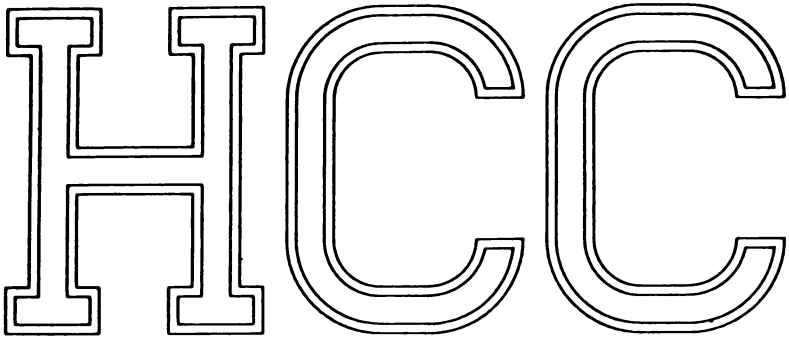


OSI Gebruikersgroep

2e OSI-UG 50 P-Boek

INHOUD

Voor vragen en/of opmerkingen:
Jos Burghouts
Montgomerylaan 112
2625 PR Delft
Tel. 015 - 56 52 91

2e OSI - UG 50 P-boek
een uitgave van Hobby Computer Club Nederland
Postbus 149 - 2250 AC Voorschoten

1e druk - april 1981 - oplage 200 ex.

```
  5 REM INITIALISERING
 10 MPROG=200:MLINE=40:MTABLE=100:MVAR=40:MIN=40
 20 DIM IN(MIN+3),T1(MLINE-1),T2(MVAR-1),PROG (MPROG),TABLE(MTABLE-1)
 25 NPROG=0:NVAR=0:RFLAG=0:GNVAR=0:EFLAG=0:GOTO5000
 29 REM DELETE SPATIES
 30 IFIN(I)=32THENI=I+1:GOTO30
 40 RETURN
 50 GOSUB80:N=-N:IFN)=0THENEFLAG=1:RETURN
 60 IFPNT=MLINETHENEFLAG=2:RETURN
 70 T1(PNT)=N:PNT=PNT+1:RETURN
 79 REM ASCII ) GETAL
 80 N=0:N1=1:IFIN(I)=45THENN1=-1:I=I+1
 90 N2=IN(I)-48:IFN2(00RN2)90RN*10+N2)9999THENN=N*N1:GOTO30
100 N=N*10+N2:I=I+1:GOTO90
109 REM CONVERTEER ASCII STRING NAAR GETAL
110 INPUT"";A$:IFMIN(LEN(A$)THENPRINT"INPUT REGEL TE LANG !":GOTO110
114 IN(0)=0:I=0
115 IFLEN(A$)THENFORA=1TOLEN(A$):IN(A-1)=ASC(MID$(A$,A,1)):NEXTA
116 IFLEN(A$)THENIN(A-1)=0
117 RETURN
120 VPNT=0:IFIN(I)(65ORIN(I))90THENEFLAG=3:RETURN
130 A=IN(I):IFA=00RA=430RA=450RA=600RA=610RA=59THEN150
131 IFA=84ANDIN(I+1)=72ANDIN(I+2)=69ANDIN(I+3)=78THEN150
140 T2(VPNT)=A:I=I+1:VPNT=VPNT+1:IFVPNT(MVARTHEN130
150 IFVPNT=0THENIN(I)=0:GOTO120
155 IFT2(VPNT-1)=32THENVPNT=VPNT-1:GOTO155
160 FORA=1TONVAR-1:IFTABLE(A)()VPNTTHEN190
170 FORB=1TOVPNT:IFTABLE(A+B)()T2(B-1)THEN190
180 NEXTB:N=A+9999:GOSUB60:IFEFLAGTHEN RETURN
181 RETURN
190 A=A+TABLE(A)+1:NEXTA
200 IFNVAR+NPNT)MTABLE-2THENEFLAG=4:RETURN
210 TABLE(NVAR)=0:TABLE(NVAR+1)=VPNT:FORA=0TOVPNT-1
211 TABLE(NVAR+A+2)=T2(A)
215 NEXTA:N=NVAR+10000:NVAR=NVAR+VPNT+2:GOSUB60:IFEFLAGTHENRETURN
217 GOTO30
220 IFIN(I)=45ORIN(I))47ANDIN(I)(58THENGOSUB80:GOSUB60:GOTO240
230 GOSUB120
240 IFEFLAGTHENRETURN
245 A=IN(I):N=-(A=43)-2*(A=45)-3*(A=60)-4*(A=61):IFN=0THENRETURN
250 GOSUB60:IFEFLAGTHENRETURN
255 I=I+1:GOSUB30:GOTO220
260 GOSUB120:IFEFLAGTHENRETURN
265 IFIN(I)()59THEN30
270 I=I+1:GOSUB30:GOTO260
280 GOSUB220:IFEFLAGTHENRETURN
285 IFIN(I)()59THEN30
290 I=I+1:GOSUB30:N=-10001:GOSUB60:IFEFLAGTHENRETURN
295 GOTO280
300 IFIN(I)()61THENEFLAG=5:RETURN
310 I=I+1:GOTO30
320 IFIN(I)()84ORIN(I+1)()72ORIN(I+2)()69ORIN(I+3)()78THENEFL=6:RETURN
325 I=I+4:GOSUB30:N=-10001:GOTO60

blz. 1
```

```
 330 EFLAG=99:RETURN
 340 PA=-10000-(NPROG=0):FORA=0TONPROG-1:IFPA()-10000THEN400
 350 A=A+1:IFPROG(A)=4THENIFPROG(A+1))=0THEN390
 360 IFPROG(A)()5THEN400
 370 A=A+1:IFPROG(A)()-10001THEN370
 380 GOTO350
 390 PROG(A+1)=PROG(PROG(A+1))
 400 PA=PROG(A):NEXTA:RFLAG=0:RETURN
 410 PA=-10000-(NPROG=0):FORA=0TONPROG-1:IFPA()-10000THEN500
 415 L=A
 420 A=A+1:IFPROG(A)=4THENIFPROG(A+1)(0THEN460
 430 IFPROG(A,()5THEN500
 440 A=A+1:IFPROG(A)()-10001THEN440
 450 GOTO420
 460 C=PROG(A+1):PA=-10000:FORB=0TONPROG-1
 465 IFPA=-10000ANDPROG(B)=CTHEN490
 470 PA=PROG(B):NEXTB:IFRFLAG=1THENEFLAG=10:RETURN
 480 GOTO500
 490 PROG(A+1)=B
 500 PA=PROG(A):NEXTA:RETURN
 510 GOSUB590:N=A
 520 B=PROG(PNT):IFB(0THENRETURN
 530 PNT=PNT+1:GOSUB590:ONBGOSUB550,560,570,580
 535 IFABS(N)(10000THEN520
 540 EFLAG=12:RETURN
 550 N=N+A:RETURN
 560 N=N-A:RETURN
 570 N=N(A:RETURN
 580 N=N=A:RETURN
 590 A=PROG(PNT):PNT=PNT+1:IFA(10000THENRETURN
 600 A=TABLE(A-10000):RETURN
 610 PNT=PNT+1
 620 LINE=PNT:PNT=PNT+2:IFLINE=NPROGTHEN5000
 630 ONPROG(PNT-1)GOTO670,680,740,760
 640 GOSUB510:IF EFLAG THEN6000
 641 PNT=PNT+2:IFNTHEN630
 650 PNT=PNT+1:IFPROG(PNT-1)()-10000THEN650
 660 GOTO620
 670 L=PROG(PNT)-10000:PNT=PNT+1:GOSUB510:IF EFLAGTHEN6000
 675 TABLE(L)=N:GOTO610
 680 GOSUB110:IFIN(I)=83THENEFLAG=13:GOTO6000
 690 IFIN(I)=0THEN680
 700 IFIN(I)()45AND(IN(I)(48ORIN(I))57)THENPRINT"FOUTE INPUT":GOTO680
 710 GOSUB80:TABLE(PROG(PNT)-10000)=N:PNT=PNT+1
 711 IFPROG(PNT)=-10000THEN610
 720 IFIN(I)()59THENPRINT"MEER":GOTO680
 730 I=I+1:GOTO690
 740 GOSUB510:PRINTN;:IFPROG(PNT)=-10000THENPRINT:GOTO610
 750 PNT=PNT+1:GOTO740
 760 PNT=PROG(PNT):GOTO620
4999 REM HAAL EEN REGEL EN VERWERK DIE
5000 NVAR=QNVAR:PRINT"-";:GOSUB110:IFIN(I)(48ORIN(I))57THEN5200
5010 IFRFLAGTHENGOSUB340
5020 PNT=0:GOSUB50:IFEFLAGTHEN6000
5025 IFIN(I)=0THENPNT=0:GOTO5090
5030 RESTORE
```

```
5040 READL:FORA=0TOL-1:READB:IFIN(I+A)=BTHENNEXTA:GOTO5070
5050 READL:IFL=255THEN5040
5060 IFLTHEN5050
5070 I=I+L:GOSUB30:READN:GOSUB60:IFEFLAGTHEN6000
5080 READN:IFN=255THEN5085
5081 ONNGOSUB120,220,260,280,300,50,320,330
5082 IFEFLAG=99THENEFLAG=0:GOTO5030
5083 IFEFLAGTHEN6000
5084 GOTO5080
5085 N=-10000:GOSUB60:IFEFLAGTHEN6000
5087 IFIN(I)THENEFLAG=7:GOTO6000
5090 A=0:PA=-10000:L=PNT:IFLAND NPROG=0THEN5160
5100 FORA=0TONPROG-1:IFPA=-10000ANDPROG(A)(=T1(0)THEN5120
5110 PA=PROG(A):NEXTA:PROG(A)=0
5120 IFPROG(A)=T1(0)THEN5170
5130 IFPNT=0THENEFLAG=8:GOTO6000
5140 IFNPROG+L)MPROGTHENEFLAG=9:GOTO6000
5150 IFLTHENFORB=NPROG-1TOASTEP-1:PROG(B+L)=PROG(B):NEXTB
5160 NPROG=NPROG+L:QNVAR=NVAR
5165 IFPNTTHENFORB=0TOPNT-1:PROG(A+B)=T1(B):NEXTB
5167 GOTO5000
5170 FORB=1TONPROG:IFPROG(A+B-1)()-10000THENNEXTB
5180 IFB(=PNTTHENL=PNT-B:GOTO5140
5190 FORL=A+BTONPROG-1:PROG(L-B+PNT)=PROG(L):NEXTL
5195 L=PNT-B:GOTO5160
5200 IFIN(I)()76THEN5420
5205 IFNPROG=0THEN5000
5210 PRINTCHR$(3);:FORA=0TONPROG-1
5215 PRINT-PROG(A);" ";:A=A+1
5220 RESTORE
5230 READL:FORB=1TOL:READT1(B):NEXTB:READB:IFB=PROG(A)THEN5260
5240 READL:IFL()255ANDL()0  THEN5240
5250 IFLTHEN5230
5255 READB
5260 IFLTHENFORB=1TOL:PRINTCHR$(T1(B));:NEXTB:PRINT" ";
5265 A=A+1
5270 READL:IFL=255THEN5280
5275 ONLGOSUB5290,5300,5340,5360,5380,5390,5410,5415
5276 IFEFLAG=99THENEFLAG=0:GOTO5220
5277 GOTO5270
5280 PRINT:NEXTA:GOTO5000
5290 L=PROG(A)-9999:FORB=1TOTABLE(L):PRINTCHR$(TABLE(B+L));:NEXTB
5291 A=A+1:RETURN
5300 IFPROG(A))9999THENGOSUB5290:GOTO5320
5310 PRINTPROG(A);:A=A+1
5320 IFPROG(A)(0THENRETURN
5330 PRINTMID$("+-(=",PROG(A),1);:A=A+1:GOTO5300
5340 GOSUB5290:IFPROG(A)()-10000THENPRINT";";:GOTO5340
5350 RETURN
5360 GOSUB5300:IFPROG(A)()-10000THENPRINT";";:A=A+1:GOTO5360
5370 RETURN
5380 PRINT"=";:RETURN
5390 IFPROG(A)(0THENPRINT-PROG(A);:A=A+1:RETURN
5400 PRINT-PROG(PROG(A));:A=A+1:RETURN
5410 A=A+1:PRINT"THEN ";:RETURN
5415 EFLAG=99:RETURN
```

```
5420 IFIN(I)()82THENEFLAG=11:GOTO6000
5430 A=IN(I+1):IFA()69THEN5460
5440 RFLAG=2:GOSUB410:L=-10:PA=-10000
5445 FORA=0TONPROG-1:IFPA=-10000THENPROG(A)=L:L=L-10
5450 PA=PROG(A):NEXTA:GOTO5000
5460 IFA()85THENIN(I)=0:GOTO5420
5465 PRINTCHR$(3);
5470 RFLAG=1:GOSUB410:IFEFLAGTHEN6000
5472 PNT=0:GOTO620
5999 END
6000 HULP=EFLAG:EFLAG=0
6005 IFHULP)10THEN6009
6007 ONHULPGOTO6010,6020,6030,6040,6050,6060,6070,6080,6090,6100
6009 ONHULP-10GOTO6110,6120,6130
6010 PRINT"FOUTIEF REGELNUMMER":GOTO5000
6020 PRINT"REGEL TE LANG":GOTO5000
6030 PRINT"ILL. VARIABELE NAAM":GOTO5000
6040 PRINT"TE VEEL VARIABELEN":GOTO5000
6050 PRINT"= VERWACHT":GOTO5000
6060 PRINT"THEN VERWACHT":GOTO5000
6070 PRINT"EINDE REGEL VERWACHT":GOTO5000
6080 PRINT"REGEL BESTAAT NIET":GOTO5000
6090 PRINT"PROGRAMMA TE GROOT":GOTO5000
6100 PRINT"NIET GEDEFINIEERDE REGEL IN REGEL";-PROG(L):GOTO5000
6110 PRINT"ILLEGAAL COMMANDO":GOTO5000
6120 PRINT"OVERFLOW IN REGEL";-PROG(LINE):GOTO5000
6130 PRINT"GESTOPT IN REGEL";-PROG(LINE):GOTO5000
7000 END
9000 DATA5,73,78,80,85,84,2,3,255
9010 DATA5,80,82,73,78,84,3,4,255
9020 DATA4,71,79,84,79,4,6,255
9030 DATA5,71,79,32,84,79,4,6,255
9040 DATA2,73,70,5,2,7,8,0
9050 DATA 1,1,5,2,255
```

```
  1 REM-----DAVID E. TEWKSBARY
  2 REM-----BATTLESHIP
  3 INPUT"Uitleg";Y$
  4 IFLEFT$(Y$,1)="J"THEN GOSUB10000
  5 FORM=1TO10:L1$(M)=CHR$(M+64):NEXTM
 10 INPUT"Geef een getal";J$:J=ASC(J$):FORI=2TO5
 20 K=K+1:PRINT"IK DENK NU NA -
 30 L=INT(2*RND(J))+1:GOSUB2000:GOSUB4000:GOSUB5000:IFJ4=1GOTO30
 40 IFK=2GOTO20
 50 NEXTI:FORM=1TO10:FORN=1TO10:C(M,N)=B(M,N):B(M,N)=0:NEXTN:NEXTM
 70 K=0:FORI=2TO5
 80 PRINT:K=K+1:M=K:GOSUB6000:OB$=O$:GOTO100
 85 PRINT:PRINT"*** FOUT ***":GOTO100
 90 PRINT:PRINT"Geen ruimte op dat punt":PRINT"en in die richting !
100 PRINT"Geef voor schip    ";OB$;":"
101 INPUT"rij, kolom, richt.";O$,N,L
105 GOSUB5000:IFN(10RN)10ORM=0GOTO85
110 GOSUB2000:GOSUB3000:IFJ4=1GOTO90
120 IFK=2GOTO80
125 NEXTI:INPUT"zal ik eerst";Y$:IFLEFT$(Y$,1)="N"GOTO200
140 GOSUB4000:IFCS(M,N)()0GOTO140
150 J4=B(M,N):B(M,N)=0:CS(M,N)=1:GOSUB6000
155 PRINT:PRINT"Computer vuurt ";O$;N:GOSUB5000:IFJ4=0GOTO190
160 M=J4:GOSUB6000:PRINT"RAAK op ";O$:GOSUB9000:IFSB=0THENPRINT"BLUF
170 J3=J3+1:IFJ3=17GOTO9500
180 GOTO200
190 PRINT"PLONS":GOTO200
195 PRINT"ONGELDIG
200 PRINT:INPUT"Uw schot";O$,N:IFO$="Z"THENGOSUB20000:GOTO200
201 GOSUB5000:IFN(10RN)10ORM=0GOTO195
202 IFB2(M,N)()0GOTO195
205 IFC(M,N)=0THENB2(M,N)=1:GOTO230
206 B2(M,N)=2
210 J5=C(M,N):C(M,N)=0:M=J5:GOSUB6000:PRINT"RAAK op ";O$:GOSUB8000
211 J6=J6+1:IFJ6=17GOTO9550
213 IFSC=0THENPRINT"BLUF...BLUF...BLUF
215 IFJ4=0GOTO140
220 GOTO240
230 PRINT"PLONS":IFJ4=0GOTO140
240 FORM=1TO10:FORN=1TO10:IFB(M,N)=J4GOTO150
250 NEXTN:NEXTM:GOTO140
2000 LM=0:LN=0:IFL=1THENLN=1
2020 IFL()1THENLM=1
2030 RETURN
3000 J4=0:FORKP=1TOI:IFM)10RN)10GOTO3100
3010 IFB(M,N)()0GOTO3100
3040 B(M,N)=K:M=M+LM:N=N+LN:NEXTKP:GOTO3200
3100 J4=1:IFKP=1GOTO3200
3101 FORKU=1TOKP-1
3110 M=M-LM:N=N-LN:B(M,N)=0:NEXTKU
3200 RETURN
4000 M=INT(9*RND(J))+1:N=INT(9*RND(J))+1:RETURN
5000 M=0:FORK1=1TO10:IFO$=L1$(K1)THENM=K1
```

```
 5010 NEXTK1:RETURN
 6000 FORK1=1TO10:IFM=K1THENO$=L1$(K1)
 6010 NEXTK1:IFM=0THENO$=" "
 6020 RETURN
 8000 SC=0:FORM=1TO10:FORN=1TO10:IFC(M,N)=J5THENSC=1
 8010 NEXTN:NEXTM:RETURN
 9000 SB=0:FORM=1TO10:FORN=1TO10:IFJ4=B(M,N)THENSB=1
 9010 NEXTN:NEXTM:RETURN
 9500 PRINT:PRINT"EINDE SPEL":PRINT"Volgende keer beter !":GOTO9990
 9550 PRINT:PRINT"EINDE SPEL":PRINT"...en ik verlies niet ":PRINT"vaak
 9990 PRINT:INPUT"nog eens";Y$:IFLEFT$(Y$,1)="J"THENRUN
 9999 GOTO49999
10000 PRINT:PRINT"Dit is het spel ZEESLAG  "
10010 PRINT" U moet de volgende":PRINT"schepen op een 10 x 10"
10011 PRINT"rooster plaatsen door":PRINT"het opgeven van RIJ
10020 PRINT"(A-J),KOLOM(1-10) en":PRINT"richting(1=hor, 2=ver)
10035 PRINT
10040 PRINT"SCHIP A--2 EENHEDEN":PRINT"SCHIP B--3 EENHEDEN"
10050 PRINT"SCHIP C--3 EENHEDEN":PRINT"SCHIP D--4 EENHEDEN"
10060 PRINT"SCHIP E--5 EENHEDEN"
10061 PRINT:INPUT"doorgaan";Y$:PRINT
10070 PRINT" Dan vuurt U schoten op"
10071 PRINT"de vloot van de computer":PRINT"door het intikken van"
10072 PRINT"de coordinaten, b.v. B,3
10080 PRINT"tik Z,1 i.p.v. een schot":PRINT"om Uw vloot te zien.
10090 PRINT"Om te zien hoe ik ervoor":PRINT"sta tik Z,1.
10110 PRINT:PRINT" Het spel is over als"
10111 PRINT"een vloot is vernietigd.":PRINT
10120 PRINT:PRINT
11000 RETURN
20000 PRINT:IFN=2THENPRINT"   SITUATIE COMPUTER":PRINT:GOTO20010
20005 PRINT"    EIGEN SITUATIE":PRINT
20010 PRINT"  1 2 3 4 5 6 7 8 9 10"
20015 PRINT"  ----------------------"
20020 FORI=1TO10:PRINTCHR$(I+64)+")";:FORI2=1TO10
20025 IFN=2GOTO21000
20030 M=B(I,I2):GOSUB6000:PRINTO$+" ";
20050 NEXTI2:PRINT:NEXTI
20100 RETURN
21000 IFB2(I,I2)=0THENO$=" ":GOTO21030
21010 IFB2(I,I2)=1THENO$="M":GOTO21030
21020 O$="H"
21030 PRINTO$+" ";:GOTO20050
```

```
 20 DIMC(13,4):FORI=1TO30:PRINT:NEXT
 21 POKE530,1:REM      ^C
 22 PRINT"INSTRUKTIES:":PRINT
 25 PRINT"1 voor een nieuwe kaart":PRINT"2 om te verdubbelen"
 30 PRINT"3 om te passen":FORI=1TO4:S(I)=228+I:NEXT:S(0)=63
 50 N=1
 60 P(1)=53320:P(2)=P(1)+13
 70 FORI=1TO13:FORJ=1TO4:C(I,J)=0:NEXTJ,I
 80 FORI=1TON+1:FORJ=1TO5:PC(I,J)=0:NEXTJ:SW(I)=0:NEXTI
100 PRINT:PRINT"Hoeveel zet U in";
101 INPUTW(1):IFW(1)<0ORW(1)>500GOTO100
110 FORI1=1TO30:PRINT:NEXT:POKE54117,32
119 I1=1
120 FORQ=1TO2
130 GOSUB1100:LO=P(Q):IFI1=1ANDQ=N+1THENS=1
140 PC(Q,I1)=IC:GOSUB1000:P(Q)=P(Q)+6*32:NEXTQ
150 S=0:I1=I1+1:IFI1<3GOTO120
160 S=0:FORQ=1TON+1:GOSUB1200:IFT=21THENGOSUB1300
170 NEXT:IFS=1GOTO690
180 Q=1:I1=3
185 L$="SPELER ??":GOSUB1310
190 POKE57088,127:C=PEEK(57088)
200 IFC=127ORC=223GOTO230
210 IFC=191THENW(Q)=W(Q)*2:GOTO230
225 GOTO190
230 L$="                    ":GOSUB1310
235 IFC=223GOTO300
240 GOSUB1100:PC(Q,I1)=IC:LO=P(Q):GOSUB1000:P(Q)=P(Q)+6*32
260 GOSUB1200:IFT>21THENL$="DOOD !!!":GOSUB1310:GOTO300
265 IFI1=5THENSW(Q)=1:GOTO700
270 I1=I1+1:GOTO185
300 I1=3:Q=1:GOSUB1200:PT=T:Q=2:GOSUB1200
305 LO=PZ:C$=PZ$+" ":S=0:IS=PS:GOSUB1000
310 IFPT>21ORPT<T+1GOTO690
320 IFI1=6THENSW(N+1)=1:GOTO700
330 GOSUB1100:PC(Q,I1)=IC:LO=P(Q):GOSUB1000:P(Q)=P(Q)+6*32
340 GOSUB1200:IFT<22THENI1=I1+1:GOTO310
350 L$="DOOD !!!":GOSUB1310
690 FORI=1TO2500:NEXT
700 FORI=53300TO54300:POKEI,32:NEXT
710 Q=1:GOSUB1200:PT=T:Q=2:GOSUB1200
720 IFPT>21THENPRINT"Het huis wint.":H(1)=H(1)-W(1):GOTO790
721 IFPT<T+1ANDT<22THENPT=22:GOTO720
730 IFSW(2)=1THENPT=22:GOTO720
740 PRINT"Speler wint!":H(1)=H(1)+W(1)
790 PRINT:PRINT:PRINT
800 Q=N+1:GOSUB1200:PRINT"Het huis had"T:PRINT:PRINT
810 Q=1:GOSUB1200
820 PRINT"Speler had"T","
821 PRINT"Uw inzet was"W(Q)",":PRINT"U hebt nu:":PRINT"$"H(Q)
830 PRINT:PRINT:INPUT"Doorgaan";Y$
840 IFLEFT$(Y$,1)="J"GOTO60
850 GOTO49999
```

```
 1000 FORI=LOTOLO+2:POKEI,128:POKEI+160,135:NEXT
 1010 FORI=LO+31TOLO+150STEP32:POKEI,143:POKEI+4,136:NEXT
 1020 IFS=1THENPZ=LO:PZ$=C$:PS=IS:IS=0:C$="??"
 1030 POKELO+32,S(IS)
 1031 POKELO+130,S(IS)
 1040 FORI=1TOLEN(C$):POKELO+I*32+33,ASC(MID$(C$,I)):NEXT
 1050 RETURN
 1100 IC=INT(RND(1)*13)+1:IS=INT(RND(1)*4)+1
 1110 IFC(IC,IS)=1GOTO1100
 1120 C(IC,IS)=1:C$=MID$(STR$(IC),2):IFIC=1THENC$="AC"
 1130 IFIC=11THENC$="JK"
 1140 IFIC=12THENC$="QU"
 1150 IFIC=13THENC$="KI"
 1160 IFIC)10THENIC=10
 1170 RETURN
 1200 IC=0:T=0:FORI=1TO5:IFPC(Q,I)=1THENIC=IC+1
 1210 T=T+PC(Q,I):NEXT
 1220 IFIC=0THENRETURN
 1230 FORI=1TOIC:IFT+10)21THENRETURN
 1240 T=T+10:NEXT:RETURN
 1300 L$="*** 21 ***":SW(Q)=1:S=1
 1310 FORI=1TOLEN(L$):POKEP(Q)+I-4,ASC(MID$(L$,I)):NEXT:RETURN
49999 END
```

BOMBER (1 K VIDEO)

```
  1 REM***DAVID E. TEWKSBARY***
  2 FORI=1TO30:PRINT:NEXT
  3 POKE530,1:REM  ^C
  5 INPUT"UITLEG NODIG";Y$:IFLEFT$(Y$,1)="N"GOTO15
  6 PRINT:PRINT"keuzes  mogelijk    :":PRINT:PRINT"1 - KLIMMEN
  7 PRINT"2 - DUIKEN":PRINT"3 - KANON":PRINT"4 - BOMMEN"
  8 PRINT:PRINT"VLIEGTUIGEN 15 PUNTEN,":PRINT"GROND DOELEN 75."
  9 PRINT:PRINT"JA, U KUNT VLIEGTUIGEN":PRINT"BOMBARDEREN!"
 10 PRINT:PRINT"OM TE BEGINNEN TIK(1-5)
 15 INPUT"(SNELHEID VIJAND)";SP:FORH=1TO30:PRINT:NEXTH
 20 FORH=54123TO54147STEP10:POKEH,14:POKEH-5,15:NEXT:POKE54117,32
 21 FORH=54145TO54179:POKEH,155:NEXT
 25 L=SP+2.1:J=99
 26 V(1)=53733:V(2)=53541
 27 V(3)=53844:V(4)=53652
 28 B5=53466
 30 FORH=1TO2:V2=0
 40 IFH=2ORABS(V(H+1)-V(H)))50GOTO60
 45 IFV(H+1)(V(H)GOTO80
 50 GOTO90
 60 K=INT(4*RND(J))+1:ONKGOTO70,100,100,100
 70 K=INT(RND(1)*2)+1:ONKGOTO80,90
 80 V2=32:GOTO100
 90 V2=-32
100 IFV(H))53990THENV2=-32
110 IFV(H)+V2(53300THENV2=32
120 POKEV(H),32:POKEV(H+2),32
125 V(H)=V(H)+V2+SP+2:V(H+2)=V(H+2)+V2+SP+2
130 POKEV(H),237:POKEV(H+2),237:GOSUB1800
135 NEXTH
140 FORH=1TO4:IFS(H)=1GOTO170
150 K=INT(3*RND(J))+1:ONKGOTO160,200,200
160 SV(H)=V(H)-1:S(H)=1
170 POKESV(H),32:SV(H)=SV(H)+SP+5
180 IFSV(H)/32-INT(SV(H)/32)(L/20THENS(H)=0
190 IFS(H)=1THENPOKESV(H),58:GOSUB1500
200 NEXTH
220 POKEB5,32:B5=B5-1
225 POKE57088,127
227 A=PEEK(57088)
230 IFA=127ANDB5)53325THENB5=B5-32
231 IFA=191ANDB5(54060THENB5=B5+32
232 IFA=223ANDSB=0THENSB=1:BV=B5+1
233 IFA=239ANDBB=0THENBB=1:B2=B5+1
234 IFB5/32-INT(B5/32)(.17THENB5=B5+23
240 POKEB5,239:IFSB=0GOTO280
250 POKEBV,32:BV=BV-5:GOSUB1000
260 IFABS(INT(BV/64)-BV/64)).85THENSB=0
270 IFSB=1THENPOKEBV,58
280 IFBB=0GOTO30
290 POKEB2,32:B2=B2+31:GOSUB1300
295 IFINT(B2/64)=B2/64THENBB=0
300 IFBB=1THENPOKEB2,46
```

```
 310 IFB2(54120GOTO30
 311 IFB2/32-INT(B2/32)).15GOTO30
 320 BB=0:FORN=96TO32STEP-1:POKEB2,N:NEXTN
 330 GOTO30
1000 FORH=1TO4:FORM=-32TO32STEP32
1100 IFABS(BV-M-V(H)+L/2)(LTHENSB=0:POKEBV,32:GOSUB2000
1200 NEXTM:NEXTH:RETURN
1300 FORH=1TO4:FORM=-32TO32STEP32
1400 IFABS(B2-M-V(H))(2.1THENBB=0:POKEB2,32:GOSUB2000
1450 NEXTM:NEXTH:RETURN
1500 FORM=-32TO32STEP32
1600 IFABS(B5-M-SV(H)+L/2)(LGOTO9000
1700 NEXTM:RETURN
1800 FORM=-32TO32STEP32
1850 IFABS(B5-M-V(H)+.2)(LGOTO9000
1860 IFABS(B5-M-V(H+2)+L/2)(LGOTO9000
1900 NEXTM:RETURN
2000 POKEV(H)-2,83:POKEV(H)-1,67:POKEV(H),79:POKEV(H)+1,82
2100 POKEV(H)+2,69:Z=Z+15
2200 FORN=1TO500:NEXTN
2300 FORN=-2TO2:POKEV(H)+N,32:NEXTN:FORN=-32TO32
2400 IFABS(INT((V(H)+N)/32)-(V(H)+N)/32)(.01THENV(H)=V(H)+N
2500 NEXTN:RETURN
9000 FORH=54123TO54147STEP5:IFPEEK(H)=32THENZ=Z+75
9005 NEXTH
9006 FORH=54145TO54179:POKEH,32:NEXT
9007 FORI=1TO25:PRINT:NEXT
9010 IFZ=0GOTO9400
9100 PRINT:PRINT:PRINT"GEFELICITEERD  !!!"
9200 PRINT:PRINT"U KREEG";Z;PRINT"PUNTEN OP UW WEG"
9300 PRINT"NAAR DE STATISTIEKEN"
9301 GOTO9500
9400 PRINT:PRINT:PRINT"U STIERF VOOR NIETS !"
9500 PRINT:INPUT"NOG EENS";Y$
9550 IFLEFT$(Y$,1)="J"THENRUN15
9600 PRINT:PRINT:PRINT"LAFAARD !!":PRINT
9999 END
```

blz. 10

BOTER KAAS EN EIEREN (1K VIDEO)

```
  1 REM "TIC-TAC-TOE" UIT BASIC COMPUTER GAMES,PAG 172.
  2 REM VOOR SUPERB. BEWERKT DOOR JOS BURGHOUTS,015-565291
  3 A$(1)=" 1 2 3":A$(2)=" 4 5 6":A$(3)=" 7 8 9"
  5 WS=0:WK=0:WG=0
  6 PRINTCHR$(3):PRINT:PRINT:PRINT"BOTER-KAAS-EIEREN":PRINT
  9 PRINT"DE NUMMERS V/H BORD":PRINT:
 10 PRINT A$(1)
 12 PRINT:PRINT A$(2)
 14 PRINT:PRINT A$(3):PRINT
 20 DIM S(9)
 30 PRINT"ALS JE JE GEWONNEN WILT ":PRINT"GEVEN TYPE DAN EEN 'O'
 40 PRINT:PRINT"WIL JY BEGINNEN";:INPUT C$
 45 FOR I=1TO9:S(I)=0:NEXT I
 50 IFLEFT$(C$,1)="J"THEN 475
 60 P$="O":Q$="X"
 70 GOSUB1000
100 G=-1:H=1:IFS(5)()0THEN 103
102 S(5)=-1:GOTO195
103 IFS(5)()1THEN 106
104 IFS(1)()0THEN 110
105 S(1)=-1:GOTO 195
106 IFS(2)=1ANDS(1)=0THEN181
107 IFS(4)=1ANDS(1)=0THEN181
108 IFS(6)=1ANDS(9)=0THEN189
109 IFS(8)=1ANDS(9)=0THEN189
110 IFG=1THEN112
111 GOTO118
112 J=3*INT((M-1)/3)+1
113 IF3*INT((M-1)/3)+1=MTHEN K=1
114 IF3*INT((M-1)/3)+2=MTHEN K=2
115 IF3*INT((M-1)/3)+3=MTHEN K=3
116 GOTO 120
118 FORJ=1TO7STEP3:FORK=1TO3
120 IFS(J)()GTHEN130
122 IFS(J+2)()GTHEN135
126 IFS(J+1)()0THEN150
128 S(J+1)=-1:GOTO 195
130 IFS(J)=HTHEN150
131 IFS(J+2)()GTHEN150
132 IFS(J+1)()GTHEN150
133 S(J)=-1:GOTO 195
135 IFS(J+2)()0THEN 150
136 IFS(J+1)()GTHEN 150
138 S(J+2)=-1:GOTO 195
150 IFS(K)()GTHEN160
152 IFS(K+6)()GTHEN165
156 IFS(K+3)()0THEN 170
158 S(K+3)=-1:GOTO195
160 IFS(K)=HTHEN 170
161 IFS(K+6)()GTHEN170
162 IFS(K+3)()GTHEN170
163 S(K)=-1:GOTO195
165 IFS(K+6)()0THEN170
```

```
 166 IFS(K+3)()GTHEN170
 168 S(K+6)=-1:GOTO195
 170 GOTO 450
 171 IFS(3)=GANDS(7)=0THEN187
 172 IFS(9)=GANDS(1)=0THEN181
 173 IFS(7)=GANDS(3)=0THEN 183
 174 IFS(9)=0ANDS(1)=GTHEN189
 175 IFG=-1THENG=1:H=-1:GOTO110
 176 IFS(9)=1ANDS(3)=0THEN182
 177 FORI=2TO9:IFS(I)()0THEN179
 178 S(I)=-1:GOTO195
 179 NEXT I
 181 S(1)=-1:GOTO195
 182 IFS(1)=1THEN 177
 183 S(3)=-1:GOTO195
 187 S(7)=-1:GOTO195
 189 S(9)=-1
 195 PRINT:PRINT"DE ZET V/D KOMPUTER":PRINT
 200 BT=1
 202 GOSUB 1000
 203 GOTO 1200
 205 GOTO 500
 450 IFG=1THEN465
 455 IFJ=7ANDK=3THEN465
 460 NEXTK,J
 465 IFS(5)=GTHEN171
 467 GOTO175
 475 P$="X":Q$="O"
 480 GOSUB1000
 500 PRINT:INPUT"JOUW ZET";M
 502 IF M=0 THEN 1500
 503 IFM)9THEN506
 505 IFS(M)=0THEN 510
 506 PRINTCHR$(3)"DAT VELD IS BEZET !":PRINT:GOTO500
 510 G=1:S(M)=1
 515 BT=2
 520 GOSUB 1000
 525 GOTO 1200
 530 GOTO 100
1000 POKE546,15:          :POKE548,10:PRINTCHR$(2);
1010 PRINT:FORI=1TO9:PRINT" ";:IFS(I)()-1THEN1014
1012 PRINTQ$" ";:GOTO1020
1014 IFS(I)()0THEN1018
1016 PRINT"  ";:GOTO1020
1018 PRINTP$" ";
1020 IFI()3ANDI()6THEN1050
1030 PRINT:PRINT"---+---+---"
1040 GOTO1080
1050 IFI=9THEN1080
1060 PRINT"!";
1080 NEXTI:PRINT:PRINT:PRINT
1090 POKE546,5:POKE548,22:PRINTCHR$(3);
1100 RETURN
1200 FOR I=1 TO 7 STEP 3
1210 IFS(I)()S(I+1)THEN 1250
1220 IFS(I)()S(I+2)THEN1250
```

```
 1230 IFS(I)=-1THEN 1500
 1240 IFS(I)=1 THEN 1400
 1250 NEXT I:FORI=1TO3:IFS(I)()S(I+3)THEN1290
 1260 IFS(I)()S(I+6)THEN1290
 1270 IFS(I)=-1THEN1500
 1280 IFS(I)=1THEN1400
 1290 NEXTI:FORI=1TO9:IFS(I)=0THEN 1310
 1300 NEXT I:GOTO 1600
 1310 IFS(5)()GTHEN 1340
 1320 IFS(1)=GANDS(9)=GTHEN 1350
 1330 IFS(3)=GANDS(7)=GTHEN 1350
 1340 IF BT=1 THEN 205
 1345 IF BT=2 THEN 530
 1350 IF G=-1 THEN 1500
 1400 PRINT:PRINT"JE HEBT ME VERSLAGEN!   ":PRINT
 1405 WS=WS+1
 1410 GOTO 2000
 1500 PRINT:PRINT"IK HEB GEWONNEN !!!     ":PRINT
 1505 WK=WK+1
 1510 GOTO 2000
 1600 PRINT:PRINT"WE HEBBEN EEN GELYK SPEL":PRINT
 1605 WG=WG+1
 2000 INPUT"NOG EEN SPELLETJE";C$
 2005 PRINT
 2010 IFLEFT$(C$,1)()"N" THEN 40
 2020 GOTO 2500
 2030 PRINT"TYPE S.V.P. JA OF NEE":PRINT
 2040 GOTO 2000
 2500 POKE548,4:PRINTCHR$(3);:PRINT
 2510 PRINT"IN TOTAAL HEBBEN WE";WS+WK+WG
 2520 PRINT"SPELLETJE(S) GESPEELD."
 2530 PRINT"JY HEBT ER";WS;"GEWONNEN"
 2540 PRINT"IK WON";WK;"SPELLETJE";
 2542 IF WK=1 THEN PRINT CHR$(13):GOTO 2550
 2545 PRINT"S"
 2550 PRINT"EN IN TOTAAL HAALDEN WE "WG"MAAL EEN GELYK SPEL"
 2560 PRINT:PRINT"EINDE PROGRAMMA":POKE548,4
 2570 END
```

BOTER KAAS EN EIEREN, 2 K AANPASSINGEN:

```
 1000 POKE546,35:POKE547,56:POKE548,10:PRINTCHR$(2);
 1090 POKE546,9:POKE548,20:PRINTCHR$(3);
 2560 PRINT:PRINT"EINDE PROGRAMMA":POKE548,4
ready
```

blz. 13

```
 10 FORI=1TO32:PRINT:NEXT
 20 PRINT"   DRIE DIMENSIONAAL":PRINT
 21 PRINT"  BOTER,KAAS EN EIEREN"
 30 FORI=1TO10:PRINT:NEXT
 35 FORI=1TO5000:NEXT
150 DIMM(63,6),N(75),E(18),D(63),W(63)
160 DIMP(63),H(7),G(63),Q(15),B(13)
165 BA=53637
180 FORA=0TO6:FORB=0TO63:READM(B,A):NEXTB,A
230 FORA=0TO15:READQ(A):NEXT
260 FORA=0TO7:READH(A):NEXT
290 FORA=0TO63:READW(A):NEXT
320 FORA=0TO10:READC(A):NEXT
350 FORA=0TO63:READP(A):NEXT
380 FORA=0TO13:READB(A):NEXT
410 FORA=0TO75:N(A)=0:NEXT
440 FORA=0TO63:G(A)=0:NEXT
470 FORA=0TO18:E(A)=0:NEXT
475 FORI=1TO10:PRINT:NEXT
479 IFFL=1THEN481
480 FORI=1TO8:READII:POKE559+I,II:NEXT
481 FL=1
485 POKE538,48:POKE539,2
500 B$=",XO":M=1:Q=76
540 FORF=0TO63:GOSUB1180:NEXT
580 FORA=1TO20
590 POKEBA+319+A,ASC(MID$("laatste zet          ",A,1))
600 NEXT
602 FORA=1TO4:POKEBA-36+6*A,A+48:NEXT
620 INPUT" zal ik eerst";A$:IFLEFT$(A$,1)="J"THEN850
630 PRINTSPC(23):PRINT:PRINT"Uw zet nr";M+Z;
640 INPUTF:F=16*F-159*INT(F/10)-6*INT(F/100)-21
690 IFF)=0ANDF(=63THEN720
700 PRINTSPC(23):PRINT:PRINT"fout,Uw zet";:GOTO640
720 IFG(F)THEN700
730 G(F)=1:GOSUB1180
760 FORB=0TO6+3*(M(F,6)=76)
770 N=N(M(F,B)):E=E(M(F,B)/4)
790 IFN=3THENPRINTSPC(23):PRINT:PRINT"OK U wint";:GOTO1300
800 IFN=INT(N/4)*4ANDN)3THENQ=Q-1
810 IFQ=0THEN1290
820 N(M(F,B))=N+1:E(M(F,B)/4)=E-((E-INT(E/8)*8)(7)
840 NEXT
850 N=INT((M-1)/4)*8:P=-10000
870 FORA=0TO63:IFG(A)THEN1020
890 T=0:E=0:B=6+3*(M(A,6)=76):G=(B/3-1)*7
930 FORC=0TOB:D=Q(N(M(A,C)))
950 T=T+W(N+D)+P(E(M(A,C)/4))*B(G+C):E=E+H(D)
970 NEXTC
980 IFE)10THENE=10
990 T=T+C(E):IFT+RND(1))PTHENP=T+.5:F=A:T2=E
1020 NEXTA
1030 G(F)=2:GOSUB1180
```

```
1060 FORB=0TO6+3* (M(F,6)=76)
1070 N=N(M(F,B)):E=E(M(F,B)/4)
1090 IFN=12THENPRINTSPC(23):PRINT:PRINT"IK WIN !";:GOTO1300
1100 IFN(4ANDN()OTHENQ=0-1
1110 IFQ=0THEN1290
1120 N(M(F,B))=N+4:E(M(F,B)/4)=E-8*(E(56)
1140 NEXTB
1150 M=M+1:GOTO630
1170 END
1180 C=INT(F/16):E=F-16*C:D=INT(E/4):E=E-4*D
1182 A=6*D+E:B=6-2*C
1240 POKEBA+A+32*B,ASC(MID$(B$,G(F)+1,1))
1250 POKEBA+332,D+49
1260 POKEBA+334,E+49
1270 POKEBA+336,C+49
1280 RETURN
1290 PRINTSPC(23):PRINT:PRINT"onbeslist";
1300 INPUT"!nog eens";A$
1310 PRINTSPC(23):PRINT:IFLEFT$(A$,1)="J"THEN410
1320 POKE538,105:POKE539,255
1390 DATA0,0,0,0,1,37,38,1,2,41,42,2,3,3,3,3
1400 DATA32,4,4,35,5,5,5,5,6,6,6,6,44,7,7,47
1410 DATA32,8,8,35,9,9,9,9,10,10,10,10,44,11,11,47
1420 DATA12,12,12,12,13,37,38,13,14,41,42,14,15,15,15,15
1430 DATA16,33,34,28,16,20,24,28,16,20,24,28,16,45,46,28
1440 DATA17,21,25,29,36,21,25,39,40,21,25,43,17,21,25,29
1450 DATA18,22,26,30,36,22,26,39,40,22,26,43,18,22,26,30
1460 DATA19,33,34,31,19,23,27,31,19,23,27,31,19,45,46,31
1470 DATA32,20,24,35,36,1,1,39,40,2,2,43,44,20,24,47
1480 DATA4,33,34,4,17,37,38,29,17,41,42,29,7,45,46,7
1490 DATA8,33,34,8,18,37,38,30,18,41,42,30,11,45,46,11
1500 DATA32,23,27,35,36,13,13,39,40,14,14,43,44,23,27,47
1510 DATA48,65,66,52,57,48,52,61,58,52,48,62,52,69,70,48
1520 DATA49,56,60,53,64,49,53,67,68,53,49,71,53,59,63,49
1530 DATA50,60,56,54,68,50,54,71,64,54,50,67,54,63,59,50
1540 DATA51,69,70,55,61,51,55,57,62,55,51,58,55,65,66,51
1550 DATA56,76,76,60,76,76,76,76,76,76,76,76,59,76,76,63
1560 DATA76,76,76,76,76,57,61,76,76,58,62,76,76,76,76,76
1570 DATA76,76,76,76,76,61,57,76,76,62,58,76,76,76,76,76
1580 DATA60,76,76,56,76,76,76,76,76,76,76,76,63,76,76,59
1590 DATA64,76,76,67,76,76,76,76,76,76,76,76,68,76,76,71
1600 DATA76,76,76,76,76,65,66,76,76,69,70,76,76,76,76,76
1610 DATA76,76,76,76,76,69,70,76,76,65,66,76,76,76,76,76
1620 DATA68,76,76,71,76,76,76,76,76,76,76,76,64,76,76,67
1630 DATA72,76,76,73,76,76,76,76,76,76,76,76,74,76,76,75
1640 DATA76,76,76,76,76,72,73,76,76,74,75,76,76,76,76,76
1650 DATA76,76,76,76,76,75,74,76,76,73,72,76,76,76,76,76
1660 DATA75,76,76,74,76,76,76,76,76,76,76,73,76,76,72
1670 DATA0,4,5,6,1,7,7,7,2,7,7,7,3,7,7,7
1690 DATA0,0,5,0,0,0,1,0,0
1710 DATA5,10,1,4000,1,4,700,0,5,10,1,4000,1,5,700,0
1730 DATA5,15,5,4000,5,6,700,0,5,15,25,4000,1,12,700,0
1750 DATA25,20,40,4000,1,15,700,0,25,30,40,4000,1,15,700,0
1770 DATA25,1,1,4000,10,40,700,0,25,1,1,4000,10,40,700,0
1800 DATA0,0,160,180,200,0,20,180,200,200,400
1820 DATA0,0,0,55,56,100,200,300,0,0,0,0,40,60,80,100
```

```
1840 DATA0,0,0,0,0,1,40,60,5,0,0,0,0,0,1,20
1870 DATA55,5,0,0,-5,0,0,0,60,50,10,0,0,-20,0,0
1890 DATA120,100,20,0,0,0,-20,0,150,120,25,0,0,0,0,-20
1910 DATA1,1,1.5,1,1,1,1,1,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5
2000 DATA201,10,240,3,32,105,255,96
```

DRIE DIMENSIONAAL BOTER KAAS EN EIEREN 2 K VIDEO AANPASSINGEN

```
21 PRINTTAB(14)"  BOTER,KAAS EN EIEREN"
30 FORI=1TO14:PRINT:NEXT
165 BA=53913
602 FORA=1TO4:POKEBA-69+6*A,A+48:NEXT
1240 POKEBA+A+64*B,ASC(MID$(B$,G(F)+1,1))
1250 POKEBA+652,D+49
1260 POKEBA+654,E+49
1270 POKEBA+656,C+49
```

DESTROYER (1K VIDEO)

```
  1 REM**DESTROYER**
  2 REM**JOVIAK**
  3 REM* IP PROGRAM *
  4 REM CONTROL-C OFF
  5 POKE530,1
  6 X=53504:X1=53505
  7 PRINT:PRINT
  8 PRINT:PRINT"VERKLARING:
  9 PRINT
 10 PRINT"U bent commandant van"
 15 PRINT"een kruiser die een met"
 20 PRINT"onderzeeboten geinfil-
 25 PRINT"treerd gebied bewaakt.
 30 PRINT"U moet ze vernietigen
 35 PRINT"met dieptebommen.
 40 PRINT"ontstekingsdiepte:
 43 PRINT:PRINT
 45 PRINT"  '1' DIEPER
 50 PRINT"  '2' HOGER
 55 PRINT"  '3' WERPT BOMMEN
 57 PRINT:PRINT
 60 PRINT"U hebt 30 dieptebommen
 63 PRINT
 65 PRINT"GOEDE JACHT  !"
 67 PRINT
 70 PRINT:INPUT"KLAAR";B$
 75 IF LEFT$(B$,1)="N"THEN1
 97 C=30:S=0
 98 FORI=1TO30:PRINT:NEXTI
 99 PRINT"SCHOT:        SCORE:"
100 FORI=53440TO53473:POKEI,155:NEXTI
110 POKE54092,51:POKE54093,48:POKE54105,48:POKE54106,48
120 DEF FND(X)=INT(15*RND(X)+1)
200 K1=53410:K2=53441
203 GOSUB850
205 FORJ=1TO30:POKEK1+J-1,32:POKEK1+J,179:POKEK1+J+1,180
207 M=0
210 GOSUB600
220 GOSUB900
250 NEXTJ
253 Z1=0
254 P1=0:P2=0:P3=0
255 FORJ=1TO30:POKEK2-J+1,32:POKEK2-J,182:POKEK2-J-1,181
257 M=1
260 GOSUB600
265 GOSUB900
299 NEXTJ
300 Z1=0
301 P1=0
302 P2=0:P3=0
500 GOTO203
600 POKE57088,127:F=PEEK(57088)
603 IFF=255THENRETURN
```

```
 605 IFF=223THEN700
 607 IFX1(53504ANDF=191THENRETURN
 608 IFX1)54016ANDF=127THENRETURN
 610 IFF=191THENR=R-32
 615 IFF=127THENR=R+32
 620 FORA=1TO315STEP2
 625 X1=X+R+A
 630 POKEX1,46
 635 IFF=191THENX2=X1+32
 640 IFF=127THENX2=X1-32
 645 POKEX2,32
 650 NEXTA
 690 GOTO800
 700 IFM=0THENQ=K1+J
 710 IFM=1THENQ=K2-J
 715 FORL=Q+64TOQ+R+96STEP32
 720 POKEL-32,32:POKEL,46
 723 POKEQ+32,155
 725 NEXTL
 730 FORT=255TO32STEP-1:POKEQ+R+96,T:NEXTT
 735 Z=Q+R+96
 737 O=S
 738 IFZ=B1+J+1ORZ=B2-J-1ORZ=B3+J+1THENS=S+1
 739 IFZ=B1+J-1ORZ=B2-J+1ORZ=B3+J-1THENS=S+1
 740 IFZ=B1+JORZ=B2-JORZ=B3+JTHENS=S+1
 741 IFZ=B1+J+2ORZ=B2-J-2ORZ=B3+J+2THENS=S+1
 742 IFS()OTHENZ1=Z
 743 C=C-1
 745 IFC=(0THEN2000
 747 IFS()OTHENGOSUB1300
 750 S1=48+INT(S/10):S2=S-INT(S/10)*10+48
 751 POKE54105,S1:POKE54106,S2
 760 C1=48+INT(C/10):C2=C-INT(C/10)*10+48
 765 POKE54092,C1:POKE54093,C2
 800 RETURN
 850 B1=53440+FND(Q)*32
 855 B2=53471+FND(B1)*32
 856 IFB2=B1-31THEN855
 857 J=24
 860 B3=53440+FND(J)*32:IFB3=B1THEN860
 861 IFB2=B3-31THEN855
 865 RETURN
 900 POKEB1+J-1,32:POKEB2-J+1,32:POKEB3+J-1,32
 905 GOSUB1200
 907 IFP1=1THEN915
 910 POKEB1+J,41:POKEB1+J+1,41:POKEB1+J+2,41
 915 IFP2=1THEN925
 920 POKEB2-J,40:POKEB2-J-1,40:POKEB2-J-2,40
 925 IFP3=1THEN1000
 930 POKEB3+J,41:POKEB3+J+1,41:POKEB3+J+2,41
1000 RETURN
1200 FORP=1TO2
1210 IFB1+J+P=Z1THENP1=1
1220 IFB2-J-P=Z1THENP2=1
1230 IFB3+J+P=Z1THENP3=1
1235 NEXTP
```

```
 1240 RETURN
 1300 FORT=Q+R+93TOQ+R+99
 1310 POKET,32
 1320 NEXTT
 1330 RETURN
 2000 INPUT"NOG EENS";A$
 2100 IFLEFT$(A$,1)="J"THEN1
49999 END
```

blz. 19.

```
  1 REM GOBANG:UIT:BYTE/NOV1979/PAG56EV
  2 REM VOOR SUPERBOARD BEWERKT DOOR JOS BURGHOUTS,TEL.015-565291
  3 PRINTCHR$(3):PRINT"GOBANG":PRINT
  4 DIM M(15,15),T(23,23),S(81)
  6 FORI=1TO81:S(I)=0:NEXTI
  8 INPUT"WILT U INSTRUKTIES";Z$
  9 IFLEFT$(Z$,1)="J"THENGOSUB2000
 19 S(20)=1:S(10)=40:S(12)=30:S(13)=47:S(27)=15:S(28)=20:S(29)=10
 26 S(30)=40:S(31)=50:S(32)=30:S(24)=1:S(36)=39:S(37)=65:S(38)=40
 33 S(39)=70:S(40)=100:S(41)=60:S(42)=30:S(43)=30:S(44)=30:S(62)=41
 41 S(72)=31:S(73)=11:S(74)=41:S(78)=51:S(80)=90:S(26)=21:S(79)=40
 48 S(60)=21:S(61)=11
 51 FORI=1TO23:FORJ=1TO23
 52 T(I,J)=0:IFI(15THENIFJ(15THENM(I,J)=0
 70 NEXTJ:NEXTI
 76 C=-1:W=12:N=12:O=12:X=12
 80 PRINTCHR$(3);
 91 GOTO300
 95 GOSUB 800
 97 POKE548,25:             PRINTCHR$(3);:INPUT"JOUW ZET";Z,Y
 99 Y=Y+4:Z=Z+4
101 IFY)19THEN97
102 IFZ)19THEN97
103 IFY(5THEN97
104 IFY(5THEN97
106 IFT(Y,Z))0THEN97
110 T(Y,Z)=2:I=Y:J=Z
120 GOSUB 800
127 GOSUB1000
128 IFC()-1THEN310
130 I=W:J=X:GOSUB1000
160 Q=-1:FORI=N-1TO0+1:FORJ=5TO19
200 IFT(I,J))0THEN220
205 A=M(I-4,J-4):IFA(QTHEN220
210 W=I:X=J:Q=A
220 NEXTJ:NEXTI
290 PRINTCHR$(3)CHR$(7)CHR$(7);"MYN ZET";X-4;",";W-4
300 T(W,X)=1:GOSUB800:IFM(W-4,X-4)(100THEN97
307 POKE548,25:PRINTCHR$(3)"IK HEB GEWONNEN"
310 IFC=0THENPOKE548,25:PRINTCHR$(3)"JY HEBT GEWONNEN !"
330 GOTO 1200
800 POKE 548,4:            PRINTCHR$(2);
806 PRINT:PRINT"              111111"
808 PRINT"    1234567890123456"
810 FORI1=5TO19:IFI1-4(10THENPRINTI1-4;" ";
811 IFI1-4)9THENPRINTI1-4;
815 FORJ1=5TO19
820 IFT(I1,J1)=0THENPRINT".";
825 IFT(I1,J1)=1THENPRINT"X";
830 IFT(I1,J1)=2THENPRINT"O";
835 NEXTJ1:PRINT"":NEXTI1
840 POKE548,25
850 RETURN
```

blz. 20

```
1000 K=1:L=-1:IFI(NTHENIFI)5THENN=I
1003 IFI)0THENIFI(19THENO=I
1005 U=I:V=J:D=0
1011 D=D+1:P=81
1026 IFU)19THEN1090
1027 IFV)19THEN1090
1028 IFU(5THEN1090
1029 IFV(5THEN1090
1030 E=U-4:G=V-4:A=M(E,G):Q=T(U+K,V+L)
1035 R=T(U-K,V-L)*27+T(U-2*K,V-2*L)*9
1036 R=R+T(U-3*K,V-3*L)*3+T(U-4*K,V-4*L)
1037 B=Q*27+T(U+2*K,V+2*L)*9+T(U+3*K,V+3*L)*3
1038 IFR=80THENIFT(U,V)=2THENC=0
1039 IFT(U,V)()0THEN1075
1042 IFR(14THENIFR)11THENIFQ=1THENP=37
1044 IFR)71THENIFB)53THENIFB(63THENP=80
1046 IFR)71THENIFB)71THENP=80
1048 IFR)53THENIFR(63THENIFQ=2THENP=72
1050 IFP=72THENIFR=60THENP=31
1052 IFQ()2THEN1058
1053 IFR=78THENP=80
1054 IFR=79THENP=80
1056 IFR=41THENR=81
1058 IFR(42THENIFR)35THENIFQ=1THENP=41
1059 IFR(33THENIFR)29THENIFQ=1THENP=41
1060 IFR)53THENIFR(63THENIFB)71THENP=80
1061 IFR)38THENIFR(42THENIFQ=1THENR=40
1062 IFR)35THENIFR(45THENIFB)35THENIFB(45THENR=40
1063 IFR)27THENIFR(54THENIFB)38THENIFB(42THENR=40
1064 IFR=79THENIFA=51THENM(E,G)=41
1065 IFR=0THENR=81
1066 IFS(P))S(R)THENR=P
1067 IFS(R)-S(R)/10*10=1THENIFA-A/10*10=1THENIFS(R)(41THENR=74
1068 IFS(R)-S(R)/10*10=9THENIFA-A/10*10=9THENIFS(R)(65THENR=37
1070 IFS(R))M(E,G)THENM(E,G)=S(R)
1075 IFD)4THEN1090
1081 U=U+K:V=V+L:GOTO1011
1090 IFK=0THENIFL=-1THENRETURN
1095 IFK=-1THENIFL=-1THENK=0
1100 IFK=-1THENIFL=0THENL=-1
1105 IFK=-1THENIFL=1THENL=0
1110 IFK=0THENIFL=1THENK=-1
1115 IFK=1THENIFL=1THENK=0
1120 IFK=1THENIFL=0THENL=1
1125 IFK=1THENIFL=-1THENL=0
1130 GOTO1005
1200 PRINT:INPUT"NOG EEN SPELLETJE";A$
1205 PRINT
1210 IF A$="JA"THEN 6
1220 IF A$="NEE"THEN 1250
1230 PRINT"TYPE SVP JA OF NEE"
1240 GOTO 1200
1250 POKE548,4:PRINTCHR$(3);:PRINT"EINDE PROGRAMMA"
1260 END
2000 PRINTCHR$(3)        "*****  G O B A N G  ****":PRINT
2010 PRINT"By dit spel is het de bedoeling om 5 stenen"
```

```
2020 PRINT"op een ry te krygen,  horizontaal, verticaal of"
2030 PRINT"diagonaal.";PRINT
2040 PRINT"De computer probeert hetzelfde te doen en vaak"
2050 PRINT"met succes !!!";PRINT
2060 PRINT"De zetten worden  opgegeven met KOLOM,RIJ
2070 PRINT
2080 PRINT
2090 PRINT"Omdat het voor de computer een ingewikkeld spel"
2100 PRINT"is moet hij wel een poosje nadenken";PRINT
2110 PRINT"Druk op de 'ESC' toets  om verder te gaan
2120 IFPEEK(57100)()222THEN2120
2140 PRINTCHR$(3);
2150 RETURN
```

EEN EN TWINTIGEN (2K VIDEO)

```
 20 DIMC(13,4):FORI=1TO30:PRINT:NEXT
 21 POKE530,1:REM      ^C
 22 PRINT"INSTRUKTIES:":PRINT
 25 PRINT"1 voor een nieuwe kaart":PRINT"2 om te  erdubbelen
 30 PRINT"3 om te passen":FORI=1TO4:S(I)=228+I:NEXT:S(0)=63
 50 N=1
 60 P(1)=53330:P(2)=P(1)+13
 70 FORI=1TO13:FORJ=1TO4:C(I,J)=0:NEXTJ,I
 80 FORI=1TON+1:FORJ=1TO5:PC(I,J)=0:NEXTJ:SW(I)=0:NEXTI
100 PRINT:PRINT"Hoeveel zet U in";
101 INPUTW(1):IFW(1)<0ORW(1)>500GOTO100
110 FORI1=1TO30:PRINT:NEXT:POKE54117,32
119 I1=1
120 FORQ=1TO2
130 GOSUB1100:LO=P(Q):IFI1=1ANDQ=N+1THENS=1
140 PC(Q,I1)=IC:GOSUB1000:P(Q)=P(Q)+6*64:NEXTQ
150 S=0:I1=I1+1:IFI1<3GOTO120
160 S=0:FORQ=1TON+1:GOSUB1200:IFT=21THENGOSUB1300
170 NEXT:IFS=1GOTO690
180 Q=1:I1=3
185 L$="SPELER ??":GOSUB1310
190 POKE57088,127:C=PEEK(57088)
200 IFC=127ORC=223GOTO230
210 IFC=191THENW(Q)=W(Q)*2:GOTO230
225 GOTO190
230 L$="                    ":GOSUB1310
235 IFC=223GOTO300
240 GOSUB1100:PC(Q,I1)=IC:LO=P(Q):GOSUB1000:P(Q)=P(Q)+6*64
260 GOSUB1200:IFT>21THENL$="DOOD !!!":GOSUB1310:GOTO300
265 IFI1=5THENSW(Q)=1:GOTO700
270 I1=I1+1:GOTO185
300 I1=3:Q=1:GOSUB1200:PT=T:Q=2:GOSUB1200
305 LO=PZ:C$=PZ$+" ":S=0:IS=PS:GOSUB1000
310 IFPT>21ORPT<T+1GOTO690
320 IFI1=6THENSW(N+1)=1:GOTO700
330 GOSUB1100:PC(Q,I1)=IC:LO=P(Q):GOSUB1000:P(Q)=P(Q)+6*64
340 GOSUB1200:IFT<22THENI1=I1+1:GOTO310
350 L$="DOOD !!!":GOSUB1310
690 FORI=1TO2500:NEXT
700 FORI=53300TO53600:POKEI,32:NEXT
710 Q=1:GOSUB1200:PT=T:Q=2:GOSUB1200
720 IFPT>21THENPRINT"Het huis wint.":H(1)=H(1)-W(1):GOTO790
721 IFPT<T+1ANDT<22THENPT=22:GOTO720
730 IFSW(2)=1THENPT=22:GOTO720
740 PRINT"Speler wint!":H(1)=H(1)+W(1)
790 PRINT:PRINT:PRINT
800 Q=N+1:GOSUB1200:PRINT"Het huis had"T:PRINT:PRINT
810 Q=1:GOSUB1200
820 PRINT"Speler had"T","
821 PRINT"Uw inzet was"W(Q)",":PRINT"U hebt nu:":PRINT"$"H(Q)
830 PRINT:PRINT:INPUT"Doorgaan";Y$
840 IFLEFT$(Y$,1)="J"GOTO60
850 GOTO49999
  blz. 23
```

vervolg EEN EN TWINTIGEN (2K VIDEO)

```
 1000 FORI=LOTOLO+2:POKEI,128:POKEI+320,135:NEXT
 1010 FORI=LO+63TOLO+300STEP64:POKEI,143:POKEI+4,136:NEXT
 1020 IFS=1THENPZ=LO:PZ$=C$:PS=IS:IS=0:C$="??"
 1030 POKELO+64,S(IS)
 1031 POKELO+258,S(IS)
 1040 FORI=1TOLEN(C$):POKELO+I*64+65,ASC(MID$(C$,I)):NEXT
 1050 RETURN
 1100 IC=INT(RND(1)*13)+1:IS=INT(RND(1)*4)+1
 1110 IFC(IC,IS)=1GOTO1100
 1120 C(IC,IS)=1:C$=MID$(STR$(IC),2):IFIC=1THENC$="AC"
 1130 IFIC=11THENC$="JK"
 1140 IFIC=12THENC$="QU"
 1150 IFIC=13THENC$="KI"
 1160 IFIC>10THENIC=10
 1170 RETURN
 1200 IC=0:T=0:FORI=1TO5:IFPC(Q,I)=1THENIC=IC+1
 1210 T=T+PC(Q,I):NEXT
 1220 IFIC=0THENRETURN
 1230 FORI=1TOIC:IFT+10>21THENRETURN
 1240 T=T+10:NEXT:RETURN
 1300 L$="*** 21 ***":SW(Q)=1:S=1
 1310 FORI=1TOLEN(L$):POKEP(Q)+I-4,ASC(MID$(L$,I)):NEXT:RETURN
49999 END
```

ENKELE VERBETERINGEN:

```
 20 DIMC(13,4)
 110 PRINTCHR$(3)
 700 POKE11,6:POKE12,254:X=USR(X)
```

blz. 24

BOMBER (2K VIDEO)

```
  1 POKE530,1:REM      ^C
  2 REM- DAVID E. TEWKSBARY
  3 PRINT"BOMBER":FORI=1TO15:PRINT:NEXT
  4 INPUT"Instrukties nodig (J/N)";Y$
  5 IFLEFT$(Y$,1)="N"THEN15
  6 FORI=1TO30:PRINT:NEXT
  7 PRINT"U bent het eenzame vliegtuig aan de rechterkant van het sch
  8 PRINT"De vijand vliegt van links naar rechts
  9 PRINT"naar U toe, al schietend.
 10 PRINT:PRINT"U kunt:";PRINT"1) klimmen";PRINT"2) duiken
 11 PRINT"3) schieten";PRINT"4) bombarderen
 12 PRINT:PRINT"vijandelijke vliegtuigen 15 punten, gronddoelen 75."
 13 PRINT:PRINT"JA, U kunt vliegtuigen bombarderen.
 14 PRINT:PRINT"Geef een redelijke (1-5)
 15 INPUT"snelheid voor de vijand";SP:FORH=1TO30:PRINT:NEXTH
 20 FORH=55045TO55095STEP10:POKEH,14:POKEH-5,15:NEXTH:POKE55040,32
 21 FORH=55105TO55164:POKEH,155:NEXTH
 25 L=SP+2.1:J=99:B5=54202
 26 V(1)=54700:V(2)=54796:V(3)=53900:V(4)=54230
 30 FORH=1TO2:V2=0
 40 IFH=2ORABS(V(H+1)-V(H)))100GOTO60
 45 IFV(H+1)(V(H)GOTO80
 50 GOTO90
 60 K=INT(4*RND(J))+1:ONKGOTO70,100,100,100
 70 K=INT(2*RND(J))+1:ONKGOTO80,90
 80 V2=64:GOTO100
 90 V2=-64
100 IFV(H)+V2)54900THENV2=-64
110 IFV(H)+V2(54000THENV2=64
120 POKEV(H),32:POKEV(H+2),32
125 V(H)=V(H)+V2+SP+2:V(H+2)=V(H+2)+V2+SP+2
130 POKEV(H),237:POKEV(H+2),237:GOSUB1800
135 NEXTH
140 FORH=1TO4:IFS(H)=1GOTO170
150 K=INT(3*RND(J))+1:ONKGOTO160,200,200
160 SV(H)=V(H)-1:S(H)=1
170 POKESV(H),32:SV(H)=SV(H)+SP+5
180 IFABS(INT(SV(H)/64)-SV(H)/64)(L/20THENS(H)=0
190 IFS(H)=1THENPOKESV(H),58:GOSUB1500
200 NEXTH
220 POKEB5,32:B5=B5-1
225 POKE57088,127
227 A=PEEK(57088)
230 IFA=127ANDB5)53650THENB5=B5-64
231 IFA=191ANDB5(54900THENB5=B5+64
232 IFA=223ANDSB=0THENSB=1:BV=B5+1
233 IFA=239ANDBB=0THENBB=1:B2=B5+1
240 POKEB5,239:IFSB=0GOTO280
250 POKEBV,32:BV=BV-5:GOSUB1000
260 IFABS(INT(BV/64)-BV/64)),85THENSB=0
270 IF SB=1 THENPOKEBV ,58
280 IFBB=0GOTO30
290 POKEB2,32:B2=B2+63:GOSUB1300
```

blz. 25

```
  295 IFINT(B2/64)=B2/64THENBB=0
  300 IFBB=1THENPOKEB2,46
  310 IFB2(55040GOTO30
  320 BB=0:FORN=96TO32STEP-1:POKEB2,N:NEXTN
  330 GOTO30
 1000 FORH=1TO4:FORM=-64TO64STEP64
 1100 IFABS(BV-M-V(H)+L/2)(LTHENSB=0:POKEBV,32:GOSUB2000
 1200 NEXTM:NEXTH:RETURN
 1300 FORH=1TO4:FORM=-64TO64STEP64
 1400 IFABS(B2-M-V(H))(2.1THENBB=0:POKEB2,32:GOSUB2000
 1450 NEXTM:NEXTH:RETURN
 1500 FORM=-64TO64STEP64
 1600 IFABS(B5-M-SV(H)+L/2)(LGOTO9000
 1700 NEXTM:RETURN
 1800 FORM=-64TO64STEP64
 1850 IFABS(B5-M-V(H)+L/2)(LGOTO9000
 1860 IFABS(B5-M-V(H+2)+L/2)(LGOTO9000
 1900 NEXTM:RETURN
 2000 POKEV(H)-2,83:POKEV(H)-1,67:POKEV(H),79:POKEV(H)+1,82
 2100 POKEV(H)+2,69:Z=Z+15
 2200 FORN=1TO500:NEXTN
 2300 FORN=-2TO2:POKEV(H)+N,32:NEXTN:FORN=-32TO32
 2400 IFABS(INT((V(H)+N)/64)-(V(H)+N)/64)(.01THENV(H)=V(H)+N
 2500 NEXTN:RETURN
 9000 FORH=55045TO55095STEP5:IFPEEK(H)=32THENZ=Z+75
 9005 NEXTH
 9010 IFZ=0GOTO9400
 9100 PRINT:PRINT:PRINT"GEFELICITEERD !!!!"
 9200 PRINT:PRINT"U HEEFT";Z;"PUNTEN
 9300 GOTO9500
 9400 PRINT:PRINT:PRINT"SUKKEL, JE STIERF VOOR NIETS !"
 9500 PRINT:PRINT:INPUT"NOG EENS (J/N)";Y$
 9550 IFLEFT$(Y$,1)="J"THENRUN15
 9600 PRINT:PRINT:PRINT"LAFAARD!!":PRINT
49999 POKE530,0: END
```

DESTROYER (2K VIDEO)

```
  1 REM**DESTROYER**
  2 REM**JOVIAK**
  3 REM 1P PROGRAM
  4 REM CONTROL-C OFF
  5 POKE530,1
  6 X=53760:X1=53761
  7 PRINT:PRINT
  8 PRINT:PRINT"VERKLARING:
  9 PRINT
 10 PRINT"U bent commandant van"
 15 PRINT"een kruiser die een met"
 20 PRINT"onderzeeboten geinfil-
 25 PRINT"treerd gebied bewaakt.
 30 PRINT"U moet ze vernietigen
 35 PRINT"met dieptebommen.
 40 PRINT"ontstekingsdiepte:
 43 PRINT:PRINT
 45 PRINT"  '1' DIEPER
 50 PRINT"  '2' HOGER
 55 PRINT"  '3' WERPT BOMMEN
 57 PRINT:PRINT
 60 PRINT"U hebt 30 dieptebommen
 63 PRINT
 65 PRINT"GOEDE JACHT  !"
 67 PRINT
 70 PRINT:INPUT"KLAAR";B$
 75 IF LEFT$(B$,1)="N"THEN1
 97 C=30:S=0
 98 FORI=1TO30:PRINT:NEXTI
 99 PRINT"SCHOT:        SCORE:"
100 FORI=53632TO53697:POKEI,155:NEXTI
110 POKE54924,51:POKE54925,48:POKE54937,48:POKE54938,48
120 DEF FND(X)=INT(15*RND(X)+1)
200 K1=53570:K2=53633
203 GOSUB850
205 FORJ=1TO62:POKEK1+J-1,32:POKEK1+J,179:POKEK1+J+1,180
207 M=0
210 GOSUB600
220 GOSUB900
250 NEXTJ
253 Z1=0
254 P1=0:P2=0:P3=0
255 FORJ=1TO62:POKEK2-J+1,32:POKEK2-J,182:POKEK2-J-1,181
257 M=1
260 GOSUB600
265 GOSUB900
299 NEXTJ
300 Z1=0
301 P1=0
302 P2=0:P3=0
500 GOTO203
600 POKE57088,127:F=PEEK(57088)
603 IFF=255THENRETURN
```

```
 605 IFF=223THEN700
 607 IFX1(53760ANDF=191THENRETURN
 608 IFX1)54859ANDF=127THENRETURN
 610 IFF=191THENR=R-64
 615 IFF=127THENR=R+64
 620 FORA=1TO63STEP2
 625 X1=X+R+A
 630 POKEX1,46
 635 IFF=191THENX2=X1+64
 640 IFF=127THENX2=X1-64
 645 POKEX2,32
 650 NEXTA
 690 GOTO800
 700 IFM=0THENQ=K1+J
 710 IFM=1THENQ=K2-J
 715 FORL=Q+128TOQ+R+192STEP64
 720 POKEL-64,32:POKEL,46
 723 POKEQ+64,155
 725 NEXTL
 730 FORT=255TO32STEP-1:POKEQ+R+192,T:NEXTT
 735 Z=Q+R+192
 737 O=S
 738 IFZ=B1+J+1ORZ=B2-J-1ORZ=B3+J+1THENS=S+1
 739 IFZ=B1+J-1ORZ=B2-J+1ORZ=B3+J-1THENS=S+1
 740 IFZ=B1+JORZ=B2-JORZ=B3+JTHENS=S+1
 741 IFZ=B1+J+2ORZ=B2-J-2ORZ=B3+J+2THENS=S+1
 742 IFS()OTHENZ1=Z
 743 C=C-1
 745 IFC=(0THEN2000
 747 IFS()OTHENGOSUB1300
 750 S1=48+INT(S/10):S2=S-INT(S/10)*10+48
 751 POKE54937,S1:POKE54938,S2
 760 C1=48+INT(C/10):C2=C-INT(C/10)*10+48
 765 POKE54924,C1:POKE54925,C2
 800 RETURN
 850 B1=53632+FND(Q)*64
 855 B2=53695+FND(B1)*64
 856 IFB2=B1-31THEN855
 857 J=24
 860 B3=53632+FND(J)*64:IFB3=B1THEN860
 861 IFB2=B3-31THEN855
 865 RETURN
 900 POKEB1+J-1,32:POKEB2-J+1,32:POKEB3+J-1,32
 905 GOSUB1200
 907 IFP1=1THEN915
 910 POKEB1+J,41:POKEB1+J+1,41:POKEB1+J+2,41
 915 IFP2=1THEN925
 920 POKEB2-J,40:POKEB2-J-1,40:POKEB2-J-2,40
 925 IFP3=1THEN1000
 930 POKEB3+J,41:POKEB3+J+1,41:POKEB3+J+2,41
1000 RETURN
1200 FORP=-1TO2
1210 IFB1+J+P=Z1THENP1=1
1220 IFB2-J-P=Z1THENP2=1
1230 IFB3+J+P=Z1THENP3=1
1235 NEXTP
   blz. 28
```

```
 1240 RETURN
 1300 FORT=Q+R+189TOQ+R+195
 1310 POKET,32
 1320 NEXTT
 1330 RETURN
 2000 INPUT"NOG EENS";A$
 2100 IFLEFT$(A$,1)="J"THEN1
49999 END
```

binnen in BASIC interpreter (1)

Regelmatig word ik geconfronteerd met het verzoek om een artikel over de werking van een BASIC-interpreter. In het vorige nummer heb ik dan ook een serie artikelen aangekondigd die de MICROSOFT interpreter van de 6502 machines zo goed mogelijk uit de doeken zou doen. Bij het doorlezen van de bestaande literatuur over dit onderwerp kwam ik in het septembernummer van CREATIVE COMPUTING 1979 een artikel van Philip Tubb *tegen dat door zijn opzet een goed begin van de serie zou vormen. Omdat Creative Computing overname van artikelen door hobbyclubs toestaat heb ik het artikel enigszins bewerkt voor plaatsing in onze HCCN. De "ik" in het nu volgende verhaal is dus Phil Tubb, president van ALF-products, een fabrikant van synthesizer boards voor de Apple.*
H.W.

De meeste gebruikers van hobbycomputers gebruiken een hogere programmeertaal, maar slechts een enkeling weet precies hoe die werken en hoe ze geschreven zijn. Omdat de meeste gebruikers met BASIC bekend zijn besloot ik als voorbeeld een kleine interpreter te schrijven voor een beperkt soort BASIC. Een "tiny" interpreter dus. Om het voor iedereen mogelijk te maken te volgen, hoe het programma werkt, besloot ik de interpreter in BASIC te schrijven in plaats van in machinetaal zoals meestal wordt gedaan. We krijgen dus een interpreter voor BASIC geschreven in BASIC.

De Tiny Interpreter specificaties

Als eerste is het van belang precies vast te leggen hoe de taal die moet worden geïnterpreteerd er uit moet zien. Ik zal daarvoor een paar BASIC opdrachten uitzoeken, juist genoeg om daarmee een aantal interpreterconcepten te illustreren.
De gekozen opdrachten zijn: LET, INPUT, PRINT, GOTO en IF. Een interpreter heeft een aantal besturingsopdrachten nodig zoals LIST, RUN en RENUMBER. De meeste BASIC'S bevatten een verschrikkelijk ingewikkeld floating-point gedeelte, dat zeker niet geschikt is voor ons voorbeeld. (We zouden er een apart nummer aan moeten wijden.)
Ik houd me bij OPTELLEN, AFTREKKEN, KLEINER DAN en GELIJK AAN met alleen gehele getallen van -9999 tot 9999. Ook moet er nog een interpreter deel geschreven worden dat *editor* wordt genoemd, waarmee we een regel uit het programma kunnen verwijderen, een regel kunnen toevoegen en een regel kunnen vervangen. Hierbij zullen we de hulp van regelnummers inroepen, zodat we dan een regel-georiënteerde editor hebben.
De volgende stap is het precies definieren van de regels van de taal, de syntax. Ik geef de voorkeur aan een gewijzigde vorm van BNF (= Backus Naur Form, een notatiemethode voor het beschrijven van de syntax van computertalen).
Als eerste definieer ik het element <programma>. De definitie is bij benadering:
<programma> :: <regel> [programma]
Het symbool: "::" wordt gelezen als "is gedefinieerd als". Dit symbool is een kleine afwijking van het standaard-symbool "::=" omdat het gemakkelijker te typen is. Alles wat tussen rechte haakjes staat is facultatief. In het Nederlands is bovenstaande definitie: "een programma is gedefinieerd als een regel eventueel gevolgd door een programma, of in andere woorden een of meer regels". Na deze lange omschrijving is het duidelijk dat we de voorkeur geven aan BNF. Vervolgens duiken we dieper in de definitie en definieren de elementen die gebruikt zijn bij de definitie van <programma>. In dit geval is dat alleen het element <regel>::
<regel> :: <regelnummer> <opdracht>
Laten we nu doorgaan met het definieren van elementen totdat er geen meer overblijft. We noemen zo'n definitie een "top-down definitie".
<regelnummer> :: een geheel getal van 1 tot 9999
Hoewel het mogelijk is een geheel getal, een integer in BNF te definieren, is het gemakkelijker dat in het Nederlands te lezen. We kunnen een "/" gebruiken om "of" aan te duiden:

```
<opdracht> :: <LET> / <INPUT> / <PRINT> / <GOTO> / <
LET :: <variabele> "=" <waarde>
INPUT :: "INPUT" <variabelelijst>
PRINT :: "PRINT" <waardenlijst>
GOTO :: "GOTO" <regelnummer>
IF :: "IF" <waarde> "THEN" <opdracht>
variabele :: <A-Z> <tekens>  )opmerking 1
waarde :: (<variabele> / <getal>) <binaire operator> <waarde>
getal :: een integer van -9999 tot 9999
binaire operator :: "+"/"-"/"*"/"<"/"="
variabelelijst :: <variabele> ";" <variabelelijst>
waardenlijst :: <waarde> ";" <waardenlijst>
```

Zo, dat is alles, maar de meeste talen gaan nog pagina's vol door. Alles wat tussen aanhalingstekens is geplaatst is letterlijk bedoeld. <A-Z> is niet verder gedefinieerd, omdat het wel duidelijk is dat hiermee de letters van A tot en met Z zijn bedoeld. "Opmerking 1" bij de definitie van <variabele> zou in een echte taaldefinitie vertellen dat <tekens> overeenkomt met een of meer van de ASCII tekens met uitzondering van "+", "-", "*", "<", "=" of ";", dat er een maximum lengte is voor de naam van een variabele en dat de naam niet het woord "THEN" mag bevatten. Men zou ook kunnen vermelden dat voor het gemak variabelenamen beginnend met "INPUT", "PRINT", "GOTO" of "IF" niet moeten worden gekozen. De redenen hiervoor zullen wel duidelijk worden als we met dit verhaal vorderen. De haakjes in de definitie van <waarde> zijn er om duidelijk te maken dat de definitie "een variabele of een getal, facultatief gevolgd door een binaire operator en een waarde" is en niet "een variabele, of een getal facultatief gevolgd door een binaire operator en een waarde.
We zetten nu de syntax eens op een rijtje, zodat we er gemakkelijk mee kunnen werken. Er zijn vijf typen opdrachten:

1. LET
2. INPUT
3. PRINT
4. GOTO
5. IF

Alle opdrachten starten met een sleutelwoord (zoals PRINT) behalve LET. Ze zijn allemaal samengesteld uit een aantal van de acht volgende syntaxelementen:

1. <variabele>
2. <waarde>
3. <variabelelijst>
4. <waardenlijst>
5. "="
6. <regelnummer>
7. "THEN"
8. <opdracht>

Op deze manier kunnen we nu een lijstje maken van het sleutelwoord met mogelijke combinaties van syntaxelementen:

(LET) 1;5, 2
INPUT 2;3
PRINT 3;4
GOTO 4:6
IF 5;2,7,8

Dit lijstje is handig bij het schrijven van de interpreter. In het algemeen moet de uitvoering van elke opdracht nog in detail worden besproken, maar daar onze taal in principe BASIC is, nemen we maar aan dat iedereen deze wel kent.

Hoe de interpreter wordt geschreven

Er zijn talloze manieren om een interpreter te schrijven. Er zijn erg veel factoren die daarbij een rol spelen. Iedereen wil een snelle interpreter, maar snelle interpreters hebben de neiging groot te worden of alleen geschikt te zijn voor beperkte talen. Een machinetaal zal waarschijnlijk sneller lopen dan een hogere taal, maar wie wil er nu een interpreter voor machinetaal? Ook wil iedereen graag de syntaxfouten op het scherm zien op het moment dat de regel wordt ingetikt, maar voor sommige talen is dat volslagen onmogelijk. Moeten de regels in de letterlijke vorm in het geheugen worden gezet of moeten ze worden gecomprimeerd. Deze en vele andere vragen moeten worden uitgewerkt met als uitgangspunt de toepassing die bij het gebruik van de taal bedoeld is. Het schema dat mij voor ogen staat is niet het meest efficient, alhoewel wel erg efficient, niet het allersnelst, maar wel erg snel, niet het meest compacte geheugen gebruik, maar wel erg compact en tenslotte is het niet het gemakkelijkst te schrijven alhoewel er moeilijker schema's te bedenken zijn.

Het grootste nadeel van mijn methode is waarschijnlijk dat het het schrijven van de interpreter moeilijker maakt dan vele andere methodes. We moeten er echter rekening mee houden dat een interpreter maar één keer wordt geschreven, maar dat hij talloze malen zal worden gebruikt..(wordt vervolgd)

binnenin BASIC interpreter (2)

Ik geef er de voorkeur aan de syntax van elke regel te analyseren op het moment, dat de regel wordt ingevoerd en de regel in het geheugen op te slaan in een speciaal gemakkelijk te hanteren formaat. Alle regelnummers zullen worden opgeborgen als gehele getallen (integers) van −9999 tot −1 (de inverse van het ingetikte regelnummer). Iedere regel zal bestaan uit een regelnummer, dan een getal van 1 tot en met 5 dat het soort opdracht aangeeft dan een aantal getallen dat van regel tot regel zal verschillen en tenslotte het getal −10000 dat het eind van de regel zal aangeven. De opdracht THEN zal worden opgeborgen als −10001, evenals puntkomma's in waardenlijsten. De binaire operatoren (bewerkingstekens) zullen als volgt worden opgeborgen:

\+ als 1
− als 2
< als 3
= als 4

Behalve de variabelen wordt verder niets meer in het geheugen opgeborgen. Variabelen zijn interessante dingen om mee te werken. Het is duidelijk dat het handig is variabele namen te hebben van een willekeurig aantal letters, zoals bijvoorbeeld de variabelenaam SOM. Handig is ook de mogelijkheid van namen bestaande uit een aantal letters gevolgd door een geheel getal, zoals bijvoorbeeld SOM1, SOM2 en SOM3. Gebruikers van Microsoft BASIC zijn bekend met de problemen die aan dit soort variabelenamen kleven. Een naam als TOOS is niet mogelijk omdat het gereserveerde woord TO erin zit. Verder neemt het gebruik van dit soort namen nogal wat geheugenruimte in beslag en maakt het de werking van de interpreter trager. Integer BASIC van de APPLE bijvoorbeeld loopt tijdens een "RUN" als de interpreter een variabelenaam tegenkomt een lange lijst namen af tot de betrokken variabele is gevonden. Hoe langer de namen zijn, hoe meer tijd dat kost. Microsoft past een andere methode toe door alleen de eerste twee letters van de naam in de lijst op te slaan, waardoor de interpreter sneller wordt. De namen in het programma worden echter nog letter voor letter opgeslagen, waardoor er wel meer geheugenruimte nodig is. Verder zijn de namen SOM1 en SOM2 voor deze interpreters identiek, wat het nut van lange namen wat begrenst. Nogal primitief lijkt me zo. Mijn interpreter zal een lijst met variabelenamen maken. Als een regel wordt ingevoerd zoekt de interpreter de lijst af. Als de naam nog niet in de lijst voorkomt, wordt hij aan het eind er aan toe gevoegd. De naam in de ingevoerde regel zelf, de zogenaamde variabele referentie wordt vervangen door een geheel getal, dat de positie van de betreffende variabelenaam in de lijst aangeeft. Dat spaart behoorlijk wat geheugenruimte uit. Er kan in de lijst ook ruimte worden gereserveerd voor de waarde van de betreffende variabele. Het afzoeken van de lijst vindt nu alleen plaats op het moment dat een regel wordt ingevoerd en niet tijdens het uitvoeren van het programma. Het gehele getal dat in de plaats is gekomen van de variabelenaam geeft aan waar een waarde moet worden gelezen of geschreven tijdens executie van het programma. De verwerkingssnelheid wordt door deze maatregel aanzienlijk opgevoerd. Microsoft moet daarentegen elke keer als er een variabelenaam tijdens executie voorkomt, de lijst afzoeken.

Wel, het is duidelijk dat we nu een snelle executie van het programma hebben, terwijl elke variabele, onafhankelijk van de lengte van de naam, evenveel geheugenruimte in het programma neemt.

De zojuist beschreven methode om een programma in het geheugen op te bergen is bruikbaar voor (a) het verminderen van de benodigde geheugenruimte om een regel "op te bergen" en (b) het verhogen van de executiesnelheid omdat de interpreter alleen naar nummers als "3" hoeft te kijken in plaats van reeksen letters als "PRINT". Er blijft echter nog één moeilijke operatie over gedurende het uitvoeren van een programma. De opdracht GOTO 50 maakt het noodzakelijk dat de plaats van regel 50 in het geheugen wordt gelocaliseerd. Er is gelukkig een eenvoudige oplossing voor dit probleem. Direct aan het begin van een "RUN" lopen we door alle regels en vervangen de regelnummers in de IF..THEN's en GOTO's door een getal dat de positie van de betreffende regel in het geheugen aangeeft. Deze "pointers" zullen worden opgeborgen als niet negatieve getallen zodat ze niet verward kunnen worden met de andere "echte" getallen, die opgeborgen zijn als negatieve getallen. Deze methode heeft tenminste twee voordelen: (a) het verhoogt de executiesnelheid en (b) het geeft de mogelijkheid te controleren op niet bestaande regelnummers, bijvoorbeeld een GOTO 50 terwijl regel 50 niet bestaat. Deze foutmelding kan nu gegeven worden voordat met de uitvoering van het programma wordt be-

gonnen zodat niet behoeft te worden gewacht op een daadwerkelijke uitvoering van de foutieve regel.
Het grootste nadeel van al dit woorden en getallen "kraken" is, dat het tijd kost. Sommige lieden zeggen dat het ook extra programmeerinspanning kost bij het maken van de interpreter, maar ik weet niet zeker hoeveel dat is. Gelukkig is deze tijd niet erg belangrijk, daar (a) elke regel maar één maal behoeft te worden "gekraakt" in plaats van elke keer dat hij uitgevoerd wordt en (b) de gebruiker de extra tijd vermoedelijk niet eens zal merken, omdat hij zich opmaakt de volgende regel te tikken. Als hij een lijnprinter gebruikt wordt het "kraken" vrijwel altijd gedaan terwijl de printer bezig is een regelopschuiving uit te voeren.

Microsoft: goede en slechte punten

We zijn nu klaar om de interpreter te schrijven. In werkelijkheid schreef ik de interpreter één maal, gooide hem weg en begon opnieuw, maar nu met een goed idee van wat er zou komen (dit is mijn standaard methode). Omdat ik een Apple computer bezit is de interpreter geschreven in applesoft, een versie van het bekende Microsoft BASIC. Omzetten naar andere Microsoft dialecten moet niet al te moeilijk zijn. Ik begon een aantal jaren geleden met programmeren met een HP 2000 timesharing systeem met een uitstekende BASIC. Het gebruik van strings in HP BASIC is veel gemakkelijker dan in Microsoft BASIC (plus het feit dat HP's methode wordt aanbevolen in de ANSI standaard).
Een voorbeeld: om het eerste teken van een string te verwijderen gebruik je bij HP: 100 A$ = A$(2). Bij Microsoft wordt dit
100 IF LEN(A$)=1 THEN A$ = " ":GOTO 120 en 110 A$ = RIGHT$(A$,LEN(A$)–1). Ik denk er zelfs maar niet aan om HP's 100 A$(4,7)=B$(20,23) te vertalen in Microsoft, vooral als de lengte van B$ ook 21 tekens mag zijn.
Dit moeilijke gedoe met Microsoft's strings bracht me er toe bij de tweede keer programmeren het gebruik van strings helemaal te vermijden (en dat geeft dan tevens een betere indruk hoe zo'n interpreter in machinetaal is geschreven). Een tweede probleem was dat bij een FOR NEXT loop de loop in Microsoft minstens éénmaal wordt doorlopen ook al is de gespecificeerde eindwaarde kleiner dan de startwaarde van de loopteller. Bij HP wordt de lus in zo'n geval helemaal niet uitgevoerd en zo hoort het ook. Om dit probleem te omzeilen moet ik een extra IF opdracht gebruiken. Om een idee te geven hoe vaak deze onnodige IF voorkomt zal ik drie *** zetten in de komende tekst op elke plaats waar ik er niet omheen kon. Om de interpreter in BASIC niet al te traag te maken moest ik ook alle subroutines aan het begin plaatsen in plaats van aan het einde, zoals iedereen zou willen doen. Dit omdat Microsoft niet eerst alle regelverwijzingen in de tekst opzoekt.
Verder had ik wat problemen met niet toegestane variabele namen, ik zal ze later verduidelijken voor diegenen die er plezier in hebben anderen interessante fouten te zien maken. Tenslotte waren er problemen met de scheiding van variabelen met komma's, zoals in INPUT A,B,C omdat Microsoft geen input van strings gescheiden door een komma toelaat. "Verbazing" komt zelfs niet in de buurt van mijn emoties toen ik dit ontdekte, als ik dit had geschreven zou ik me van schaamte verbergen in een stil hoekje.
(wordt vervolgd)

binnenin BASIC interpreter (3)

Dit is het derde deel van ons verhaal over een BASIC interpreter in BASIC. Het zal worden gevolgd door een beschrijving van de Microsoft BASIC interpreter voor de 6502. Tot zover is de beschrijving klaar voor de PET, CBM, OSI ROM en OSI DISK BASICS. Ondanks een aantal toezeggingen is helaas echter nog geen disassembly listing ontvangen van andere 6502 BASIC's. Voor een zo compleet mogelijk verhaal en vooral ook om het voor iedereen met een 6502 BASIC mogelijk te maken programma's voor zijn/haar machine gereed te maken hebben we die wel nodig. Iedereen, die beschikt over een dergelijke listing voor een 6502 BASIC gelieve contact op te nemen met de redactie software.
H.W.

Wat boekhouden als start

Laten we verder gaan met de eigenlijke interpreter die ons voor ogen stond. We moeten als eerste een aantal grenzen stellen aan de variabelen-grootte en dergelijke parameters. Door al dergelijke parameters bij elkaar te zetten aan het begin van het programma is een latere wijziging, indien nodig, gemakkelijker. We moeten in ieder geval de volgende parameters kennen: maximum programma lengte (MPROG), maximum regellengte (MLINE), maximum grootte van de variabelentabel (MTABLE), het maximum aantal tekens in een variabele-naam (MVAR) en de maximum regellengte gedurende een input (MIN). We moeten ook arrays dimensioneren en een aantal beginwaarden van variabelen gereedmaken. Een pointer is nodig om het begin van de vrije ruimte in PROG %, het programma geheugen (NPROG) aan te wijzen; evenals een pointer voor de start van de vrije ruimte in TABLE %, de variabelentabel (NVAR). Ook moeten we gereedmaken: RFLAG, een vlag voor de RUN-mode/ SYNTAX-mode en een tijdelijke variabele voor NVAR, QNVAR.
Het BASIC-interpreterprogramma begint nu als volgt:

```
10  MPROG=500 : MLINE=40 : MTABLE=500 :
    MVAR=40 : MIN=80
20  DIM IN(MIN+3),T1(MLINE-1),T2(MVAR-1),
    PROG%(MPROG), TABLE%(MTABLE-1)
25  NPROG=0 : NVAR=0 : RFLAG=0 : QNVAR=0 :
    EFLAG=0: GOTO 5000
```

Het %-teken is in sommige microsoft-BASICS niet beschikbaar. Het geeft aan, dat er sprake is van een integervariabele en bespaart bij arrays nogal wat geheugenruimte. Heeft Uw BASIC deze optie niet, laat dan het %-teken weg.
De arrays T1 en T2 worden gebruikt voor tijdelijke gegevenopslag. Als we de interpreter in machinetaal zouden schrijven (zoals eigenlijk gebruikelijk), dan zouden de arrays PROG% en TABLE% alle ongebruikte geheugenruimte bevatten.
De eerste stap is nu om een input-regel van de gebruiker te krijgen. We zullen een subroutine gebruiken om een string van de gebruiker te halen en deze in een array getallen om te zetten. In een machinetaal-interpreter zou deze string al direct een aantal getallen bevatten. Dit programmadeel is relatief eenvoudig:

```
110 INPUT" ";A$ : IF MIN<LEN(A$) THEN PRINT
    "INPUT REGEL TE LANG" : GOTO 110
114 IN(0)=0 : I=0 : IF LEN(A$) THEN FOR A=1 TO
    LEN(A$) : IN(A-1) = ASC(MID$(A$,A1)) : NEXT
    A : IN(A-1)=0
116 RETURN
```

Ik tikte eerst voor regel 110 : IF LEN(A$)>MIN THEN... maar dat vertaalde de computer als : IF LEN(A$)>M INT HEN, probeer het zelf maar eens. Ook *** in 114. Een nul wordt op de laatste plaats in het array gezet als afsluiter (in machinetaal zou dit bijvoorbeeld ook een CR-teken kunnen zijn). De variabele I wijst nu naar het eerste teken en wordt opgehoogd iedere keer als een teken is verwerkt.
Het eerste dat we met deze input-regel moeten doen is controleren of het eerste teken wel een cijfer is. Is dit het geval, dan is de input-regel een programma-regel, zo niet, dan is het een opdracht.

```
5000 NVAR=QVAR : PRINT"-"; : GOSUB 110 : IF
     IN(I)<48 OR IN(I)>57 THEN 5200
```

Laten we voorlopig aannemen dat de input-regel begint met een cijfer en regel 5000 dus niet de computer naar regel 5200 dirigeert. Het eerste wat ons nu te doen staat is de (ASCII) cijfers omzetten in één getal, zoals bijvoorbeeld 1, 2 en 3 omzetten naar het getal 123. Omdat dit omzetten wel vaker nodig zal zijn kunnen we er beter een subroutine van maken. De volgende routine accepteert getallen van -9999 tot 9999:

```
80  N=0 : N1 = 1 : IF IN(I)=45 THEN N1=-1 : I=I+1
90  N2=IN(I)-48 : IF N2 0<OR N2>9 OR N*10+N2>
    9999 THEN N=N*N1 : GOTO 30
100 N=N*10+N2 : I=I+1 : GOTO 90
```

De ASCII-waarde voor het '-' teken is 45, en de code voor '0' is 48. De routine wordt verlaten naar regel 30, die de start is van een subroutine om spaties (ASCII-waarde 32) te verwijderen:

```
30  IF IN(I)=32 THEN I=I+1 : GOTO 30
40  RETURN
```

De reden hiervoor is, dat er iets moet worden gedaan aan spaties. De meeste BASIC's verwijderen alle spaties die niet tussen aanhalingstekens staan of achter een REM opdracht. Dit geeft dan dwaasheden als het omzetten van MIN THEN in MINTHEN en dan in M INT HEN. Ik prefereer de spaties alleen te verwijderen tussen syntax-elementen. Omdat een getal een syntaxelement is, wordt de subroutine startend met regel 30 gebruikt om alle spaties na het getal te verwijderen.

Regelnummers

Ik breng in herinnering, dat alleen positieve getallen zijn toegestaan als regelnummers, en dat ze met behulp van negatieve getallen in het geheugen worden opgeslagen. De subroutine op regel 80 geeft N als geheel getal, dat nog moet worden gecontroleerd op positief of negatief en dan moet worden opgeborgen. De regel die zojuist is ingevoerd wordt voorlopig opgeborgen in het array T1 en de pointer PNT wijst het aan de beurt zijnde element aan. Omdat de regelnummers ook door een GOTO opdracht worden gebruikt, is er een subroutine nodig om een regelnummer te vertalen en op te bergen in het geheugen:

```
50  GOSUB 80 : IF N =OTHENEFLAG=1:RETURN
60  IF PNT=MLINE THENEFLAG=2:RETURN
70  T1(PNT)=N:PNT=PNT+1:RETURN
```

Regel 60 is een handige subroutine die N in het array T1 opbergt, na gecontroleerd te hebben of de regel niet te lang is. De variabele EFLAG wordt gebruikt om aan de subroutine oproeper te signaleren dat er een foutmelding is gedaan, zodat maatregelen kunnen worden genomen.
Hoe wordt nu deze regel-opberg-routine aangeroepen? Wel, eerst moeten we controleren of er regelnummers zijn, die omgezet zijn in pointers. Als dat het geval is moeten we ze terug omzetten naar regelnummers omdat er een nieuwe regel bij komt, zodat de posities van de regels in het geheugen kunnen veranderen en de pointers dan moeten worden gewijzigd. RFLAG wordt ongelijk aan nul gemaakt als er pointers zijn en gelijk aan nul als ze er niet zijn, zodat we de volgende regel kunnen gebruiken:

```
5010 IF RFLAG THEN GOSUB 340
```

Deze regel roept een subroutine op regel 340 aan die pointers omzet naar regelnummers als RFLAG ongelijk aan nul is. Nu krijgen we (we hebben alleen nog maar regelnummers):

```
5020 PNT=0 : GOSUB 50 : IF EFLAG THEN 6000
5025 IF NOT IN(I) THEN PNT=0 : GOTO 5090
```

Als EFLAG gelijk is aan 1 is er een foutmelding geweest en gaan we naar regel 5000, waar weer een nieuwe regel aan de gebruiker wordt gevraagd.
Als de afsluiter is gevonden na een regelnummer, dan moet er een regel worden verwijderd en wordt er doorverwezen naar regel 5090. Er was dan bijvoorbeeld alleen 100 (CR) ingetikt om regel 100 te verwijderen.
Op dit punt wordt de regel, of het nu een vervanging is van een bestaande programma-regel of een nieuwe programma-regel, gecontroleerd op syntaxfouten en daarna in het geheugen opgeborgen. Omdat alle regels met een opdracht beginnen, behalve LET, wordt de eerste stap te controleren of de regel begint met enig bekende opdracht. Is dat niet het geval, dan nemen we aan dat het LET is. De syntax die we hebben gegeven in deel 1, zie HCCN 17 pagina 21 wordt nu opgeborgen in de volgende DATA-regels:

```
9000 DATA 5,73,78,80,85,84,2,3,255
9010 DATA 5,80,82,73,78,84,3,4,255
9020 DATA 4,71,79,84,79,4,6,255
9030 DATA 5,71,79,32,84,79,4,6,255
9040 DATA 2,73,70,5,2,7,8,0
9050 DATA 1,1,5,2,255
```

Het eerste getal in elke regel is het aantal letters in het sleutelwoord. Deze worden gevolgd door dit aantal getallen, die uit de ASCII code voor het sleutelwoord bestaan. Hierna volgen de getallen zoals vermeld in deel 1, HCCN-17, pag. 21. Regel 9030 is dezelfde als regel 9020 maar met een extra getal in het sleutelwoord. Dit maakt het mogelijk zowel GOTO (regel 9020) als GO TO (regel 9030) in te voeren. Elke syntaxregel eindigt met of 255, of 0. De 0 betekent dat de volgende opdracht LET is, die geen sleutelwoord heeft.

Als eerste moeten we de DATA 'restoren', voor het geval we niet bezig zijn het allereerste sleutelwoord te ontcijferen. Daarna wordt de lengte van het sleutelwoord gelezen en gecontroleerd of dit overeenstemt met het sleutelwoord in de inputregel:

```
5030 RESTORE
5040 READ L: FOR A = 0 TO L-1: READ B : IF IN
     (I+A)=B THEN NEXT A:GOTO5070
```

Regel 5040 springt naar regel 5070 als de sleutelwoorden overeenstemmen. Is dit niet het geval, dan gaat de 'READ' door tot 255 of 0 is gevonden. Is het 255 dan gaan we terug naar 5040 om het volgende sleutelwoord te proberen.

```
5050 READ L : IF L = 255 THEN 5040
5060 IF L THEN 5050
```

We weten nu, op regel 5070, het type opdracht. Het volgende getal in de DATA regel bevat het opdrachten nummer, de volgende getallen de syntax. 'L' is het aantal tekens in het sleutelwoord en 0 als we geen sleutelwoord hebben gevonden dat overeenstemt met dat in de inputregel. In het laatste geval wordt nu verondersteld dat LET is bedoeld.
De DATA voor LET, regel 9050, bevat geen sleutelwoord en begint daarom met het opdrachtnummer (1). De volgende stap is in de input-regel over het sleutelwoord 'heen' te tellen en de eventuele spaties eveneens over te slaan. Dan moet het opdrachtennummer uit de DATA-regels worden gelezen en opgeborgen in het programmageheugen:

```
5070 I = I+L : GOSUB 30 : READ N : GOSUB 60 :
     IF EFLAG THEN 6000
```

Nu kan de syntax worden gelezen en gebruiken we een ON GOSUB om naar de benodigde subroutines te springen. Dit wordt dan herhaaldelijk gedaan tot een 255 is gevonden in de syntax regel.

```
5080 READ N : IF N = 255 THEN 5085
5081 ON N GOSUB 120,220,260,280,300,50,320,330:
     IF EFLAG = 99 THEN EFLAG=0 GOTO 5030
5083 IF EFLAG THEN 6000
5084 GOTO 5080
```

Als er een '255' is gevonden zakken we door tot de volgende regel, die -10000 in het programmageheugen opbergt. Deze -10000 markeert het einde van een regel. Als de input-regel geen einde heeft, springen we naar de foutroutine (regel 6000), waar de foutmelding wordt afgedrukt (in dit geval foutmelding nummer 7):

```
5085 N=-10000 : GOSUB 60 : IF EFLAG THEN 6000
5087 IF IN(I) THEN EFLAG=7: GOTO 6000
```

(wordt vervolgd)

binnenin BASIC interpreter (4

We vervolgen ons verhaal over een BASIC-BASIC interpreter overgenomen uit creative computing, september 1979. Aan het eind van deze serie zal het complete programma van de interpreter worden opgenomen.
H.W.

Syntax elementen

Laten we nu eens kijken naar elk van de syntax elementen. De eerste en meest ingewikkelde is de "variabele". De naam van de variabele moet in de opdracht worden geïsoleerd door te zoeken naar het eerste niet toegestane teken of het woord THEN. De niet toegestane tekens zijn: "+", "-", "<", "=" en ";". De eerste vier zijn niet toegestaan omdat ze binaire operatoren zijn. We willen immers de tekst "SOM1+SOM2" geïnterpreteerd zijn als de variabele "SOM1", de binaire operator "+", en de variabele "SOM2" en niet als één variabele "SOM1+SOM2". De puntkomma is illegaal omdat we de tekst "INPUT A;B" willen zien geïnterpreteerd als de opdracht "INPUT", de variabele "A", het scheidingsteken ";" en de variabele "B". Het woord THEN mag niet in de naam van een variabele voorkomen om "IFSUM1THEN ..." als de opdracht "IF SUM1 THEN ..." te kunnen zien. Verder mogen variabelen alleen met letters beginnen. Dit laatste dient om te voorkomen dat getallen als "-5" en "16" worden gezien als variabelenamen. Het programmeren van dit alles begint bij het controleren of het eerste teken van een variabelenaam een letter is:

```
● 120   VPNT = O : IF IN(I)< 65 OR IN(I)>90 THEN
          EFLAG = 3 : RETURN
```

"VPNT" is een pointer die in het voorlopige array T2 wijst waar de naam van de variabele wordt samengesteld (gecopieerd van de inputregel). Het wordt ook gebruikt voor het bepalen van het aantal tekens van een naam. Indien niet aan de voorwaarde wordt voldaan, wordt de foutvlag EFLAG gelijk aan 3 gemaakt. Het hoofdprogramma, dat deze subroutine aanroept kan dan een foutmelding als "ILLEGALE VARIABELE" geven en verdere noodzakelijke actie nemen. EFLAG = O betekent dat er geen fouten zijn geconstateerd, vóór het aanroepen van deze subroutine is deze vlag ergens in het hoofdprogramma op nul gezet.
Elk teken moet nu worden gecontroleerd op illegaliteit en het woord THEN:

```
● 130   A = IN(1) : IF NOT A OR A = 43 OR A = 45
●       OR A = 60 OR A = 61 OR A = 59 OR A = 84 AND
●       IN(I + 1) = 72 AND IN(I + 2) = 69 AND
●       IN(I + 3) = 78 THEN 150
● 140   T2(VPNT) = A : I = I + 1 : VPNT = VPNT + 1 :
●       IF VPNT > MVAR THEN 130
```

Merk op, dat O ook als een niet toegestaan teken geldt, daar deze het einde van een regel aangeeft. Het controleren gaat door tot er een illegaal teken of het woord THEN is gevonden, óf tot het maximaal aantal toegestane tekens in een naam is geaccumuleerd in variabele VPNT. Als er geen tekens in de naam zijn, d.w.z. de naam begon met THEN, dan moet er een fout worden afgedrukt, zie boven. Zo niet,

dan moeten de spaties, die aan de naam voorafgaan, worden verwijderd:

```
150     IF NOT VPNT THEN IN(I)=0: GOTO 120
155     IF T2 (VPNT - 1)=32 THEN VPNT=VPNT-1
        : GOTO 155
```

Nu moeten we controleren of de variabelenaam zich reeds in TABLE bevindt. Het eerste dat we in TABLE vinden is de waarde van de eerste variabele, het volgende de lengte van de variabelenaam, gevolgd door de variabelenaam, waarna al deze informatie wordt herhaald voor de volgende variabelen. TABLE begint met het element 0.

```
160     FOR A = 1 TO NVAR - 1 : IF TABLE(A)
        <= VPNT THEN 190
```

Als de lengtes niet gelijk zijn, kan het niet de goede naam zijn, als de lengtes wel kloppen, dan moet elk teken worden gecontroleerd:

```
170     FOR B=1 TO VPNT : IF TABLE(A+B)<>
        T2(B-1) THEN 190
180     NEXT B : N=A+9999 : GOSUB 60 : IF
        EFLAG<>0 THEN RETURN
181     GOTO 30
```

Als subroutine 60 een foutmelding geeft, wordt niet verder gegaan met verwerking. Als alle tekens van de variabelenaam overeenstemmen, dan wordt de waarde van de pointer opgeborgen en de variabele-subroutine wordt verlaten via regel 30. De opgeborgen waarde minus 9999 wijst naar de lengte van de variabelenaam en de opgeborgen waarde minus 1000 wijst naar de waarde van de variabele. Als de naam niet klopt, dan moet de volgende naam worden gecontroleerd:

```
190     A=A+TABLE(A)+1 : NEXT A
```

Als de naam nog niet in de tabel staat kijken we of er genoeg ruimte is om hem erbij te zetten. Zo ja, dan copiëren we de naam in de tabel en zetten de waarde van de variabele op nul:

```
200     IF NVAR+NPNT>MTABLE-2 THEN
        EFLAG=4: RETURN
210     TABLE(NVAR)=0 : FOR A=0 TO VPNT-1 ';
        TABLE(NVAR+A+2)=T2(A) : NEXT A :
        N=NVAR+1000 : NVAR=NVAR+VPNT+2
        : GOSUB 60 : IF EFLAG 0 THEN RETURN
211     GOTO 30
```

Oorspronkelijk schreef ik regel 200 als NVAR+NPNT+2 MTABLE THEN, wat gemakkelijker te begrijpen is, maar het einde daarvan werd MTAB LET HEN. We merken op, dat dit niet kan voorkomen met de subroutine die we zojuist hebben besproken, zelfs als we hadden gecontroleerd op LET in plaats van THEN, daar de spatie tussen MTABLE en THEN het vormen van LET voorkomt.

Het volgende syntaxelement is "waarde". Vaak gebruikt men hiervoor de termen "uitdrukking" of "wiskundige uitdrukking", maar "waarde" is veel eenvoudiger uit te spreken en te tikken. We beginnen met het controleren op "–" (ASCII 45) en de cijfers (ASCII 48 tot en met 57). Als één van deze wordt gevonden moet het een getal zijn, anders een variabele.

```
220     IF IN(I)=45 OR IN(I)>47 AND IN(I)<58
        THEN GOSUB 80 : GOSUB 60 : GOTO 240
```

De subroutine die een getal "vertaalt" bestaat al en begint met regel 80. Het getal wordt dan opgeborgen, waarbij we gebruik maken van de subroutine beginnend met regel 60. Als het een variabele is i.p.v. een getal wordt de "berg een variabele op" subroutine gebruikt, deze routine begint op regel 120:

```
230     GOSUB 120
```

Vervolgens wordt gecontroleerd of alles tot zover goed is verlopen d.m.v. EFLAG. Zo ja, dan controleren we, zowel voor een getal als een variabele of er nu een binaire operator aanwezig is. Als er geen is, dan is "waarde" klaar, anders wordt de waarde opgeborgen en beginnen we opnieuw met de controle op een getal of een variabele:

```
240     IF EFLAG<>0 THEN RETURN
245     A=IN(I) : N=(A=43)+2*(A=45)+3*(A=60)
        +4*(A=61): IF NOT N THEN RETURN
250     GOSUB 60 : IF EFLAG 0 THEN RETURN
255     GOSUB 30 : GOTO 220
```

Normaal gesproken gebruikt men DATA voor de verschillende operatoren, i.p.v. een aantal logische vergelijkingen (regel 245), maar de meeste Microsofts hebben geen methode om DATA te RESTORE-en vanaf een bepaalde regel. Ik wilde in ieder geval niet alle DATA RESTOREn en dan door de syntaxdata heenlezen tot ik de operator DATA had.

De volgende aan de beurt zijnde syntax is "variabelen lijst". De syntax is vrij eenvoudig en roept voornamelijk de "variabele" subroutine aan op regel 120. We controleren op "punt-komma", die de variabelen scheidt, maar er is geen enkele reden om die in het geheugen te zetten, dat kost alleen maar ruimte:

```
260     GOSUB 120: IF EFLAG<> THEN RETURN
265     IF IN(I)<>59 THEN 30
270     I=I+1 : GOSUB 30 : GOTO 260
```

"waarden lijst" is vrijwel hetzelfde, maar roept "waarde" aan (regel 220), puntkomma's moeten worden opgeborgen omdat anders kleine positieve getallen worden verward met operatoren:

```
280     GOSUB 220 : IF EFLAG 0 THEN RETURN
285     IF IN(I)<>59 THEN 30
290     I=I+1 : GOSUB 30 : N=-10001 : GOSUB 60
        : IF EFLAG<>0 THEN RETURN
295     GOTO 280
```

Het volgende syntaxelement is het "="-teken in de LET opdracht, we merken het wel op, maar bergen het niet in het programmageheugen op:

```
300     IF IN(I)<>61 THEN EFLAG = 5 : RETURN
310     I=I+1 : GOTO 30
```

De foutmelding in het hoofdprogramma geeft op "5" iets als: "= VERWACHT".

Het volgende syntaxelement is "regelnummer" die eenvoudig genoeg de subroutine aanroept die we al hebben op regel 50. (zie de ON GOSUB in regel 5080). THEN is het volgende element, dat wordt opgeborgen als –10001:

```
320     IF IN(I)<>84 OR IN(I+1)<>72 OR IN(I+2)
        <>69 OR OR IN(I+3)<>78 THEN EFLAG
        =6 : RETURN
325     I=I+4 : GOSUB 30 : N=-10001 : GOTO 60
```

De foutmelding op 6 geeft: "THEN VERWACHT" in het hoofdprogramma. En tenslotte hebben we nog "opdracht". Alles wat hier nodig is, is te vergeten, dat een subroutine werd aangeroepen (in regel 5080) en opnieuw te beginnen met de opdrachten routine. Daar PNT is verhoogd zal de opdracht automatisch worden opgeborgen na het begin van de IF opdracht, die we al hadden opgeborgen. IF is immers de enige opdracht die het syntaxelement "opdracht" gebruikt. We signaleren de "opdracht" syntax aan het hoofdprogramma door de foutflag EFLAG op 99 te zetten en terug te keren naar het hoofdprogramma:

```
330     EFLAG= 99 : RETURN
```

In het hoofdprogramma reageren we daarop met:

```
5081    ON N GOSUB 120,220,260,280,300,50,320,330
5082    IF EFLAG =99 THEN EFLAG=0 : GOTO
        5030
5083    IF EFLAG<>0 THEN 6000
5084    GOTO 5080
```

In dit deel hebben we dus nu alle benodigde subroutines voor de syntaxelementen behandeld. Ze worden in het hoofdprogramma aangeroepen in regel 5081.

```
● 6000    ON E FLAG GOTO 6001,6002,6003,6004,6005,
●         .....
● 6001    PRINT"FOUTMELDING 1" : GOTO 5000
● 6002    PRINT"FOUTMELDING 2" : GOTO 5000
```

enzovoorts. Pas als we het gehele programma klaar hebben kunnen we alle foutmeldingen aanvullen en weten we hoeveel dat er gaan worden. We zullen ons de volgende keer gaan bezig houden met het verwerken van een regel.

(wordt vervolgd)

binnenin BASIC interpreter (5)

Regelverwerking

Als de syntax van een regel geheel is gecontroleerd en de regel als een serie getallen is opgeslagen in array T1 zijn we gereed om hem op de juiste plaats in PROG% op te nemen. PROG% bevat immers het gehele programma. T1(0) bevat het regelnummer, of we nu een regel toevoegen aan of verwijderen uit het programma, zodra we bij regel 5090 zijn gearriveerd. PNT bevat de lengte van de regel (en is nul als we een regel deleten). Dit getal wordt gecopieerd in L (de afkorting voor "lengte") omdat we straks mogelijk een variabele nodig hebben die de hoeveelheid extra plaatsen bevat dat nodig is om bijvoorbeeld een regel te vervangen door een langere. Als we op dit ogenblik nog geen regels hebben in het programma zakken we door naar regel 5160 (•••). Anders zoeken we PROG% af naar de juiste plaats voor de regel, het z.g. "scannen" van PROG%.

```
O 5090    A=0 : PA = -10000 : L=PNT : IF L AND
          NOT NPROG THEN 5160
O 5100    FOR A = 0 TO NPROG-1 . IF PA =
          - 10000 AND PROG%(A) < = T1(0) THEN
          5120
O 5110    PA = PROG%(A) : NEXT A : PROG%(A) =
          0
```

Dat PA gedoe is ter voorkoming van problemen met PROG%(A - 1) voor het geval A = 0. De meeste BASIC's staan dat niet toe, en ik was dom genoeg om inderdaad het element 0 in het array te gebruiken. Als ik was begonnen met als eerste element van PROG% de geheugenplaats PROG% (1) te nemen, had ik PROG%(0) op - 10000 kunnen zetten en dan had "IF PA = -10000 AND PROG%(A) < = T1(0)" kunnen worden geschreven als het meer zinvolle "IF PROG%(A - 1) = -10000 AND PROG%(A) < = T1(0)", waardoor ik het gebruik van PA helemaal zou kunnen vermijden. PROG%(0) zou maar eenmaal helemaal aan het begin van het programma behoeven te worden gevuld. Hoe het ook zij, gewend als ik was aan het programmeren in machinetaal, waar we niet met arrays te maken hebben, zag ik het probleem niet totdat ik al een groot deel van het programma af had. Het werkt echter, dus laat ik deze niet elegante oplossing er maar in, zodat iedereen iets heeft om aanmerkingen op te maken.

Als de juiste regel helemaal niet wordt gevonden dan wordt de eerste net gebruikte geheugenplaats in PROG% op nul gezet (regel 5110) om het vervangen van regels die er niet zijn te voorkomen.

Als het einde van het programma of een regel met een hoger nummer dan de nieuwe regel is gevonden, komen we terecht op regel 5120, met A wijzend naar de hogere regel of het eerste ongebruikte element in PROG%. Als het regelnummer op plaats A in PROG% overeenkomt met het regelnummer van de nieuwe regel, dan moet er een bestaande regel worden verwijderd. Als de regelnummers niet overeenstemmen, dan moet er een nieuwe regel worden toegevoegd of de regel die verwijderd moet worden bestaat niet. In het laatste geval drukken we een foutmelding af:

```
O 5120    IF PROG%(A) = T1(0) THEN 5170
O 5130    IF NOT PNT THEN EFLAG = 8 : GOTO
          6000
```

Regel 6000 is de start van de routine die de foutmelding verzorgt waardoor EFLAG = 8 er toe leidt dat de melding "REGEL BESTAAT NIET!" wordt afgedrukt.

Regel 5140 is het begin van het tussenvoegproces, terwijl regel 5170 de start is van de routine die het vervangen en verwijderen regelt. Als er een regel moet worden toegevoegd, wordt eerst gecontroleerd of er voldoende ruimte is. Als dat het geval is, wordt er een "gat" gemaakt in PROG% door middel van regel 5150 (•••), en de regel wordt in dit "gat" gezet door opdracht 5160 (•••).

```
O 5140    IF NPROG + L > MPROG THEN EFLAG =
          9 : GOTO 6000
```

Fout nummer 9 luidt: "PROGRAMMA TE GROOT!"

```
O 5150    IF L THEN FOR B = NPROG - 1 TO A
          STEP -1: PROG%(B+L) = PROG%(B) :
          NEXT B
O 5160    NPROG = NPROG + L : QNVAR = NVAR
O 5165    IF PNT THEN FOR B = 0 TO PNT -1 :
          PROG%(A+B) = T1(B) : NEXT B
O 5167    GOTO 5000
```

Een wat merkwaardig gedoe in regel 5160 is "QNVAR = NVAR". We zagen iets dergelijks ook al helemaal aan het begin in regel 5000 ("NVAR = QNVAR"). Dit komt voornamelijk door een probleem in BASIC, dat we het "implied LET" statement noemen. Ik had oorspronkelijk de bedoeling QNVAR PNVAR te noemen, maar dat gaf problemen met PNT (microsoft brr....).

Als er een syntaxfout wordt geconstateerd, wordt NVAR teruggezet tot de voorgaande NVAR (QNVAR) door regel 5000. Dit verwijdert alle variabelen die bij TABLE% zijn gekomen gedurende het verwerken van de foutieve regel. Indien echter een goede regel wordt verwerkt maakt regel 5160 QNVAR gelijk aan NVAR zodat alle variabelen die aan TABLE% zijn toegevoegd behouden blijven als we regel 5000 uitvoeren.

Wat heeft dit allemaal nu te doen met de LET opdracht? Wel, veronderstel dat ik 10 GOSUB 50 intik. Er is helemaal geen GOSUB opdracht in de BASIC die we hier maken, dus daar geen van de sleutelwoorden overeenkomt met GOSUB wordt verondersteld dat we te maken hebben met een LET opdracht. GOSUB 50 is een toegestane naam voor een variabele, zodat deze wordt toegevoegd aan TABLE%. Daar echter nu een " = " wordt verwacht en er geen wordt gevonden zal de foutmelding " = VERWACHT " worden afgedrukt. (Dit is een nogal raadselachtige foutmelding, maar het is wat het syntax gedeelte van ons programma "denkt"). Het toevoegen van al deze variabelenamen aan TABLE% wordt met deze QNVAR truc voorkomen.

We kunnen de foutmelding aanzienlijk verbeteren door bij het afdrukken van de foutmeldingen er een PRINT LEFT$(A$, I+1) aan toe te voegen, zoals PRINT LEFT$(A$, I+1); " = VERWACHT ". We kunnen dit doen bij alle syntax fouten. In dit geval krijgen we dan de foutmelding:

```
10 GOSUB 50 = VERWACHT
```

Als U de fout verwijdert door de regel over te tikken en het = teken toe te voegen, dan krijgen we de volgende foutmelding:

```
10 GOSUB 50 = ILLEGALE VARIABELE
```

Dan voegt U er een legale variabele aan toe zoals 10 GOSUB 50 = SOM1, en "presto", daar hebben we een toegestane opdracht. Het zal U nu zolangzamerhand wel duidelijk worden dat het programma GOSUB 50 ziet als een variabele.

Terug naar de regelverwerking. Regel 5170 is de plaats waar de regels worden vervangen of verwijderd (in computer jargon: deleten). De variabele A wijst naar de oude regel of de te deleten regel. Het is nu noodzakelijk de lengte van de oude regel te bepalen.

Meestal beginnen interpreterschrijvers een regel in het ge-

leugen met de regellengte voor een groot aantal goede redenen. In dit geval deed ik dat niet:

```
O 5170    FOR B = 1 TO NPROG :IF PROG%(A+B
          -1) < > -10000 THEN NEXT B
```

De lengte van de oude regel staat nu in B. Als de nieuwe regel groter is dan die van de oude (of ze hebben dezelfde lengte), dan kunnen we de routine gebruiken, die we al voor het tussenzetten (computerjargon: "inserting") hebben geschreven. L moet worden gelijk gemaakt aan het verschil in lengte, en er wordt dan een "gat" gemaakt om de nieuwe regel te kunnen bevatten:

```
O 5180    IF B < = PNT THEN L = PNT - B :GOTO
          5140
```

Regel 5140 kijkt zelfs of er voldoende ruimte is om de nieuwe regel te bevatten. Als de nieuwe regel korter is dan de oude, of als er een regel moet worden verwijderd dan moeten we alles in PROG% opschuiven, zodat het ontstane gat wordt opgevuld. We laten voldoende ruimte over om de nieuwe regel op te nemen:

```
O 5190    FOR L = A + B TO NPROG - 1 :
          PROG%(L - B + PNT) = PROG%(L):
          NEXT L
O 5195    L = PNT - B : GOTO 5160
```

L moet worden gelijk gemaakt aan het verschil in lengte zodat NPROG juist wordt aangepast.

Foutmeldingen

We zijn al een aantal malen de foutmelding tegengekomen, nu alles codes te noteren voor toekomstig gebruik.

De programmering is vrijwel rechttoe rechtaan, met uitzondering van de foutcode 99, die afzonderlijk wordt behandeld. Deze foutmelding is eigenlijk geen fout, maar een methode om te vergeten dat er een subroutine is aangeroepen. De melding ontstond in regel 300, aan het einde van de vorige aflevering.

```
O 6000    HULP = EFLAG : EFLAG = 0
O 6005    IF HULP = 99 THEN 5030
O 6007    ON HULP GOTO 6010,6020,6030,6040,6050,
          6060,6070,6080,6090,6100,6110,6120,6130
O 6010    PRINT "FOUTIEF REGELNUMMER":
          GOTO 5000
O 6020    PRINT "REGEL TE LANG": GOTO 5000
O 6030    PRINT "ILL. VARIABELENAAM":
          GOTO 5000
O 6040    PRINT "TE VEEL VARIABELEN":
          GOTO 5000
O 6050    PRINT "= VERWACHT": GOTO 5000
O 6060    PRINT "THEN VERWACHT": GOTO 5000
O 6070    PRINT "EINDE REGEL VERWACHT":
          GOTO 5000
O 6080    PRINT "REGEL BESTAAT NIET":
          GOTO 5000
O 6090    PRINT "PROGRAMMA TE GROOT":
          GOTO 5000
O 6100    PRINT "NIET GEDEFINIEERDE REGEL IN
          REGEL"; - PROG%(L):GOTO 5000
O 6110    PRINT "ILLEGAAL COMMANDO":
          GOTO 5000
O 6120    PRINT "OVERFLOW IN REGEL";
          - PROG%: GOTO 5000
O 6130    PRINT "GESTOPT IN REGEL"; - PROG%:
          GOTO 5000
O 7000    END
```

We zullen de betreffende foutcodes behandelen op het moment dat ze worden gegenereerd.

(wordt vervolgd).

Nog meer boekhouden

In het vorige nummer hebben we de "syntaxing" afgesloten. We hebben nu een programma, dat opdrachten accepteert, deze in nummers omzet en de getallen in de juiste volgorde wegzet. We hebben nu een aantal subroutines nodig die regelnummers converteren in pointers en vice versa. De subroutine die getallen omzet in pointers ziet er als volgt uit:

```
O 410     PA = - 10000 + NOT NPROG : FOR A = 0 TO
          NPROG - 1: IF PA <'> - 10000 THEN 500
O 415     L = A
O 420     A = A + 1:IF PROG%(A) = 4 THEN IF
          PROG%(A + 1) < 0 THEN 460
O 430     IF PROG%(A) < >5 THEN 500
O 440     A = A + 1:IF PROG%(A) < > - 10001 THEN
          440
O 450     GOTO 420
O 460     C = PROG%(A + 1): PA = - 10000: FOR B = 0
          TO NPROG - 1: IF PA = - 10000 AND
          PROG%(B) = C THEN 490
O 470     PA = PROF%(B):NEXT B:IF RFLAG = 1
          THEN EFLAG = 10: RETURN
O 480     GOTO 500
O 490     PROG%(A + 1) = B
O 500     PA = PROG%(A):NEXT A:RETURN
```

Dat " + NOT NPROG" in regel 410 controleert op een zogenaamd null programma, d.w.z. een programma zonder opdrachten. (•••). Regel 420 controleert op opdracht type 4, de GOTO (de enige opdracht die een regelnummer heeft). Als GOTO is gevonden wijzigt regel 460 het regelnummer in een pointer, als dat al niet gebeurd was. Regel 430 controleert op opdracht type 5 (IF) omdat deze opdracht een nieuwe opdracht kan bevatten met daarin weer een GOTO of een andere IF. Nog onduidelijk is de controle op RFLAG in regel 470. Voorlopig is het voldoende te weten dat de opdracht RUN RFLAG op 1 zet. Foutcode 10 wil zoveel zeggen als "ongedefinieerde regelreferentie in regel ...".

De subroutine die het omgekeerde doet van de voorgaande ziet er zo uit:

```
O 340     PA = - 10000 + NOT NPROG: FOR A = 0
          TO NPROG - 1: IF PA < > - 10000 THEN 400
O 350     A = A + 1:IF PROG%(A) = 4 THEN IF
          PROG%(A + 1) > = 0 THEN 390
O 360     IF PROG%(A) < >5 THEN 400
O 370     A = A + 1:IF PROG%(A) < > - 10001 THEN
          370
O 380     GOTO 350
O 390     PROG%(A + 1) = PROG%(PROG%(A + 1))
O 400     PA = PROG%(A):NEXT A:RFLAG = 0:
          RETURN
```

Ook hier controleert regel 340 door " + NOT NPROG" op een null-programma (•••). Misschien herinnert U zich nog dat deze routine werd aangeroepen in regel 5010. Regel 400 reset RFIAG op 0 zodat deze routine niet opnieuw door regel 5010 wordt aangeroepen totdat het nodig is.

Alles wat we nu nog hebben te doen is de drie commando's uit te voeren. Als de eerste letter van een commando een L is, dan moet het het LIST commando zijn. Als er geen programma is, dat moet worden gelist, dan gaan we door middel van regel 5205 terug naar de routine die een input regel vraagt (•••).

```
O 5200    IF IN(I) < >76 THEN 5420
O 5205    IF NOT NPROG THEN 5000
```

Op regel 5420 komen we terecht als de eerste letter geen L is. Het regelnummer moet uiteraard vooraan de regel getikt worden:

```
O 5210    PRINT : FOR A = 0 TO NPROG - 1
O 5215    PRINT - PROG%(A);" "; : A = A + 1
```

Nu moeten we de opdrachtcode weer omzetten in het sleutelwoord door middel van dezelfde DATA opdrachten als bij het syntaxen van een inputregel:

```
O 5220    RESTORE
O 5230    READ L : FOR B = 1 TO L : READ T1(B) :
          NEXT B : READ B : IF B = PROG%(A)
          THEN 5260
O 5240    READ L : IF L< >255 AND L THEN 5240
O 5250    IF L THEN 5230
O 5255    READ B
```

Het sleutelwoord voor de opdracht is gevonden zodra we op regel 5260 zijn beland. We hebben het voorlopig opgeborgen in T1. L is de lengte van het sleutelwoord (of 0 voor LET). De volgende DATA die gelezen gaan worden geven de syntax voor de opdracht. Laten we eerst maar het sleutelwoord afdrukken (•••):

```
5260    IF L THEN FOR B = 1 TO L: PRINT
        CHR$(T1(B)); : NEXT B : PRINT" ";
5265    A = A + 1
```

Nu lezen we de syntax in en drukken de rest van de regel af. We gebruiken een ON GOSUB opdracht om de subroutines voor elke syntax aan te roepen:

```
5270    READ L : IF L = 255 THEN 5280
5275    ON L GOSUB 5290,5300,5340,5360,5380,5390,
        5400,5415
5276    IF EFLAG = 99 THEN EFLAG = 0:GOTO
        5220
5277    GOTO 5000
5280    PRINT : NEXT A : GOTO 5000
```

Als de hele regel klaar is komen we terecht op regel 5280, waardoor de zaak opnieuw begint, totdat het gehele programma is gelist.

Nu bekijken we elk syntax element eens.
Het eerste element is "variabele". We zetten geheugenplaats L zodat deze wijst naar de plaats van de lengte en naam van de variabele en drukken vervolgens met behulp van L de naam af:

```
O 5290    L = PROG%(A) - 9999 : FOR B = 1 TO
          TABLE%(L) : PRINT CHR$(TABLE%
          (B+L); : NEXT B : A = A + 1 : RETURN
```

Vervolgens hebben we "waarde". Als het eerste getal groter is dan 9999 dan hebben we een variabele en gebruiken we bovenstaande subroutine, zo niet, dan hebben we een getal en drukken we dat ongewijzigd af:

```
O 5300    IF PROG%(A)>9999 THEN GOSUB 5290 :
          GOTO 5320
O 5310    PRINT PROG%(A); : A = A + 1
O 5320    IF PROG%(A)<0 THEN RETURN
```

Op regel 5320 betekent een getal kleiner dan nul het einde van "waarde". Is dat niet het geval dan is er sprake van een binaire operator (zoals +) en beginnen we opnieuw.

```
O 5330    A$ = "+-<=" : PRINT MID$(A$,PROG%
          (A),1); : A = A + 1 : GOTO 5300
```

Vervolgens hebben we "waardenlijst", wat nogal triviaal is:

```
O 5340    GOSUB 5290 : IF PROG%(A)< >-10000
          THEN PRINT ";"; : A = A + 1 : GOTO
          5340
O 5350    RETURN
```

Op gelijke manier voor "waardenlijst" worden de waarden afgedrukt gescheiden door punt-komma's tot de regelafsluiter (-10000) is gevonden:

```
O 5360    GOSUB 5300 : IF PROG%(A) -10000 THEN
          PRINT ";" : A = A + 1 : GOTO 5360
O 5370    RETURN
```

Nu volgt een slimme oplossing voor het is gelijk teken in de LET opdracht:

```
O 5380    PRINT "=" ; : RETURN
```

Het volgende onderwerp is "regelnummer" en we moeten er rekening mee houden dat de regelnummers als negatieve getallen of als niet negatieve pointers zijn opgeborgen. We zouden GOSUB 340 hebben kunnen doen om ze allemaal om te zetten naar negatieve getallen, maar dat zou wel erg grof zijn.

```
O 5390    IF PROG%(A)<0 THEN PRINT
          -PROG%(A); :A = A + 1:RETURN
O 5400    PRINT -PROG%(PROG%(A)); : A = A +
          1 : RETURN
```

Nog een simpele:

```
O 5410    A = A + 1 : PRINT"THEN"; : RETURN
```

Tenslotte doen we "opdracht" op de gebruikelijke manier:

```
O 5415    EFLAG = 99:RETURN
```

(slot volgt)

binnenin BASIC interpreter (7)

Hieronder volgt het laatste deel van de tekst van ons lange artikel binnenin de basic interpreter. *Ik hoop dat U het interessant hebt gevonden. In het volgende nummer van onze HCCN volgt een kopie van het gehele programma. We zullen deze kopie plaatsen zoals deze door de computer is afgedrukt om zetfouten te voorkomen. Zoals U wellicht gemerkt zult hebben heeft het zetduiveltje ons hier en daar aardig parten gespeeld. Hoe het ook zij, de listing is afkomstig van een programma dat door mij uitvoerig is getest en geheel in orde bevonden. Het is bijzonder leerzaam het geheel in te tikken en uit te proberen. In het nieuwe jaar zullen we dan tenslotte starten met de bespreking van een aantal 6502 microsoft interpreters.*
Veel plezier met dit slot en de komende listing!
H.W.

Zetduiveltje

Dat het moeilijk is een programma foutloos langs de zetterij en de correctie te loodsen bewijst ook deze artikelen reeks. We zullen nu alvast alle correcties tot en met nummer 21 van de HCCN opgeven, zodat de ongeduldigen onder U nu het

komplete programma kunnen intikken. Een absoluut correcte versie volgt in het volgende nummer.

```
HCCN 19 blz 19: 5000 NVAR = QNVAR etc
                20: 50 GOSUB 80 : N = -N : IF N > = 0 etc
HCCN 20 blz 22: IF VPNT < MVAR in regel 140
                23: regel 210 voor FOR A etc tussenvoegen
                    TABLE (NVAR + 1) = VPNT
                    in dezelfde regel moet 1000 worden ver-
                    vangen door 10000
                23: regel 255 moet beginnen met I = I + 1
                    regel 260 vergelijkingstekens verwijderen
                    regel 270 moet starten met I = I + 1
HCCN 21 blz 23: regel 5100 < vervangen door >
                24: regel 6005 kan vervallen
```

Ten slotte nog dit: tik op Uw computer eens in de "direct" mode PRINT 1 = 1. Indien er dan 1 wordt afgedrukt op het scherm dan kunt U de gepubliceerde code ongewijzigd overnemen. Krijgt U echter -1, dan moet U alle opdrachten IF NOT XXX vervangen door IF XXX = 0 . . . ., de te publiceren code in het volgende nummer werkt voor beide soorten microsoft interpreters.

Vervolg

We vervolgen nu het artikel van Philip Tubb uit het september nummer van Creative Computing 1979:
Regel 5420 is de plaats waar we eindigen als er een commando is getikt dat niet met de letter L begint. Als het commando óók niet met R begint moet er iets mis zijn.
5420 IF IN(1) < > 82 THEN EFLAG = 11 : GOTO 6000
EFLAG = 11 geeft een foutmelding als "illegaal commando".

Nu hebben we nog de keus uit RENUMBER of RUN. Als de tweede letter een E is, dan hebben we te maken met RENUMBER, zo niet dan springen we naar de code voor RUN.

```
5430 A = IN(I + 1) : IF A < > 69 THEN 5460
```

U zult zich misschien afvragen waarom ik besloot RENUMBER op te nemen. Iedereen weet, dat renumber behoorlijk moeilijk te doen is, omdat je ook alle GOTO verwijzingen moet veranderen. In feite is het zo moeilijk, dat de meeste personal computers het niet hebben.

Ik deed het op een eenvoudige wijze, om ze allemaal voor gek te zetten. Eerst roepen we dezelfde routine aan, die RUN zal gebruiken om regelnummers om te zetten in pointers, daarna veranderen we alleen de regelnummers en vergeten gewoon alles van de GOTO verwijzingen.

```
5440 RFLAG = 2 : GOSUB 410 : L = —10 : PA = —10000
5445 FOR A = O TO NPROG —1 : IF PA = —10000
THEN PROG% (A) = L: L = 1—10
5450 PA = PROG% (A) : NEXT A : GOTO 5000
```

RFLAG dient ervoor om te zorgen dat er geen foutmelding wordt gegeven bij een ongedefinieerde regel verwijzing. Wordt dezelfde routine door RUN aangeroepen (met een andere RFLAG) dan wordt er wel een foutmelding gegeven. Ongedefinieerde regelnummers worden negatieve getallen gelaten als RFLAG gelijk is aan twee.

Daar de GOTO regelnummers naar de echte regelnummers verwijzen, en we deze laatste zojuist gewijzigd hebben, worden de regelnummers in de GOTO's veranderd, zonder dat we er iets voor hoeven te doen.

Bovenstaande routine hernummert met stappen van 10, maar kan eenvoudig worden gewijzigd.

Nu volgt de code voor het RUN commando, te beginnen met regelnummer 5460:

```
5460 IF A < > 85 THEN IN(I) = 0 : GOTO 5420
5470 RFLAG = 1 : GOSUB 410 : IF EFLAG THEN 6000
5472 PNT = 0 : GOTO 620
```

Regel 5460 controleert of de tweede letter van het commando wel een U is. Het is duidelijk, dat we het grootste gedeelte van de code voor het RUN commando aan het begin van het programma hebben geplaatst. Op regel 620 wordt de pointer PNT naar het regelnummer, dat in bewerking is, gecopieerd in LINE voor toekomstig gebruik.

Als er geen regels meer te verwerken zijn wordt er gesprongen naar het gedeelte, dat de invoer van een nieuwe regel verzorgt, regels 5000 en verder:

```
620 LINE = PNT : PNT = PNT + 2 : IF LINE = NPROG
    THEN 5000
```

PNT wordt met twee verhoogd, zodat deze pointer nu naar de eerste parameter wijst (voorbij het opdrachttype nummer). We gebruiken een ON GOTO opdracht om naar de juiste routines te springen:

```
630 ON PROG% (PNT—1) GOTO 670, 680, 740, 760
```

We beginnen met het IF statement. Dit type opdracht moet de waarde van een gegeven "waarde" bepalen. Een subroutine beginnend op regel 510 wordt hiervoor gebruikt. Deze routine evalueert "waarde" en "returns" met het antwoord in de variabele N.

```
510 GOSUB 590 : N = A
```

Regel 590 "returns" met de waarde van een getal of variabele in A.

```
520 B = PROG% (PNT) : IF B < 0 THEN RETURN
```

De waarde is compleet als er geen binaire operator op volgt. Anders evalueren we de tweede variabele (óf getal) en gebruiken een ON GOSUB om naar de juiste code voor de binaire operator te springen.

```
530 PNT = PNT + 1 : GOSUB590 : ON B GOSUB 550,
    560, 570, 580
535 IF ABS(N) < 10000 THEN 520
540 EFLAG = 12 : RETURN
550 N = N + A : RETURN
560 N = N — A : RETURN
570 N = N    A : RETURN
580 N = N = A : RETURN
```



```
590 A = PROG% (PNT) : PNT = PNT + 1 : IFA < 10000
    THEN RETURN
600 A = TABLE% (A—10000) : RETURN
```

Als we eenmaal het IF gedeelte hebben gedaan vervolgen we met de opdracht na THEN, indien de waarde evaluatie van de evaluatie ongelijk aan nul is:

```
640 GOSUB 510 : IF EFLAG THEN 6000
641 PNT = PNT + 2 : IF N THEN 630
```

Indien de evaluatie van de IF niet op nul uit kwam vervolgen we met de volgende regel:

```
650 PNT = PNT + 1 : IF PROG% (PNT—1) < > —10000
    THEN 650
660 GOTO 620
```

Nu het LET statement. Dit is relatief eenvoudig. L is de pointer naar de variabele waaraan een waarde wordt toegekend, dus links van het = teken:

```
670 L = PROG% (PNT)—10000 : PNT = PNT + 1 :
    GOSUB 510 : IF EFLAG THEN 6000
675 TABLE% (L) = N : GOTO 610
```

Regel 610 is een regel om de pointer PNT op te hogen, daarna begint de hele zaak weer van voren af aan:

```
610 PNT = PNT + 1
```

Nu INPUT. Deze opdracht leest een regel in (routine op regel 110 e.v.). Als de eerste letter van de input een S is stoppen we de uitvoering van het programma en springen naar het gedeelte dat de invoer van een nieuwe opdracht verzorgt (regels 5000 e.v.):

```
680 GOSUB 110 : IF IN(I) = 83 THEN EFLAG = 13 : GO-
    TO 6000
```

Nu onderzoeken we vervolgens of er nog letters in de inputregel zijn overgebleven, zo niet dan gaan we terug om er nog wat meer te krijgen:

```
690 IF NOT IN(I) THEN 680
```

Als het eerste teken niet een "-" is (ASCII 45) of een cijfer (ASCII 48 tot en met 57) dan drukken we een foutmelding af, en vragen opnieuw input:

```
700 IF IN(I) < > 45 AND(IN (I) < 48 OR IN(I) > 57) THEN
    PRINT "FOUTE INPUT" : GOTO 680
```

Anders vertalen we het getal (regel 80) en bergen het op in de juiste variabele. Als er geen andere variabelen zijn, dan zijn we klaar en springen naar regel 610:

```
710 GOSUB 80 : TABLE% (PROG% (PNT)—10000) = N :
    PNT = PNT + 1 : IFPROG% (PNT) = —10000
    THEN 610
```

Als dat niet het geval is, kijken we of er een punt-komma in de INPUT regel staat en gaan we door met de volgende variabele als dat het geval is. (de punt-komma dient om de variabelen te scheiden in een INPUT opdracht, ik zou komma's prefereren, maar dat staat microsoft niet toe).

```
720 IF IN(I) < > 59 THEN IN(I) = 0 : GOTO 700
730 I = I + 1 : GOTO 690
```

De PRINT opdracht is erg eenvoudig:

```
740 GOSUB 510 : PRINT N : IF PROG% (PNT) = —10000
    THEN PRINT : GOTO 610
750 PNT = PNT + 1 : GOTO 740
```

De GOTO opdracht is eveneens erg eenvoudig:

```
760 PNT = PROG% (PNT) : GOTO 620
```

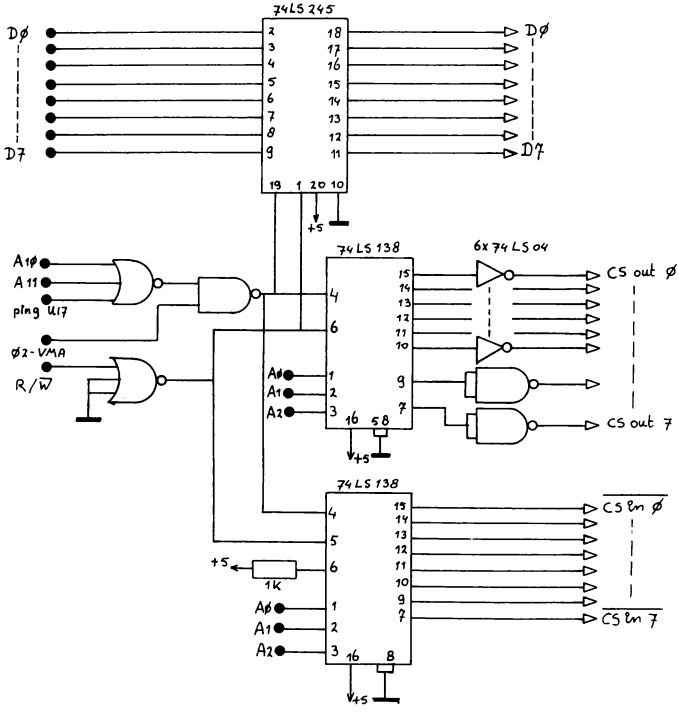
Besluit

En dat was het dan. In werkelijkheid is het natuurlijk een stuk gecompliceerder, maar deze interpreter is bedoeld om de basis principes te laten zien. U kunt de interpreter intikken en uitproberen als U dat wilt, mogelijk dat U daardoor een (nog) beter inzicht krijgt in de gang van zaken. De verwerking is erg traag, en U zult merken dat het aardig wat tijd kost om een ingetikte regel te controleren op een juiste syntax. Bedenk, dat een goed geschreven interpreter in machinetaal deze hele controle gemakkelijk in een fractie van een seconde kan doen. Dit zet de snelheid van de opdracht RUN in juiste proporties.

De volgende aflevering zullen we als slot de listing in zijn geheel plaatsen van deze interpreter, zoals die op de meeste microsoft interpreters zal draaien. We zullen dan tevens een aantal ondervonden problemen bespreken om de zaak draaiende te krijgen.

Input/Output-poort voor het OSI-Superbord

Met bijgaand schema is het mogelijk het Superbord te voorzien van maximaal 8 input- en/of 8 outputpoorten. Met een 74LS245 worden de datalijnen gebufferd. Het grof selecteren gebeurt met de adreslijnen A10 en A11 en één van de uitgangen van IC U17 (een 74LS139, zie schema in user manual). Van dit IC zijn de pinnen 9,10 en 11 nog niet in gebruik zodat er zonder meer 3 van deze schakelingen aan het Superbord zijn vast te knopen. Voor het selecteren van één van de 8 poorten zijn dan nog 3 lagere adreslijnen nodig : A0, A1 en A2. Deze sturen 2 74LS138 IC's (één voor de aansturing van de output- en één voor de inputpoorten). Het R/$\overline{W}$ signaal bepaalt dan of de input- of de outputpoorten geselecteerd worden. De 74LS138 IC's produceren een (laaggaand) signaal op één van de 8 uitgangen, zodat dat gebruikt kan worden als chip select voor de poorten.

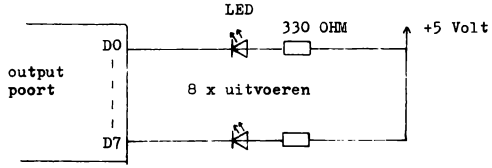


buffering en decodering

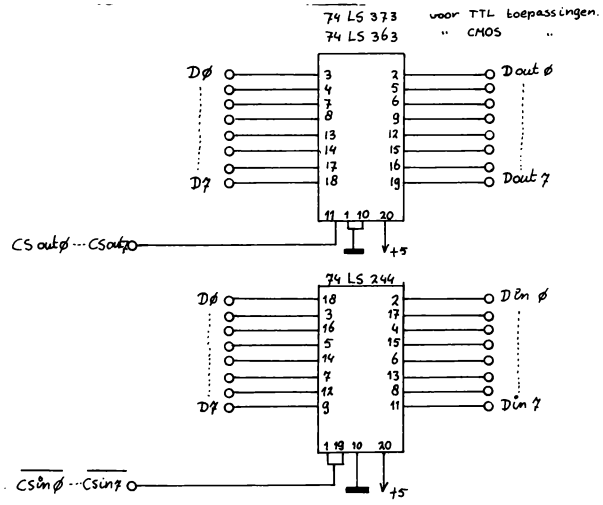
Voor de outputpoorten dienen deze laaggaande signalen helaas nog geïnverteerd te worden. Op welk adres de poorten terecht komen hangt af van het gebruikte pootje van U17 :

	adres	
	decimaal	hexadecimaal
pin 11	61696...61703	$F100...$F107
pin 10	61952...61959	$F200...$F207
pin 9	62208...62215	$F300...$F307

Een voorbeeld van een "zinnig" gebruik van de I/O poorten is een looplicht metregelbare snelheid. Hiervoor moet op de outputpoort van adres 61696 een achttal LED's met weerstanden aangesloten worden. De schakeling ziet er als volgt uit :



De snelheid kan geregeld worden door bij de inputpoort met hetzelfde adres als de outputpoort een aantal ingangen aan massa te leggen. Het BASIC programma loopt als volgt :



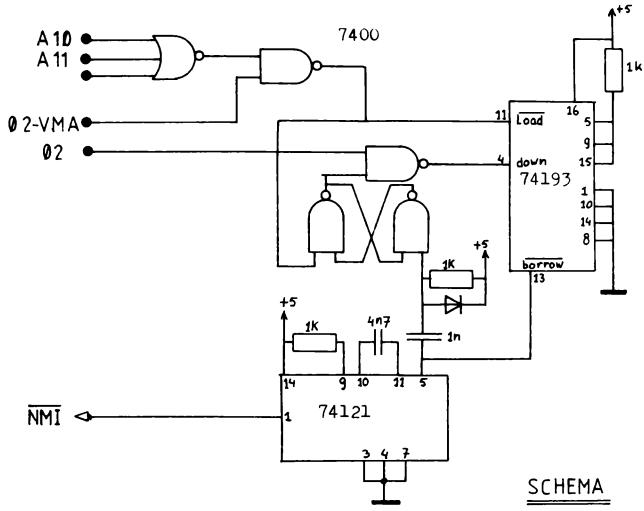
1 input + outputpoort

```
 10 INPUT "Aantal keer ";I
 20 FOR J=1 TO I
 30 FOR T=0 TO 7
 40 POKE 61696, (255-2^T)
 50 SN=(256-PEEK(61696))*10
 60 FOR G=0 TO SN : NEXT G
 70 NEXT T
 80 PRINT "Loop nummer :";I-J+1
 90 NEXT J
100 END
```

SINGLE-STEP voor het OSI-Superboard

Met bijgaand programma+schema kunnen machinetaal programma's instructie voor instructie worden uitgevoerd, terwijl de registers op het scherm worden geprint.
De benodigde electronica is ook door liefhebbers met een laag budget te bekostigen aangezien een en ander uit slechts 3 TTL IC's is opgebouwd. (de NOR-poort is niet meegerekend aangezien die overblijft wanneer de I/o-poorten worden gebouwd).
Verder zijn er een aantal signalen uit de computer nodig t.w. het kloksignaal Ø2, de adreslijnen A1Ø en A11, het besturingssignaal Ø2-VMA en één v/d uitgangen van IC U17 (een 74LS139).
Wordt de schakeling geselecteerd dan gebeurd het volgende:
De up/down teller 74193 leest z'n data in (op de pinnen 1,9,10 en 15) en begint de klokcycles te tellen, na 10 cycles volgt een laaggaand signaal op de borrow-uitgang. Dit signaal wordt met de 74121 verlengd tot een voor de computer bruikbaar NMI-signaal.
Door dit NMI-signaal wordt de instructie waar het programma mee bezig was nog afgemaakt maar daarna wordt via de NMI-vector naar het hulpprogramma gesprongen waar de registers op het scherm worden gezet.
Dit hulpprogramma maakt daarbij gebruik v/d print subroutine die ook door BASIC gebruikt wordt.
Die staat voor het normale beeldscherm op adres $FF69 en voor het uitgebreide beeldscherm op $Ø222.
Het NMI-signaal moet worden aangesloten op aansluiting 2 v/d uitbreidingsconnector (rechts naast die chip met de vele pootjes).
Verder moet er bij de NMI-vector een jump naar het hulpprogramma staan dus:

NMI: Ø13Ø 4C 2Ø Ø9



Software voor SINGLE-STEP

Volgorde van handelingen bij het opstartprogramma :
Eerst de registers voor het gebruikersprogramma opladen, dan de teller op adres $F200 starten en naar het gebruikersprogramma gaan. de inhouden van de registers moeten de eerste keer op de desbetreffende geheugenplaatsen gezet worden. De gebruikte geheugenplaatsen voor de registers zijn :

LABEL	ADRES	OPMERKING
SSAV	$00F0	inhoud stackpointer
YSAV	F1	Y register
XSAV	F2	X register
PSAV	F3	flags (P register)
ASAV	F4	accumulator
VPSAV	F5	intern gebruikt voor de opcode van de volgende instructie
PLSAV	F6	laagste byte PC
PHSAV	F7	hoogste byte PC
PSAV2	F8	intern gebruikte buffer voor de flags

Opstartprogramma :

```
0900 A6 F0    SINST  LDX SSAV ;X=SP
0902 E8              INX ;X=X+3
0903 E8              INX
0904 E8              INX
0905 9A              TXS ;nu naar SP
0906 A5 F7           LDA PHSAV ;zet hoogste byte PC in accu
0908 48              PHA ;en op de stack
0909 A5 F6           LDA PLSAV ;en laagste byte
090B 48              PHA ;en op de stack
090C A5 F3           LDA PSAV ;ook zo met de flags
090E 48              PHA
090F A6 F2           LDX XSAV ;laad X register
0911 A4 F1           LDY YSAV ;en Y register
0913 8D 00 F2        STA TELLER ;start teller door adres $F200
                                ;te selecteren
0916 A5 F4           LDA ASAV   ;laad accu
0918 40              RTI        ;laad de flags (P register)
                                ;en ga naar gebruikersprogramma
```

Het volgende programma moet na een NMI doorlopen worden. Hiervoor moet op adres $0130 JMP NMISS staan. In dit geval dus :
0130 4C 20 09
Het programma zet de registers en de geheugenplaatsen.

```
0920 85 F4    NMISS  STA ASAV ;bewaar accu
0922 86 F2           STX XSAV ;en X
0924 84 F1           STY YSAV ;en Y
0926 68              PLA      ;haal flags van stack
0927 85 F3           sta PSAV ;bewaar in geheugen
0929 68              PLA      ;laagste byte PC van stack
092A 85 F6           STA PLSAV ;en naar geheugen
092C 68              PLA      ;en hoogste byte PC van stack
092D 85 F7           STA PHSAV ;etc
092F A2 00           LDX #0   ;maak index 0
0931 A1 F6           LDA (PLSAV,X) ;haal opcode byte van de
                                   ;volgende instructie
0933 85 F5           STA VPSAV     ;en naar geheugen
0935 BA              TSX      ;SP naar X
0936 CA              DEX      ;X=X-3 voor de echte SP
0937 CA              DEX
0938 CA              DEX
0939 9A              TXS      ;zet weer in SP
093A 86 F0           STX SSAV ;en in geheugen
093C 4xEA            4xNOP
```

Het volgende programmagedeelte print de registers die op hun desbetreffende geheugenplaatsen staan op het scherm.

De volgorde is : PC (PC) A P X Y S flags

```
0940 20 1A 08 PRTREG JSR CRLF  ;nieuwe regel
0943 A2 06           LDX #6    ;aantal registers
0945 A5 F7           LDA PHSAV ;eerst hoogste byte PC
0947 20 00 08        JSR PRTA  ;print accu hexadecimaal
094A B5 F0    LOOP   LDA (SSAV,X) ;haal register byte op
094C 20 00 08        JSR PRTA  ;en print het
094F 20 15 08        JSR SPACE ;spatie erachter
0952 CA              DEX       ;en de volgende
0953 D0 F5           BNE LOOP  ;laatste ?
0955 A9 01           LDA #1    ;hoogste byte SP is 1
0957 20 00 08        JSR PRTA  ;en print het
095A A5 F0           LDA SSAV  ;en nu het laagste byte
095C 20 00 08        JSR PRTA
095F 20 15 08        JSR SPACE
0962 A5 F3           LDA PSAV  ;haal de flags op
0964 85 F8           STA PSAV2 ;zet in buffer
0966 A2 08           LDX #8    ;aantal flags
0968 66 F8    LP     ROR PSAV2 ;haal een bit in de carry
096A 90 0A           BCC CONT  ;geset ?
096C BD 80 09        LDA TABFL,X ;ja haal ASCII waarde in
                                 ;accu
096F 86 F5           STX VPSAV ;bewaar X register
0971 20 22 02        JSR PRINT ;print accu op scherm
0974 A6 F5           LDX VPSAV ;herstel X weer
0976 CA       CONT   DEX       ;volgende flag
0977 D0 EE           BNE LP    ;al klaar ?
0979 20 00 FD        JSR GETKEY ;wacht op toets
097C 4C 00 09        JMP SINST ;en begin helemaal opnieuw
097F EA              NOP

0980 4E 56    TABFL  N V       ;tabel met ASCII waarden
0982 00 42           - B       ;van de flags
0984 44 49           D I
0986 5A 43           Z C
```

De volgende subroutines worden door het programma gebruikt voor de uitvoer naar het scherm.
De eerste print de accu hexadecimaal op het scherm.

```
0800 48        PRTA   PHA        ;bewaar accu op de stack
0801 4A               LSR A      ;roteer hoogste vier bits
0802 4A               LSR        ;naar de onderste vier
0803 4A               LSR
0804 4A               LSR
0805 20 0B 08         JSR PRTAL  ;en print ze op scherm
0808 68               PLA        ;herstel de accu
0809 29 0F            AND #15    ;alleen laagste vier bits
080B F8        PRTAL  SED        ;voor BCD rekenen
080C 18               CLC        ;maak carry nul voor ADC
080D 69 90            ADC #$90   ;test groter dan 9
080F 69 40            ADC #$40   ;maak ASCII waarde
0811 D8               CLD        ;reset BCD mode
0812 4C 22 02         JMP PRINT  ;en zet het op het scherm

0815 A9 20     SPACE  LDA #" "   ;print een spatie
0817 4C 22 02         JMP PRINT

081A A9 0A     CRLF   LDA #10    ;nieuwe regel
081C 20 22 02         JSR PRINT
081F A9 0D            LDA #13    ;naar begin van regel
0821 4C 22 02         JMP PRINT
```

De programma's en de hardware zijn getest op een OSI-Superboard met een klokfrequentie van 2 Mhz en uitgebreid video-display van $D000-$D700. Bij lange aansluitdraden dienen pull-up weerstanden gebruikt te worden.

Auteurs van deze artikelen:

M.J. Looyen
Hendrikstr. 25
8121 BE Olst
tel. 05708-1280

A.G. Megens
Castorstr. 22
7771 XR Hardenberg
tel. 05232-4532

Wijzigingen en aanvullingen op de ledenlijst van het 1e OSI-UG NO P-Boek
naar de stand van de HCC administratie per 26 februari 1981.

Een volledige ledenlijst van de OSI-GC zal weer verschijnen als de resultaten
van de enquete in de hoofdnieuwsbrief van maart 1981 beschikbaar zijn.
Werkt allen mee aan deze enquete.