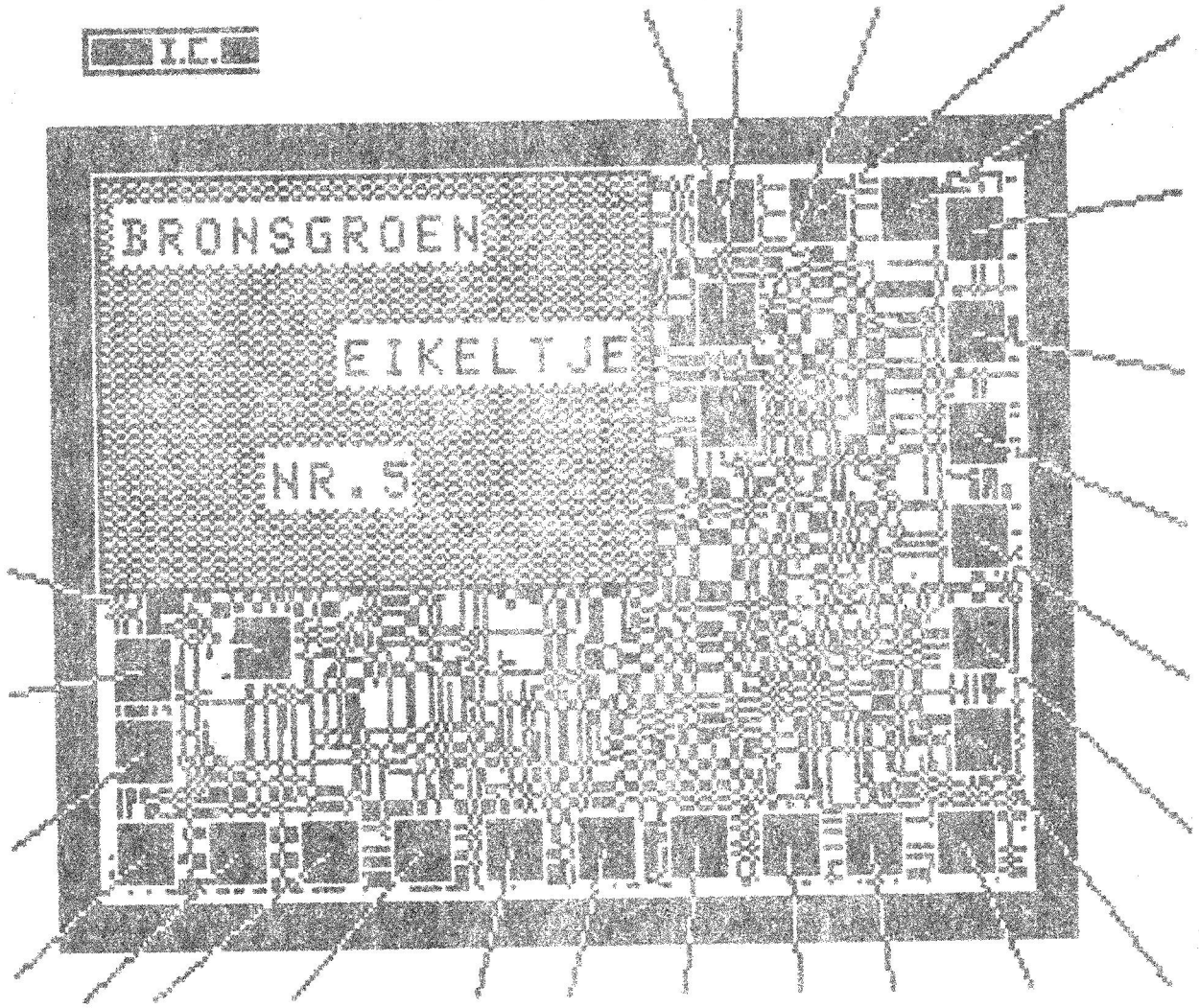


ACORN COMPUTERCLUB

LIMBURG

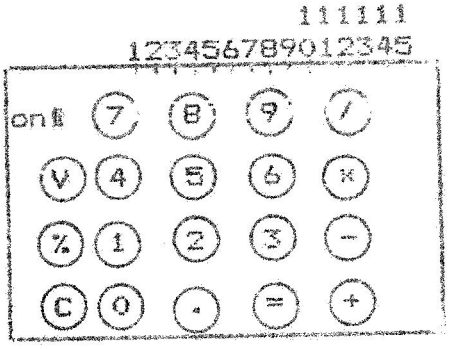
****
NUMERIEK TOETSENBORD J. WIJNEN
****

Onderstaande omschrijving werd overgenomen uit het clubblad van "BRABANT-OOST".

Als er op een gegeven moment veel getallen ingetypt moeten worden dan is het gemakkelijk als alle cijfertoetsen bij elkaar staan.
Veel computers zijn daarom ook uitgevoerd met een numeriek toetsenbord.
Dit programmatje maakt het mogelijk om zo'n bordje aan te sluiten op de ATOM.
Ik heb een toetsenbordje met de volgende aansluitingen:

```
      1 2 3 4 5 6 7 8 9
   ┌──────────────────
10 │   V %
11 │ C   = + - x / . 0
12 │ 1 2 3 4 5 6 7 8 9
```

13---- geen aansluiting
14-15 on/off schakelaar



Ik heb het als volgt aan de de VIA 6522 aangesloten:

1 aan A0, 2 aan A1, enz.
9 aan B0, ..., 12 aan B3
overige (13,14,15) niet aangesloten.

Het programma vervangt de normale read character routine, zodat er alleen nog maar getallen (geen letters) met het numerieke bordje worden ingevoerd.
De ESC-toets zorgt dat je normaal het toetsenbord kunt gebruiken.

p.s. ik wil het aan gaan sluiten op de 6 niet geselecteerde rijen in de toetsenbord matrix. Daar is wel een extra IC voor nodig maar je houdt de VIA dan wel vrij.

```
 10 DIME30;$E="P.""INCORRECTE OPGAVE""";G.240"
 20 ?16=E;?17=E/256
 30 DIMLL10;FORI=0TO10;LLI=#FFF;N.
 40 P.$21;FORI=0TO1;DIMP-1;[
 50 :LL3PHP;CLD;STX#E4;STY#E5
 60 :LL0JSRLL5;BCCLL0
 70 :LL1JSRLL5;BCSLL1
 80 JSRLL5;BCSLL1
 90 JMP#FE60
100 :LL5LDA@14;STA#B802;LDA@#36;STA#B800
110 LDX@3;LDY@27
120 :LL6LDA#B800;AND@1;BEQLL10;DEY;LDA@128
130 :LL7BIT#B801;BEQLL10;DEY;LSRA;BNELL7
140 LSR#B800;DEX;BNELL6
150 LDA@32;BIT#B001;BEQLL4;TYA;SEC;RTS
160 :LL4LDA@#94;STA#20A;LDA@#FE;STA#20B;LDA@#1B;CLC;RTS
170 :LL10LDALL2,Y;CLC;RTS
180 :LL2;J;N.;P.$6
190 $LL2="  ^%        C =+-*/.0123456789"
200 ?(LL2+12)=13
210 ?(LL2+10)=127
220 P.$12
230 ?#20A=LL3;?#20B=LL3/256
240 FIN.%B;FP.%B';G.240
```

****
WEG RUIS bron onbekend
****

WEG MET DE RUIS IN DE GRAPHIC MODE'S 1 T/M 4

Iedereen zal zich wel vaak geergerd hebben aan de vele ruis die voorkomt in de graphic's , denk hierbij b.v. aan de MOVING SHAPE'S met de JOSBOX. Er zijn enkele mogelijkheden om deze ruis weg te werken, zie hiervoor o.a. enkele publicaties in Acorn Nieuws 1982 - 1983. Al deze methodes hadden het bezwaar dat ze veel extra tijd vergden waardoor de computer veel trager werd. De hier door mij voorgestelde methode heeft daar minder last van. De vertraging is bijna niet merkbaar, hij is er dus wel.
Waar komt de ruis vandaan?
Wel als zowel de 6502 als de 6847 gelijktijdig in het geheugen blok van #8000 tot #9800 werken dan krijgt de 6502 voorrang waardoor de 6847 tijdelijk geen video kan verzorgen. Door er nu voor te zorgen dat de 6502 wacht tot de 6847 het geheugen niet nodig heeft dat is b.v. tijdens de vertikale en horizontale lijnpuls, is de ruis weg.
Intern kan dat meteen door de vertikale sync.puls te gebruiken, deze heeft een frequentie van 60 Hz. maar met een extra Pia poortje is er de mogelijkheid om de vertikale sync.puls met een frequentie van meer dan 16000 Hz. in te lezen en te gebruiken. De 6502 kan nu vaker werken in het geheugen terwijl de ruis weg blijft.
In de standaard Atom is dit mogelijk door van de Via een bitje van poort B in te zetten of door b.v. een van de vrije pootjes van Big-Benny als u die tenminste heeft te gebruiken.
Voor beide mogelijkheden is de software gegeven.
Indien de Via gebruikt wordt is het aan te bevelen een schakelaar in de verbinding op te nemen, dit om ook de Via voor andere aktiviteiten te kunnen gebruiken. Denk hierbij b.v. aan de Eprom-programmer of een extra soundboard.
Nu de aansluitingen:
Ik ga er even van uit, dat u de Via gebruikt hiervan stel ik voor PB7 te gebruiken. De lijnsyncpuls is te vinden op pootje 38 van I.C. 31, dus de 6847.
Er moet nu een draadje worden gelegd tussen pootje 38 van de 6847 en pootje 17 van I.C. 1, de 6522. (hier moet dus wel eventueel nog een schakelaar in). De lijnsyncpuls is eventueel ook te vinden op pootje 8 van PL4.
De routine moet nu gebruikt worden na iedere declaratie van CLEARx of COLOUR. Dit betekent dus als u de code assembleert naar b.v. #6800 na ieder CLEARx of COLOUR LINK#6800. Daaom is het misschien beter om een extra statement met deze routine te maken die na elke CLEARx of COLOURx komt, of nog wat het beste is hele nieuwe graphic commando's schrijven, kortom mogelijkheden te over om te experimenteren. In MODE 0 werkt dit niet maar dat zal meestal geen bezwaar zijn omdat hier de ruis toch minimaal is.

```
10 REM NO NOISE
20 REM DEZE VERSIE WERKT MET DE VIA
30 REM PB7 WORDT NU GEBRUIKT
40 REM DENK AAN EEN EVENTUELE SCHAKELAAR
```

```
 50 J=5
 60 DIM LL(J)
 70 F.N=1 TO J;LL(J)=#FFF;N.
 80 LL1=#80;LL2=#3FE;LL3=#3FF
 90 LL4=#B800
100 P.$12" NO NOISE"''
110 IN."WAAR MOET DE CODE HEEN"Z
120 FOR I=1 TO 2;P=Z;@=O
130[
140:LL0 LDALL2;STA#80;LDALL3;STA#81
150LDA@LL5%256;STALL2;LDA@LL5/256;STALL3
160RTS
170:LL5 LDALL4;BMILL5;JMP(LL1)
180]
190 N.;P."CODE VAN "&Z" TOT "&P';E.


 10 REM NO NOISE
 20 REM DEZE VERSIE WERKT ALLEEN MET BIG-BENNY
 30 REM EEN VRIJ POOTJE VAN DE PIA IS NU INGEZET
 40 REM NU IN GEBRUIK PB5 IS POOTJE 15
 50 J=8; DIMLLJ;F.I=0 TO J;LLI=#FFF;N.;DIMA4;@=0
 60 P.$12" NO NOISE"''
 70 IN."WAAR MOET DE CODE HEEN"Z
 80 LL1=#80;LL2=#3FE;LL3=#3FF;LL8=#B402;LL7=#B403
 90 P.$21;F.I=1 TO 2;P=Z;IFI=2;P.$6;IF?A=74P.$2
100[
110:LL0 LDALL2;STA#80;LDALL3;STA#81
120LDA@#04;STALL7;LDA@LL6%256;STALL2;LDA@LL6/256;STALL3;RTS
130:LL6
140LDA@#20
150:LL5 BITLL8;BNELL5;JMP(LL1)
160];N.;P.'"CODE VAN "&Z" TOT "&P';E.

OPSTARTEN NA IEDERE CLEARx OF COLOUR MET LINK LL0
```

BESTUURSMEDEDELINGEN

Schijvenarchief: dit is overgegaan naar Hans Steenan te Geleen. Bestellen op de club-avond.

Redactie-commissie: toegevoegd aan de redactie-commissie Evert Sanders en Hans Heuts. De redactie-commissie hebben verlaten Harry Princen en Wim Ernst.

Bestuur: N.a.v. geruchten die de ronde doen deelt het bestuur mede, dat GEEN der bestuursleden voornemens is de aktiviteiten binnen de club te staken tot de eerstvolgende ledenvergadering, begin 1985.

****
BEGINNERSHOEKJE E. SANDERS
****

HOOFDSTUK 7 ARRAYS EN VECTORS

Tot nu toe hebben we net 26 variabelen ontmoet, nl. de letters A tm Z.
Stel, dat je een grafiek wilt laten zien, die de gemiddelde temperatuur van iedere maand van het jaar laat zien. Je zou, in noodgevallen, de 12 letters A tm L kunnen gebruiken om de diverse gemiddelde temperaturen voor te stellen en ze in te lezen door te zeggen:

```
INPUT A,B,C,D,E,F,G,H,I,J,K,L
```

Er is echter een handigere manier. Een wiskundige zou de reeks temperaturen als volgt voorstellen:

$t_1, t_2, t_3 \ldots\ldots t_{12}$

waarin het 'subscript', het cijfer onder de regel, het nummer van de maand van het jaar is. Deze manier van voorstellen is veel duidelijker dan met het gebruik van 12 verschillende letters en er bestaat geen twijfel over welk simbool de temperatuur van bijv. de derde maand moet voorstellen.
Een soortgelijke serie variabelen kent de ATOM BASIC ook, ze worden 'arrays' genoemd. Een lid van een array bestaat uit een array 'identifier', oftewel de naam die correspondeert met de naam 't' in bovengenoemd voorbeeld en een subscript.
Op de meeste computers is geen faciliteit aanwezig voor het schrijven van subscripts, zodat een andere manier van voorstellen moet worden gebruikt.
Ieder lid van de array kan optreden als een volledig onafhankelijke variabele welke in staat is een waarde te bevatten net als de variabelen A tm Z. De leden van een array worden 'elementen' genoemd. Het maximale aantal elementen hangt af van hoe de array was opgezet cq gedemensioneerd, in het bovengemoemde voorbeeld waren het 12 elementen met subscripts van 1 - 12.
Als extra boven de normale arrays kent ATOM BASIC nog twee andere soorten arrays nl. 'byte vectors' en 'word vectors'. Byte vectors zijn handig als slechts kleine getallen nodig zijn en gebruikt minder opslagruimte dan word arrays. Word vectors gebruiken dezelfde hoeveelheid opslagruimte als normale arrays, maar zijn wat flexibeler in gebruik.

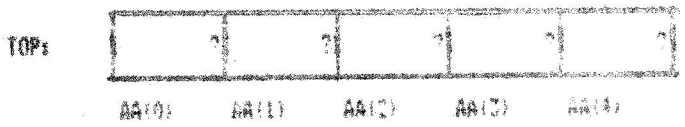
7.1 Arrays - AA tot ZZ

De arrays in ATOM BASIC bestaan uit een paar identieke (hoofd) letters, gevolgt door een subscript tussen haakjes, bijv.: EE(3).
Ieder element van dit type array kan dezelfde getallen bevatten als de gewone variabelen A tm Z nl. de getallen tussen -2000 miljoen en 2000 miljoen.
Voordat een array kan worden gebruikt moet er (geheugen) ruimte voor gereserveerd worden door een DIM, of 'dimension', statement welke de BASIC moet vertellen hoe groot de array is.
Om bijvoorbeeld ruimte te reserveren voor een array genoemd AA met de vijf elementen AA(0), AA(1), AA(2), AA(3) en AA(4) zou

het statement moeten zijn:

```
DIM AA(4)
```

Het DIM statement maakt ruimte voor arrays te beginnen op de eerste vrije geheugenlocatie achter het programma.
Als DIM AA(4) het eerste DIM statement zou zijn dat werd tegengekomen in het programma dan zou het element AA(0) op TOP staan, dus direct achter het programma.
TOP bevat het adres van de eerste vrije gehuegenlocatie achter een BASIC-programma.



Het vraagteken stelt een ongedefinieerde waarde voor, afhankelijk van wat de array bevatte op het moment van dimensioneren. Als nog een array zou worden gedimensioneerd met de opdracht:

```
DIMBB(3)
```

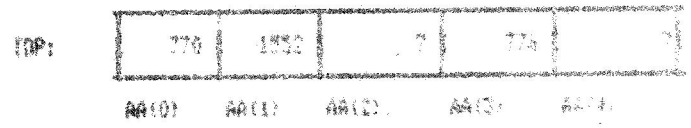
dan wordt de ruimte voor de array BB onmiddellijk achter die van AA gereserveerd.
Array elementen kunnen voorkomen in expressions, ze krijgen een waarde toegekend op dezelfde manier als de gewone variabelen A tm Z. Om bijv. AA(3) de waarde 776 te geven typen we:

```
AA(3)=776  of:
```

We zouden verder kunnen gaan met:

```
AA(1)=AA(3)\2  of:
AA(0)=AA(3)-6
```

enz. Het resultaat van de array in het geheugen zou zijn:



Er zijn echter twee plaatsen in BASIC programma's waar array elementen niet mogen worden gebruikt, te weten:
1) Als een controle variabele in een FOR...NEXT loop.
2) In een INPUT statement.
In deze twee gevallen moeten de gewone variabelen A tm Z worden gebruikt.

7.1.1 HISTOGRAM (staafdiagram)

Het volgende programma laat het gebruik van arrays zien om een staafdiagram van de temperaturen over de 12 maanden van het jaar te laten tekenen.
De temperaturen, deze liggen tussen de 0 en 100 graden, worden eerst ingetoetst en opgeslagen in de array TT(1..12).

```
 1 REM HISTOGRAM
10 DIM TT(12)
20 FOR J=1 TO 12:INPUT K
30 TT(J)=K: NEXT J
```

```
 40 PRINT $12; CLEAR 0; @=5
 50 MOVE 60,12; DRAW 12,12
 60 DRAW 12,42
 70 FOR N=11 TO 0 STEP -1
 80 IF N=7 PRINT "TEMP."
 90 IF N%2=0 PRINT N$10
100 PRINT';NEXT N
110 PRINT "       JAN MAR MAY JUL SEP NOV"'
120 PRINT "         FEB APR JUN AUG OCT DEC"'
130 PRINT "                 MONTH"'
140 FOR N=1 TO 12; J=11+4$N
150 MOVE J,12; DRAW J,(TT(N)$3/10+12)
160 NEXT N; END
```

Beschrijving van het programma:

20-30	Input van de 12 waarden
40	maak scherm schoon
50-60	Teken de assen
70-100	Voorzie vert. as van tekst
110-130	Voorzie hor. as van tekst
140-160	Teken de staven van het diagram

Programmalengte: 415 bytes
Array opslag : 52 bytes

7.1.2 SORTING PROGRAM (sorteer programma)

Het volgende programma laat het gebruik van arrays zien om een serie getallen te sorteren op oplopende waarden. Het maakt gebruik van een zeer efficiente sorteerprocedure die bekend staat als de "Shell" sorteermethode. Het onderstaande programma leest 20 getallen in, roept een subroutine aan om de getallen te sorteren, en print ze vervolgens keurig gesorteerd uit.

```
  1 REM Sorting
  5 DIM AA(20)
 10 FOR N=1 TO 20; INPUT J
 20 AA(N)=J; NEXT N
 30 N=20; GOSUB s
 40 FOR N=1 TO 20; PRINT AA(N)'
 50 NEXT N
 60 END
100s:M=N
110 DO M=(M+2)/3
120   FOR I=M+1 TO N
130     FOR J=I TO M+1 STEP -M
140       IF AA(J)>=AA(J-M) GOTO b
150       T=AA(J); AA(J)=AA(J-M); AA(J-M)=T
160     NEXT J
170b  NEXT I
180 UNTIL M=1; RETURN
```

Beschrijving van het programma:

5-20	Leest de cijfers in (in arrays)
30	Roept de Shell sort routine aan
40-50	Print de gesorteerde array uit
100-180	s: Shell sort subroutine

140-150 Verwisselt de elementen die niet aan de beurt zijn.

Variabelen:
AA(1..20) - Array om de cijfers in op te slaan
I,J - Loop tellers
N - Nummer van een element in de array AA
M - Subset step size
T - Tijdelijke variabele

Programmalengte: 332 bytes
Array opslag : 84 bytes

7.1.3 ARBITRARY-PRECISION ARITHMETIC

Met het volgende programma kunt U machten berekenen tot iedere nauwkeurigheid, mits je voldoende geheugenruimte hebt. Als je het programma intikt zoals het hier staat, berekent het alle machten van 2 waarvan de uitkomst minder dan 32 cijfers heeft.

```
  5 REM POWERS OF TWO
 10 DIM AA(31)
 20 @=11 P=0
 30 AA(0)=1
 40 FOR J=1 TO 31
 50   AA(J)=0
 60 NEXT J
 70 DO J=31
 80     DO J=J-1: UNTIL AA(J)<>0
 85     PRINT"2^" P "="
 90     FOR K=J TO 0 STEP -1
 94       PRINT AA(K)
 96     NEXT K
110     C=0
120     FOR J=0 TO 31
130       A=AA(J)*2+C
140       C=A/10
150       AA(J)=A%10
160     NEXT J
170     P=P+1
180 UNTIL AA(31)<>0
190 END
```

Beschrijving van het programma:
40-60 Vullen van de arrayelementen met 0
80 Negeer eerste nullen
85-96 Print de macht
110-160 Vermenigvuldig het huidige getal met 2
180 Stop als de array te groot wordt

Variabelen:
AA - Array van cijfers, een cijfer per element
C - Decimale carry van het ene element naar het andere
I - Lusteller
J - Lusteller
P - De te berekenen macht

```
Programmalengte  : 356 bytes
Array opslag     : 124 bytes
Totaal geheugen  : 480 bytes
```

7.1.4 DIGITAL WAVEFORM PROCESSING

Het volgende programma gebruikt een uit 256 elementen bestaande array om een golfvorm in op te slaan, die, door hem door een low passfilter te laten gaan, wordt omgezet in een blokgolf. Hierna worden de drie golven uitgeprint.

```
    1 REM DIGITAL WAVEFORM PROCESSING
    5 DIM AA(255)
   10 H=2000
   15 CLEAR4
   23 GOS.s: GOS.q
   25 Z=160: GOS.p
   28 GOS.l
   30 Z=96: GOS.p
   32 GOS.s
   34 Z=32:GOS.p
   90 END
 1000pREM PLOT WAVEFORM
 1005 MOVE 0,96
 1010 FOR N=0 TO 255
 1020 PLOT 13,N,(Z+AA(N)/H)
 1030 NEXT N
 1040 RETURN
 2000sREM MAKE SINE WAVE
 2010 S=0:C=40000
 2020 FOR N=0 TO 255
 2030 AA(N)=-S
 2040 C=C-S/10
 2050S=S+C/10
 2060 NEXT N
 2070 RETURN
 3000qREM MAKE SQUARE WAVE
 3010 FOR N=0 TO 255
 3020 IF AA(N)>=0 AA(N)=40000
 3030 IF AA(N)<0 AA(N)=-40000
 3035 NEXT N
 3040 RETURN
 4000lREM LOW PASS FILTER
 4010 B=0
 4020 FOR N=0 TO 255
 4030 B=AA(N)*360/1000+B*697/1000
 4040 AA(N)=B: NEXT N
 4050 RETURN
```

Beschrijving van het programma:

```
23          Bereken een blokgolf
25          Teken hem bovenaan op het scherm
28          Laat de blokgolf door een low-pass filter gaan
            (althans theoretisch, beiden)
30          Teken hem hierna in het midden op het scherm
32          Bereken een sinusgolf
34          Teken hem aan de onderzijde van het scherm
1000-1040   ps Teken golfvorm
```

2000-2070 s: Bereken een sinusgolf
3000-3040 q: Maak van de golfvorm een blok
4000-4050 l: Laat de golfvorm door een low-passfilter gaan

Variabelen:
AA(0...255) - Array van punten, waarden tussen -40000 en 40000.
B - Voorafgaande waarde voor de low-passfilter
C - Cosinus van de golfvorm
H - Schaalfactor voor het tekenen van de golfvormen
N - Teller
S - Sinus van e golfvorm
Z - Vertikale coordinaten om de golfvorm te centreren.

Programmalengte : 564 bytes
Arrayopslag   : 1024 bytes
Totaal geheugen : 1588 bytes

7.1.5 SUBSCRIPT CHECKING

Diverse BASIC interpreters voeren een uitgebreidde controle uit wanneer er een arrayelement wordt gebruikt in een programma. Als er bijv. een array was gedemensioneerd met:

```
DIM RR(10)
```

dan zou er iedere keer, als de array wordt gebruikt, getest worden of de subscript 0 of groter is en of de subscript 10 of kleiner is. Het is echter wel zo, dat de uitvoering van het programma door deze tests langzamer zou worden. In onze ATOM wordt daarom dan ook alleen maar de eerste test uitgevoerd. Hiermee wordt bereikt dat je alleen maar positieve subscripts kunt gebruiken. Het wordt dus aan de programmeur overgelaten om te zorgen dat de subscripts niet buiten de demensionering vallen.
Als je toch een subscript zou gebruiken die groter is als in de demensionering opgegeven is dan worden de waarden in de andere arrays, of strings, die achter die arrays gedemnsioneerd zijn, veranderd.

Als je toch wil, kun je op een eenvoudige manier toch de subscripts testen. Als een arrayelement de waarde:

```
RR(A)=35
```

krijgt, dan zou het statement:

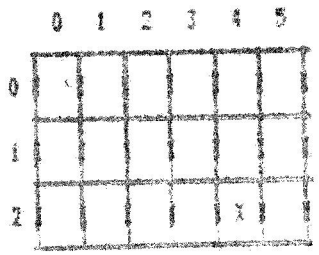
```
IF A>10 THEN PRINT "ERROR"
```

kunnen worden toegevoegd zodat een melding gegeven wordt als de arraysubscript te groot zou worden.

7.1.6 MULTI-DEMENSIONAL ARRAYS

De standaard type arrays in ATOM BASIC zijn 1-demensionaal. Met andere woorden, ze hebben slechts een subscript en kunnen dus worden voorgesteld alsof ze op een lijn liggen, vandaar de naam 'ARRAY'.

Soms kan het makkelijk zijn de te doen alsof ieder element van een array een cel in een vierkant 'matrix' voorstelt; ieder element zou dan 2 subscripts hebben die correspanderen met de kolom en de rij van dat vierkant.
Zulke 2-demensionale arrays worden "matrices" genoemd.
Bekijk maar eens het volgende plaatje van een 3 bij 6 matrix:

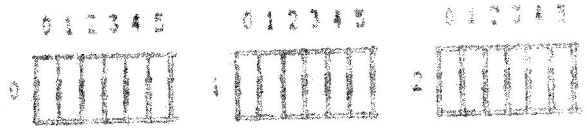


De hele matrix heeft 3 x 6 = 18 elementen en het element aangeduidt met X heeft de subscripts (2,4).

ATOM BASIC heeft geen mogelijkheden tot het rechtstreeks maken van 2 of meer demensionale arrays, maar ze kunnen gemakkelijk worden geconstrueerd door gebruik te maken van de 1-demensionale arrays AA tm ZZ zoals in de volgen delen zal worden verteld.

7.1.7. CALCULATION OF SUBSCRIPTS

Om een 2-demensionale matrix voor te stellen maken we gebruik van de 1-demensionale voorstelling en delen de matrix in rijen zoals hieronder:



Het eerste element van rij 1, met subscript (1,0), volgt onmiddellijk op het laatste element van rij 0 , met de coordinaten 0,5.
We gaan uit van het algemene geval waarin de matrix M rijen, genummerd van 0 - M-1, en N kolommen, genummerd van 0 - N-1, heeft.
De matrix kan dan worden gedemensioneerd, door gebruik te maken van een 1-demensionale array, door het statement:

```
DIM XX(M*N-1)
```

Dus in het algemeen: ieder array element, met de subscripts A en B, kan worden teruggebracht tot:

```
XX(A*N+B)
```

In het vorige voorbeeld had de array een demensie van 3 x 6 en moet dus als volgt gedemensioneerd worden:

```
DIM XX(17)
```

Het arrayelement met de subscripts 2,4 wordt dan:

```
XX(16)          [XX(2*6+4)=16]
```

7.1.8 SOLVING SIMULTANEOUS EQUATIONS

Het volgende programma loost een willekeurig aantal vergelijkingen van de 1ste graad op, door gebruik te maken van

een matrix om de coefficienten van de vergelijkingen in op te slaan, en een matrix omzettings methode om de uitkomsten te vinden. Het programma print de oplossingen, indien mogelijk, als een geheel getal of als een breuk.
Deze methode heeft het voordeel boven de standaard "pivotal condesation techniqus" dat voor hele coefficienten de antwoorden precies hele getallen zijn of gedeelten daarvan.

Het voorbeeld achter het programma geeft de oplossing van de 2 vergelijkingen:

```
 a + 2b + 1 = 0
4a + 5b + 2 = 0
```

```
 10 REM SIMULTANEOUS EQUATIONS
 50 INPUT"NUMBER OF EQUATIONS="N
 60 I=N*N;J=N*(N+1)
 65 DIM AA(I),CC(J),II(N)
 70 @=0;FOR I=1TON;FOR J=1TO N+1
 80 PRINT"C("I","J")=";INPUT C
 90 CC((I-1)*(N+1)+J)=C;NEXT J;NEXT I
100 L=N+1;GOSUB c;E=D;M=1-2*(N%2)
110 PRINT"SOLUTION:"'
112 IFE<0 E=-E;M=-M
115 IF E=0;PRINT"DEGENERATE!"";END
120 FOR L=1TON;GOSUB c
125 PRINT"X("L")=  "
130 A=M*D;B=E;DO A=A%B
140 IF ABS(B)>ABS(A) THEN T=B;B=A;A=T
150 UNTIL B=0;A=ABS(A)
151 P.(M*D)/A;IF E/A<>1 PRINT"/"E/A
155 M=-M;PRINT';NEXT L;END
160cFOR I=1TON;FOR J=1TON;K=I*N-N+J
170 IFJ<L AA(K)=CC(K+I-1)
180 IF J>=L AA(K)=CC(K+I)
190 NEXT J;NEXT I
200dD=0;F=1;S=1
210 FOR J=1TON;II(J)=J;F=F*J;NEXT J
215 GOSUB f
220 FOR H=2TOF;GOSUB e;NEXT H;RETURN
230eI=N-1;J=N
240gIF II(I)>=II(I+1) I=I-1;GOTO g
250hIF II(I)>=II(J) J=J-1;GOTO h
260 GOSUB i;I=I+1;J=N;IF I=J GOTO f
270 DO GOSUB i;I=I+1;J=J-1;UNTIL I>=J
280fP=1;FOR K=1TON;P=P%AA(N*K-N+II(K))
290 NEXT K;D=D+S*P;RETURN
300iK=II(I);II(I)=II(J);II(J)=K
310 S=-S;RETURN
```

Beschrijving van het programma:

50-60	Reserveer ruimte voor de matrix
70-90	Lees de coefficienten in in de matrix
120-155	Print de oplossingen
130-150	Vindt de GCD van de oplossingen, het wordt dus geprint als de kleinst mogelijke termen
160-190	c: verwisselt termen om de volgende optelling bij de determinant te verkrijgen
280-290	f: Telt het volgende produkt op bij de determinant
300-310	i: verruilt termen in de omzetting

Voorbeeld:

```
>RUN
NUMBER OF EQUATIONS=?2
C(1,1)=?1
C(1,2)=?2
C(1,3)=?1
C(2,1)=?4
C(2,2)=?5
C(2,3)=?2
SOLUTION:
X(1)=  1/3
X(2)=  -2/3
```

Dit was het dan voor deze keer. De volgende keer gaan we verder met de word- en bytevectoren en de strings.

E.Sanders

```
**************************************
TELROUTINE VOOR DE INFOMASTER  K.RULAND
**************************************

      1 AANGEPASTE TELROUTINE VOOR
      2 INFOMASTER.
      3 KAN OOK ASCII VERWERKEN ALS
      4 FLOITING POINT GETAL.
      5 K. RULAND
      6 AUGUSTINUSSTRAAT 84
      7 6161 AM GELEEN
      8 TEL 04494-54243
      9 VERVANG REGEL 21 DOOR:
     21 P.'"7=OPTELLLEN";GOS.o;P."8=OVERZICHT"
    100 DELETE REGEL 714 EN VERDER
    101 EN ZET DEZE ROUTINE DAAR NEER :
    714GOS.690;IN.''"WELK ITEMNUMMER OPTELLEN"$S;IFL.S<1;G.a
    724J=VALS;IFJ>A;G.714
    728%@=0
    734X=0;T=RRO;DDR=T+0?J;GOS.o
    738%@=%@+VALD
    744X=X+D;T=T+B;U.T>=RR1
    748IFL?J=0;P.'"TOTAAL  "$NNJ" = ";FP.%@';G.a
    754P.'"TOTAAL  "$NNJ" = "X';G.a
```

****
ATOM STEREO GELUIDS PROCESSOR C. de MOOR
****

Deze geluids processor is zeer zeker niet de eerste maar zal dan ook wel niet de laatste zijn.
Het verschil tussen deze en andere geluids processors is dat hier de opgewekte geluiden werkelijk in stereo zijn.
De PSG (Programmable Sound Generator) kan gebouwd worden op een reeds ontwikkeld printje dat een formaat heeft van een halve eurokaart (10x8 cm).
Op deze print zit een 64 polige conector waarmee de PSG direkt achter in de bus van de ATOM gestoken kan worden, wel moet de VIA (6522 ic.1) in het daarvoor bestemde ic voetje zitten. Er wordt gebruik gemaak van de hele poort A en 5 bits van poort B.
De aansluiting naar een versterker mag nu ook geen problemen meer opleveren. Dit gebeurt rechtstreeks met een 5 polige DIN kabel die van de DIN uitgang van het PSG printje naar de tape of aux ingang gaat van een willekeurige stereo versterker.

HARDWARE BESCHRIJVING:

De hele stereo PSG draait om het bekende ic de AY-3-8910 van General Instruments. Gegevens over dit ic staan in de Acorn Nieuws van maart 1984. Het daar beschreve sound board is opgebouwd met maar een ic, maar bezit wel een eigen versterker.
De beide PSG'S worden parallel gekoppeld aan de poort-uitgangen van de VIA met uitzondering van de CS (A8 pen 25) en de drie analoge uitgangen A,B en C (resp. pen 4,3 en 38).
De data ingangen van de PSG's worden doorverbonden met poort A van de via, dus in totaal 8 stuks. Van poort B worden de eerste 3bits gebruikt voor de controle signalen van de PSG. Deze zijn resp. BC1 BC2 en BDIR. Bit 3 is vrij maar kan gebruikt worden voor een software reset van de PSG's. Bit 4 en 5 zijn de CS lijnen van de verschillende PSG's.
Verder moeten de PSG's nog van een clock signaal worden vorzien. Deze is bij de standaard print 1 MHz, het is ook mogelijk om de clock frequentie van de VDU controller (6847 ic.31) pen 33 af te takken. De frequentie bedraagt nu 3.57 MHz waardoor de nauwkeurigheid en het aantal octaven van de PSG's aanzienlijk stijgt. Voor de goede werking zijn verder nog een paar elco's, condensators en weerstanden nodig. Ook is de print voorzien van een led indicatie over de voedingsspanning.
De AY-3-8910 is voorzien van 2 I/O poorten die op de print door middel van 16 polige connectors berijkbaar zijn.
Ook bruikbaar is de AY-3-8912 deze is echter niet pin-compatible(28 pens) en heeft een I/O poort ,zodat men zelf de schakeling op een stukje experimenteerprint moet monteren of zelf een print ontwerpen.

MOGELIJKHEDEN:

De mogelijkheden lijken wel onbeperkt, men heeft de keuze uit 3 onafhankelijke VCO's en 1 ruisgenerator per kanaal (links rechts). De programmering gebeurt door middel van 15 interne registers in de PSG. Register 0 t/m 13 worden gebruikt door de

PSG zelf en de registers 14 en 15 zijn voor de I/O poorten.

Kort overzicht van de registers:
Register 0 t/m 5: bepalen de frequentie van de 3 VCO's die verdeeld is over een 8 bits register en een 4 bits register per VCO.
register 6: bepaalt de ruiskleur van de ruisgenerator met 5 bits.
Register 7: bepaalt welke VCO met de uitgang wordt door verbonden met of zonder. De laatste 2 bits zijn voor het besturen van de 2 I/O poorten.
Register 8 t/m 10: bepalen de amplitude van de VCO's met de eerste vier bits. Is het vijfde bit "1" dan wordt de amplitude bepaalt door de envelope generator.
Register 11 en 12: bepalen de frequentie van de envelope generator (dus 16 bits).
Register 13: bepaalt de envelope mode. Deze wijkt af van de in Acorn Nieuws geplaatse envelope modes. Daarom hier nog een kort overzicht van de envelope modes van register 13:

Register 14: I/O poort A.
Register 15: I/O poort B.

SOFTWARE:

De software die gebruikt wordt bij het soundboard (Acorn Nieuws maart 1984) is niet zonder meer te gebruiken. Daarom volgen hieronder 3 korte assembler listings voor het initialiseren van de VIA, het resetten van de registers van de PSG en het schrijven van data naar een bepaald register. Voor dit wegschrijven van data naar een of beide PSG's hebben we 3 variabelen nodig:

C: chip select, deze variabelebepaalt welke PSG wordt aangesproken.

R: register, deze variabele bepaalt welk register van de geselecteerde PSG's wordt aangesproken.

D: data, deze variabele bepaalt de data die naar het geselecteerde register gaat.

Hieronder volgt een listing van de gebruikte assembler routines:

```
 0 REM PSG ASSEMBLER ROUTINE
 1 REM  505 BYTES
 2 DIM LL(2),NN(2),P(-1)
 3 F.I=0TO2;NNI=-1;LLI=-1;N.
 4 P.$21;F.N=1TO2;[
 5:LLO\ PROGRAM VIA
 6 LDA#0;STA&B80C
 7 LDA#&FF;STA&BB02;STA&B803
 8 RTS
 9:LL1\ RESET PSG
10 LDA#48;STA&324;LDX-1
11:NN1\ SUBR.
12 STX&333;JSRLL2
13 LDX&333;INX;CPX#15
14 BNE NN1
15 RTS
16:NN2\ SUBR.
17 ADC&324;STA&B800;RTS
```

```
18:LL2\ PROGRAM PSG
19 LDA#333;STA#B801
20 LDA@07;JSRNN2
21 LDA@02;JSRNN2
22 LDA#325;STA#B801
23 LDA@06;JSRNN2
24 LDA@02;JSRNN2
25 RTS
26];N.;P.$6
```

Deze routine is eenvoudig aanspreekbaar vanuit basic. Zo moet bijvoorbeeld het basic programma na de assembler altijd beginnen met LINK LL0 voor het initialiseren van de VIA. Daarna moet men voor de zekerheid alle registers resetten; dit gebeurt door D=0;LINK LL1. Willen we beide kanalen tegelijk programmeren dan moet men de variabele C (chip select) de waarde 48 geven, als men maar een kanaal wil aanspreken, links of rechts, dan moet C de waarde van resp. 16 of 32 bevatten.
Nu hoeft men allen nog het te programmeren register te bepalen met de data die daar in komt te staan. Dit gebeurt door de variabelen R en D een waarde toe te kennen bijv. R=0;D=119. Hierna linkt men naar de routine LL2 en het register 0 bevat nu de waarde 119 (dit komt overeen met de noot C bij een clockfrequentie van 1 MHz). Een eenvoudig testprogramma dat na de assembler komt te staan kan er als volgt uitzien:

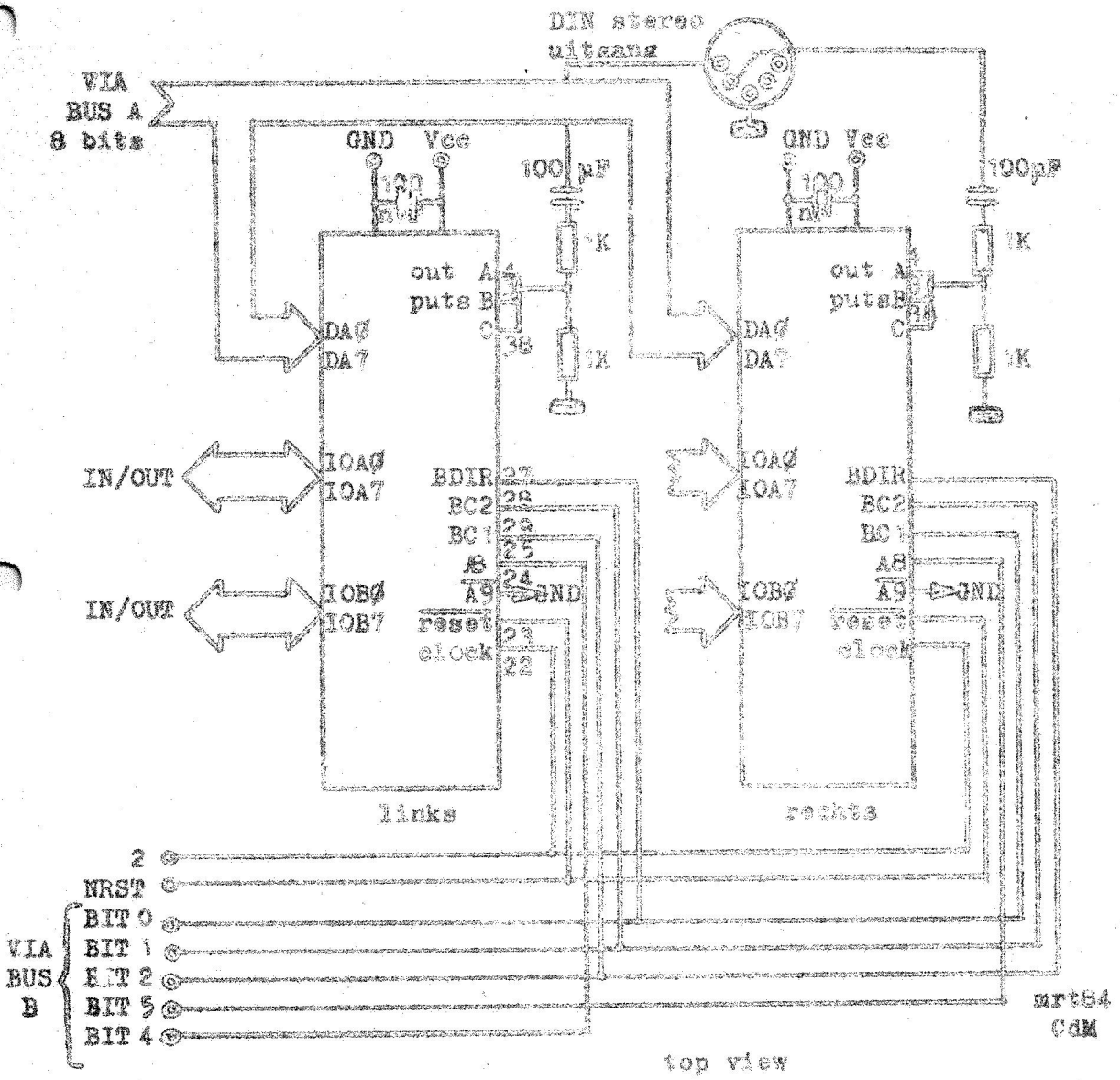
```
100 LINK LL0
101 D=0;LINKLL1
102 C=48
103 R=7;D=62;LINKLL2
104 R=8;D=15;LINKLL2
105 FOR N=1 TO 2;C=16*N
106 R=0
107 FOR D=0 TO 255;LINKLL2
108 NEXT;R=8;D=0;LINKLL2
109 NEXT;END
```

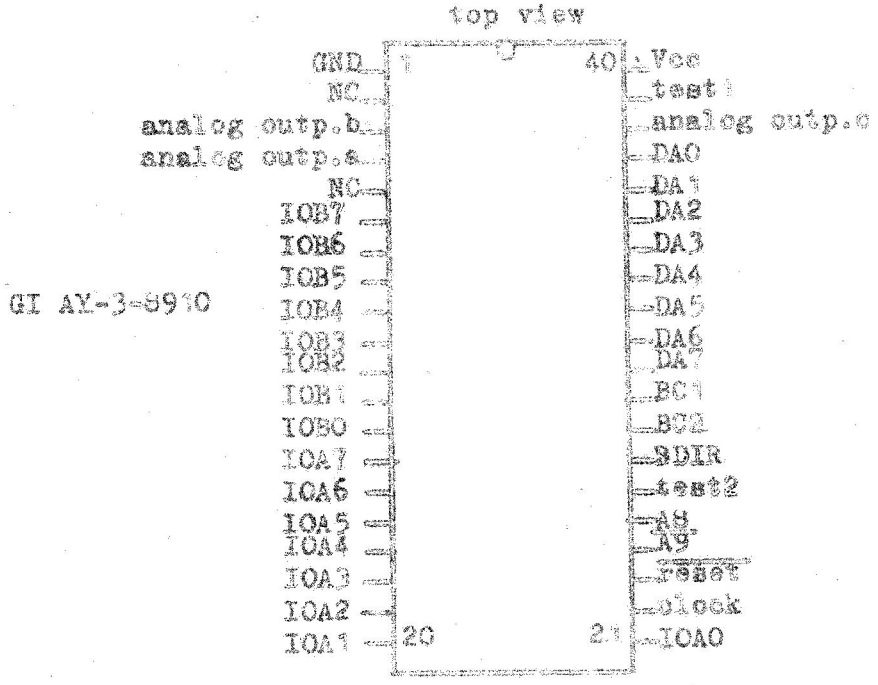
Dit programma produceert een toon met een aflopende frequentie op eerst het linker dan het rechter kanaal van de versterker.

Er zijn verder nog enkele voorbeeld programma's door mij geschreven zoals:

PIANO: maakt van uw ATOM een piano met programeerbare delay tijd.
STEREO SUNSET: dit is het bekende SUNSET programma, echter nu met geluid; en nog wel in stereo !
MANUAL: Programma met tevens een korte inleiding en programmeer instrukties voor het zelf programmeren van alle mogelijke geluiden.
Voor diegenen die interesse hebben in het printontwerp met bouwbeschrijving, een printje (zonder onderdelen) of de programma's kunnen kontakt op nemen met:

Charl de Moor
Op gen Hoes 62
Brunssum
045 - 272358 Na 18.00 uur.


DIN stereo uitgang
VIA BUS A 8 bits
GND Vcc
100 nF
1K
out A
puts B
C
DA0
DA7
IOA0
IOA7
IN/OUT
IOB0
IOB7
BDIR 27
BC2 26
BC1 25
A8
A9
GND
reset
clock
links
rechts
2
NRST
BIT 0
BIT 1
BIT 2
BIT 5
BIT 4
VIA BUS B
mrt84 CdM


top view
GI AY-3-8910
GND 1
NC
analog outp.b
analog outp.a
NC
IOB7
IOB6
IOB5
IOB4
IOB3
IOB2
IOB1
IOB0
IOA7
IOA6
IOA5
IOA4
IOA3
IOA2
IOA1 20
40 Vcc
test1
analog outp.c
DA0
DA1
DA2
DA3
DA4
DA5
DA6
DA7
BC1
BC2
BDIR
test2
A8
A9
reset
clock
21 IOA0

****
OPROEPEN B. TOSSAINT
****

747-FANS
========

Er zijn verschillende leden die veel belangstelling hebben voor de vliegsimulatie-spelen 747 en Schiphol. Met het groter worden van de geheugens,ontstaat er ruimte om een aantal elementen aan deze programma's toe te voegen, ik zelf denk bv. aan:
- meerdere vliegvelden met bijbehorende aanvliegbakens (o.a. Brussel, Maastricht);
- verbeteren van de landingsbaan.
Een deel van de benodigde documentatie is reeds in mijn bezit.
Wie doet mee?

Bruno Tossaint.

FORTH
=====

Reeds geruime tijd is deze programmeertaal voor de Acorn Atom beschikbaar.(in tenminste twee versies).
Wie heeft ervaringen cq. toepassingen????

Bruno Tossaint.

******
INHOUD
******

· SLIPPERTJES

B.C.E. 4-84 pag. 17 De betreffende wijziging werkt NIET altijd.