
C	1046.5	22		D1	68	34
B	988	24		C6	63	31
A#	932.3	26		BA	5D	2F
A	880	29		BØ	58	2C
G#	830.6	2B		A6	53	2A
G	784	2E		9D	4E	27
F#	740	30		94	4A	25
F	698.4	33		8C	46	23
E	659.2	36		84	42	21
D#	622.2	39	F9	7C	3E	1F
D	587.3	3D	EB	75	3B	1D
C#	554.3	41	DE	6F	37	1C
C	523.2	44	D1	69	34	1A
B	494	48	C6	63	31	19
A#	466.1	4D	BA	5D	2F	17
A	440	51	BØ	58	2C	16
G#	415.3	56	A6	53	2A	15
G	392	5B	9D	4E	27	14
F#	370	61	94	4A	25	12
F	349.2	66	8C	46	23	11
E	329.6	6C	84	42	21	1Ø
D#	311.1	73	7C	3E	1F	1Ø
D	293.6	79	75	3A	1D	ØE
C#	277.2	81	6F	37	1C	ØE
C	261.6	89	69	34	1A	ØD
B	247	91	63	31	19	ØC
A#	233.1	99	5D	2F	17	ØC
A	220.6	A2	58	2C	16	ØB
G#	207.6	AC	53	2A	15	ØB
G	196	B6	4E	27	14	ØA
		ØØ	E0	7Ø	38	IC
			ØØ			

Tabel 1. De kodes voor toonhoogte en toonduur, waarmee men een stuk muziek kan "omzetten" in junior-taal.

Tabel 2

```
JUNIOR

M
HEXDUMP: 2ØØ,25D
      Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
Ø2ØØ: A9 7F 8D 81 1A A9 Ø8 8D 82 1A A9 ØØ 85 ØØ A9 Ø2
Ø21Ø: 85 Ø2 A6 ØØ BD Ø1 Ø3 85 Ø1 FØ E5 A9 4Ø 8D 8Ø 1A
Ø22Ø: 2Ø 5Ø Ø2 A6 ØØ BC ØØ Ø3 FØ Ø8 A9 BF 8D 8Ø 1A 2Ø
Ø23Ø: 5Ø Ø2 C6 Ø1 DØ E5 C6 Ø2 DØ D8 E6 ØØ E6 ØØ A2 FF
Ø24Ø: CA EA EA EA DØ FA 4C ØE Ø2 ØØ ØØ ØØ ØØ ØØ ØØ ØØ
Ø25Ø: A6 ØØ BC ØØ Ø3 A2 Ø2 CA DØ FD 88 DØ F8 6Ø
```

Tabel 2. Het programma dat via de 6532 en de ULN 2003 het geluidssignaal voor het luidsprekertje opwekt. Een interface voor het luidsprekertje is hier niet nodig.

Tabel 3

```
JUNIOR

M
HEXDUMP: 3ØØ,36B
      Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
Ø3ØØ: 51 58 3D EA 41 DE 3D 75 36 84 51 58 48 63 5B 9C
Ø31Ø: 61 4A 5B 4E 6C 84 61 94 79 3A 51 58 3D EA 41 DE
Ø32Ø: 3D 75 36 84 51 58 48 63 5B 9C 61 4A 5B 4E 6C 84
Ø33Ø: 61 94 79 3A 61 94 5B 4E 51 BØ 51 58 48 63 48 63
Ø34Ø: 56 53 51 BØ 51 58 3D EA 48 63 41 6F 41 6F 51 58
Ø35Ø: 3D 75 3D 75 48 63 41 DE 3D 75 51 58 48 63 5B 9C
Ø36Ø: 61 4A 5B 4E 6C 84 79 74 ØØ 7Ø ØØ ØØ
```

Tabel 3. En hier is een voorbeeld van een melodietje gegeven. Op de even adresnummers staat de toonhoogte en op de oneven nummers de toonduur. De kode ØØ op adres $Ø36B zorgt er voor dat de processor weer terugkeert naar het begin. Het muziekstuk wordt dus eindeloos herhaald totdat het programma wordt onderbroken.

64 K op de dynamische RAM-kaart

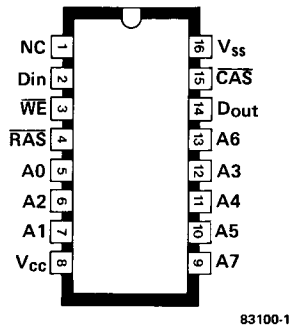
één enkele voedingsspanning voor 524288 bits

naar een idee van
K.D. Lorig

Ruim een jaar na de publikatie van de dynamische RAM-kaart in Elektuur (april 1982) kan deze geheugenkaart zich nog steeds in een levendige belangstelling verheugen. Dit komt vooral door het feit dat vele lezers deze 16 K-kaart willen gebruiken als 64 K-kaart, door de 16 Kbit-RAM's te vervangen door 64 Kbit-typen. Hierbij stoten ze dan op moeilijkheden omdat er in dat geval nog verschillende veranderingen op de print moeten worden aangebracht. In dit artikel wordt een lijst met veranderingen gegeven waarmee elke lezer zijn (oude) kaart kan ombouwen tot een even goed lopende kaart met een geheugenkapaciteit van 64 kilobytes.

Tegenwoordig zijn dynamische RAM's met een kapaciteit van 64 Kbits goed verkrijgbaar. Bovendien zijn deze RAM's de laatste tijd zodanig in prijs gedaald dat ze voor de hobbyist zeer aantrekkelijk zijn geworden. Tel daar nog bij dat de meeste 64 K-RAM's slechts één enkele voedingsspanning nodig hebben, dan is er eigenlijk nog maar een konklusie mogelijk: De 16 K RAM-kaart moet 64 K worden! Zo'n ombouw brengt nogal wat voordelen met zich mee. Zo wordt de prijs per bit veel lager dan bij een 16 K-kaart, er is nog maar een enkele voedingsspanning nodig en voor het hele RAM-bereik is nog maar een enkele konnektor nodig, zodat de andere konnektoren vrij blijven (een 8-bits microprocessor kan gewoonlijk maar 64 K adresseren, dus één RAM-kaart is dan voldoende voor het hele bereik). Het enige nadeel: er moet hier en daar wat veranderd worden op de dynamische RAM-kaart. Enkele koperbanen moeten worden doorgekrast en er moeten enkele nieuwe verbindingen worden gelegd. Maar al die moeite resulteert wel in een viervoudige geheugenkapaciteit op dezelfde print-oppervlakte.

1



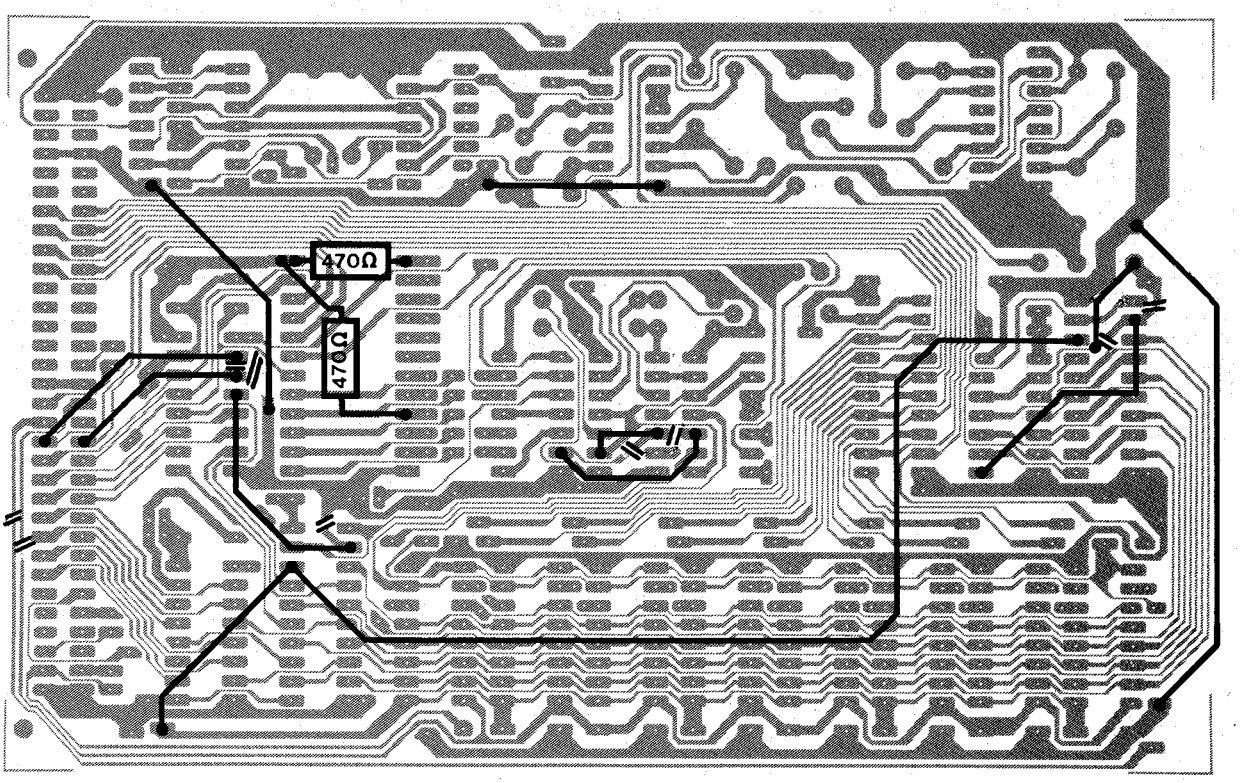
Figuur 1. Aansluitgegevens van de dynamische RAM 4164. Het verschil tussen dit IC en de 4116 zit kwa aansluitpunten in pen 1, 8 en 9. Bij de 4164 is adres A7 toegevoegd en zijn de −5 V- en +12 V-aansluitingen vervallen.

A0-A7	Address Inputs
$\overline{CAS}$	Column Address Strobe
Din	Data In
Dout	Data Out
$\overline{RAS}$	Row Address Strobe
$\overline{WE}$	Read/$\overline{Write}$ Input
Vcc	Power (+5 V)
Vss	Ground

Eerst afbreken . . .

Om niets aan het toeval over te laten en fouten uit te sluiten hebben we alle veranderingen mooi op een rijtje gezet. Als men zich daaraan houdt bij de rekonstruktie en alles punt voor punt uitvoert kan er niets mis gaan. In figuur 2 en 3 zijn de veranderingen in het schema en op de print voor de duidelijkheid ook nog eens aangegeven. We beginnen met het verwijderen van verschillende onderdelen en het doorkrassen van diverse printbanen.

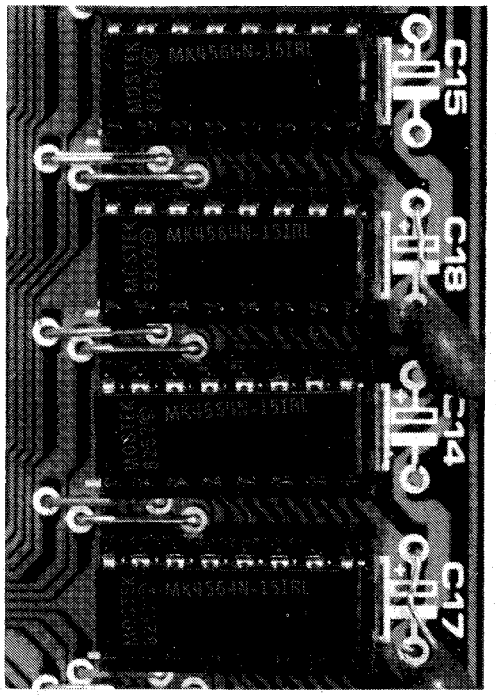
- Verwijder IC11, IC12 . . . IC19 uit hun voetjes.
- De kondensatoren C3, C12 . . . C15, C19 en C20 worden los gesoldeerd en van de print gehaald.
- Verwijder de draadbrug die vlak naast pen 9 . . . 16 van IC9 ligt.
- Onderbreek de volgende koperbanen:
  - de verbinding tussen pen 2 van IC4 (N18) en massa;
  - de verbinding tussen pen 2 van IC5 (N19) en massa (vergeet niet pen 2 en 12 van IC6 daarna weer apart met massa te verbinden);
  - de verbinding tussen pen 8 van IC12 . . . IC19 en +12 V;
  - de verbinding tussen pen 1 van IC12 . . . IC19 en −5 V;
  - de verbinding tussen pen 6 van IC7 (N29) en pen 5 van IC2 (N47);
  - de verbinding tussen pen 5 van IC2 en pen 10 van IC8 (N31);



83100-3

rest van het adresbereik is nodig voor ROM van de junior). De dekodering zoals aangegeven wordt gebruikt bij de DOS-junior.

- Nu moeten nog twee massaverbindingen worden gelegd, uitgaande van de op pen 4a/4c van de konnektor aangesloten koperbaan. Dit is duidelijk in figuur 3 aangegeven.



Enkele nieuwe komponenten

Als alle onderbrekingen en nieuwe verbindingen goed zijn uitgevoerd zit het zwaarste gedeelte van het karwei erop. Nu hoeven alleen nog maar enkele nieuwe IC's op de print te worden geprikt. Op de plaats van IC11 komt een 74LS159 (een type met open-kollektor-uitgangen, vandaar de aanwezigheid van die twee 470 Ω-weerstanden). Daarna volgt nog een laatste inspektie om te zien of men niets vergeten is. Kijk alles zorgvuldig na!

Tenslotte kunnen de nieuwe RAM's in de voetjes worden geplaatst. De 64 K-RAM's worden geleverd door verschillende (voornamelijk Japanse) fabrikanten, waarvan de typenummers verschillen maar de laatste twee cijfers altijd "64" zijn: F 4164 (Fairchild), MB 8264 (Fujitsu), HM 4864 (Hitachi), ITT 4164, M5K 4164 (Mitsubishi), MK 4564 (Mostek), NMC 4164 (National Semiconductor), UPD 4164C/D (NEC), enzovoorts. Keus genoeg!

Het lijkt ons nuttig om tot slot nog eens te wijzen op het programma voor het testen van het geheugen, zoals beschreven op pagina 4-56 van Elektuur april 1982. Bij 524288 bits is het testen van het geheugen beslist geen overbodige luxe!

Figuur 3. Hier ziet men welke koperbanen moeten worden onderbroken en welke nieuwe verbindingen moeten worden gelegd. Vergeet niet de draadbrug vlak bij IC9 te verwijderen!

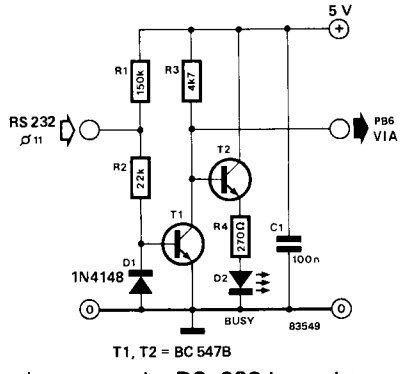
49

W. Schaaij

busy-indikator voor junior

Op de junior computer kan vrij eenvoudig een busy-indikator voor een printer worden aangesloten. Alles wat daarvoor nodig is, is een kleine schakeling die wordt toegevoegd aan de printer-interface. Verder moet een kleine modifikatie worden aangebracht in de PM-software.
Het schema van de schakeling is bijzonder eenvoudig. Het busy-signaal van de printer is beschikbaar aan pen 11 van de RS 232-aansluiting.

Het gedeelte rond T1 zorgt voor de nivo-aanpassing van RS 232 naar TTL. Bij een positieve spanning aan de ingang (''0'') gaat T1 geleiden, zodat op aansluiting PB6 van de VIA een logische nul komt te staan. Bij een negatieve spanning (''1'') spert T1, zodat PB6 +5 V krijgt toegevoerd via R3. Diode D1 zorgt er voor dat de basis-emitter-spanning van T1 in dat geval wordt begrensd tot −0,6 V. Via T2 gaat de LED branden als de printer een busy-signaal (negatieve) spanning geeft. Weerstand R1 zorgt er voor dat de LED niet brandt en ook geen busy-signaal aan de junior wordt gegeven als niets is aange-



T1, T2 = BC 547B

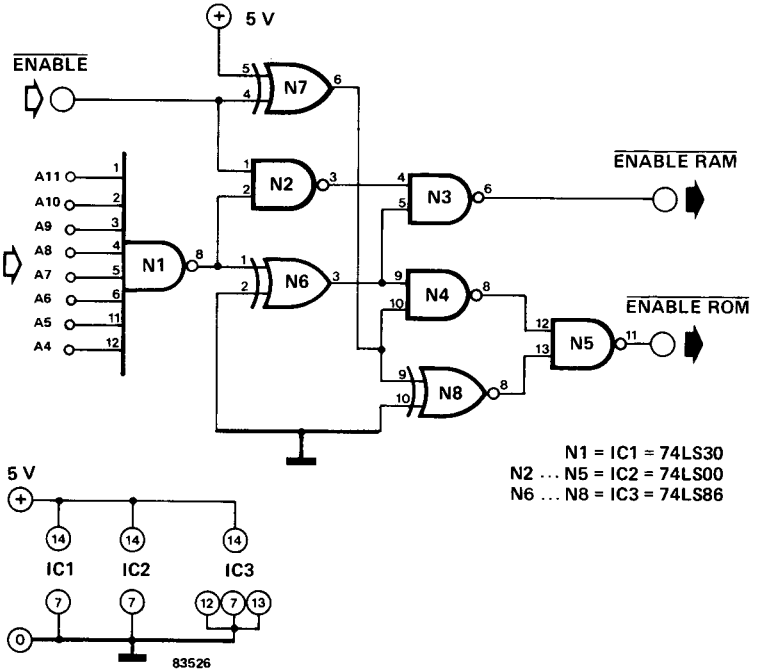
sloten op de RS 232-konnektor. In de PM-EPROM moeten de volgende plaatsen veranderd/toegevoegd worden:

```
133A          2Ø F4 14   JSR BUSY (veranderen)
                  .
                  .
                  .
14F4 BUSY     AD ØØ 18   LDA PBDVIA (toevoegen)
14F7          29 4Ø      ANDIM $4Ø        ↓
14F9          DØ F9      PB6  VIA = Ø?
14FB          AD 82 1A   LDA PBDPIA
14FE          6Ø         RTS
```

58

N. Humphreys

vektor-aansturing voor junior



Met de hier afgebeelde schakeling is het mogelijk bij de junior computer de noodzakelijke vektordata uit de standaard-EPROM te lezen zonder dat daarvoor een flink gedeelte van het adresbereik hoeft te worden ingeleverd en/of een extra ROM nodig is. De vektoren voor NMI, RES en IRQ staan op de adressen FFFA . . . FFFF. Bij de gewone oplossing voor het halen van die vektoren uit de standaard-EPROM, zoals in het junior computer boek deel 3 is beschreven, wordt het hele geheugenbereik van F000 . . . FFFF voor 6 bytes "opgeofferd". Dat is 4 K geheugenruimte! Bij de hier beschreven oplossing worden voor het lezen van die 6 bytes slechts 16 bytes gereserveerd. De schakeling is natuurlijk alleen nodig als men RAM wil leggen in het bovenste geheugenbereik van F000 . . . FFFF; bij gebruik van de mini-EPROM-kaart bijvoorbeeld (Elektuur april '82) is er sowieso geen opoffer-probleem.

De schakeling bestaat uit slechts acht poorten. Ze genereert uit het "gewone" enable-signaal voor het hoogste 4 K-bereik twee aparte enable-signalen. Bij de adressen F000 . . . FFEF wordt een nieuw $\overline{\text{ENABLE RAM}}$-signaal gegeven, terwijl bij de adressen FFF0 . . . FFFF geen $\overline{\text{ENABLE RAM}}$ wordt gegeven, maar een $\overline{\text{ENABLE ROM}}$-signaal naar de standaard-EPROM van de junior wordt gestuurd.

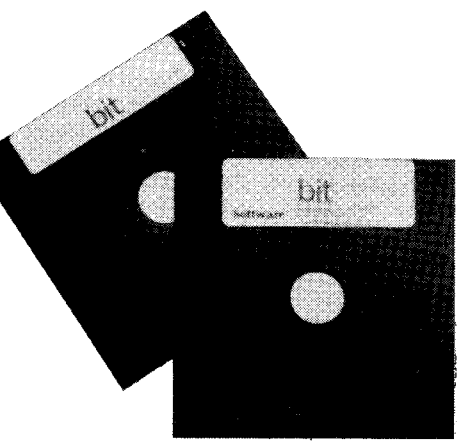
De schakeling kan worden bijgebouwd op de betreffende RAM-kaart, het gaat hier om slechts drie IC's. De oude $\overline{\text{ENABLE}}$-lijn voor het bereik F000 . . . FFFF van de adresdekoder op de RAM-kaart (bij de dynamische RAM-kaart is dat uitgang F van IC11) wordt op de "toevoeg"-schakeling aangesloten en de acht ingangen van N1 worden verbonden met de adreslijnen A4 . . . A11. Naar de RAM-kaart gaat dan een nieuw $\overline{\text{ENABLE RAM}}$-signaal (bij de dynamische RAM-kaart wordt die lijn dus aan één van de punten V, W, X of Y gelegd). De $\overline{\text{ENABLE ROM}}$-lijn moet worden verbonden met aansluiting K7 (pen 14a) van de konnektor.

G. de Cuijper

floppy-disk interface

voor junior en andere 6502-computers

De floppy disk is momenteel het belangrijkste massageheugen voor een computer. Als men ziet hoe zo'n systeem werkt, dan lijkt het wel een klein wonder dat met zo'n grote snelheid digitale gegevens op een eenvoudig flexibel schijfje kunnen worden opgeslagen en gelezen. Dit artikel beschrijft wat er allemaal moet gebeuren voordat een bit op zo'n floppy-schijf kan worden gezet. De eveneens beschreven hardware voor de floppy-interface is bijzonder universeel van opzet. Niet alleen junior computer-fans kunnen deze interface gebruiken, maar ook bezitters van onder andere een KIM, SYM, AIM 65 en ACORN kunnen met deze interface hun systeem uitbreiden. Tenslotte wordt ook nog een interface beschreven voor het aansluiten van een Epson-printer op de junior computer.



Ongeveer acht jaar geleden brak hier in Europa de computer-koorts uit. Toen werden de eerste 8 bit-processoren voor amateurs een beetje betaalbaar. Massageheugens waren echter peperduur. In 1976 kwam de Amerikaanse firma Shugart met een 8 inch-loopwerk op de markt. De computerhobbyisten, die zich tot die tijd hadden moeten behelpen met ponsband en magneetband, kregen daarmee de beschikking over de zogenaamde floppy disk. Met ponsband is bij het lezen een maximale snelheid van 15 kilobaud (1 baud = 1 bit/s) mogelijk, terwijl de snelheid bij het ponsen hooguit 700 Bd is. Met een gewone cassette-recorder zit men met een snelheid van 4,75 cm/s bij 1200 Bd al aan de grens van de mogelijkheden. Floppy's zijn in vergelijking hiermee stukken sneller. Maar ook een floppy disk heeft nadelen. Zo'n plaatje is niet veel duurder dan een goede chroomdioxide-cassette, maar het floppy-loopwerk is niet bepaald goedkoop, vrij duur zelfs. Voor een mini-floppy-drive moet de computer-liefhebber toch wel zo'n 500 tot 1000 gulden op tafel leggen. Als men bedenkt dat men voor één computer eigenlijk twee floppy-drives nodig heeft om "komfortabel" te kunnen werken, dan is dat bij elkaar een hele investering.

Het einde van de prijzenstorm

De prijzen voor floppy-loopwerken zijn vrij stabiel en het ziet er dan ook niet naar uit dat deze de komende tijd veel zullen dalen. Daarom is getracht op een andere plaats te sparen bij de DOS (= Disk Operating System) voor de junior computer. We hadden de keus tussen het gebruik van een echte floppy disk controller en een "zelfgemaakte" controller, bestaande uit enkele TTL-IC's en wat software. Als controller-IC's komen de 1771 of 1791 van Western Digital en de 6843 van Motorola in aanmerking. Het nadeel van alle IC's: de prijs, die tussen 80 en 200 gulden ligt. Een ander nadeel is het feit dat voor deze controller-IC's heel weinig 6502-software verkrijgbaar is.

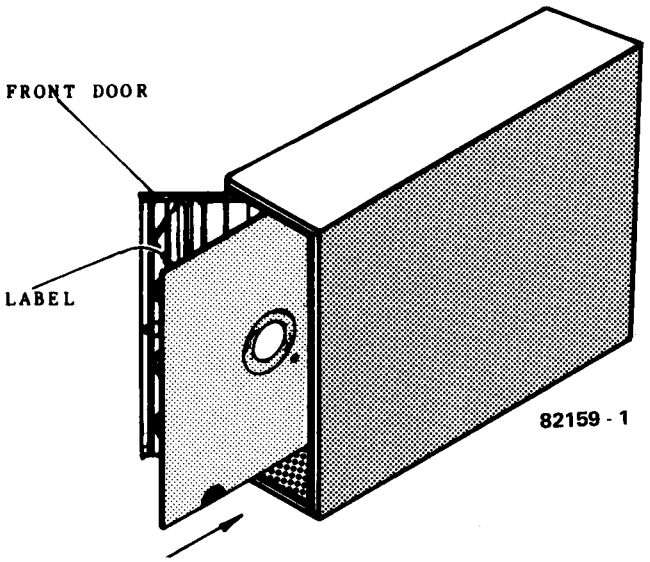
Het was ons doel de junior computer uit te rusten met een disk operating system dat ook geschikt is voor KIM, SYM en AIM 65. Daarnaast mocht de floppy-interface niet meer dan 200 gulden gaan kosten. Hierbij moest de DOS aan de volgende punten voldoen:

1. De programmeur hoeft zich niet meer druk te maken over de absolute adressen in de computer.
2. De DOS moet kunnen samenwerken met een Microsoft-BASIC. De BASIC-interpreter moet de DOS-makrokommando's kunnen "begrijpen".
3. De DOS moet ook kunnen samenwerken met een komfortabele debugger. Een debugger is een programma waarmee software kan worden getest.
4. Een assembler en editor moeten eveneens aanwezig zijn. Zij dienen de DOS-makrokommando's ook te "verstaan".
5. Als de programmeur de computer een keer fout programmeert, dan moet de computer nauwkeurige foutmeldingen geven om een direkte analyse van de syntax- en bedieningsfouten mogelijk te maken.
6. Voor de DOS dient veel goede en goedkope software op floppy verkrijgbaar te zijn:
   - spelprogramma's
   - boekhoudprogramma's
   - programma's in BASIC en assembler
7. De DOS moet eenvoudig kunnen worden aangepast voor elke 6502-computer.
8. De DOS moet random-files kunnen maken. Random-files zijn data-files op de floppy disk waarin data wordt geschreven die tijdens de uitvoering van een BASIC-Programma ontstaat.

Zoals uit alle opgesomde punten blijkt hebben we nogal hoge eisen aan het disk operating system gesteld. Op grond hiervan is gekozen voor een in Amerika en Europa wijd verbreid operating system: het "Ohio Scientific OS-65D Operating System".

De firma Ohio Scientific is ook de fabrikant van de populaire computers C1P, C4P en C8P. De software die voor deze computers is ontwikkeld (en dat is een hele hoop!) kan men door het veranderen van enkele bytes in het DOS-hoofdprogramma genaamd KERNEL gemakkelijk aanpassen voor de junior computer en andere 6502-systemen. Van de Ohio-DOS zijn op het moment twee versies beschikbaar:

1



Figuur 1. Zo wordt een diskette in de floppy-drive geschoven. Het label van de diskette moet aan de kant van de klep zitten.

1. OS-65D V3.1 bestaat uit
   - een 5 inch diskette (diskette = floppy met beschermhoes).
   - een handboek in de Engelse taal met een omvang van circa 75 pagina's.

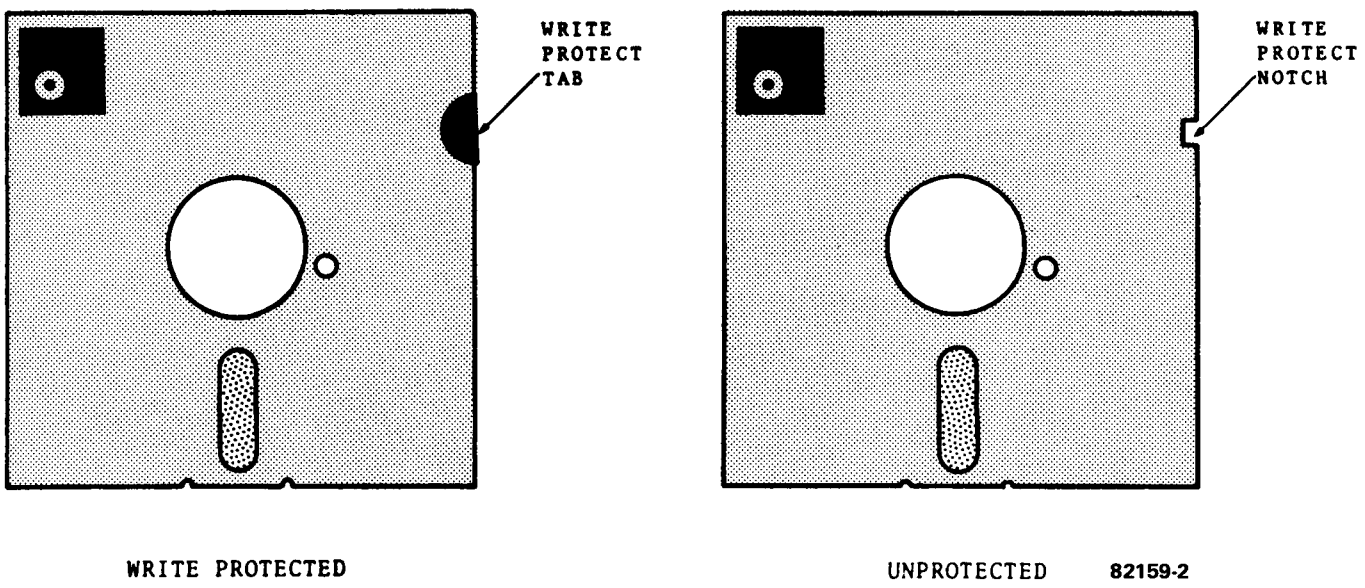
De prijs voor de hele set is ongeveer 150 gulden, dus vrij goedkoop.

2. OS-65D V3.3 bestaat uit
   - vijf 5 inch-diskettes, waarop verschillende gebruikerhulpprogramma's staan (in totaal meer dan 17 utility programs, die het programmeren heel gemakkelijk maken). Alle programma's zijn in BASIC geschreven en kunnen dus indien nodig eenvoudig worden gemodificeerd door de gebruiker.
   - een lege diskette.
   - een handboek in de Engelse taal van 250 pagina's, dat een gedetailleerde beschrijving voor het werken met de DOS, BASIC en assembler bevat. Daarnaast zijn er nog een BASIC-handboek en een assembler-handboek.

De prijs voor OS-65D V3.3 inklusief alle boeken: ongeveer f 300,—. Als men de uitgebreide dokumentatie bekijkt, dan is deze prijs zeker gerechtvaardigd, om niet meteen te zeggen spotgoedkoop. We hebben de twee versies aangepast voor de junior computer en ze werken beide al enkele maanden tot volle tevredenheid.

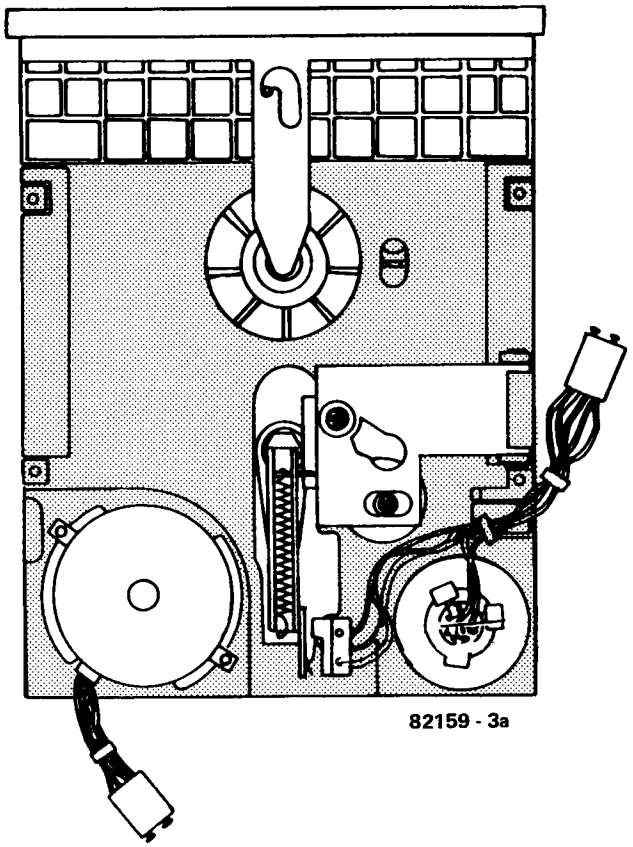
Voordat men met de uitgebreide DOS van Ohio Scientific kan werken, moet men wel het een en ander van het operating system afweten. De beschrijvingen in de handboeken van Ohio zijn weliswaar heel goed, maar om alles te begrijpen dient men toch wel een oude rot in het (computer) vak te zijn. Daarom hebben we besloten de lezer stap voor stap met zo'n disk operating system vertrouwd te maken.

2

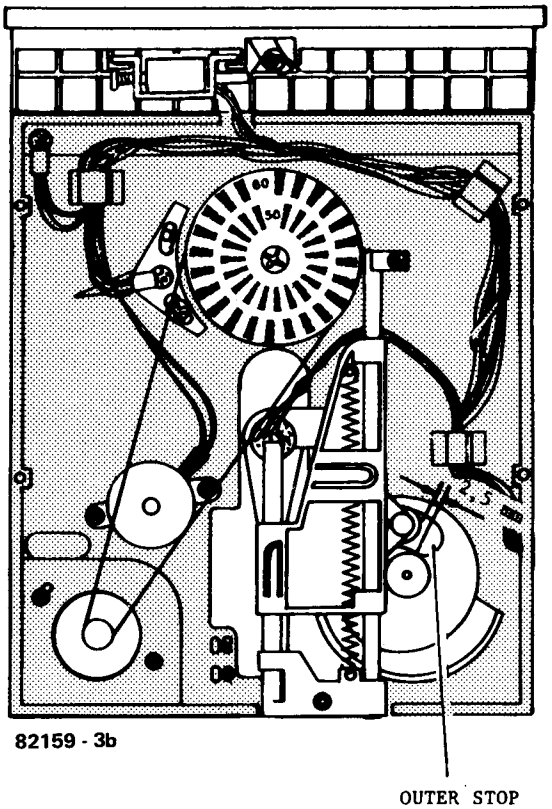


Figuur 2. De diskette heeft aan een zijde een inkeping. Als men deze inkeping bedekt met een plakkertje dat geen licht doorlaat, kan er niet meer op de diskette worden geschreven.

3



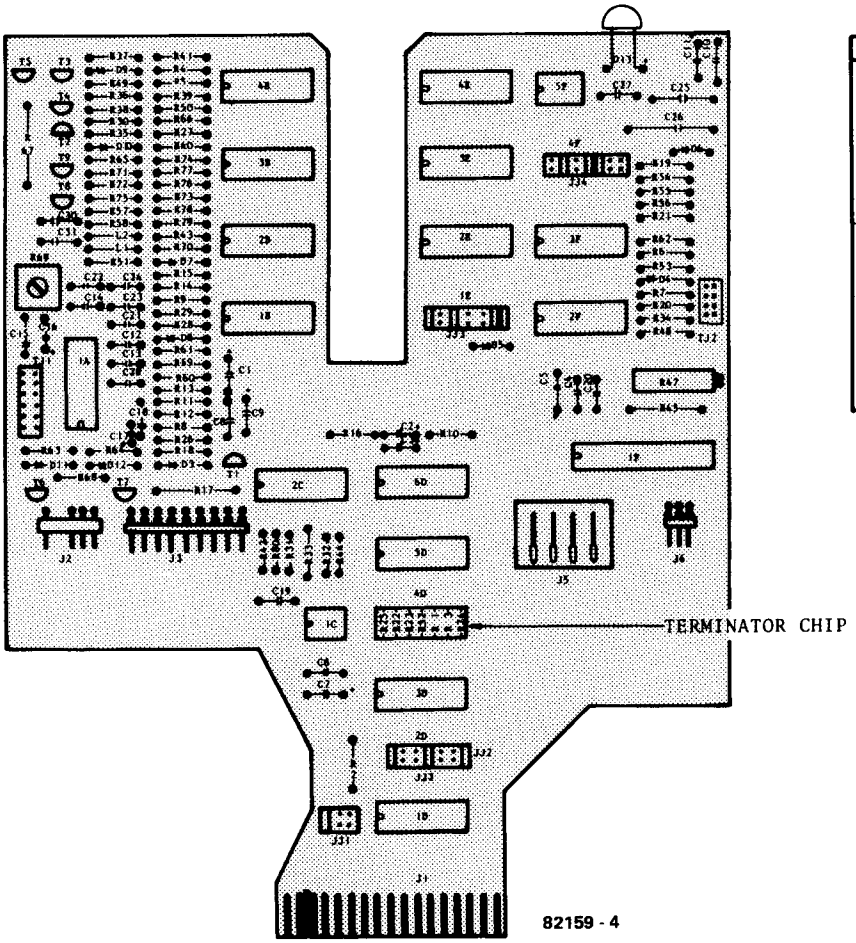
82159 - 3a



82159 - 3b

Figuur 3. De opbouw van een floppy-drive. Figuur 3a toont een drive van Shugart en figuur 3b laat een drive van BASF zien.

4



82159 - 4

Test Points		Signal
TJ1	1,2	Write Current Signal
	3,5	Read Signal (Preamp. Output)
	6	GND
	7,9	Read Signal (Differentiator Input)
	8	Jitter Voltage
	10	Erase Current T.P.
	11,12	Write Current T.P.
TJ2	1	DISK CHANGE FF/
	2	PWRONRESET/
	3	N.O. TRACK ZERO SWITCH
	4	IN USE- FF
	5	MOTOR ON
	6	TRACK 00
	7	GND
	8	INDEX

POTI	FUNCTION
R 47	Drive Motor Adjust
R 69	Jitter Adjust

Connector		Function
J1		Signal - Interface
J2		Read/Write - Head
J3	1,3	Head Load Solenoid
	2,4	Door Lock Solenoid
	5,6	Write Protect Phototransistor
	7,8	Index Phototransistor
	11-18	Stepper Motor
J5		DC- Connector
J6	2,4	Drive Motor
	3,5,6	Track Zero Switch

Figuur 4. Zo ziet de print uit een BASF 6106-drive er uit. Via konnektor J1 loopt de data-overdracht tussen drive en computer. Op konnektor JJ1 zit een kortsluitstekertje, waarbij de plaats van de steker afhankelijk is van het aantal drives dat aangesloten is. In de getekende stand wordt maar één drive gebruikt.

Eerst geven we een algemene beschrijving over de wijze waarop de data op een floppy opgeslagen wordt, daarna gaan we kijken naar de opslagmethode van Ohio Scientific. Ook wordt een beschrijving gegeven van de mechanische opbouw van een floppy-drive.

De floppy-drive

Zoals figuur 1 laat zien heeft elke floppy-drive een klep aan de voorzijde. Deze klep moet men eerst openen om een diskette in of uit het loopwerk te schuiven. De klep mag pas gesloten worden als de diskette helemaal in de sleuf van de drive is geschoven, anders bestaat de kans dat de diskette beschadigd wordt. Bij de klep van de drive is een schakelaar gemonteerd die een kontakt sluit als de klep helemaal gesloten is. Op die manier wordt voorkomen dat de computer data op de diskette leest of schrijft als de klep nog open staat.

Diskettes kunnen worden beveiligd tegen abusievelijk overschrijven. Figuur 2 toont een diskette met aan de rechterkant een inkeping. Een opto-elektronische (soms ook mechanische) schakeling in de floppy-drive kijkt of deze inkeping open of afgedekt is. Als de inkeping bedekt is met bijvoorbeeld een plakkertje, dan is de diskette beveiligd tegen overschrijven. Als de programmeur in dat geval probeert data op de diskette te schrijven, dan geeft de DOS een foutmelding.

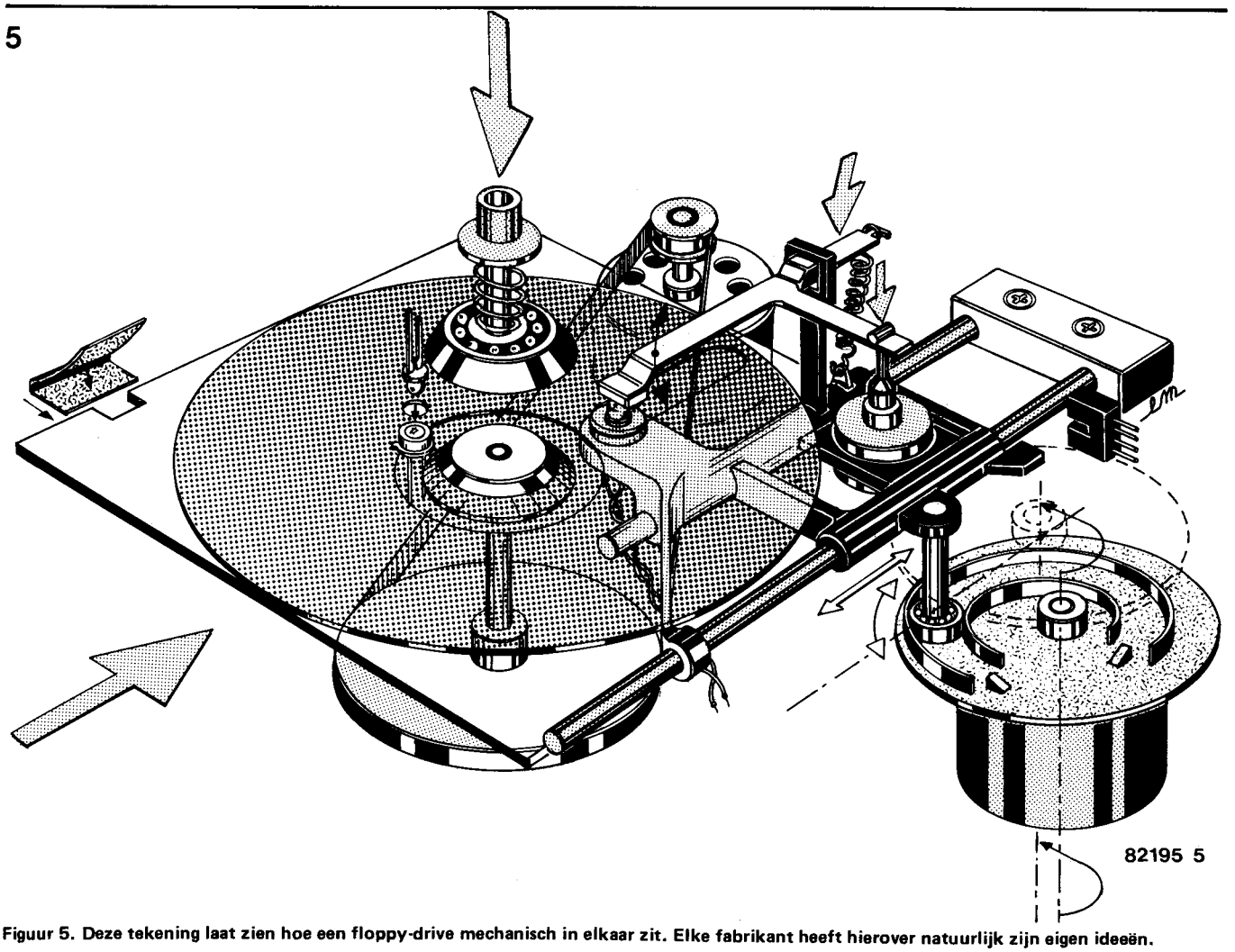
In principe kan men elke floppy-drive bij de hier beschreven interface gebruiken. De enige voorwaarde is, dat de input/output-konnektor van de drive Shugart-kompatibel is. De meeste 5 1/4 inch-drives voldoen wel aan deze voorwaarde. We hebben de DOS in het lab getest met Shugart- en BASF-drives. Het verschil tussen beide soorten drives is alleen dat bij Shugart de lees/schrijf-kop in positie wordt gebracht door middel van een worm, terwijl de positionering bij BASF gebeurt door middel van een schijf met een spiraalvormige inkeping. In figuur 3a is de mechanische opbouw van een Shugart-drive te zien en in figuur 3b een BASF-drive. Beide drives hebben twee motoren:

– een aandrijfmotor (drive motor)
– een stappenmotor (stepper motor)

De aandrijfmotor doet de floppy-schijf met een konstant toerental van 360 omwentelingen per minuut ronddraaien. Een elektronische regeling zorgt ervoor dat het toerental ook bij belastingvariaties konstant wordt gehouden. Het toerental kan bij beide drives tussen bepaalde grenzen gevarieerd worden. De tweede motor, de stappenmotor, zorgt voor de positionering van de lees/schrijf-kop. Ook deze motor wordt door de nodige elektronica gestuurd. Die elektronica is verbonden met de computer en bij elke puls die de computer geeft laat de elektronica de motor één stap maken. Een andere verbinding met de computer geeft het kommando of de motor vooruit of achteruit moet lopen, dus of de kop van buiten naar binnen of van binnen naar buiten moet bewegen.

Verder zitten er nog drie elektromechanische onderdelen op het drive-chassis. De head-load-magneet is, zoals de naam al zegt, verantwoordelijk voor het neerlaten van de lees/schrijf-kop op de magnetische oppervlakte van de diskette (Head Load Solenoid = koplaadmagneet). Als de head-load-magneet niet geaktiveerd is drukt een veer de kop van de diskette af. Bij de BASF-drive is de kop star opgehangen en drukt een aandrukschijf van vilt de diskette tegen de lees/schrijf-kop. Op het chassis zitten verder twee fotosensoren. De ene sensor geeft een puls als de lees/schrijf-kop zich precies boven "track zero" (daarover straks meer) bevindt. Track zero is bij praktisch alle floppy-drives een bijzonder opnamespoor. De tweede sensor "bewaakt" het index-gat, dat in de diskette zit. Het index-gat (zie ook

5

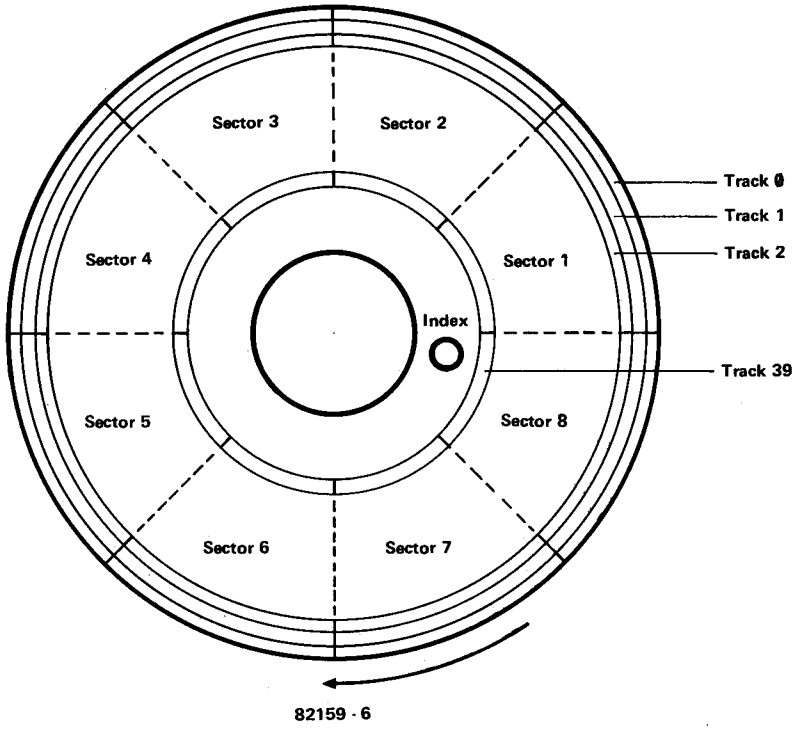


Figuur 5. Deze tekening laat zien hoe een floppy-drive mechanisch in elkaar zit. Elke fabrikant heeft hierover natuurlijk zijn eigen ideeën.

figuur 2) is het absolute nulpunt van de diskette. Anders gezegd: het index-gat vormt het referentiepunt op de floppy-schijf, zodat de computer weet wanneer de diskette een hele omwenteling heeft gemaakt. Bij 360 toeren per minuut geeft de sensor dus om de 166,66 ms een index-impuls.
Samenvattend bestaat een floppy-drive uit de volgende delen:
— een stappenmotor voor de positionering van de lees/schrijf-kop,
— een aandrijfmotor die de diskette met een konstante snelheid laat ronddraaien,
— een sensor die kontroleert of de lees/schrijf-kop precies boven "track zero" is geplaatst,
— een sensor die kijkt of de diskette is beveiligd tegen overschrijven,
— een sensor bij het index-gat, die bij elke omwenteling van de schijf een inpuls geeft,
— een head-load-magneet, die de lees/schrijf-kop neerlaat op de diskette.
Het is wel duidelijk dat voor de sturing van al die onderdelen een flinke brok elektronica nodig is. Daarnaast zijn er ook nog lees- en schrijfversterkers aanwezig voor de kop. Die versterkers lijken veel op de opname- en weergaveversterkers van recorders, alleen met dat verschil dat de drive-versterkers geschikt zijn voor frekwenties van zo'n 125 kHz. De baud-rate van onze floppy-interface bedraagt namelijk 125 KBd!
Al die stuur- en regelelektronica is in de drive ondergebracht. Deze elektronica is mede verantwoordelijk voor de vrij hoge prijs van zo'n floppy-drive. Meestal is het elektronische gedeelte van een drive al afgeregeld, zodat het aansluiten van de drive op de junior computer verder geen problemen geeft.

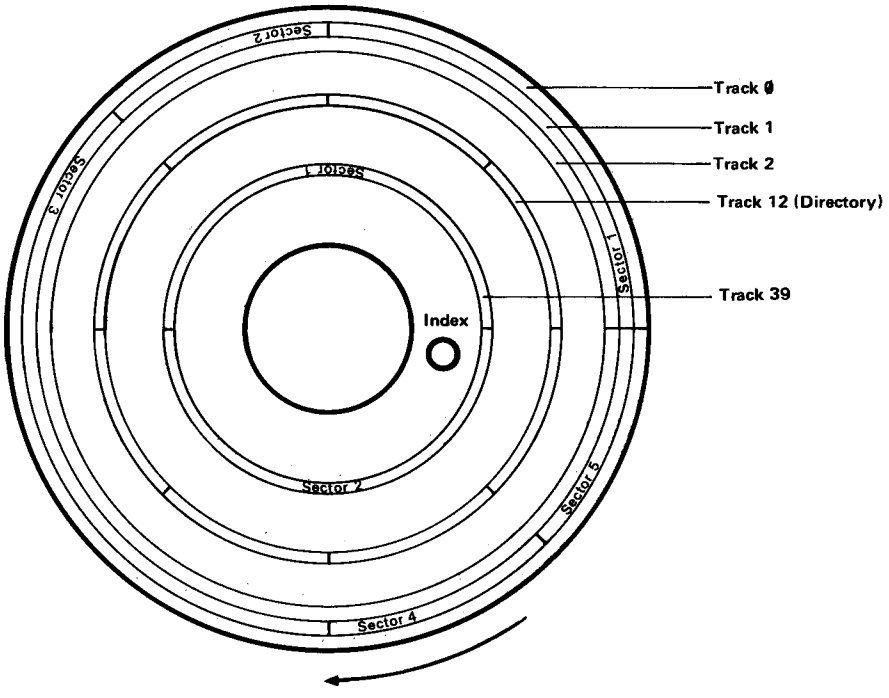
Figuur 4 geeft de opbouw van de print uit een BASF-loopwerk. Voor de gebruiker zijn twee konnektor-aansluitingen van belang: J1 en J5. J1 is de "Shugart-kompatibele" konnektor van het loopwerk. Alle stuursignalen op deze konnektor werken met TTL-nivo. Alle stuursignalen die de floppy-interface uitzendt gaan via konnektor J1 naar de elektronica in het loopwerk. Konnektor J5 is eveneens Shugart-kompatibel; deze zorgt voor de voeding van de hele drive. Het loopwerk heeft twee spanningen nodig: 12 V/800 mA en 5 V/300 mA. De stroomopname is vrij groot, maar in een van de volgende Elektuurs zullen we nader ingaan op het punt voeding voor de floppy-drives.
Wie twee of meer loopwerken op zijn computer wil aansluiten moet ook nog iets weten over konnektor JJ1 en de zogenaamde "terminator-chip". De terminator chip bevat acht weerstanden en bij het gebruik van meer dan een loopwerk is deze chip altijd alleen in de laatste drive aanwezig. Als er bijvoorbeeld twee drives op de junior computer zijn aangesloten bevat de eerste drive, drive A, geen terminator chip en de laatste drive, drive B, wel een terminator chip. Worden er vier drives aangesloten,

6



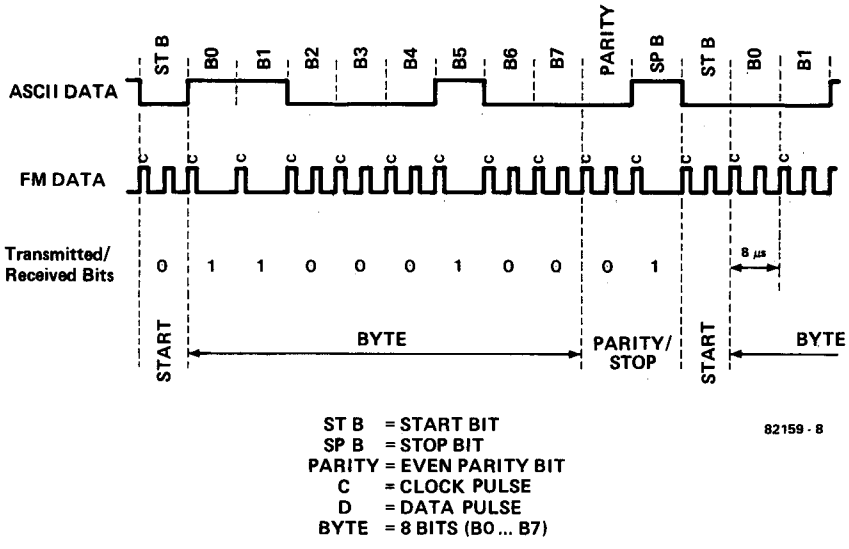
Figuur 6. De informatie wordt in de vorm van cirkels op de diskette gezet. Zo'n spoor noemt men een "track". Bij een minidiskette zitten op een kant 40 sporen. Elk spoor is nog eens onderverdeeld in verschillende sektoren. Een index-gat in de diskette geeft het beginpunt aan van de sektoren van een spoor.

7



Figuur 7. Bij een zogenaamde soft-sektor-indeling kunnen de sektoren verschillende lengtes hebben. Het is zelfs mogelijk een spoor te verdelen in één of in acht sektoren. Spoor 12 is bij de junior computer gereserveerd voor de "directory" (inhoudsopgave).

8



Figuur 8. Zo ziet het signaal er uit dat de floppy-interface verzendt en ontvangt.

dan bevat alleen drive D een terminator chip. De volgorde van de drives kan men instellen met behulp van de kortsluitsteker op JJ1. In de getekende stand is gekozen voor "drive A". Als men de steker een plaats verder naar rechts zet betekent dat drive B. Nog een plaatsje verder naar rechts is drive C. Voor drive D wordt de kortsluitsteker weggelaten, want deze drive is duidelijk geïdentificeerd door de terminator chip. Instelpotmeter R69 dient voor de fijnafregeling van de leesversterker. Draai nooit aan deze potmeter! Hij bepaalt de kwaliteit van het gelezen signaal (jitter-vrijheid).

De loopwerkmechanica

Figuur 5 toont schematisch de mechanische opbouw van een loopwerk. De floppy-schijf wordt op dezelfde manier als bij een platenspeler door de aandrijfmotor rondgedraaid. De kop zit op de oppervlakte van de diskette en zet de magneetveldjes op de schijf om in elektrische signalen. Het opnemen en afspelen van de diskette gebeurt volgens hetzelfde principe als bij de band- en cassette-recorder.

In de oppervlakte van de diskette zijn geen rillen aanwezig, zodat het niet mogelijk is de kop via rillen naar een gewenste plaats op de schijf te voeren (zoals bij een langspeelplaat).

De kop moet dus boven het juiste spoor worden gebracht door middel van een stappenmotor. Deze motor voert de kop, die zich op een wagen bevindt, van spoor tot spoor. Die hele wagen met daarop de kop kan dus naar links en naar rechts bewegen. Na elke lees- of schrijfoperatie wordt de kop weer onmiddellijk van het oppervlak van de schijf afgehaald, anders zou de oppervlakte van de floppy disk vrij vlug versleten zijn. Als de kop op het diskette-oppervlak zit noemt men dat in het Engels "loaded head". Staat de kop omhoog, dan heet dat "unloaded head". Zou men de kop kontinu op de schijf laten staan, dan is het spoor van de diskette na 50 uur helemaal weggesleten. Bij normaal gebruik bedraagt de levensduur van een diskette meerdere jaren bij veelvuldig gebruik.

Sektor-indeling bij een diskette

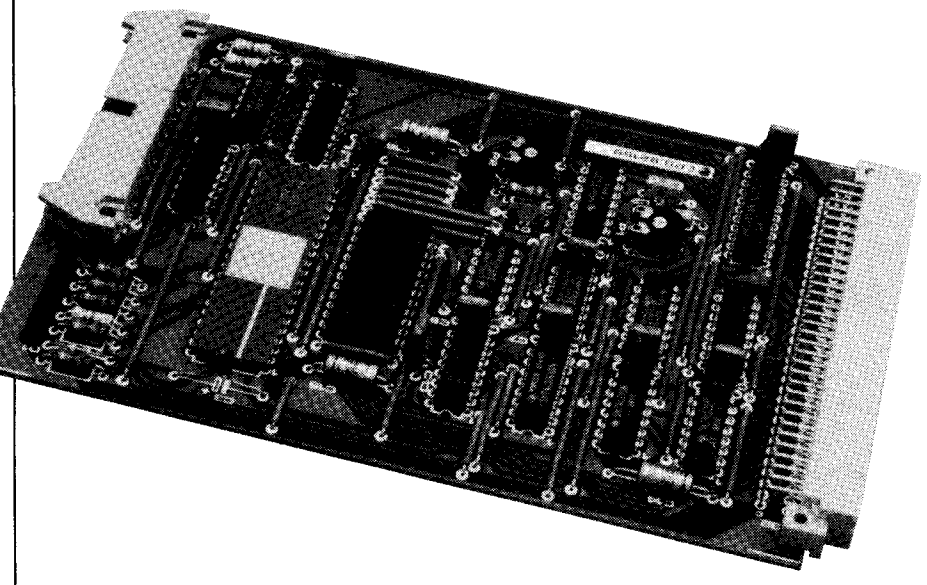
We zullen nu beschrijven op welke manier de data op een diskette opgeslagen wordt. Bij minifloppy's, zoals we die bij de junior computer gebruiken, wordt de data op 40 koncentrische ringen (tracks) vastgelegd. De spoorbreedte van elke ring is slechts 0,2 mm! De buitenste ring van de diskette wordt "track zero" of nulspoor genoemd. Dit spoor doet bij de meeste disk operating systemen (ook bij OS-65D) dienst als referentiespoor voor de andere sporen op de diskette. Figuur 6 laat zien dat de diskette ook nog op een andere manier in stukken gedeeld is. Naast de 40 sporen zijn er nog de sektoren. Voor de duidelijkheid hebben we in de figuur acht sektoren getekend. Sektor 1 ligt altijd vlak achter het index-gat (gezien vanuit de draairichting). Die sektor ligt een stukje achter het index-gat omdat bij OS-65D altijd eerst een millisekonde na het ontvangen van de index-impuls gewacht wordt en daarna pas wordt begonnen met het lezen/schrijven van data.

Wat de sektor-indeling van een floppy betreft bestaan er diverse formaten. Het meest bekende is het IBM-3740-formaat, maar dat wordt niet door Ohio Scientific gebruikt. Daarom gaan we niet verder op het IBM-formaat in en beperken we ons tot het Ohio-formaat.

Een datablok op de diskette is duidelijk geïdentificeerd door het spoor-nummer en het sektor-nummer. De diskette die in figuur 6 is getekend heeft allemaal sektoren van gelijke lengte. Het is echter ook mogelijk op een diskette sektoren van verschillende lengte onder te brengen. De minimum lengte van de data, die bij Ohio Scientific op een sektor kan worden gezet, bedraagt een 6502-pagina oftewel 256 bytes. Bij de diskette vormen het sektor-nummer en het spoornummer dus de koördinaten waarmee een data-block op de schijf bliksemsnel kan worden opgespoord.

In figuur 7 is een diskette met variabele sektorlengte getekend. Dit formaat wordt ook gebruikt door de DOS die we voor de junior computer hebben aangepast. Het buitenste spoor van de diskette, track zero, heeft een heel bijzonder formaat dat straks nog verklaard wordt. Spoor 1 is in meerdere sektoren verdeeld: sektor 1 bevat 2 pagina's, dus 2 maal 256 bytes. Sektor 2 van spoor 1 is maar half zo lang als sektor 1 en bevat dan ook maar 256 bytes. Sektor 3 bevat 3 pagina's dus 3 maal 256 bytes. Sektor 4 en 5 bevatten tenslotte elk 256 bytes. Een draaiing van 45° van de diskette komt overeen met een pagina van 256 bytes, dus met deze drie sektoren is spoor 1 helemaal vol. Het is ook mogelijk een enkele sektor op een spoor te plaatsen. Dat is gedaan bij spoor 2 in figuur 7. Die sektor kan dan maximaal 8 pagina's (2048 bytes) bevatten. Aangezien per sektor ook nog formaat-informatie moet worden toegevoegd op de diskette is het uit veiligheidsredenen aan te bevelen niet meer dan zeven sektoren op een spoor te zetten.

Een bijzondere funktie heeft spoor 12. Op dit spoor staat de "directory" (inhoudsopgave) van de diskette. Met behulp van de BASIC-interpreter is het mogelijk een file (bijvoorbeeld een BASIC-programma of een tekst) in de computer op te slaan en die dan een eigen naam te geven. Het zou anders voor de programmeur onmogelijk zijn om te onthouden op welk spoor en in welke sektor een programma of een file is weggezet. Daarom heeft de Ohio-DOS de mogelijkheid om programma's een naam te geven. Een programma- of file-naam mag uit maximaal zes alfanumerieke tekens



bestaan, waarbij het eerste teken een letter (A . . . Z) moet zijn. Stel dat we een BASIC-programma hebben geschreven voor het berekenen van een cirkelomtrek, en we willen dit programma op diskette zetten. We noemen het programma dan "CIRKEL", waarna het programma op diskette kan worden weggeschreven met het eenvoudige kommando: DISK! "PUT CIRKEL". De computer zet het programma dan op diskette. Het programma kan men daarna weer in de computer laden met het kommando: DISK! "LOAD CIRKEL". De kommando's van de DOS zullen we later nog behandelen.
Voordat de computer een file op de diskette kan schrijven of lezen moet de file-naam in de inhoudsopgave staan. Ohio levert diverse hulpprogramma's op diskette om file-namen in de "directory" te kunnen zetten.

Data-pulsen voor de floppy-drive

Op deze plaats willen we even tonen hoe de elektrische signalen er uit zien die de computer naar het loopwerk stuurt. Ohio gebruikt hiervoor een heel eenvoudig formaat. De data wordt, evenals bij de printer-interface van de junior computer, asynchroon verstuurd. De printer-interface kan echter hooguit met 2400 baud werken, terwijl de floppy-interface werkt met 12500 baud. Een ACIA van het type MC 6850, een IC van zo'n tien gulden, maakt deze hoge overdrachtssnelheid mogelijk. De data die de ACIA (Asynchronous Communications Interface Adapter) levert ziet er als volgt uit:
— een startbit
— acht databits
— een even-pariteitsbit
— een stopbit

Het even-pariteitsbit is een kontrolebit waarmee eventuele fouten bij de overdracht kunnen worden opgespoord. Dit bit wordt geset als het aantal "1"-bits in het verzonden byte een even aantal is.
De elektronica in het loopwerk kan het seriële signaal dat de ACIA levert niet verwerken. Daarom wordt het seriële signaal omgezet in een frekwentiegemoduleerd signaal. Figuur 8 laat zien hoe deze omzetting gebeurt. Bij elk databit wordt een zeer korte clock-impuls gegeven, die slechts enkele honderden nanosekonden duurt. Als het verzonden bit logisch nul is, dan wordt tussen twee clock-pulsen "C" nog een data-puls "D" uitgezonden. Bij een logische één staat er niets tussen twee clock-pulsen. De tijd die nodig is voor het verzenden van een bit bedraagt slechts acht mikrosekonden.
Bij de ontvangst van door de floppy-drive verstuurde data moet het frekwentiegemoduleerde signaal door de interface weer worden terugvertaald in een serieel data-signaal. Een data-separator op de floppy-interface zorgt hiervoor.
We hebben nu alle belangrijke punten bij de floppy en de floppy-drive bekeken. Voordat we nu een beschrijving gaan geven van de interface-hardware eerst nog even een opsomming van alles wat men nodig heeft om een junior computer of een andere 6502-computer om te bouwen tot een DOS-computer:
— Er zijn minstens twee dynamische RAM-kaarten nodig (zie Elektuur april 1982). Drie RAM-kaarten zijn nodig bij de ontwikkeling van grotere programma's.
— Een junior computer, bestaande uit basis-print, interface-kaart en een bus-print met vijf konnektoren.
— een floppy-interface-kaart, waarop een paar TTL-IC's en een MC 6850 met een MC 6821 zitten.
— Eén of liever twee floppy-loopwerken met Shugart-kompatibele aansluitingen. Bijvoorbeeld 5 1/4-inch-loopwerken van BASF, Shugart, Teac, enz. Zelfs goedkope loopwerken uit aanbiedingen zijn bruikbaar.
— Een netvoeding die de volgende spanningen en stromen levert:
+ 5 V/5 A
+12 V/2,5 A
+12 V/400 mA
−5 V/400 mA
−12 V/400 mA

De hardware voor de floppy-interface

In figuur 9 is het schema van de floppy-interface afgedrukt. Bij het bekijken van het schema valt waarschijnlijk op dat alleen maar standaard-onderdelen zijn gebruikt. We zijn daar ook best een beetje trots op: deze universele floppy-interface is de goedkoopste interface die momenteel verkrijgbaar is. Alle KIM-1-, AIM-65- en SYM-bezitters kunnen nu ook hun computer "omschakelen" van cassette op floppy. Maar voordat de floppy-interface wordt gebouwd en aangesloten gaan we eerst vertellen hoe die hardware funktioneert.

Data-transfer tussen computer en floppy

Het dataverkeer tussen computer en floppy loopt via de volgende lijnen:

- De STEP- en DIR-lijnen (uitgangen)

Door de Peripheral Interface Adapter IC5 (afgekort PIA) wordt de lees/schrijfkop van de disk-drive op het gewenste spoor geplaatst. Via poort PB3 zendt de computer stepper-impulsen uit (voor de stappenmotor), waarbij de buffer N18 zorgt voor de aanpassing op de drive-elektronica. Bij elke impuls wordt de lees/schrijfkop een spoor verder naar binnen of naar buiten verplaatst. PB2 van de PIA en N19 leveren het DIR-signaal. Het logische nivo op de DIR-lijn bepaalt of de kop van binnen naar buiten beweegt of juist omgekeerd.

- De TRØ-lijn (ingang)

De TRØ-lijn is een terugmeldingslijn van de drive naar de computer. Het logische nivo op deze lijn geeft aan of de lees/schrijfkop boven track zero staat.

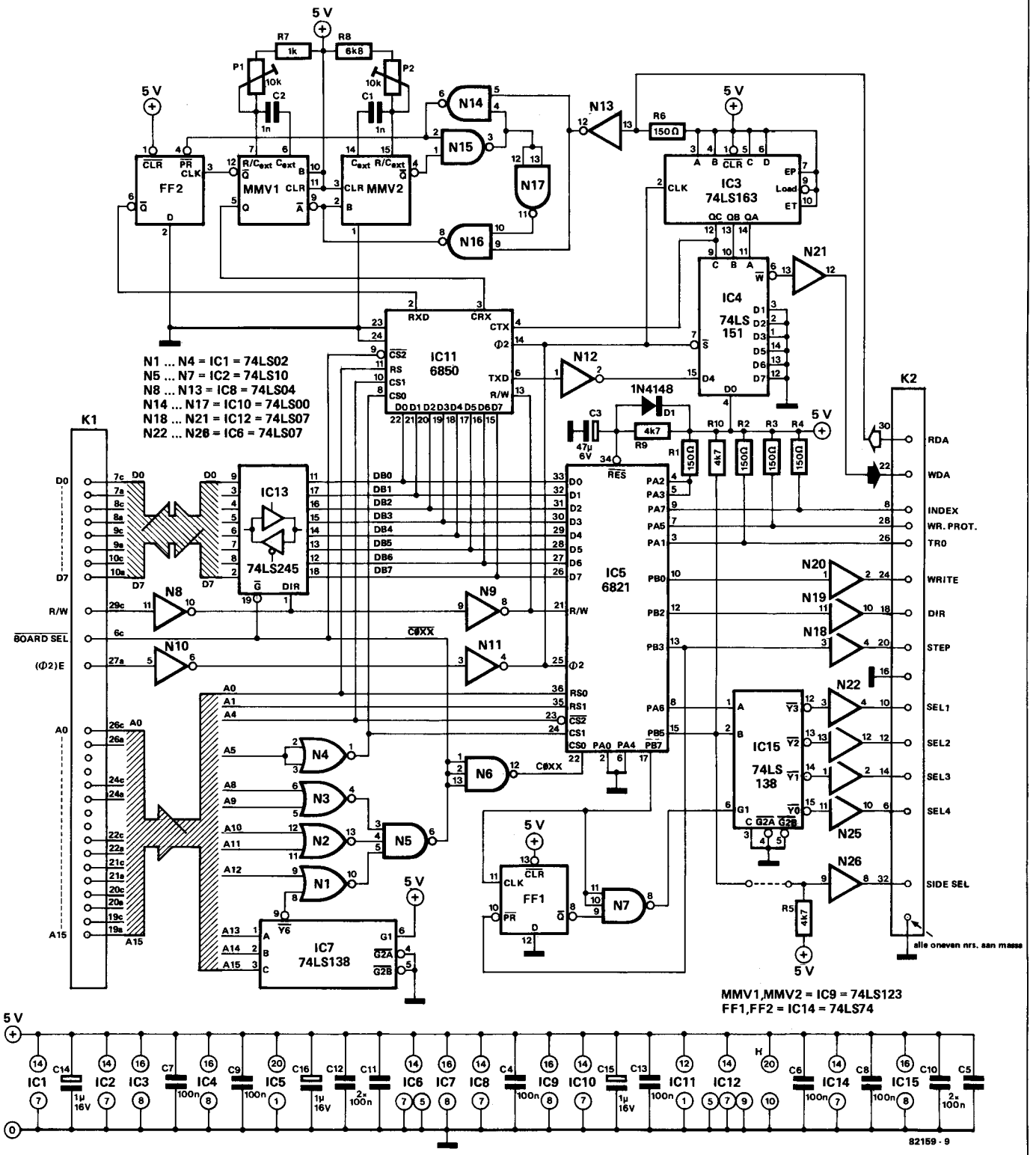
- De INDEX-lijn (ingang)

Ook dit is een terugmeldingslijn van de drive naar de computer. Telkens als het index-gat van de diskette de sensor passeert wordt via deze lijn een impuls gegeven. Deze puls wordt gebruikt voor de sektor-indeling.

- De WR.PROT-lijn (ingang)

Via deze lijn krijgt de computer medegedeeld of het toegestaan is op de diskette van de gekozen drive te schrijven. Alleen als de WR. PROT-lijn niet aktief is (dus als de diskette niet beschermd is tegen overschrijven) kan de computer op de floppy schrijven.

9



Figuur 9. Het schema van de floppy-interface. Doordat de data-uitwisseling tussen computer en drives door software gestuurd wordt is een dure floppy-disk-controller niet nodig. In de schakeling worden alleen gewone onderdelen gebruikt.

- De WRITE-lijn (uitgang)

De WRITE-lijn schakelt de elektronica van de floppy-drive om van de leesnaar de schrijftoestand. Voordat deze lijn aktief wordt gemaakt kontroleert de computer eerst via de WR. PROT-lijn of de diskette is beschermd tegen overschrijven. Als de diskette daartegen is beschermd kan de WRITE-lijn niet aktief worden gemaakt.

- De lijnen SEL1, SEL2, SEL3 en SEL4 (uitgangen)

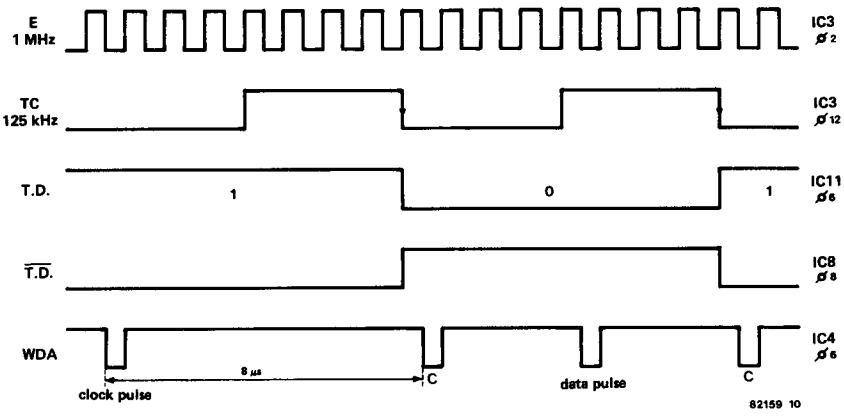
Via de SEL-lijnen kan de computer een van de vier drives kiezen. Gewoonlijk worden alleen de lijnen SEL1 en SEL2 gebruikt, waarbij SEL1 dan "drive A" en SEL2 "drive B" stuurt. Bij het gebruik van de Ohio-software moet op lijn SEL1 altijd een floppy-drive zijn aangesloten.

- De SIDE SEL-lijn (uitgang)

Deze lijn is bedoeld voor latere uitbreidingen en wordt momenteel niet gebruikt bij de junior computer. Met deze lijn kan men speciale drives met twee lees/schrijfkoppen sturen. Zulke drives kunnen beide zijden van de diskette beschrijven en lezen.

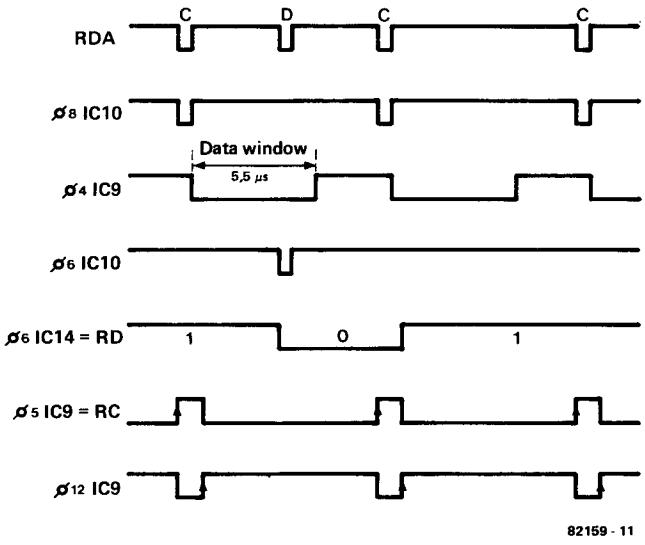
- De lijnen WDA (uitgang) en RDA (ingang)

10



Figuur 10. Dit tijdvolgordediagram toont verschillende signalen uit de "write data"-encoder. Dat is het gedeelte voor de signalen die naar de drive verstuurd worden.

11



Figuur 11. Het tijdvolgordediagram voor signalen van de "read data"-separator (de schakeling voor het verwerken van de impulsen die van de drive afkomstig zijn).

Via de WDA-lijn stuurt de computer de seriële data naar de drive-elektronica. De seriële data van de drive naar de computer loopt via de RDA-lijn. De baudrate op deze lijnen is 125 kilobaud. De data-overdracht van de computer naar de drive kan men vergelijken met een eenvoudige seriële V24/RS232-interface. Deze interface is al uitvoerig besproken in de boeken junior computer deel 3 en deel 4. Via deze interface loopt de data van computer naar Elekterminal en van Elekterminal naar computer.

De ACIA IC11 zet de parallel-data van de computer om in seriële data die dan met een snelheid van 125 Kbd wordt afgegeven door de T x D-uitgang. De seriële data kan niet in die vorm naar de diskette worden gestuurd, maar moet eerst nog frekwentiegemoduleerd worden. Op welke manier dat gebeurt is al bij de beschrijving van figuur 8 ter sprake gekomen. Als de computer data van de diskette leest moet het gemoduleerde signaal van de drive weer omgezet worden in "gewone" data. Dit werk wordt verricht door een data-separator die bestaat uit de poorten N13 . . . N17, de twee monoflops MMV1 en MMV2 en flipflop FF2. Uitgang Q van MMV1 levert de clock-pulsen voor de ACIA en uitgang $\overline{Q}$ van FF2 voorziet de ACIA weer van seriële V24/RS232-data. De ACIA zet het signaal daarna weer om in parallel-data die door de computer via de databus kan worden gelezen. Aangezien er bij de seriële data-overdracht tussen computer en drive gebruik wordt gemaakt van een I/O-chip die gewoonlijk voor V24/RS232-doeleinden wordt gebruikt, zijn er in het verstuurde data-patroon enkele bekende trekjes te vinden:

1. Elk verstuurde byte begint met een startbit en eindigt met een pariteits-bit.
2. Tussen twee bytes bevindt zich nog een stopbit, dat het geïnverteerde is van het startbit.

Voor de overdracht van een byte zijn dus elf bits nodig: een startbit, acht databits, een pariteitsbit en een stopbit. Als er geen data wordt verstuurd staan alleen stopbits op de lijn (logisch een). Door de toepassing van de pseudo-FM-modulatie worden bij de overdracht van een byte (= 8 bits) dat alleen maar uit "enen" bestaat (= $ FF) 22 impulsen op de diskette geschreven (zie ook figuur 8).

In totaal kunnen er acht pagina's van elk 256 bytes op een spoor, zoals we al bij de indeling van de floppy disk hebben gezien. Aangezien er enkele sporen nodig zijn voor de "directory" blijven van de 40 sporen nog maar 35 over voor "direkt" gebruik. Dat komt neer op een totale opslagkapaciteit op een diskette van 2 x 35 = 70 Kbytes. Meer dan genoeg voor een hobby-computersysteem.

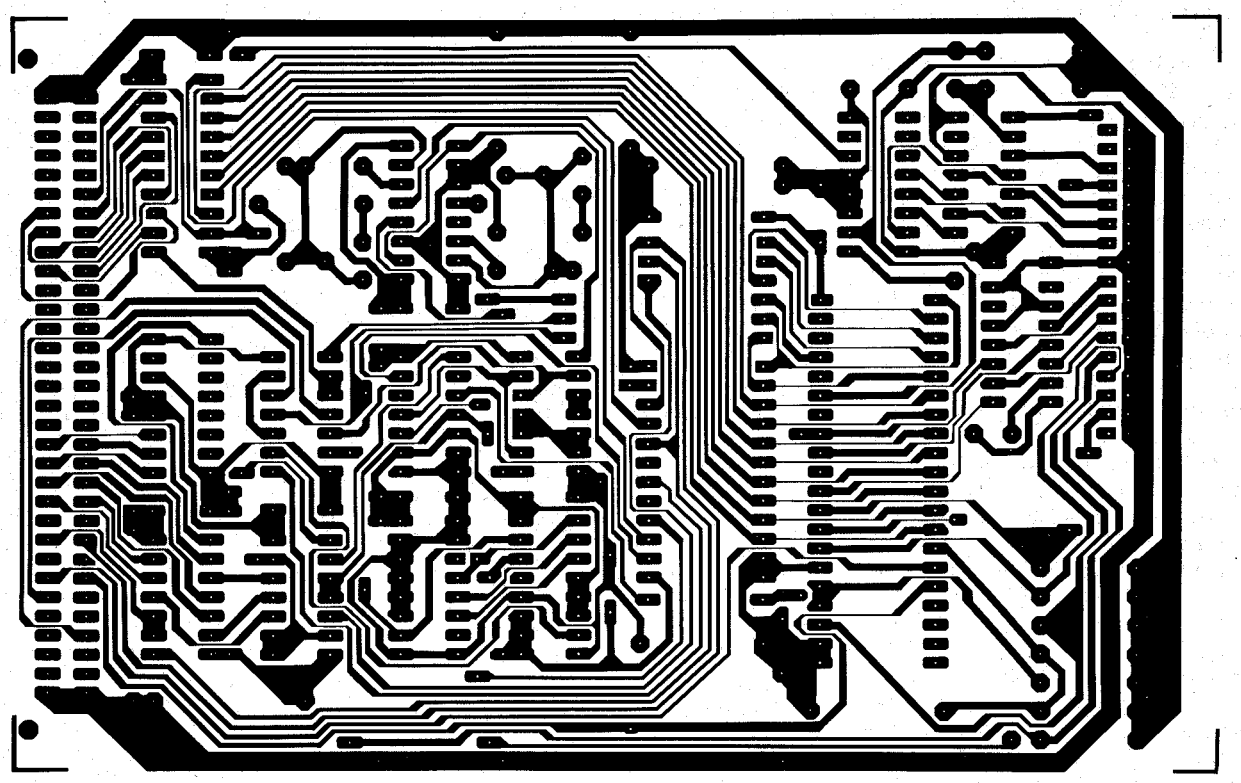
Het schema in detail

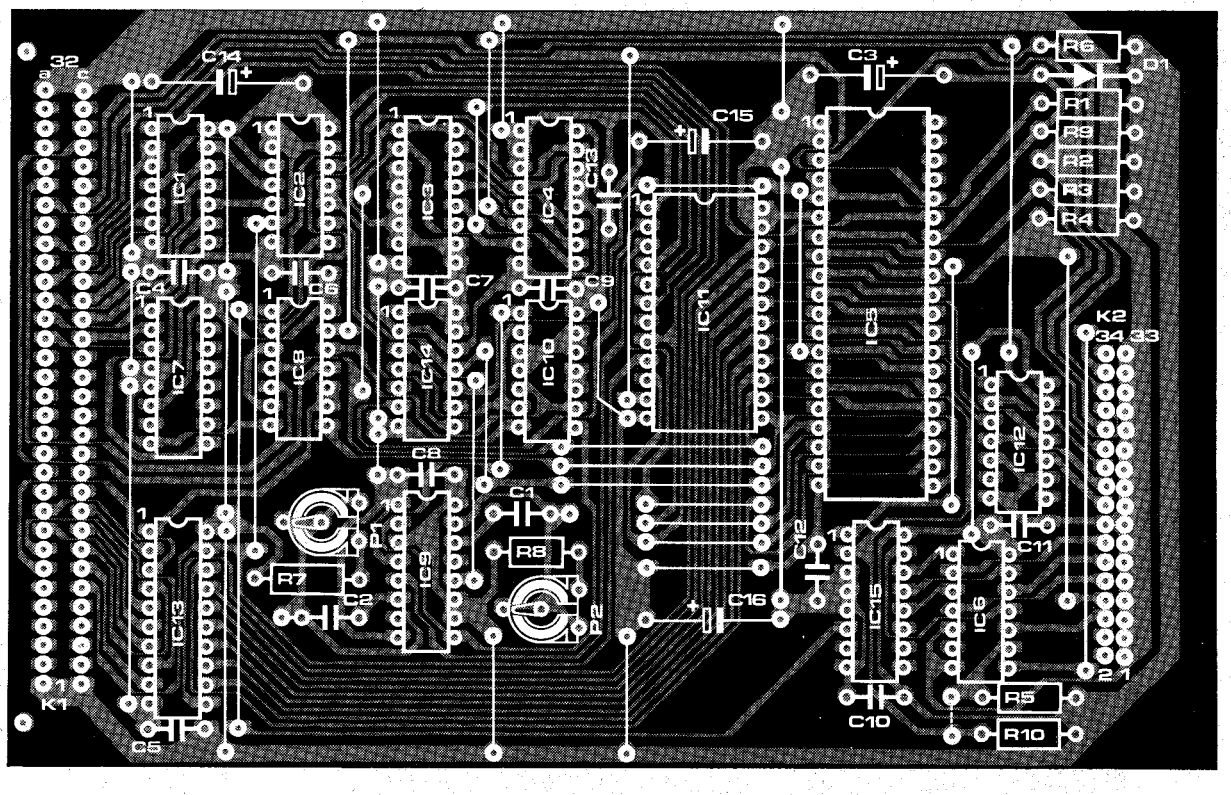
De schakeling van de floppy-interface kan men opsplitsen in verschillende delen:

a. Adreskodering en databuffer

De ingangen van de adresdekoder IC7 zijn verbonden met de adreslijnen A13, A14 en A15, waardoor het uitgangsnivo van uitgang Y6 elke 8 Kbyte verandert. Uitgang Y6 is door middel van de poorten N1, N2, N3 en N5 zodanig verknoopt met de adreslijnen A8 . . . A12 dat de uitgang van N5 logisch nul is tussen de adressen $ C000 . . . $ C0FF. Het uitgangssignaal van de poort N5 wordt gebruikt voor het aktiveren van de databusbuffer IC13. Verder is dit signaal $\overline{(COXX)}$ uitgevoerd naar pen 6c van buskonnektor K1. De doorlaatrichting van de databusbuffer wordt bepaald door het R/W-signaal dat via N8 naar IC13 gaat en via N9 naar IC5. Het door N6 geleverde C0XX-signaal gaat naar de CSØ-aansluiting van de PIA IC5. De andere chip-select-aansluitingen

12





Figuur 12. Print-layout en komponentenopstelling voor de schakeling van de floppy-disk-interface.

Onderdelenlijst

Weerstanden:

R1 . . . R4,R6 = 150 Ω
R5,R9,R10 = 4k7
R7 = 1 k
R8 = 6k8
P1,P2 = 10 k instelpotmeter

Kondensatoren:

C1,C2 = 1 n MKT
C3 = 47 µ/6,3 V
C4 . . . C13 = 100 n
C14,C15,C16 = 1 µ/16 V

Halfgeleiders:

D1 = 1N4148
IC1 = 74LS02
IC2 = 74LS10
IC3 = 74LS163
IC4 = 74LS151
IC5 = 6821
IC6,IC12 = 74LS07
IC7,IC15 = 74LS138
IC8 = 74LS04
IC9 = 74LS123
IC10 = 74LS00
IC11 = 6850
IC13 = 74LS245
IC14 = 74LS74

Diversen:

1 64-polige konnektor volgens DIN 41612, male
2 23-polige bandkabelkonnektors met bijbehorende haakse stekers

zijn verbonden met de adreslijnen A4 en A5, zodat het basis-adres van de PIA $ CØØØ is.

b. De verbindingen tussen drive en interface

De uitgangen naar de floppy-drive worden gebufferd door de poorten N18 . . . N26. Deze buffers hebben een open-kollektor-uitgang. De benodigde pull-up weerstanden zitten in de laatste drive (in de terminator chip). Ook de uitgangen van de floppy-drive hebben buffers met een open kollektor. R1 . . . R4 en R6 dienen hierbij als pull-up-weerstanden.

IC15 multiplext de lijnen PA6 en PB5 van de PIA, zodat het mogelijk is vier drives via een 34-polige kabel te sturen. Door middel van een kleine modifikatie op de interface-print is het zelfs mogelijk twee dubbelzijdige drives op de computer aan te sluiten. Die modifikatie bestaat uit het doorverbinden van de ingang van N26 met PB5 en het onderbreken van de verbinding tussen PB5 en pen 2 van IC15.

De multiplexer wordt gestuurd door N7. De ingangen van deze poort zijn aangesloten op de head-load-uitgang PB7 van de PIA en op de Q-uitgang van FF1. Deze flipflop wordt geset door de step-impulsen en gereset door de neergaande flank van de head-load-impulsen. N7 en FF1 zijn niet per se nodig, maar ze zitten toch op de interface-print in verband met de aparte head-load-lijn. Die lijn is door Ohio overgenomen van de 8 inch-drives. Mini-disk-drives gebruiken de select-lijnen voor het aktiveren van de lees/schrijf-kop. Om de kop niet voortdurend over de oppervlakte van de diskette te laten "schuren" wordt de select-lijn gestuurd door de head-load-lijn. Op die manier wordt de diskette zo veel mogelijk gespaard.

De poortlijnen van de PIA

De poortlijnen van de PIA worden als volgt gebruikt:

A-kant: adres $ CØØØ; disk status poort		
PAØ: drive 0 ready	ingang	
PA1: track 0	ingang	X
PA2: fault	ingang	
PA3: vrij voor gebruiker		
PA4: drive 1 ready	ingang	X
PA5: write protect	ingang	X
PA6: drive select L	uitgang	X
PA7: index-impuls	ingang	X
B-kant: adres $ CØØ2; disk control poort		
PBØ: write enable	uitgang	X
PB1: erase enable	uitgang	
PB2: step direction	uitgang	X
PB3: step-impuls	uitgang	X
PB4: fault reset	uitgang	
PB5: drive select H	uitgang	X
PB6: low current	uitgang	
PB7: head load	uitgang	X

Alle I/O-lijnen werken met negatieve logika

"X" = gebruikte I/O-lijn

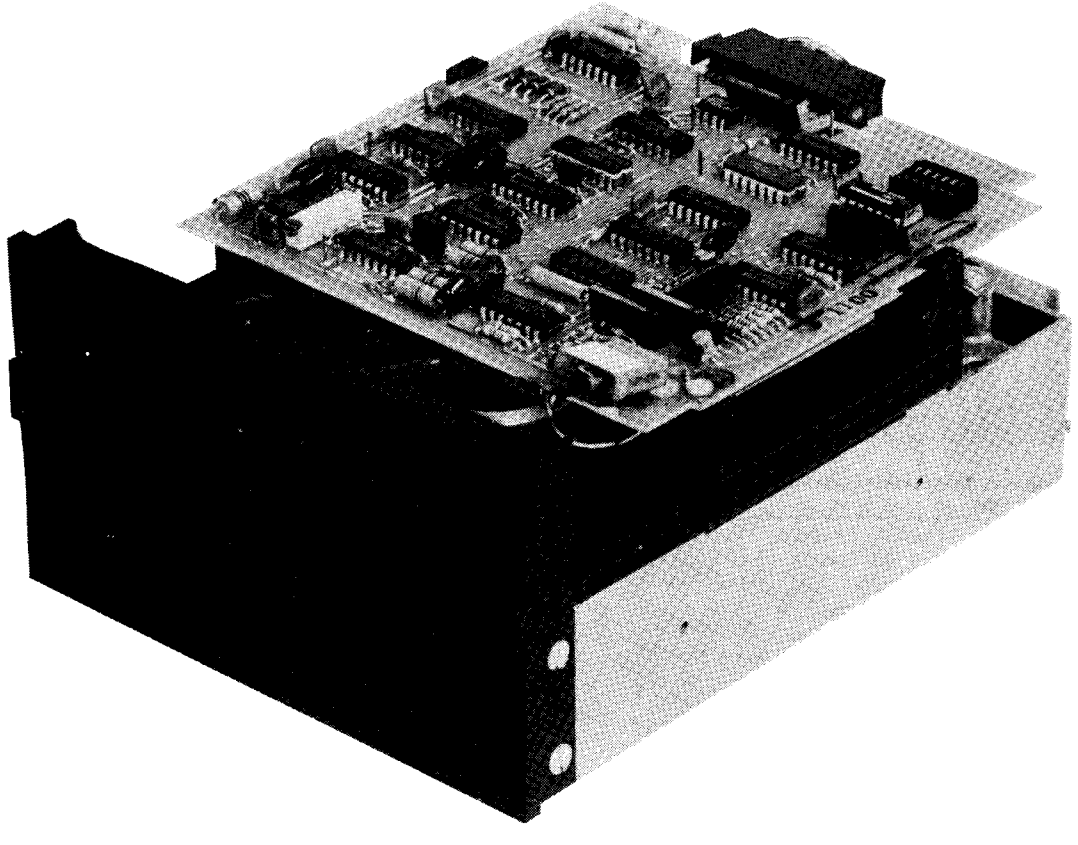
De elektronica voor de data-overdracht

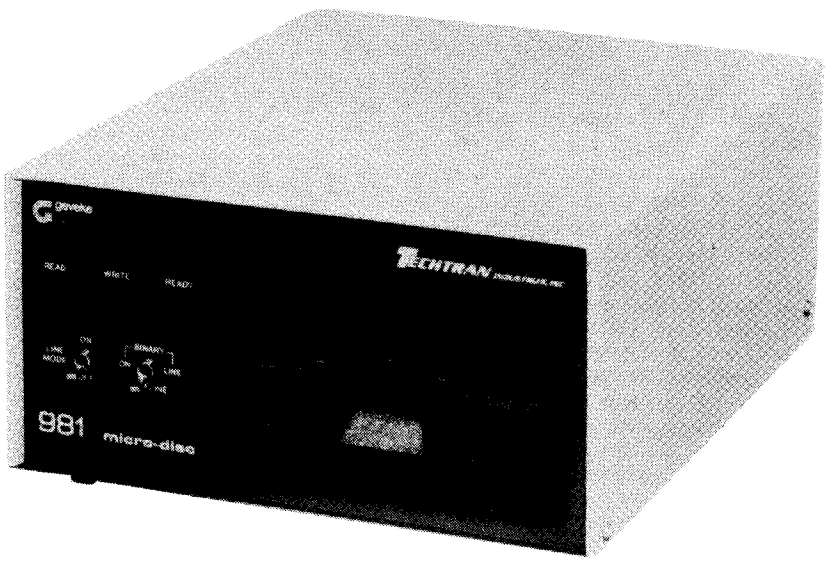
De data-overdracht wordt in de eerste plaats verzorgd door de ACIA 6850 (IC11). De computer schrijft eerst het te versturen byte via de databus in het zendregister van de ACIA. Vervolgens schuift IC11 dat byte in seriële vorm naar buiten via zijn T x D-uitgang. Het seriële signaal dat van de diskette komt ontvangt de ACIA op zijn R x D-ingang. De clock-ingang voor het ontvangen signaal is aangeduid met CRx. Als de ACIA een serieel woord heeft ingelezen kan de computer het in parallelvorm uit het ontvangregister lezen. De registerstruktuur van de ACIA zal in een volgende Elektuur worden besproken.

De door IC11 uitgezonden data wordt door N12 geïnverteerd en gaat dan naar ingang D4 van de dataselector IC4. Alle andere data-ingangen, met uitzondering van DØ, zijn aan massa gelegd. De select-ingang van IC4 en ingang Φ 2 van IC11 worden gestuurd door het signaal "E", dat via de poorten N10 en N11 komt van pen 27a van konnektor K1. De teller IC3 deelt het clock-signaal door 8 en stuurt dan met zijn uitgangen Q_A, Q_B en Q_C achtereenvolgens de acht adres-ingangen van IC4 aan. Aan pen 6 van het IC ontstaat dan een signaal zoals in figuur 8 te zien is. Pen 6 van IC4 is namelijk altijd "0" als het "E"-signaal logisch nul is en de data op ingang D4 "1" is. Dat betekent dat op adres nul steeds een impuls wordt gegeven. Dit is de clock-impuls, die elke acht mikrosekonden herhaald wordt. Als de ACIA (IC11) een logisch één-signaal afgeeft is T x D "1" en D4 van de dataselector "0". Dat heeft tot gevolg:

— Bij de overdracht van een logische één geeft de W-uitgang van IC4 geen data-puls.

— Bij de overdracht van een logische nul wordt aan de W-uitgang van IC4 een data-impuls tussen twee clock-



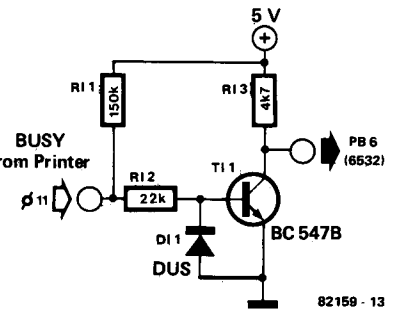


pulsen gegeven.
– Elke clock- en data-impuls duurt slechts 500 nanosekonden.

Het gekodeerde signaal aan de W-uitgang van de dataselector gaat via de buffer N21 naar de floppy-drive. Het "write data"-tijdvolgordediagram is te zien in figuur 10.

De data die terugkomt van de floppy-drive moet ook weer gedekodeerd worden voordat ze verder verwerkt kan worden. Daartoe moeten de clock- en data-impulsen van elkaar worden gescheiden. Nadat dit gebeurd is worden de clock-pulsen gebruikt om de seriële data-pulsen in de ACIA te schuiven met een snelheid van 125 KBd. Het scheiden van de clock- en data-pulsen wordt gedaan door een data-separator die is opgebouwd uit de komponenten N13, N14 . . . N17, MMV1, MMV2 en FF2. De data van de floppy wordt eerst door N13 geïnverteerd. Aangezien de uitgang van N17 "1" is wordt de eerste clock-puls door N16 doorgegeven en kunnen de monostabiele multivibrators MMV1 en MMV2 getriggerd worden. MMV1 reageert op de neergaande flank van de clock-puls en MMV2 op de opgaande flank. De $\overline{Q}$-uitgang van MMV2 blijft ongeveer 5,5 mikrosekonde "0", waardoor N14 vrijgegeven en N16 geblokkeerd wordt. Als er een data-puls tussen twee clock-pulsen aanwezig is kan deze via N14 flipflop FF2 presetten. De Q-uitgang van MMV1 geeft een impuls van 1 μs aan de CRx-ingang van de ACIA. Bij de opgaande flank van deze puls wordt het databit in het seriële ingangsregister van IC11 geschoven. De databits worden geleverd door de $\overline{Q}$-uitgang van FF2. Bij een data-impuls op zijn preset-ingang maakt FF2 zijn $\overline{Q}$-uitgang namelijk "0". De volgende clock-puls zet deze nul dan in de ACIA. Als de tijd van MMV1 om is wordt FF2 gereset, zodat de flipflop klaar staat voor het ontvangen van het volgende databit. Het tijdvolgorde-diagram voor de "read data"-separator is getekend in figuur 11.

13



(ϕ11 V24/RS232- connector, Junior Interface Print)

Figuur 13. Bij het gebruik van een Epson-printer met seriële interface is een V24/RS232-naar-TTL-nivo-omzetter nodig. PB6 van de 6532 op de basis-print van de junior computer dient als busy-lijn voor de printer.

Bouw en afregeling

Het bouwen en afregelen van de floppy-interface is vrij eenvoudig. Eerst worden alle draadbruggen op de print gemonteerd. Aangezien sommige printsporen vrij dicht naast elkaar liggen moet het solderen met de nodige voorzichtigheid gebeuren. Na de draadbruggen kunnen dan de weerstanden, de kondensatoren, de diode D1 en de beide konnektoren op de print worden gesoldeerd. Daarna zijn de instelpotmeters aan de beurt; deze worden voorlopig in de middenstand gezet. Voor de IC's wordt het gebruik van een goede kwaliteit voetjes aanbevolen. Dat geldt vooral voor IC5 en IC11.

Nadat de voedingsspanning is ingeschakeld zal de floppy-interface gewoonlijk direkt werken als de twee instelpotjes in de middenstand staan. Is toch een afregeling nodig, dan dient deze als volgt te geschieden:

1. De steker van konnektor K2 verwijderen.
2. Daarna wordt de WDA-uitgang aan de koperzijde van de print door middel van een stukje draad doorverbonden met de RDA-ingang.
3. Uitgang $\overline{Q}$ van MMV2 wordt op een skoop aangesloten en daarna kan men P2 zo instellen dat de duur van de pulsen precies 5,5 μs is.
4. Als laatste wordt P1 zo verdraaid dat uitgang Q van MMV1 pulsen met een duur van 1 μs geeft. Deze laatste instelling is echter niet kritisch. Voor deze afregeling is trouwens een kort programma nodig voor de initialisatie van de ACIA en de PIA. Maar dat bewaren we voor de volgende keer, als de software voor de floppy-interface wordt besproken.

Epson-interface

In het Elektuur-lab wordt bij de junior computer een Epson-printer gebruikt. Dat is de reden waarom we hier een interface geven voor deze kombinatie.

De Epson-printer moet voor gebruik met de junior beschikken over een seriële interface-adapter en niet over de "normale" Centronics-konnektor. Zo'n seriële interface-adapter is te koop bij de computer-vakhandel tegen een redelijke prijs. De baudrate moet dan wel op de print worden ingesteld op een snelheid van 1200 baud. Ook de Elekterminal moet op 1200 baud "lopen". De Epson-printer wordt dan parallel op de V24/RS232-uitgang van de Elekterminal aangesloten.

Via de busy-lijn laat de Epson de computer weten of er data naar de printer kan worden gestuurd. Daar de cassette-sturing bij de DOS-computer toch niet meer nodig is hebben we PB6 van de 6532 op de basis-print van de junior computer gebruikt als busy-ingang. Relais Re2 op de junior-interface-kaart kan dan vervallen. De groene LED D5 kan verder dienst doen als transmit-indikator.

Op de busy-lijn wordt gebruik gemaakt van een V24/RS232-signaalnivo. Dit nivo moet weer worden omgezet naar TTL-nivo. Figuur 13 toont een kleine schakeling die nog wel een plaatsje op de junior-interface-print kan krijgen. Als men geen Epson-printer aansluit op de junior moet PB6 van de 6532 aan massa worden gelegd, anders kan de computer geen data verzenden.

Tenslotte willen we nog even opmerken dat de printer met de interface pas goed werkt in kombinatie met de nieuwe software. Die software voor de floppy-interface komt volgende maand aan bod!

G. de Cuijper

floppy-disk interface

voor junior en andere 6502-computers

software en aanpassingen

In dit tweede en tevens laatste deel over de floppy-disk-interface wordt beschreven wat er aan de hardware van de junior computer moet worden veranderd om de software van Ohio Scientific goed te laten lopen op deze computer (en andere 6502-computers). Daarbij is een nieuwe EPROM noodzakelijk om software van de diskette te kunnen laden bij de initialisatie (reset) van de computer. De source-listing van het monitorprogramma in de EPROM is verkrijgbaar als "paperware". Verder wordt in dit artikel de nodige aandacht besteed aan het werken met de nieuwe software.

deel 2

De hardware voor de junior-DOS

Om de junior computer te veranderen in een DOS-computer zijn enkele hardware-modifikaties nodig. Maar geen paniek! Daarvoor hoeft men geen printbanen door te krassen of andere mechanische ingrepen uit te voeren. Het enige wat men hoeft te doen is het solderen van een IC boven op één van de IC's van de interface-print van de junior computer. Verder is een interface voor de busy-lijn nodig bij het aansluiten van een Epson-matrixprinter. Aangezien deze interface slechts bestaat uit drie weerstanden, een transistor en een diode kan men dit schakelingetje gemakkelijk als "spin" naast de V 24/RS 232-konnektor bouwen.

Laten we nog eens kijken naar het schema van de interfaceprint van de junior computer (figuur 1). In het hier afgebeelde schema zijn de poorten N33 en N34 vervangen door een NAND-poort. Lijn $\overline{8K0}$ (of EX) is nu niet meer aktief in het adresbereik $ 0000 . . . 1FFF, maar in het nieuwe bereik E000 . . . FFFF. Deze adressering heeft voor de geheugens van de junior computer de volgende konsekwenties:
In het adresbereik $ 0000 . . . BFFF ligt 48 Kbyte dynamische RAM. Een dynamische RAM heeft het voordeel dat dit geheugen goedkoop is en niet veel stroom nodig heeft. Voor een totale bezetting van dit RAM-bereik zijn drie dynamische RAM-kaarten voldoende (zie Elektuur april 1982). De adresdekoder op de basis-print van de junior computer (IC6) dekodeert nu het adresbereik $ E000 . . . FFFF. De geheugens op de basis-print krijgen dan de volgende adressen toegewezen:

EPROM IC2, type 2708:
$ FC00 . . . FFFF
PIA, RAM, timer, type 6532:
$ FA00 . . . FBFF (paperware!)
RAM IC4 en IC5, type 2114:
$ E000 . . . E3FF
De geheugens op de interface-print van de junior hebben de volgende adressen:
VIA IC1, type 6522:
$ F800 . . . F9FF (paperware!)
RAM IC2 en IC3, type 2114:
$ E400 . . . E7FF
EPROM IC4 en IC5, type 2716:
$ E800 . . . F7FF

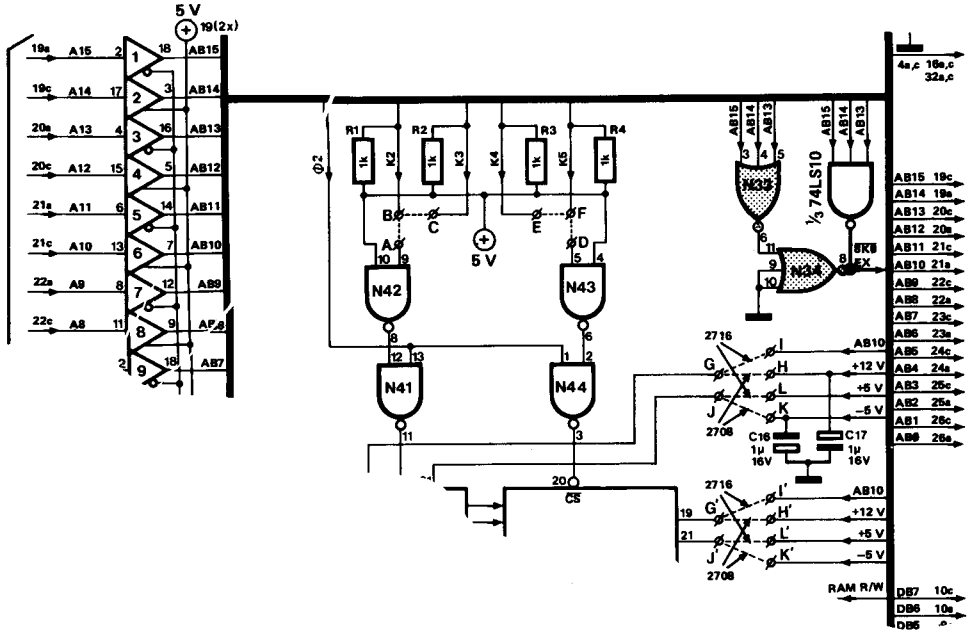
De tweede hardware-modifikatie bestaat uit de interface voor de busy-lijn bij een Epson-printer. Figuur 2 laat zien waar deze interface in de schakeling wordt opgenomen. Relais Re1 kan dan vervallen. De LED D4 werkt nu als busy-indikator; ze licht gelijktijdig met de busy-lamp van de Epson-printer op.

De tekeningen in figuur 3 en 4 laten zien hoe de hardware-modifikaties kunnen worden uitgevoerd. Nadat men deze werkzaamheden heeft verricht moet nog een 2708-EPROM met de juiste inhoud (ESS 515) in het betreffende voetje op de basis-print van de junior worden gestoken. De twee EPROM's IC4 en IC5 op de interface-print (PM en TM) zijn nu niet meer nodig, aangezien de input/output-programma's voor de printer-sturing in de 2708 zijn ondergebracht. De adresruimte die eerst voor IC4 en IC5 was gereserveerd kan nu worden gebruikt voor user-programma's. Tenslotte nog een punt waar men goed op moet letten: tussen de soldeereilandjes "R" en "S" op de junior-interface-print moet een draadbrugje worden gelegd (WITH)!
De junior computer is nu omgebouwd tot een DOS-computer. Nu moet men minstens twee en liever drie dynamische RAM-kaarten op de bus-print van de junior computer steken. De verbindingen voor de adresdekodering op de RAM-kaarten worden als volgt gelegd:

RAM-kaart 1: U – 0
V – 1
X – 2
Y – 3
RAM-kaart 2: U – 4
V – 5
X – 6
Y – 7
RAM-kaart 3: U – 8
V – 9
X – A
Y – B

Steek de floppy-disk-interface nog niet op de bus-print van de junior! Eerst wordt de spanning voor het computersysteem ingeschakeld en daarna drukt men op de RST-toets van het hexkeyboard. Het display moet dan oplichten. De kommandotoetsen AD, DA, + en GO hebben dezelfde funktie als vroeger, alleen de PC-toets heeft nu een andere funktie gekregen. Als het disk-operating-system namelijk in de computer is geladen kan men met de PC-toets de DOS-command-interpreter oproepen. Maar daarover straks meer.

1



Figuur 1. Een gedeelte van het schema van de interface-kaart voor de junior computer. De poorten N33 en N34 worden vervangen door een NAND-poort uit een 74LS10. Daarmee wordt het geheugen van de standaard junior computer en de interface-kaart op een ander adresbereik gelegd.

Tabel 1 toont de geheugenindeling bij de DOS-junior. Deze indeling geldt ook voor alle andere 6502-computers waarop de floppy-interface wordt aangesloten. Daarbij kan de indeling van de adressen $ C000 . . . FBFF bij elke computer anders zijn. Maar dat maakt niets uit. Belangrijk is alleen dat de computer in het onderste adresbereik minstens 32 Kbyte RAM aan één stuk kan adresseren. Als het adresbereik FC00 . . . FFFF bij een niet-juniorcomputer al voor iets anders wordt gebruikt moet de zogenaamde bootstrapsoftware op een andere plaats in het geheugen worden gezet. Met behulp van de "paperware" zal dat wel geen problemen geven.

De software van de DOS-junior

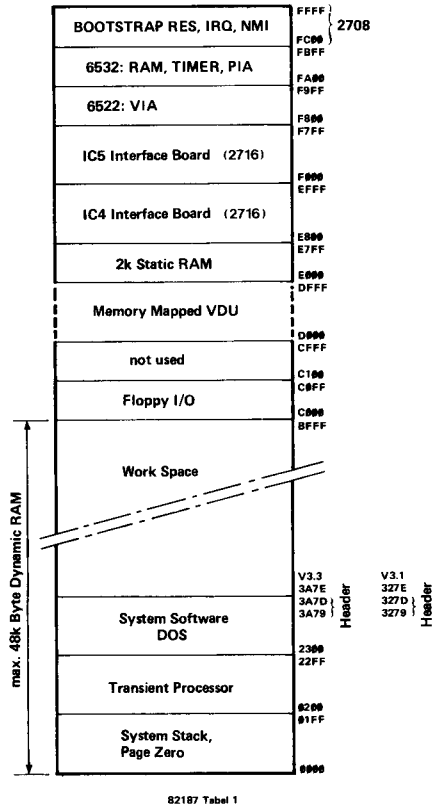
De software voor de DOS van de junior computer werkt volgens de modernste computermaatstaven. Dat betekent dat de computer wordt uitgerust met een minimum aan ROM-intelligentie en zoveel mogelijk RAM krijgt. De voordelen van deze opzet liggen voor de hand:

Aangezien bij gebruik van een diskette de systeem-software (BASIC, FORTH, assembler of een woordprocessor) heel snel in de computer kan worden geladen is er ook niet veel adresruimte voor de ROM nodig. De ROM hoeft slechts zo veel intelligentie te bevatten dat de computer het hex-display en het hex-toetsenbord kan bedienen en kan zenden/ontvangen naar/van de Elekterminal. Verder zorgt een gedeelte van de ROM-intelligentie er voor dat track 0 van de diskette in de junior computer wordt geladen.

BASIC, assembler, enz. zijn dus niet meer in ROM opgeslagen, maar worden van de floppy in de computer geladen. In zo'n geval spreekt men van "portable software". Deze software heeft het voordeel dat ze heel eenvoudig kan worden veranderd. Was er vroeger een nieuwe set EPROM's nodig als men software-fouten ontdekt had of iets wilde aanpassen aan de nieuwste gegevens, nu hoeft er alleen maar een nieuwe diskette in de drive te worden gestopt. Van die mogelijkheid om eenvoudig iets te kunnen veranderen in de software maken we bij de DOS-junior dankbaar gebruik. Kijk maar eens naar tabel 1. Pagina Ø en de stack van het systeem liggen in het adresbereik $ ØØØØ . . . Ø1FF. Het bereik $ Ø2ØØ . . . . 22FF is gereserveerd voor de "transient processor". Met die moeilijke term wordt de software bedoeld die de computer in staat stelt te werken. Werkt men bijvoorbeeld met BASIC, dan is de BASIC-interpreter de transient processor. Als men in machinetaal werkt is de assembler of de extended monitor van Ohio Scientific de transient processor.

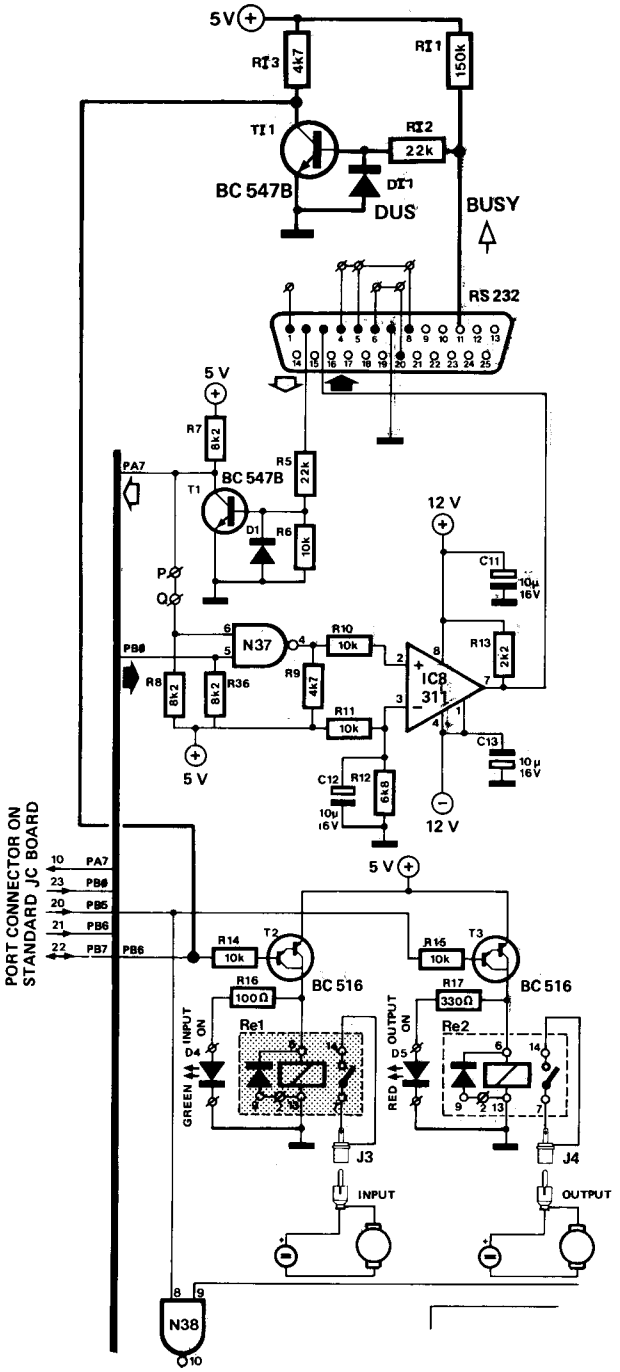
De systeem-software voor het sturen van de floppy-drive, voor het sturen van de printer en voor een "memory mapped video display unit" heeft een omvang van ongeveer 4 Kbyte RAM. Bij de versie OS-65D V3.1 zit de DOS-software op de geheugenplaatsen $ 23ØØ . . . . 3278. Op adres 327E begint de "work space", het gedeelte van het geheugen waarin het programma van de gebruiker wordt opgeslagen. Als men een BASIC-programma intikt, dan wordt dit vanaf adres 327E in het geheugen gezet. Vanaf dit adres wordt de data dan weer op de floppy geschreven. Data die van de diskette naar de computer wordt gestuurd wordt ook weer vanaf adres 327E opgeslagen.

2



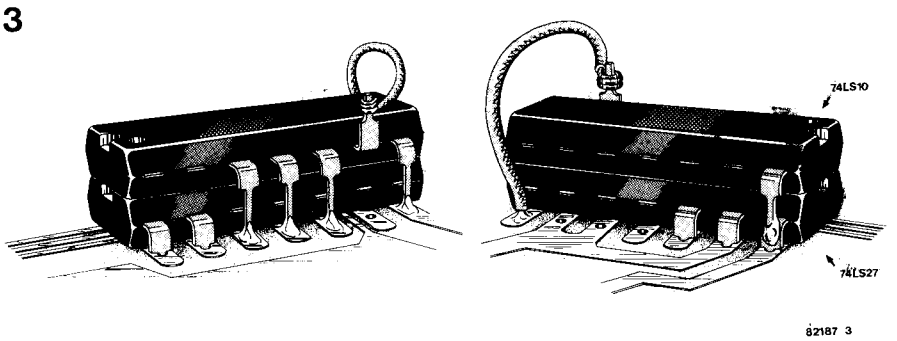
Tabel 1. De nieuwe geheugen-indeling voor de DOS-junior. Ten gevolge van de uitbreidingen is de work space bij OS-65D V3.3 2 Kbyte korter. De transient processor overschrijft het adresbereik $ 22ØØ . . . 22FF van de DOS-software.

2



Figuur 2. Deze figuur toont op welke punten de interface voor een Epson-printer moet worden aangesloten. Als PB6 logisch één is wordt de zend-uitgang van de computer (pen 3 van de RS 232-konnektor) geblokkeerd. Als er geen printer is aangesloten zorgt weerstand RI1 dat de zend-uitgang niet geblokkeerd wordt.

3



Figuur 3. Door deze hardware-modifikatie krijgt de junior computer een nieuwe geheugenindeling. Op de tekening zijn de boven op elkaar gesoldeerde IC's duidelijk zichtbaar.

Bij de versie OS-65D V3.3 begint de work space bij adres 3A7E. De eerste vijf bytes van de work space worden "header" genoemd. Die vijf bytes bevatten de volgende informatie:

1. startadres van de file (2 bytes)
2. eindadres van de file (2 bytes)
3. lengte van de file in pagina's (1 byte, 1 pagina = 256 bytes)

Na de work space volgt de 4 Kbyte-ruimte $ DXXX, die de "memory mapped video display unit" van Ohio Scientific bevat. Zo'n video unit onderscheidt zich van de Elekterminal door het feit dat de computer nu direkt via de databus data naar de monitor kan sturen. De data-uitwisseling tussen scherm en computer gaat bij een "memory mapped video display unit", afgekort VDU, veel sneller dan bij een video-interface met V 24/RS 232-aansluiting.

In het adresbereik $ E000 . . . E7FF ligt 2 Kbyte statische RAM, die vroeger in het bereik $ 0000 . . . 07FF lag. Dit RAM-bereik gebruiken we in de toekomst voor de object code die door de assembler wordt geproduceerd. Men kan nu een source file met de assembler samenstellen, naar wens een listing maken en de door de assembler geproduceerde machine-kode direkt door de computer laten uitvoeren. Meer informatie hierover is te vinden in het assembler manual van Ohio Scientific.

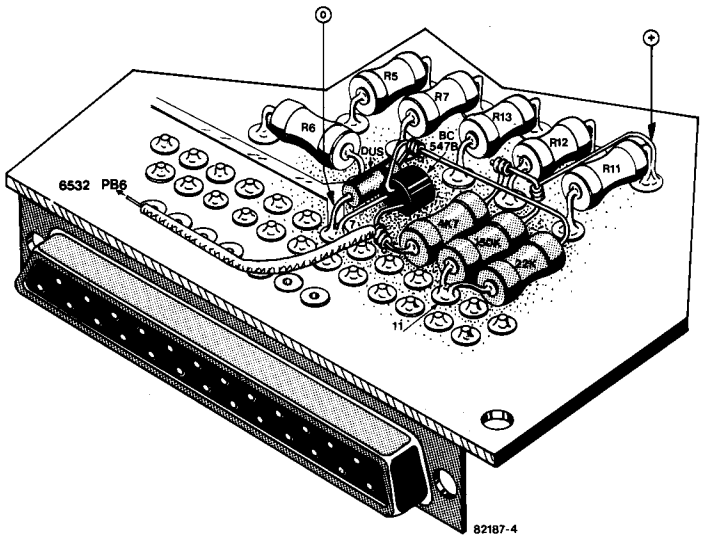
Het adresbereik $ E800 . . . FFFF omvat IC4 en IC5 van de interface-print van de junior, de twee interface-IC's 6522 en 6532 en de bootstrap-EPROM 2708. De twee EPROM's IC4 en IC5 op de interface-print zijn nu vrij voor gebruik door de programmeur. Hierin kan men bijvoorbeeld programma's zetten die erg veel gebruikt worden.

De bootstrap-EPROM (ESS 515)

De bootstrap-EPROM is ondergebracht in het adresbereik $ FC00 . . . FFFF. Het IC bevat slechts 1 Kbyte software. Deze software kan in acht stukken verdeeld worden:

1. Hex-display-monitor. Dit programma lijkt erg veel op de oorspronkelijke monitor van de junior. De kommando's AD, DA, GO en + zijn hetzelfde gebleven, alleen de PC-toets heeft een andere funktie gekregen. Met de PC-toets kan men de DOS-command-interpreter direkt vanaf het hex-keyboard oproepen. De hex-display-monitor wordt in de eerste plaats gebruikt om de software op de diskettes van Ohio Scientific te kunnen modificeren. Zo kan Ohio-Software worden veranderd in junior-software. Met de hex-display-monitor kan men belangrijke startadressen oproepen en starten met de GO-toets:
   - RESBAS* $ FF17
   - RESDOS* $ FF34
   - VONE* $ FFE2
   - VTHREE* $ FFE8
2. Laden van de BASIC-interpreter van de diskette. Met het kommando <AD> FF17 <GO> (RUBOUT) kan

4

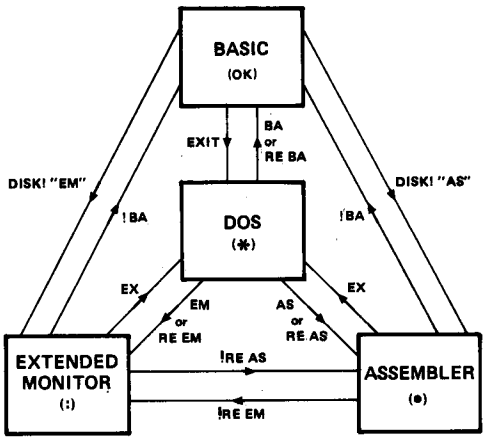


Figuur 4. Op deze manier wordt de Epson-interface op de interface-kaart van de junior gemonteerd.

men de BASIC van de diskette in de junior computer laden. Om de kommandotoetsen te kunnen onderscheiden van de toetsen van de Elekterminal zullen we voortaan de toetsen van het hex-keyboard tussen de tekens <> plaatsen en de toetsen van de Elekterminal tussen de tekens ( ). Als de BASIC is geladen en de computer zich op de terminal "meldt" is de transient processor klaar voor programma-onderbrekingen. Een BASIC-programma kan worden onderbroken door een druk op de (BREAK)-toets van de terminal. Als de (BREAK)-toets tijdens een LIST-kommando wordt bediend geeft de BASIC-interpreter "Break". Heeft men een BASIC-programma gestart met het RUN-kommando en wil men de uitvoer stoppen, dan kan dat ook met de (BREAK)-toets. De BASIC-interpreter antwoordt dan met "BREAK IN LINE X". Alle programma-variabelen en de pointer worden nu "gesaved" in de stack. Met behulp van het CONT-kommando kan men het programma daarna weer verder laten lopen. De indirekte sprongvektor voor de (BREAK)-toets wordt bij het starten van het BASIC-programma automatisch door de computer via adres $ FF17 geset. De break-vektor is opgeslagen in adres $ FA7C en $ FA7D.

3. Laden van DOS-software van de diskette. Met het kommando <AD> FF34 <GO> (RUBOUT) kan men voortaan eigen software van Elektuur in de junior computer laden. Dit adres is bedoeld voor toekomstige uitbreidingen en "niet-Ohio"-software.
4. Aanpassen van een Ohio-diskette OS-65D V3.1. Een Ohio-Scientific-diskette OS-65D V3.1 kan men aan de junior computer aanpassen met het kommando <AD> FFE2 <GO> (RUBOUT). Als de monitor wordt gestart op adres VONE* $ FFE2 gebeurt het volgende:
   – De computer leest de data van track Ø en zet die data vanaf adres $ 2ØØ in het geheugen.
   – De computer zet de lees/schrijf-kop van de drive op track 1.
   – De computer leest de data van track 1 en zet deze data vanaf adres $ 2AØØ in het geheugen. Het hele disk operating system is nu in de computer geladen en kan door de programmeur door middel van het hex-keyboard gemodificeerd worden.
   – Als track Ø en track 1 in de computer zijn geladen volgt een sprong naar de hex-display-monitor. De computer meldt zich dan terug op de printer met *Track 0 & 1*.
5. Aanpassing van een Ohio-diskette OS-65D V3.3 – tutorial disk 5. Een Ohio Scientific diskette OS-65D V3.3 – tutorial disk 5 kan men met het kommando <AD> FFE8 <GO> (RUBOUT) aanpassen aan de junior computer. Voor track Ø en track 1 geldt hetzelfde als bij 4. Nadat deze twee tracks in de computer zijn geladen gebeurt echter het volgende:
   – De computer zet de lees/schrijf-kop van de drive op track 6. De data van sektor 2 op track 6 wordt vanaf adres $ 32ØØ in het geheugen gezet. Sektor 2 is één pagina lang.
   – De data van sektor 3 op track 6 wordt vanaf adres $ ØØØØ in het geheugen opgeborgen. Ook deze sektor is één pagina lang.
   – De computer stuurt de lees/schrijf-kop van de drive naar track 13. De data van sektor 1 op track 13 wordt vanaf adres $ 3A79 in de computer geladen. Sektor 1 van track 13 is 8 pagina's lang. Nadat alle vermelde tracks in de computer zijn geladen verschijnt evenals bij OS-65D V3.1 de melding *Track 0 & 1* en volgt een sprong naar de hex-display-monitor.
6. De <PC>-toets. Met de <PC>-toets van het hex-keyboard kan men de hex-display-monitor verlaten en naar de DOS-command-interpreter springen. Daarbij wordt de printer-I/O geïnitiali-

5



Figuur 5. De kommando's voor de transfer tussen de verschillende transient processors: van BASIC naar assembler, van assembler naar extended monitor, enz., en omgekeerd. De "schakelcentrale" is de DOS. Hiermee kan men de transient processors BASIC, assembler en extended monitor van de diskette in de computer laden.

seerd, maar er wordt geen nieuwe bittijdmeting uitgevoerd. De computer meldt zich terug met de prompt A* of B*, enz.

7. De printer-input-routine. Deze routine zorgt voor de ontvangst van een karakter van de terminal. Na de terugkeer uit de input-subroutine staat het ontvangen ASCII-karakter in de accu van de CPU. Bit 7 van het karakter is altijd logisch nul. De inhoud van het indexregister wordt door het aanspreken van deze subroutine niet beïnvloed. Het startadres van de printer-input-routine is: RECCHA * $ FE1B
(oproep: JSR RECCHA).
8. De printer-output-routine. Deze routine zendt het karakter in de accu van de CPU naar de terminal. Het overdrachtformaat is:
— een startbit
— zeven databits
— geen pariteitsbit (no parity)
— twee stopbits
De inhoud van het indexregister wordt ook door het oproepen van deze subroutine niet beïnvloed. Het startadres van de printer-output-routine is: PRCHA * $ FEA3
(oproep: JSR PRCHA).

Hoe werkt de bootstrap?

Nadat we hebben bekeken uit welke delen de software van de bootstrap-EPROM bestaat gaan we nu verklaren hoe de bootstrap werkt. Nadat men

<RST>
<AD> FF17
<GO>

op de computer heeft ingegeven gebeuren de volgende dingen:

1. De computer zet de indirekte sprongvektor voor de (BREAK)-toets op de terminal. De NMI-vektor wijst naar de hex-display-monitor.
2. De computer initialiseert de I/O op de floppy-interface-print (6821 en 6850) en de seriële I/O voor de terminal (6532). Nadat dit gebeurd is wacht de computer op het RUBOUT-karakter van de terminal, voor het meten van de bittijd. De bittijd wordt in de geheugenplaatsen $ FA5A . . . FA5D opgeslagen (zie floppy disk paperware).
3. Nadat de bittijd (baud rate) van de terminal is gemeten laadt de computer track Ø (2 Kbyte software in machinetaal). Dat laden van track Ø gebeurt als volgt:

— De lees/schrijf-kop van de drive wordt boven track Ø geplaatst. Een sensor in de drive deelt de computer mee dat de kop boven track Ø staat.
— Dan zendt de computer een head-load-impuls naar de drive, met als gevolg dat de kop wordt neergelaten op de oppervlakte van de diskette. De computer wacht daarna op een impuls van de sensor bij het index-gat.
— Na het verstrijken van de index-impuls wordt het control register in de ACIA geset. Het overdrachtformaat van de ACIA is: een startbit, acht databits, een even-pariteitsbit en een stopbit.
— De computer leest het eerste byte van de diskette. Dit byte is het hoge startadres van het geheugenbereik waarin track Ø moet worden opgeslagen ($ 22). Het tweede byte van de diskette is het lage byte van het startadres ($ ØØ). Beide bytes worden in de bump-pointer (laad-pointer) geladen. De bump-pointer wijst dus naar adres $ 22ØØ. Daarna leest de computer het derde byte van de diskette. Dit geeft informatie over het aantal pagina's dat track Ø lang is ($ Ø8).
— Nu volgen 2 Kbytes software in machinetaal die van de diskette in de computer worden geladen. Het geheugenbereik is $ 22ØØ . . . 29FF.
— Nadat track Ø in het geheugen is gezet wordt de lees/schrijf-kop weer van het diskette-oppervlak afgehaald en volgt een sprong naar adres $ 22ØØ. Vanaf dit adres vindt de computer verdere instrukties, namelijk welke tracks en sektoren op welke adressen

2

```
<RST>
<AD>   FFE2
<GO>

(RUBOUT)

*TRACK Ø&1*

<AD>   2217
<DA>   4C
<+>    40
<+>    22

<AD>   2245
<DA>   4C
<+>    76
<+>    22

<AD>   2283
<DA>   4C
<+>    A6
<+>    22

<PC>

A*CA Ø2ØØ=13,1
```

Tabel 2. Het aanpassen van OS-65D V3.1 aan de junior computer (eerste gedeelte), bij gebruik van een enkele drive.

3

```
A*IN
ARE YOU SURE?Y
A*GO Ø2ØØ

- DISKETTE UTILITIES -

SELECT ONE:
1) COPIER
2) TRACK Ø READ/WRITE
? 2

- TRACK ZERO READ/WRITE UTILITY -

COMMANDS:
Rnnnn - READ INTO LOCATION nnnn.
Wnnnn/gggg,p - WRITE FROM nnnn FOR p PAGES
               WITH gggg AS THE LOAD VECTOR
E - EXIT TO OS-65D

COMMAND? W22ØØ/22ØØ,8

- TRACK ZERO READ/WRITE UTILITY -

COMMANDS:
Rnnnn - READ INTO LOCATION nnnn.
Wnnnn/gggg,p - WRITE FROM nnnn FOR p PAGES
               WITH gggg AS THE LOAD VECTOR
E - EXIT TO OS-65D

COMMAND? E

A*HOME

A*SA Ø1,1=2AØØ/8
```

Tabel 3. Het aanpassen van een OS-65D V3.1-diskette (kopie) aan de junior computer, tweede gedeelte.

4

```
A*CA 4ØØØ=Ø2,1
A*CA 48ØØ=Ø3,1
A*CA 5ØØØ=Ø4,1
A*CA 58ØØ=Ø5,1
A*CA 6ØØØ=Ø6,1
A*CA 68ØØ=Ø7,1
A*CA 7ØØØ=Ø8,1
A*CA 78ØØ=Ø9,1
A*SA Ø2,1=4ØØØ/8
A*SA Ø3,1=48ØØ/8
A*SA Ø4,1=5ØØØ/8
A*SA Ø5,1=58ØØ/8
A*SA Ø6,1=6ØØØ/1
A*SA Ø7,1=68ØØ/8
A*SA Ø8,1=7ØØØ/8
A*SA Ø9,1=78ØØ/5
```

```
A*CA 4ØØØ=1Ø,1
A*CA 48ØØ=11,1
A*CA 5ØØØ=12,1
A*CA 51ØØ=12,2
A*CA 52ØØ=12,3
A*CA 53ØØ=12,4
A*CA 58ØØ=13,1
A*CA 6ØØØ=14,1
A*CA 68ØØ=15,1
A*CA 7ØØØ=16,1
A*CA 78ØØ=17,1
A*SA 1Ø,1=4ØØØ/8
A*SA 11,1=48ØØ/8
A*SA 12,1=5ØØØ/1
A*SA 12,2=51ØØ/1
A*SA 12,3=52ØØ/1
A*SA 12,4=53ØØ/1
A*SA 13,1=58ØØ/5
A*SA 14,1=6ØØØ/8
A*SA 15,1=68ØØ/8
A*SA 16,1=7ØØØ/8
A*SA 17,1=78ØØ/8
```

```
A*CA 4ØØØ=18,1
A*CA 48ØØ=19,1
A*CA 5ØØØ=2Ø,1
A*CA 58ØØ=21,1
A*CA 6ØØØ=22,1
A*CA 68ØØ=23,1
A*CA 7ØØØ=24,1
A*CA 78ØØ=25,1
A*SA 18,1=4ØØØ/8
A*SA 19,1=48ØØ/8
A*SA 2Ø,1=5ØØØ/8
A*SA 21,1=58ØØ/8
A*SA 22,1=6ØØØ/8
A*SA 23,1=68ØØ/8
A*SA 24,1=7ØØØ/8
A*SA 25,1=78ØØ/8
```

```
A*CA 4ØØØ=26,1
A*CA 48ØØ=27,1
A*CA 5ØØØ=28,1
A*CA 58ØØ=29,1
A*CA 6ØØØ=3Ø,1
A*CA 68ØØ=31,1
A*CA 7ØØØ=32,1
A*SA 26,1=4ØØØ/8
A*SA 27,1=48ØØ/8
A*SA 28,1=5ØØØ/8
A*SA 29,1=58ØØ/8
A*SA 3Ø,1=6ØØØ/8
A*SA 31,1=68ØØ/8
A*SA 32,1=7ØØØ/8
A*
```

```
<RST>
<AD>   FF17
<GO>

(RUBOUT)

OS-65D V3.Ø
OSI 9 DIGIT BASIC
COPYRIGHT 1977 BY MICROSOFT
 36225 BYTES FREE
```

Tabel 4. De aanpassing van OS-65D V3.1 aan de junior computer, derde gedeelte.

5

```
RUN"BEXEC*"

 BASIC EXECUTIVE FOR
 OS-65D V3.1

 JUNE 25, 1980 RELEASE

 FUNCTIONS AVAILABLE:

   CHANGE- ALTER WORK-
     SPACE LIMITS

   DIR- PRINT DIRECTORY

   UNLOCK- UNLOCK SYSTEM
     FOR END USER MODI-
     FICATIONS

 FUNCTION? UNLOCK

 SYSTEM OPEN

 OK
 RUN

 BASIC EXECUTIVE FOR
 OS-65D V3.1

 JUNE 25, 1980 RELEASE

 FUNCTIONS AVAILABLE:

   CHANGE- ALTER WORK-
     SPACE LIMITS

   DIR- PRINT DIRECTORY

   UNLOCK- UNLOCK SYSTEM
     FOR END USER MODI-
     FICATIONS

 FUNCTION? DIR
 LIST ON LINEPRINTER INSTEAD OF DEVICE # 1 ? NO

 OS-65D VERSION 3.0
  -- DIRECTORY --

 FILE NAME     TRACK RANGE
 ------------------------
 OS65D3        0  - 12
 BEXEC*        14 - 14
 CHANGE        15 - 16
 CREATE        17 - 19
 DELETE        20 - 20
 DIR           21 - 21
 DIRSRT        22 - 22
 RANLST        23 - 24
 RENAME        25 - 25
 SECDIR        26 - 26
 SEQLST        27 - 28
 TRACE         29 - 29
 ZERO          30 - 31
 ASAMPL        32 - 32

  50 ENTRIES FREE OUT OF 64

 OK
```

Tabel 5. Proefdraaien met een aangepaste OS-65D V3.1-diskette. Het utility-programma BEXEC* wordt opgeroepen na het laden van de BASIC-interpreter.

6

```
OK
LIST

  10 REM DIRECTORY UTILITY FOR OS-65D VERSION 3.0
  20 REM
  30 NF=0
  40 PN=11897
  50 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
BREAK
OK

NEW

OK
RUN"DIR"
LIST ON LINEPRINTER INSTEAD OF DEVICE # 1 ? NO

OS-65D VERSION 3.0
  -- DIRECTORY --

FILE NAME     TRACK RANGE
------------------------
OS65D3        0  - 12
BEXEC*        14 - 14

BREAK IN  11110
OK
CONT
CHANGE        15 - 16
CREATE        17 - 19
DELETE        20 - 20
DIR           21 -
BREAK IN  11100
OK
```

Tabel 6. Het uitproberen van de (BREAK)-toets bij een LIST-kommando en bij een RUN-statement.

7

```
                                                  02
                                                  03
                                                  04
                                                  05
                                                  06
A*CA 0200=13,1                                    07
                                                  08
A*GO 0200                                         09
                                                  10
                                                  11
                                                  12
- DISKETTE UTILITIES -                            13
                                                  14
SELECT ONE:                                       15
1) COPIER                                         16
2) TRACK 0 READ/WRITE                             17
? 1                                               18
                                                  19
                                                  20
- DISKETTE COPIER -                               21
                                                  22
FROM DRIVE (A/B/C/D)? A                           23
                                                  24
TO DRIVE (A/B/C/D)? B                             25
                                                  26
                                                  27
STARTING TRACK? 2                                 28
                                                  29
ENDING TRACK (INCLUSIVE)? 32                      30
                                                  31
READY (Y/N)? Y                                    32

                                ANOTHER (Y/N)? N

                                A*
```

Tabel 7. Bij gebruik van twee floppy-drives (drive A en drive B) kan men een OS-65D V3.1-diskette heel eenvoudig aanpassen. Het kopiëren wordt door de computer automatisch uitgevoerd.

geladen moeten worden. Gewoonlijk wordt de 2 Kbyte software van track 1 op de adressen $ 2A00 . . . 31FF gezet. Track 0 en track 1 bevatten samen 4 Kbyte software in machinetaal, die het hele disk operating system operationeel maakt.

— Na het disk operating system (DOS) wordt de BASIC-interpreter van de diskette in het geheugen geladen. De BASIC-interpreter staat bij OS-65D V3.1 op track 2 . . . 4. Bij OS-65D V3.3 staan de BASIC-interpreter en diverse uitbreidingen voor de editor op track 2, 3, 4, 6 en 13. Na het laden van de interpreter volgt een sprong naar de cold-start-entry van de BASIC-interpreter ($ 20E4). Het systeem meldt zich dan met de prompt "OK".

— Het systeem is nu nog niet klaar voor de statements LIST, CONT, etcetera. Ook een BASIC-file kan nog niet gemaakt worden. Het enige BASIC-statement dat de computer nu kent is RUN. Als men een BASIC-file wil maken moet eerst nog het programma BEXEC* in de computer worden geladen met het kommando:
RUN "BEXEC*".
De computer biedt dan een heel stel opties. Bij de versie OS-65D V3.1 moet men dan de optie "UNLOCK" kiezen en bij OS-65D V3.3 de optie "9". De computer meldt zich dan met de prompt "SYSTEM OPEN". Als men daarna een "NEW"-statement geeft is de work space van de junior geformatteerd voor de BASIC-file.

De DOS en zijn mogelijkheden

Met het disk operating system van Ohio, dat op track 0 en 1 van de diskette staat, kan men de junior computer eenvoudig veranderen in diverse transient processors. Figuur 5 toont hoe de verschillende verbindingen tussen de DOS en de verschillende processors kunnen worden gemaakt en verbroken. Als de DOS bijvoorbeeld wordt gestart op adres $ FF17 verandert de junior in een BASIC-computer. Wil men de junior daarna veranderen in een assembler-computer, dan moet men eerst EXIT intypen om de BASIC-interpreter te verlaten. De junior meldt zich dan terug op de printer met de DOS-prompt A* of B*, enz. Als men daarna AS of ASSEMBLER intypt verandert de junior in een assembler-computer. Nu kan men een assembler-file maken en met de DOS op de diskette schrijven. Ook kan men een source file samenstellen en de object code met de EPROMmer (Elektuur januari 1982) direkt in een EPROM programmeren, zonder dat men een enkel byte zelf hoeft in te tikken. De computer doet dat allemaal automatisch. Verdere gegevens over het werken met de assembler zijn te vinden in het assembler manual van Ohio Scientific.

Opgelet! Bij de versie OS-65D V3.1 staan de assembler en de uitgebreide monitor niet op de diskette. Alleen bij OS-65D V3.3 staan de assembler en de uitgebreide (extended) monitor (EM) standaard op de diskette.

Aanpassingen voor één drive bij OS-65D V3.1

Als men met één enkele drive werkt kost het nogal wat werk om OS-65D V3.1 op de junior computer aan te passen. De tabellen 2, 3 en 4 laten zien wat er dan moet gebeuren. Eerst starten we het systeem door middel van het hex-keyboard. Op adres $ FFE2 begint een programma waarmee een V3.1-diskette "gemakkelijk te modificeren" in de junior kan worden geladen. Na het indrukken van de (RUBOUT)-toets laadt de computer track 0 en track 1 in het geheugen en meldt hij zich met *TRACK 0 & 1*. Het display licht op, de vektoren voor de input- en output-routines worden automatisch geladen en de programmeur kan nu op de adressen $ 2217, 2245 en 2283 enkele bytes veranderen. Nadat deze bytes volgens tabel 2 veranderd zijn volgt een sprong naar de DOS-command-interpreter door middel van de <PC>-toets. De DOS meldt zich dan met de prompt A.

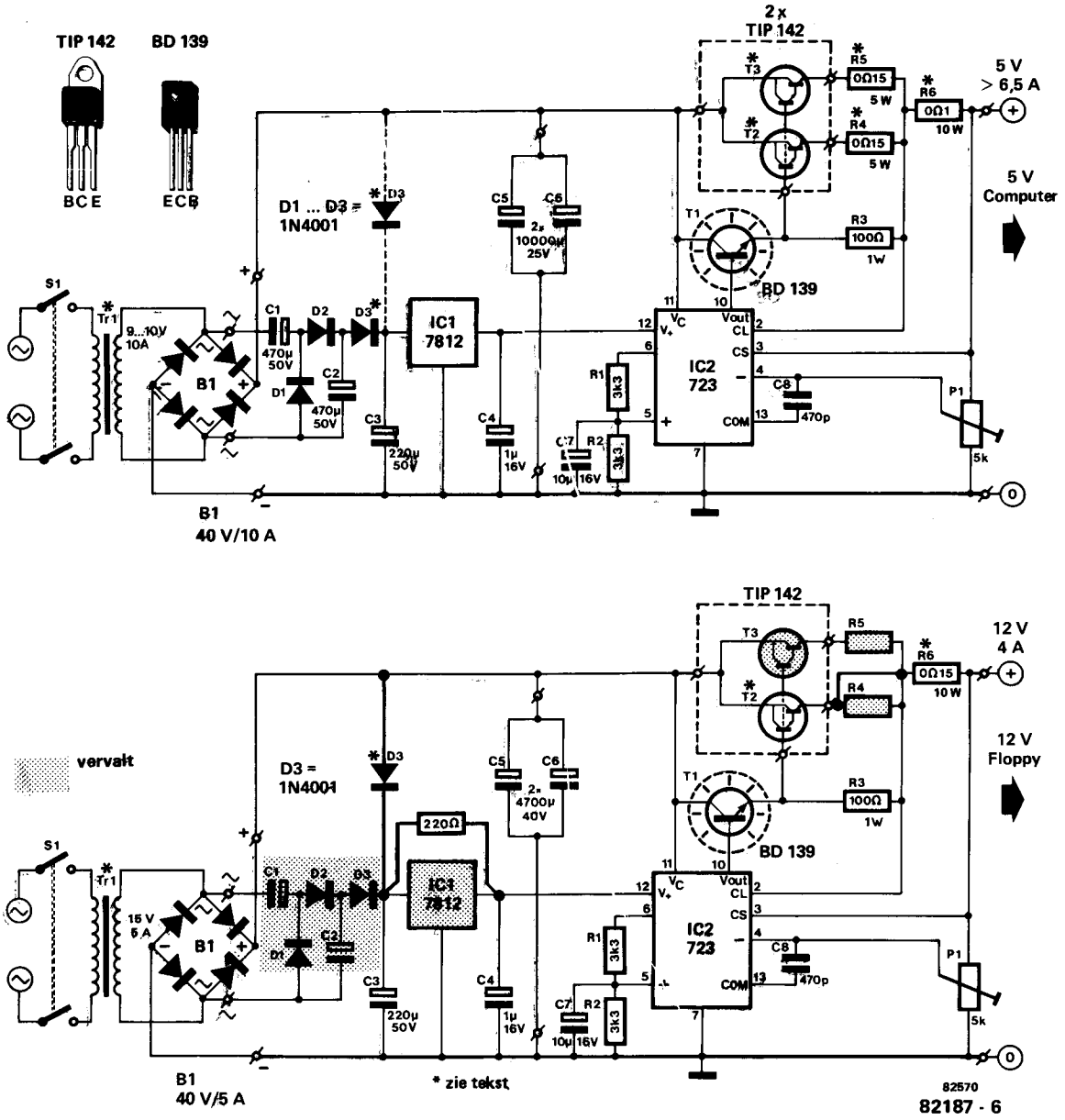
Het kommando CA 0200 = 13,1 laadt sektor 1 van track 13 van de diskette. De data wordt vanaf adres 0200 in de computer opgeslagen. Het programma dat nu in de junior is geladen bevat een diskette-kopieerprogramma en een track-0-read/write-utility-programma. Met dat laatste kan men de gemodificeerde DOS op track 0 van zijn eigen diskette schrijven. Maar eerst moet de diskette nog geïnitialiseerd worden. De Ohio-diskette wordt uit de drive gehaald en

8

Tabel 8. Het aanpassen van een OS-65D V3.3-diskette. Via een sprong naar de DOS-command-interpreter (GO 2276) wordt BASIC geladen en daarna wordt gekopieerd met behulp van het utility-programma BEXEC*.

```
<RST>
<AD>   FFE8
<GO>

(RUBOUT)

*TRACK 0&1*

<PC>

A*GO 2276

OS-65D V3.0
OSI 9 DIGIT BASIC
COPYRIGHT 1977 BY MICROSOFT
 33921 BYTES FREE

Ok
RUN"BEXEC*"

OS-65D  Tutorial disk five - Sept. 16, 1981
 1 > Directory
 2 > Create a new file
 3 > Change a file name
 4 > Delete file from diskette
 5 > Create blank data diskette
 6 > Create data diskette with files
 7 > Create buffer space for data files
 8 > Single or dual disk drive copier
 9 > Enter OS-65D system
Type the number of your selection
and depress RETURN ? 8

- Diskette copier -

Copy from which drive (A/B/C/D) ? A

Copy to which drive (A/B/C/D) ? A

What is the last track to be copied (Inclusive) <0-39> ? 39

Are you ready to start copying (Y/N) ? Y

Insert master diskette -- press <RETURN> ?

Reading --

Insert blank diskette -- press <RETURN> ?

Initializing --

Track 01 - 01/08
Track 02 - 01/08
Track 03 - 01/08
Track 04 - 01/08
Track 05 - 01/08
Track 06 - 01/01 - 02/01 - 03/01 - 04/02
Track 07 - 01/08
Track 08 - 01/08
Track 09 - 01/08
Track 10 - 01/08
Track 11 - 01/01 - 02/01 - 03/01 - 04/01 - 05/01 - 06/01 - 07/01
Track 12 - 01/01 - 02/01 - 03/01 - 04/01
Track 13 - 01/08
Track 14 - 01/08

Insert master diskette -- press <RETURN> ?

Reading --

Insert blank diskette -- press <RETURN> ?

Track 15 - 01/08
Track 16 - 01/08
Track 17 - 01/08
Track 18 - 01/08
Track 19 - 01/08
Track 20 - 01/08
Track 21 - 01/08
Track 22 - 01/08
Track 23 - 01/08
Track 24 - 01/08
Track 25 - 01/08
Track 26 - 01/08
Track 27 - 01/08
Track 28 - 01/08

Insert master diskette -- press <RETURN> ?

Reading --

Insert blank diskette -- press <RETURN> ?

Track 29 - 01/08
Track 30 - 01/08
Track 31 - 01/08
Track 32 - 01/08
Track 33 - 01/08
Track 34 - 01/08
Track 35 - 01/08
Track 36 - 01/08
Track 37 - 01/08
Track 38 - 01/08
Track 39 - 01/05 - 02/02

Insert master diskette -- press <RETURN> ?

Please, put the tutorial disk in drive A and depress <RETURN>.
<RST>
<AD>   FFE8
<GO>
(RUBOUT)

*TRACK 0&1*


<AD>   2217
<DA>   4C
<+>    40
<+>    22

<AD>   2245
<DA>   4C
<+>    76
<+>    22

<AD>   2285
<DA>   8E
<+>    C6
<+>    2A
<+>    4C
<+>    B3
<+>    22

<AD>   2E84
<DA>   4C
<+>    B0
<+>    2E

<PC>


A*CA 0200=06,4

A*GO 0200


- TRACK ZERO READ/WRITE UTILITY -

COMMANDS:
Rnnnn - READ INTO LOCATION nnnn.
Wnnnn/gggg,p - WRITE FROM nnnn FOR p PAGES
               WITH gggg AS THE LOAD VECTOR
E - EXIT TO OS-65D

COMMAND? W2200/2200,8

- TRACK ZERO READ/WRITE UTILITY -

COMMANDS:
Rnnnn - READ INTO LOCATION nnnn.
Wnnnn/gggg,p - WRITE FROM nnnn FOR p PAGES
               WITH gggg AS THE LOAD VECTOR
E - EXIT TO OS-65D

COMMAND? E

A*SA 01,1=2A00/8

A*
<RST>
<AD>   FF17
<GO>

(RUBOUT)

OS-65D V3.0
OSI 9 DIGIT BASIC
COPYRIGHT 1977 BY MICROSOFT
 34177 BYTES FREE

Ok
RUN"BEXEC*"

OS-65D  Tutorial disk five - Sept. 16, 1981
 1 > Directory
 2 > Create a new file
 3 > Change a file name
 4 > Delete file from diskette
 5 > Create blank data diskette
 6 > Create data diskette with files
 7 > Create buffer space for data files
 8 > Single or dual disk drive copier
 9 > Enter OS-65D system
Type the number of your selection
and depress RETURN ? 1

     Directory utility
Directory of which drive ?
Type A,B,C or D and depress RETURN <A> ?

Do you want to list the directory
to the printer (Yes or No) <No> ?

   -- Directory --
File name   Track range
-----------------------
 OS65D3       0  - 13
 BEXEC*      14 - 16
 COPIER      17 - 18
 CHANGE      19 - 20
 CREATE      21 - 22
 DELETE      23 - 23
 DIR         24 - 24
 RANLST      25 - 26
 RENAME      27 - 27
 SECDIR      28 - 28
 SEQLST      29 - 30
 TRACE       31 - 31
 ZERO        32 - 33
 ASAMPL      34 - 34
 ATNENB      35 - 35
 COLORS      36 - 36
 MODEM       37 - 38
 COMPAR      39 - 39
 46 Entries free out of 64

Depress RETURN to continue ?
```

6



Figuur 6. Deze twee "krachtcentrales" verzorgen de DOS-computer met voldoende stroom. Beide schakelingen kunnen op printen uit de EPS worden opgebouwd. De voedingen zijn zo gedimensioneerd dat ze groot genoeg zijn voor een komplete junior computer met vier drives.

dan steekt men zijn eigen diskette in de drive. De klep van de drive wordt weer gesloten en dan vervolgen we met datgene wat in tabel 3 is vermeld: eerst wordt IN getypt. Dit kommando initialiseert de diskette. Het systeem vraagt dan: "ARE YOU SURE?". Het antwoord dat dan moet worden gegeven is Y(ES). Daarna begint de drive klikgeluiden te produceren. Als het klikken ophoudt is op alle 39 tracks van de diskette de formatteer-informatie geschreven. Na de initialisatie is de diskette klaar voor read/write-operaties. We gaan nu verder volgens tabel 3. Eerst worden met het kommando W2200/2200,8 acht pagina's software op track 0 geschreven. Het startadres is $ 2200 en de laadvektor voor de bootstrap is eveneens $ 2200. Daarmee is de eerste helft van de DOS op de diskette gezet.

Met het kommando SA 01,1 = 2A00/8 wordt op sektor 1 van track 1 een datablok met een lengte van acht pagina's gesaved, beginnend bij startadres $ 2A00. Met dit kommando "redt" men dan het tweede deel van de DOS op de diskette.

Aangezien men met het kopieerprogramma van V3.1 niet van drive A naar drive A kan kopiëren, moet track voor track met de hand gekopieerd worden. Hoe dat gaat laat tabel 4 zien. Daarbij moet men er aan denken dat bij de "CA"-kommando's de Ohio-diskette in drive A moet zitten en bij de "SA"-kommando's de eigen diskette. Als de tracks 2 . . . 32 van de Ohio-diskette naar de eigen diskette gekopieerd zijn kan men eens proefdraaien om te zien of alles goed funktioneert. Schakel de voedingsspanning van de computer uit en wacht enkele sekonden. Schakel dan de voedingsspanning weer in en steek de eigen (pas volgeschreven) diskette in drive A. De bootstrap wordt nu gestart op adres $ FF17, zoals tabel 4 aangeeft. Het systeem meldt zich dan met een "bericht" en de prompt OK. Het aantal vrije bytes hangt af van de grootte van het RAM-bereik. Bij een dynamische RAM van 48 Kbyte is het aantal vrije geheugenplaatsen 36225.

Dan starten we met het kommando RUN"BEXEC*" het utility-programma BEXEC*. Dit programma is in BASIC geschreven. De computer drukt nu een serie opties uit. Kies eerst de optie UNLOCK. De computer is dan klaar voor het ontvangen van alle BASIC-statements.
We gaan hierna verder volgens tabel 5. Start het directory-utility-programma. De computer drukt nu de hele "directory". Tenslotte proberen we nog of de (BREAK)-toets van de terminal goed funktioneert. Tabel 6 laat zien hoe een LIST-kommando en daarna een lopend programma met de (BREAK)-toets onderbroken wordt. Met het CONT-kommando kan men het programma vanaf de onderbroken plaats weer verder laten lopen.

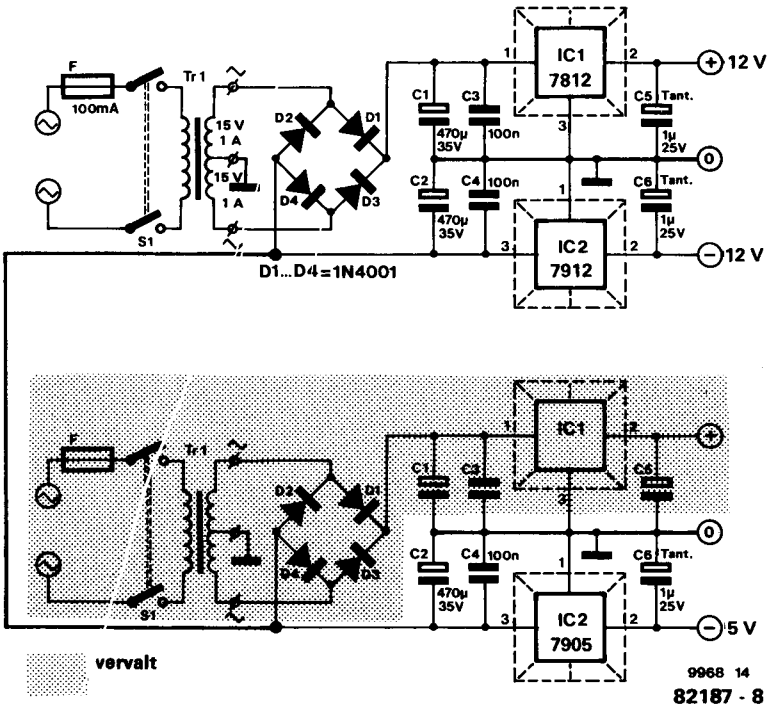
Aanpassingen bij OS-65D V3.1 met twee drives

Een stuk eenvoudiger gaat de aanpassing van V3.1 op de junior computer als men twee drives heeft. Eerst modificeert en kopieert men track Ø en track 1 volgens tabel 2 en 3. Als deze twee tracks op de nieuwe diskette zijn gezet neemt men deze diskette uit drive A en stopt ze in drive B. De Ohio-diskette wordt nu in drive A gestopt. Dan gaan we verder volgens tabel 7. De computer kopieert nu automatisch alle tracks, namelijk de nummers 2 . . . 32, van de Ohio-diskette in drive A naar de nieuwe diskette in drive B. Na deze procedure heeft men een eigen, aangepaste diskette met Ohio-software voor de junior computer.

Aanpassen van een diskette OS-65D V3.3

De diskette V3.3 kan veel eenvoudiger worden aangepast dan de diskette V3.1. Men heeft daarvoor slechts één drive nodig. Tabel 8 toont wat er moet gebeuren. Eerst laden we weer track Ø en track 1 in de computer. Als de DOS in het geheugen zit laden we met het kommando GO 2276 de BASIC-interpreter. Met de prompt OK meldt de computer dat de interpreter is geladen. Dan laden we het utility-programma BEXEC* met het kommando:
RUN"BEXEC*"
De computer geeft dan een serie opties en we kiezen daaruit optie "8". Hiermee kiezen we het kopieerprogramma. Als alle tracks zijn gekopieerd moeten we nog enkele modifikaties in het disk operating system aanbrengen. Daartoe drukken we nogmaals op de <RST>-toets, stoppen de eigen disk in drive A en starten we het programma op adres $ FFE8. Track Ø en track 1 worden nu weer in de computer geladen. De geheugenruimte $ 22ØØ . . . 22FF wordt nu echter niet meer gebruikt door de BASIC-interpreter. Op de adressen $ 2217, 2245, 2285 en 2E84 worden de modifikaties ingebracht die in tabel 8 staan. Nadat dit is gebeurd laden we van sektor 4 van track 6 het track-Ø-read/write-programma. Met dit programma schrijven we de gemodificeerde DOS weer op track Ø van de (eigen) diskette. Met het kommando SA Ø,1 = 2AØØ/8 worden vervolgens acht pagina's DOS-software op track 1 gesaved. En nu hebben we een diskette V3.3 die is aangepast aan de junior computer. Op adres $ FF17 kan men BASIC starten. De DOS en de BASIC-interpreter worden automatisch na het bedienen van de (RUBOUT)-toets in de junior computer geladen (zie tabel 8).

7



Figuur 7. De voedingen voor de dynamische RAM's en de printer-interface.

De DOS-command-interpreter

Zoals we al eerder in dit artikel hebben vermeld heeft het disk operating system een eigen command-interpreter. Op deze plaats zullen we in het kort de belangrijkste kommando's bespreken. Als een kommando niet goed wordt ingetypt verschijnt een foutmelding. Telkens als de computer de DOS-prompt A* of B* afdrukt kan men achter deze prompt een DOS-kommando geven. De computer herkent van elk kommando alleen de eerste twee hoofdletters. Bij een SAVE-kommando maakt het dus niks uit of men SA, SAV of SAVE intikt.

Het kommando AS of ASM

De computer laadt van de momenteel geselekteerde floppy de assembler en de extended monitor. Als dit machinetaal-programma is geladen volgt een sprong naar de cold-start-entry van de assembler. De assembler is verbonden met een regelgeoriënteerde editor.

Het kommando EM

De computer laadt van de momenteel geselekteerde floppy de assembler en de extended monitor. Als dit machinetaal-programma is geladen volgt een sprong naar de extended monitor. Met behulp van deze extended monitor kan men heel eenvoudig een programma in machinetaal kontroleren. De extended monitor heeft zijn eigen command-interpreter. De belangrijkste kommando's zijn:
!STRING
"STRING" wordt naar de DOS-command-interpreter gezonden als een kommando.
@NNNN.
De geheugenplaats met het adres NNNN wordt geopend voor het uitvoeren van een van de volgende subkommando's:

(LF)	Open de volgende geheugenplaats.
(CR)	Sluit de momenteel geadresseerde geheugenplaats.
(D) (D)	Schrijf de data DD in de momenteel geadresseerde geheugenplaats.
(")	Print het ASCII-karakter van de momenteel geadresseerde geheugenplaats.
(/)	Maak de momenteel geadresseerde geheugenplaats klaar voor een nieuwe data-ingave.
(^)	Open de reeds geadresseerde geheugenplaats voor een data-ingave.

BN,LLLL
Zet het breakpoint met nummer N op adres LLLL. Het getal N mag liggen tussen 1 en 8.
EN

Wis het breakpoint met nummer N.

A

Print de accu-inhoud zoals die was bij het laatste breakpoint.

C

Start het programma na het laatste breakpoint.

DNNNN, MMMM

Maak een memory-dump van adres NNNN tot aan adres MMMM.

EX

Verlaat de extended monitor en spring naar de DOS.

FNNNN,MMMM=DD

Zet in het geheugenbereik van adres NNNN tot en met MMMM-1 de data DD.

GNNNN

Spring naar adres NNNN en werk het programma vanaf die plaats verder af.

HNNNN,MMMM(OP)

Oproepen van de hexadecimale "rekenmachine". De "rekenmachine" berekent NNNN (OP) MMMM, waarbij (OP) +, –, * en / mag zijn. Op deze manier kan men eenvoudig rekenen met hexadecimale getallen.

MNNNN=MMMM,LLLL

Verplaats het geheugenbereik MMMM . . . . LLLL-1 naar het geheugenbereik dat begint bij adres NNNN (move-kommando).

RMMMM=NNNN,LLLL

De extended monitor heeft een zogenaamde relocator, waarmee programma's in een ander geheugenbereik kunnen worden gezet. De computer zorgt ook voor het aanpassen van alle absolute adressen: verplaats het geheugenbereik tussen NNNN . . . LLLL-1 naar het geheugenbereik dat begint bij adres MMMM.

De extended monitor heeft nog meer kommando's, die men kan vinden in het manual van Ohio (wel in de Engelse taal).

Het kommando BA

De computer laadt van de momenteel geselekteerde floppy de BASIC-interpreter. Als deze interpreter is geladen volgt een sprong naar de cold-start-entry. De interpreter geeft informatie over het aantal vrije geheugenplaatsen in het systeem en meldt zich terug met de prompt OK.

Het kommando CA NNNN=TT,S of CALL NNN=TT,S

De data van sektor S van track TT wordt in de computer geladen in het geheugenbereik NNNN . . . Het getal TT mag liggen tussen 1 en 39 en het getal S tussen 1 en 8.

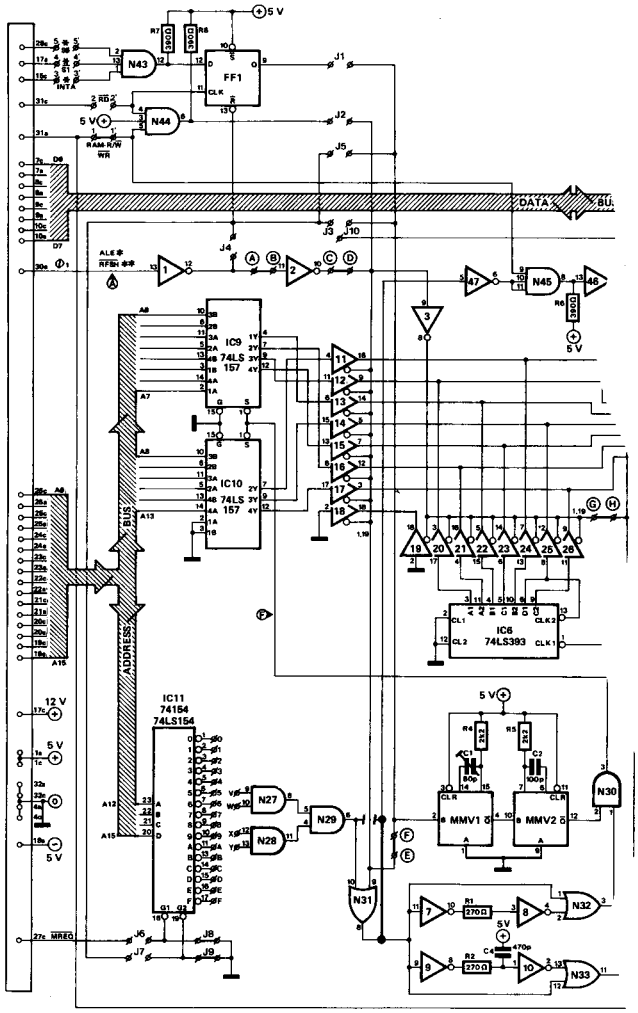
Het kommando SA TT,S=NNNN/8PP of SAVE TT,S=NNNN/P

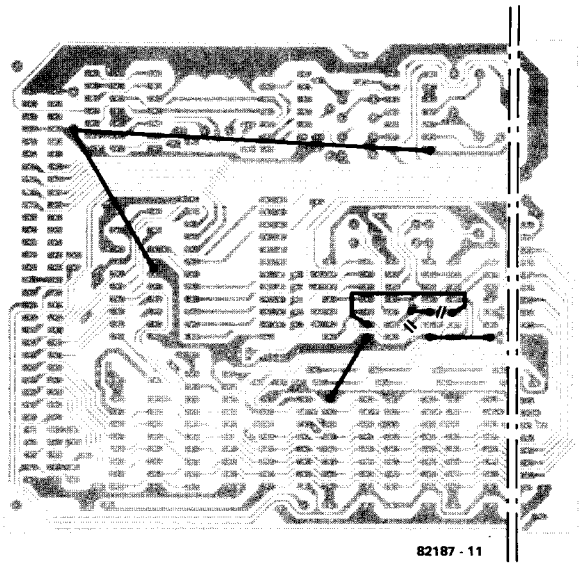
Save de data in het geheugenbereik NNNN . . . voor P pagina's op de diskette. Het track-nummer is TT, het sektornummer S en het aantal pagina's dat op de diskette wordt geschreven P. TT mag een getal zijn tussen 1 en 39, S een getal van 1 tot 8 en P ook een getal tussen 1 en 8.

Het kommando DI TT of DIR TT

Met dit kommando kan men kontroleren hoeveel sektoren op track TT staan. TT is een getal tussen 1 en 39.

8





Figuur 8. Deze veranderingen zorgen er voor dat de dynamische RAM-kaart met alle 6502- en 6809-systemen goed werkt. Tevens is aangegeven hoe de veranderingen aan de koperzijde van de RAM-kaart moeten worden aangebracht. Verder wordt C1 vervangen door een trimmer van 80 p en vervalt C3.

Het kommando IN of INIT

Dit kommando initialiseert een nieuwe diskette waarop nog geen data is geschreven. Als men een beschreven diskette wil wissen is dat ook mogelijk met het IN-kommando.

Het kommando IN TT

Dit initialiseert alleen track TT

Het kommando SE X of SELECT X

Hiermee kan men een van de vier drives kiezen. De computer kan slechts met één gekozen drive samenwerken. X = A, B, C of D.

Het kommando LO FILE of LOAD FILE

Met dit kommando kan men een file met een naam in het werkgeheugen laden. De naam moet echter in de "directory" van de diskette staan. Een file-naam kan men met behulp van het CREATE-utility-programma in de directory (track 12) zetten. Verdere gegevens staan in het manual dat bij de diskette wordt geleverd. De file-naam moet beginnen met een letter en mag 1 . . . 6 karakters lang zijn.

Het kommando PU FILE of PUT FILE

Hiermee kan men een file met een naam van het werkgeheugen op de diskette schrijven. Dit is alleen mogelijk als men eerst de file-naam in de directory heeft gezet.

De kommando's PU TT of LO TT

Met deze kommando's kan men een file uit het werkgeheugen van de computer zonder file-naam op de diskette schrijven of van de diskette in de computer lezen. De file in het werkgeheugen mag daarbij niet langer zijn dan 2 Kbyte. Met deze kommando's moet men heel voorzichtig zijn, aangezien men zo gemakkelijk bestaande software op de diskette kan overschrijven zonder dat er een foutmelding wordt gegeven.

Het kommando RE of RETURN

Met de RETURN-kommando's kan men vanuit de DOS-command-interpreter terugkeren naar de momentele transient processor:

RE AS return to assembler
RE BA return to BASIC
RE EM return to extended monitor

Hardware-bijzonderheden bij een 6502-DOS-computer

Computers waarbij gebruik wordt gemaakt van een disk operating system moeten een goede voeding hebben. De DOS-junior moet dan ook worden voorzien van een netvoeding die voldoende reserves heeft en die impulsen goed kan verwerken. Als men al een uitgebreide junior computer heeft kan men de bestaande netvoeding uitstekend voor de DOS-junior gebruiken. Voor de drives is dan wel nog een aparte voeding van 12 V/4 A nodig. Heeft men geen geschikte voeding, dan kunnen de schakelingen uit figuur 6 en 7 worden gebouwd. Deze voedingen zijn in staat om het hele systeem van voldoende spanning en stroom te voorzien: bovendien zijn ze vrij eenvoudig te bouwen en bedrijfszeker. De hele voeding kan worden ondergebracht op vier printen: twee maal EPS 82570 (voor de voeding uit figuur 6) en twee maal EPS 9968-5a (voor de voeding uit figuur 7). Er zijn twee trafo's nodig, een trafo die 9 . . . 10 V/10 A levert en een tweede die 15 V/4,4 . . . 5 A levert. Heel geschikt voor deze toepassing zijn ringkerntrafo's. Ringkerntrafo's zijn weliswaar iets duurder dan "normale" trafo's, maar ze zijn ook lichter en hebben een kleiner strooiveld. Bovendien is de warmte-ontwikkeling zelfs bij volle belasting vrij gering.

Voor de dynamische RAM-kaarten en de printer-interface is nog een derde trafo nodig die twee maal 15 V/1 A levert. Aan deze trafo worden geen bijzondere eisen gesteld. De netvoedingen uit figuur 6 en 7 zullen bij de DOS-junior en elke andere DOS-computer nooit problemen geven, aangezien de voedingen flink overgedimensioneerd en kortsluitvast zijn. Alle voedingen kunnen samen in een kast worden gemonteerd. Bij de bedrading moet kabel van 1,5 mm^2 gebruikt worden.

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

FC00: A9 1E 8D 83 FA A9 01 85 FD A2 FF 9A 78 D8 20 7C
FC10: FC D0 FB 20 7C FC F0 FB 20 7C FC F0 F6 20 E1 FC
FC20: C9 13 D0 07 A2 FF 9A D8 6C FA 00 C9 10 D0 06 A9
FC30: 01 85 FD D0 14 C9 11 D0 06 A9 00 85 FD F0 0A C9
FC40: 12 D0 09 E6 FA D0 02 E6 FB 4C 0D FC C9 14 D0 03
FC50: 4C FD FE C9 15 10 F2 85 FF A4 FD D0 0D B1 FA 0A
FC60: 0A 0A 0A 05 FF 91 FA 4C 49 FC A2 04 06 FA 26 FB
FC70: CA D0 F9 A5 FA 05 FF 85 FA 4C 49 FC A0 00 B1 FA
FC80: 85 F9 A9 7F 8D 81 FA A2 08 A5 FB 20 B8 FC A5 FA
FC90: 20 B8 FC A5 F9 20 B8 FC A9 00 8D 81 FA A0 03 A2
FCA0: 00 A9 FF 8E 82 FA E8 E8 2D 80 FA 88 D0 F5 A0 06
FCB0: 8C 82 FA 09 80 49 FF 60 48 84 FC 4A 4A 4A 4A 20
FCC0: CB FC 68 29 0F 20 CB FC A4 FC 60 A8 B9 08 FD 8D
FCD0: 80 FA 8E 82 FA A0 FF 88 D0 FD 88 8C 82 FA E8 E8
FCE0: 60 A2 21 A0 01 20 A1 FC D0 07 E0 27 D0 F5 A9 15
FCF0: 60 A0 FF 0A B0 03 C8 10 FA 8A 29 0F 4A AA 98 10
FD00: 03 18 69 07 CA D0 FA 60 40 79 24 30 19 12 02 78
FD10: 00 10 08 03 46 21 06 0E 20 1E FD 6C FD 00 20 28
FD20: FD 20 CF FD 20 4F FD 60 A0 00 8C 01 C0 A9 40 8D
FD30: 00 C0 A9 04 8D 01 C0 A9 40 8D 00 C0 A2 04 8E 01
FD40: C0 8C 03 C0 88 8C 02 C0 8E 03 C0 8C 02 C0 60 A9
FD50: FB D0 09 A9 02 2C 00 C0 F0 1C A9 FF 8D 02 C0 20
FD60: CE FD 29 F7 8D 02 C0 20 CE FD 09 08 8D 02 C0 A2
FD70: 18 20 BA FD F0 DD A2 7F 8E 02 C0 20 D7 FC AD 00
FD80: C0 30 FB AD 00 C0 10 FB A9 03 8D 10 C0 A9 58 8D
FD90: 10 C0 20 C5 FD 85 FE AA 20 C5 FD 85 FD 20 C5 FD
FDA0: 85 FF A0 00 20 C5 FD 91 FD C8 D0 F8 E6 FE C6 FF
FDB0: D0 F2 86 FE A9 FF 8D 02 C0 60 A0 F8 88 D0 FD 55
FDC0: FF CA D0 F6 60 AD 10 C0 4A 90 FA AD 11 C0 60 D8
FDD0: 78 A9 67 8D 82 FA A9 00 8D 80 FA A2 FC 9E 5A FA
FDE0: A2 FF 8E 5B FA EA A9 7F 8D 81 FA 4A 8D 83 FA A2
FDF0: 03 8E 59 FA 2C 80 FA 30 FB 20 4F FE 4E 5F FA 6E
FE00: 5E FA AD 5E FA 8D 5C FA AD 5F FA 8D 5D FA A2 08
FE10: 20 72 FE 20 2B FE C9 7F D0 B5 60 2C 80 FA 30 FB
FE20: 8E 61 FA A2 08 20 72 FE 20 81 FE 2C 80 FA 10 09
FE30: 38 6E 62 FA CA D0 F1 F0 07 18 6E 62 FA CA D0 E8
FE40: 20 81 FE AD 62 FA 29 7F 8D 63 23 AE 61 FA 60 18
FE50: AD 5A FA 69 01 8D 5A FA AD 5B FA 69 00 8D 5B FA
FE60: 2C 80 FA 10 EA AD 5A FA 8D 5E FA AD 5B FA 8D 5F
FE70: FA 60 AD 5C FA 8D 5E FA AD 5D FA 8D 5F FA 38 B0
FE80: 0C AD 5A FA 8D 5E FA AD 5B FA 8D 5F FA 38 AD 5E
FE90: FA E9 01 8D 5E FA AD 5F FA E9 00 8D 5F FA EA EA
FEA0: B0 EB 60 8E 60 FA 8D 62 FA AD 82 FA 29 40 D0 F9
FEB0: AD 82 FA 29 FE 8D 82 FA 20 81 FE A2 07 4E 62 FA
FEC0: 90 30 AD 82 FA 09 01 8D 82 FA 20 81 FE CA D0 ED
FED0: AE 59 FA AD 82 FA 09 01 8D 82 FA 20 81 FE CA D0
FEE0: F2 2C 80 FA 10 04 AE 60 FA 60 2C 80 FA 10 FB 6C
FEF0: 7C FA AD 82 FA 29 FE 8D 82 FA 18 90 CD 20 03 FF
FF00: 4C 51 2A A9 27 8D 82 FA A9 00 8D 80 FA A9 7F 8D
FF10: 81 FA 4A 8D 83 FA 60 A9 2E 8D 7C FA A9 FF 8D 7D
FF20: FA A9 00 8D 7A FA A9 FC 8D 7B FA 4C 18 FD A9 03
FF30: 8D 25 23 60 A9 51 8D 7C FA A9 2A 8D 7D FA A9 00
FF40: 8D 7A FA A9 FC 8D 7B FA 4C 18 FD 6C 7A FA 6C 7E
FF50: FA 20 1E FD A9 28 8D A3 26 A9 01 8D 5E 26 20 BC
FF60: 26 A9 2A 85 FF 20 54 27 86 FE 20 67 29 A9 01 8D
FF70: 21 23 8D 22 23 8D C6 2A 20 C6 29 A9 1A 8D 01 23
FF80: 8D 03 23 A9 A2 8D 11 23 8D 13 23 A9 FE 8D 02 23
FF90: 8D 04 23 8D 12 23 8D 14 23 60 EE 5E 26 A9 06 20
FFA0: BC 26 20 67 29 EE 5E 26 A9 00 85 FE 85 FF 20 67
FFB0: 29 A9 01 8D 5E 26 A9 13 20 BC 26 A9 32 85 FF A9
FFC0: 74 85 FE 20 54 27 20 67 29 20 61 27 20 73 2D 00
FFD0: 0A 2A 54 52 41 43 4B 20 30 26 31 2A 0D 0A 00 4C
FFE0: 00 FC 20 51 FF 4C C9 FF 20 51 FF 4C 9A FF FF FF
FFF0: FF FF FF FF FF FF FF FF FF FF 4B FF 00 FC 4E F5
```

Tabel 9. De hexdump-listing voor de bootstrap-EPROM ESS 515. De source-listing is verkrijgbaar als "paperware".

De dynamische RAM-kaart

Uit ervaringen in de afgelopen maanden is gebleken dat de dynamische RAM-kaart met sommige 6502-processors niet goed werkt. Ook de kombinatie van 6809-processor met dynamische RAM-kaart wil niet zo best funktioneren. Daarom hebben we een paar veranderingen op de dynamische RAM-kaart aangebracht, waardoor deze geheugenkaart nu geschikt is voor elk 6502/6809-systeem. Figuur 8 toont alle veranderingen die in het schema en op de print zijn aangebracht.

Dat was het hele verhaal over de hardware en software voor de floppy-disk-interface. We wensen iedereen veel plezier met de DOS-junior, de personal-zelfbouw-computer met disk operating system!

De diskettes met OS-65D V3.1 en V3.3 worden geleverd door:
Ingenieursbureau Koopmans
Sluisweg 2h — Postbus 176
3370 AD Hardinxveld — Giessendam
tel. 01846-6833