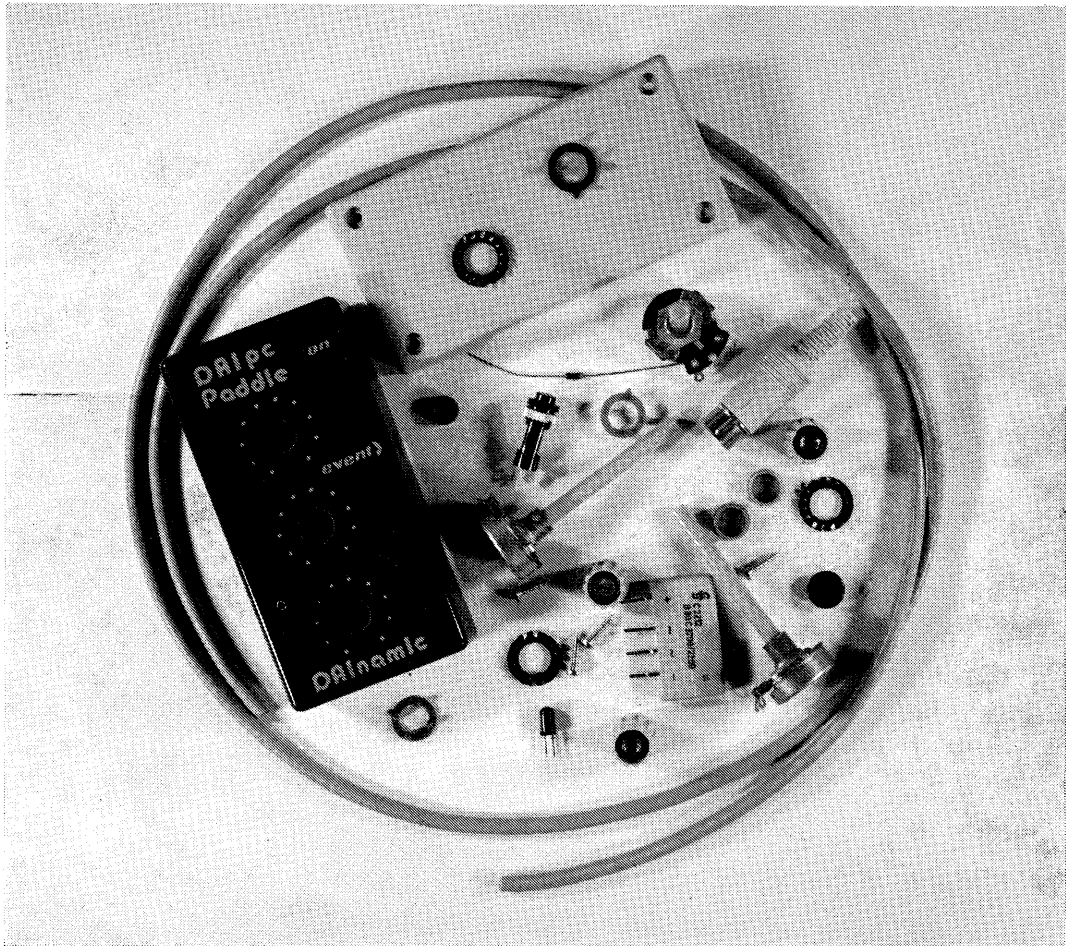


DAI

Dynamic

Nr 7 SEPT-OKT 1981



GEDRUKTE PERIODIEK verschijnt tweemaandelijks

Verantw. Uitgever : W. HERMANS HEIDE 98 3171 WESTMEERBEEK

COLOFON

DAInamic verschijnt tweemaandelijks.
 abonnementsprijs is inbegrepen in de
 jaarlijkse contributie:

750 Bfr 50 Gld 50 Dm

Bij toetreding worden de verschenen
 nummers van de jaargang toegezonden.

DAInamic redactie:

Dirk Bonné

Freddy De Raedt

Wilfried Hermans

Jules Meulenbergs

Jos Schepens

Roger Theeuws

Bruno Van Rompaey

Jef Verwimp

vormgeving :Ludo van Mechelen

U wordt lid door storting van de
 contributie op nr406-3016141-33 van
 KREDIETBANK WESTMEERBEEK, via bank-
 instelling of POSTGIRO.

Abonnement loopt van januari tot
 december.

U kan telefonisch contact nemen op
 nr 016/698623.

correspondentieadres:

DAInamic

Heide 98

3171 WESTMEERBEEK BELGIE

DAInamic verschijnt de eerste week van
 de pare maanden.

Bijdragen zijn steeds welkom.

4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15

belangrijke ASCII-waarden in DAInamic

functie/symbool	HEX	DEC
back-space	8	8
TAB	9	9
linefeed	A	10
clear screen	C	12
CURSOR UP	10	16
CURSOR DOWN	11	17
CURSOR LEFT	12	18
CURSOR RIGHT	13	19
space-bar	20	32
Ø	30	48
A	41	65
a	61	97
pijltje rechts	89	137
pijltje links	88	136
pijltje boven	5E	94
pijltje onder	8C	140
volle blok	FF	255
verticale lijn	A	10
horizontale lijn	B	11
6 hor.lijnen	1D	29

ASCII - HEX - ASCII CONVERSION TABLE

LSD	MSD	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	,	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	§	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	*	7	G	W	g	w
8	1000	BS	CAN	(8	H	X	h	x
9	1001	HT	EM)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[k	{
C	1100	FF	FS	,	<	L	\	l	
D	1101	CR	GS	-	=	M]	m	}
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

Dear members,

Less paper, more information : that's what we try to bring with our new lay-out format. We hope that the smaller charactersize of some articles doesn't give any inconvenience, otherwise try glasses...

We hebben ons best gedaan om deze uitgave voor 4 oktober te bezorgen.

De heer Van Lieshout meldde ons namelijk:

De NOS zendt in het radioprogramma "HOBBYSCOOP" elke week een computer-programma uit, telkens voor een ander merk. Na enige correspondentie met de NOS en andere computerfanaten is er een werkgroep ontstaan.

Hieruit is een universele codering ontstaan voor computerprogramma's. Ieder toestel heeft een "bootstrap"-programma nodig om deze code in te lezen en daarna om te zetten in de specifieke BASIC-code.

De uitzending van deze programma's is gepland voor volgende data: 23 sept EXIDY, 30 sept PET/PHILIPS en 4 okt DAI. Op 4 okt kan U dus het programma opnemen dat uitgezonden wordt in normaal DAI-formaat.

Hiermee zal U dan de toekomstige universele programma's kunnen inlezen.

*Er wordt die dag ook een documentatie-programma uitgezonden. Heden hebben we nog niet de volledige informatie, we zullen dit onderwerp verder behandelen in NEWSLETTER 8. Voor dringende informatie kan U terecht bij dH Van Lieshout tel 02297-2648.

Golflengte en tijdstip : HILVERSUM 1 19.30 u.

In HILLEGOM zijn DAInamic-bijeenkomsten georganiseerd: geïnteresseerden kunnen contact nemen met TOM NIEUWENHUIZEN POSTBUS 28 2100 AA HEEMSTEDE. DAInamic zal vertegenwoordigd zijn op de 5e HCCMICROCOMPUTERDAG op 28 november 81, Julianahal, Jaarbeurs UTRECHT 10.00 - 17.00 uur.

Ondertussen zullen DCR en Floppy geleverd zijn, onze ervaringen (en de uw) vindt U in NEWSLETTER 8.

Een tyfout in het vorige nummer : de titels van MUSIC COLLECTION 1 & 2 waren verwisseld. We zijn hard aan het experimenteren met screen-foto's, mogelijk biedt NS U een klare kijk op de beschikbare programma's! Onze klassieke dank aan alle medewerkers, we hopen dat dit nummer U brengt wat U ervan verwachtte...

In verband met familieleven, nachtrust en andere noodzakelijkheden ontvangen we graag telefoontjes tussen 18.00 en 20.00 .

tot kijk, tot schrijfs, tot aan de lijn

de redactie

179	remark	bladwijzer
180	8080 simulator +++ comments	8080 simulator +++ variabelen atlas
181	8080 simulator BASIC program	8080 simulator BASIC page 2
182	8080 simulator BASIC page 3	8080 simulator page 4 + object code
183	8080 simul. SOURCE page 1	8080 simul SOURCE page 2
184	8080 simul. TABLE 1 : list of instruction	
185	PATTERNS A DAI-story by Alan Sutcliffe	(PERSONAL COMPUTER WORLD)
186	PATTERNS page 2	
187	PATTERNS page 3	
188	HEAP, INTEGER, FLOATING POINT, STRING	by J.Boerriqter.
189	HEAP....	
190	Binary coding of numbers part 1	: integer format
191	Binary coding page 2	
192	MICRO-ELECTRONICA	: a new TV-course by TELEAC
193	MICRO-ELECTRONICA page 2	
194	SPEECH SYNTHESISERS	an introduction by Jos Schepens
195	EC-01: a VOICE SYNTHESISER UNIT	from EASICOMP UK
196	BIG EARS : a VOICE RECOGNITION SYSTEM	from Stuart Systems
197	VOTRAX : a UNIT from the speech synthesiser	leading company
198	SPRAAKSYNTHESE : with many thanks to ELEKTUUR	
199	SPRAAKSYNTHESE page 2	
200	SPRAAKSYNTHESE page 3	
201	SPRAAKSYNTHESE page 4	
202	SPRAAKSYNTHESE page 5	
203	SPRAAKSYNTHESE page 6	
204	SPRAAKSYNTHESE page 7	
205	8080 Software Design	: a book review by Inno Broekman
206	+++ 16 colours in 4 colour MODE	by J.Mol
	The IMP STR command	is faulty by Thomas Krebs
	+++ CLEAR in SUBROUTINE	is dangerous ...
207	DAIpc PCB-layout	and component numbering J.J.Dessart
208	DAIpc Keyboard	
209	PRINT CHR\$(12) in hardware...	it's possible !
210	Keyboard extension	
211	TALK EDITOR	to make life easier...
212	MX-80 DAI connection	through serial interface, dipswitch setting
213	Screen copies on MX-80 FI/2	by A.De Dauw, more info in N8
214	Cijfertabel	C.Van Dijk

This 8080 simulator could be a fine and easy introduction into the world of assembly language.

There is a BASIC part (some kind of disassembler) and a machine language part.

Please save the BASIC program after typing in, before RUN !
After initialisation of the screen and DATA, the programs asks for a string on line 550.

560-570 : check for special command CLEAR or INIT

CLEAR : all registers (except PSW) zero.

INIT : user has to fill in reg. values.

please note the format : 50 = 50 decimal

:50 = 50 HEX

Most mnemonics are used following INTEL standard, except:

, becomes . (MOV A.B)

DATA is preceded by space, slash : MVI A /:FF

ADDRESS is preceded by space, slash : LXI H /:BFEE

1020-1170 : translate the mnemonic into HEX instruction.

1117 : special case : bit shift left

1118 : special case : bit shift right

After the matching instruction has been found, this value(s) is (are) POKED into some reserved locations of the ml program. (hex 32B- hex 32D).

If the user instructions (1,2 or 3 bytes) are on the right locations, there is a call to hex312 (line 540), to execute the ml program, including the users instruction.

The ml program takes the old regs values from locations hex300- hex308, executes the users instructions and updates actual and previous values on the screen.

The bit-contents on the left are AFTER the instruction, the bit-contents on the right are BEFORE the instruction.

We have to reserve some room for the ml program:

POKE HEX 29C,4: CLEAR 1000.

Now we can enter the ml program with substitute. (from hex 300- hex 3A4).

FPT	:	L1	\$SCREEN.PGM.LOCATION.....	10000-
FPT	:	L2	\$SCREEN.COM.LOCATION.....	10000-
FPT	:	X	LOOP-VARIABLE.....	
FPT	:	Y	LOOP-VARIABLE.....	
INT	:	AD	ADDRESS.....	30000-
INT	:	ARROW	X-POS FOR ARROW (BIT SHIFT).....	
INT	:	D	COMPUTE-VARIABLE IN SUB HEX.....	2500-
INT	:	EF	ENDFLAG : TO CHECK IF INSTRUCTION IS COMPLETE.....	
INT	:	EFH	ADDITIONAL ENDFLAG.....	3500-
INT	:	F	LOOP-VARIABLE.....	500
INT	:	FOUND	MATCHFLAG " / ".....	1000-
INT	:	H	COMPUTE-VARIABLE IN SUB HEX.....	2500-
INT	:	HM	" " " " " ".....	
INT	:	I	LOOP-VARIABLE.....	
INT	:	INSTR	HEX-VALUE OF INSTRUCTION.....	
INT	:	L	LENGTH...STRING.....	1010
INT	:	LOC	LOCATION IN USER RESERVED BYTES.....	
INT	:	OPE	VARIABLE IN HEX.SUB.....	2500-
INT	:		ROUTINE ADDRESS OF USER RESERVED LOCATIONS...535.	
INT	:	V	VALUE OF REGISTER.....	
INT	:	X	LOOP-VARIABLE.....	
INT	:	XP	LOOP-VARIABLE.....	
INT (ARR)	:	HEXL	NUMBER OF BYTES FOR INSTRUCTION.....	
STR	:	A\$	STRING.....	
STR	:	COM\$	COMMAND STRING.....	
STR	:	F\$	INITIATE SCREEN.....	500-
STR	:	M\$	MID\$ OF \$\$.....	
STR	:	REG\$	INPUT IN INIT.....	3000-
STR	:	S\$	GENERAL INPUT.....	
STR (ARR)	:	HEXL\$	MNEMONICS.....	

```

1  REM ***8080 SIMULATOR*** V4.2
2  REM DATA LIST WITH MNEMONICS, LENGTH OF OP CODE, PLACE =
3  OPCODE
4  10  DATA MCF,0,LXI B,2,STAX B,0,INX B,0,INR B,0,DCR B,0,MVI B,1,
5  RLC,0,EMP,0,DAD B,0,LDA B,0,DCX B,0,INR C,0,DCR C,0,MVI C,1,
6  RRC,0
7  20  DATA EMP,0,LXI D,2,STAX D,0,INX D,0,INR D,0,DCR D,0,MVI D,1,
8  RAL,0,EMP,0,DAD D,0,LDA D,0,DCX D,0,INR E,0,DCR E,0,MVI E,1,
9  RAR,0
10 30  DATA EMP,0,LXI H,2,SHLD,2,INX H,0,INR H,0,DCR H,0,MVI H,1,
11 DAA,0,EMP,0,DAD H,0,LHLD,2,DCX H,0,INR L,0,DCR L,0,MVI L,1,
12 CMA,0
13 40  DATA EMP,0,LXI SP,2,STA,2,INX SP,0,INR M,0,DCR M,0,MVI M,1,
14 STC,0,EMP,0,DAD SP,0,LDA,2,DCX SP,0,INR A,0,DCR A,0,MVI A,1,
15 CMC,0
16 50  DATA MOV B,B,0,MOV B,C,0,MOV B,D,0,MOV B,E,0,MOV B,H,0,MOV
17 B,L,0,MOV B,M,0,MOV B,A,0
18 55  DATA MOV C,B,0,MOV C,C,0,MOV C,D,0,MOV C,E,0,MOV C,H,0,MOV
19 C,L,0,MOV C,M,0,MOV C,A,0
20 60  DATA MOV D,B,0,MOV D,C,0,MOV D,D,0,MOV D,E,0,MOV D,H,0,MOV
21 D,L,0,MOV D,M,0,MOV D,A,0
22 65  DATA MOV E,B,0,MOV E,C,0,MOV E,D,0,MOV E,E,0,MOV E,H,0,MOV
23 E,L,0,MOV E,M,0,MOV E,A,0
24 70  DATA MOV H,B,0,MOV H,C,0,MOV H,D,0,MOV H,E,0,MOV H,H,0,MOV
25 H,L,0,MOV H,M,0,MOV H,A,0
26 80  DATA MOV L,B,0,MOV L,C,0,MOV L,D,0,MOV L,E,0,MOV L,H,0,MOV
27 L,L,0,MOV L,M,0,MOV L,A,0
28 85  DATA MOV M,B,0,MOV M,C,0,MOV M,D,0,MOV M,E,0,MOV M,H,0,MOV
29 M,L,0,MOV M,A,0,MOV M,A,0
30 90  DATA MOV A,B,0,MOV A,C,0,MOV A,D,0,MOV A,E,0,MOV A,H,0,MOV
31 A,L,0,MOV A,M,0,MOV A,A,0
32 95  DATA ADD B,0,ADD C,0,ADD D,0,ADD E,0,ADD H,0,ADD L,0,ADD M,0,
33 ADD A,0
34 100 DATA ADC B,0,ADC C,0,ADC D,0,ADC E,0,ADC H,0,ADC L,0,ADC M,0,
35 ADC A,0
36 105 DATA SUB B,0,SUB C,0,SUB D,0,SUB E,0,SUB H,0,SUB L,0,SUB M,0,
37 SUB A,0
38 110 DATA SBB B,0,SBB C,0,SBB D,0,SBB E,0,SBB H,0,SBB L,0,SBB M,0,
39 SBB A,0
40 115 DATA ANA B,0,ANA C,0,ANA D,0,ANA E,0,ANA H,0,ANA L,0,ANA M,0,
41 ANA A,0
42 120 DATA XRA B,0,XRA C,0,XRA D,0,XRA E,0,XRA H,0,XRA L,0,XRA M,0,
43 XRA A,0
44 125 DATA ORA B,0,ORA C,0,ORA D,0,ORA E,0,ORA H,0,ORA L,0,ORA M,0,
45 ORA A,0
46 130 DATA CMP B,0,CMP C,0,CMP D,0,CMP E,0,CMP H,0,CMP L,0,CMP M,0,
47 CMP A,0
48 135 DATA RNZ,0,POP B,0,INJ,2,JMF,2,CNZ,2,PUSH B,0,ADI,1,RST 0,0
49 DATA RZ,0,RET,0,JZ,2,EMP,0,CZ,2,CALL,2,ACI,1,RST 1,0
50 145 DATA RNC,0,POP D,0,JNC,2,OUT,1,CNC,2,PUSH D,0,SUI,1,RST 2,0
51 DATA RC,0,EMP,0,JC,2,IN,1,CC,2,EMP,0,SBI,1,RST 3,0
52 155 DATA RPO,0,POP H,0,JPO,2,XTHL,0,CPO,2,PUSH H,0,ANI,1,RST 4,0
53 DATA RP,0,PCHL,0,JPE,2,XCHG,0,CPE,2,EMP,0,XRI,1,RST 5,0
54 160 DATA RP,0,POP PSW,0,JP,2,DI,0,CP,2,PUSH PSW,0,ORI,1,RST 6,0
55 170 DATA RM,0,SPLH,0,JM,2,EI,0,CM,2,EMP,0,CPI,1,RST 7,0

```

```

175 DATA 9,S,11,Z,15,AC,20,P,24,CV
180 REM DATA FOR WORD REG
190 DATA C,B,E,D,A,PSW,L,H,M,-----
300 COLURT B 0 12 0:MODE 0:POKE #75,#20
310 PRINT CHR$(12):CURSOR 17,12:PRINT "B 0 0 SIMULATOR"
315 POKE #75,32
320 CURSOR 3,2:PRINT "w.h. V4.1":CURSOR 40,2:PRINT "from
325 DAINamic"
330 CLEAR 4000:DIM HEXL$(255,0),HEXL(255,0)
500 FOR I=0 TO 255
510 READ HEXL$(I),HEXL(I)
520 NEXT
525 PRINT CHR$(12):FOR X=#8565 TO #B569 STEP 2:POKE X,#FF:POKE
530 X-3,#FF:NEXT
535 POKE #B9A6,#CD:POKE #B920,#CB:POKE #B5FC,#C7
540 CALL #312:FOR F=1 TO 5:READ F,F#:CURSOR F,11:PRINT F#
545 CURSOR F+30,11:PRINT F#:NEXT:CURSOR 30,23:PRINT "##"
550 FOR X=0 TO 8:CURSOR 3,22-X#2:READ A#:PRINT A#:NEXT
555 CURSOR 40,3:PRINT "BYTE "
560 GOSUB 10000:REM REGISTER COLORS
565 LOC=#328:ROUTINE=#312
570 GOSUB 5000:REM # & d
575 IF EF=1 THEN EF=0:CALL #312:GOTO 535
580 CURSOR 21,3:PRINT SPC(15)
585 SOUND 1 0 15 0:FREQ(1000,0):WAIT TIME 5: SOUND OFF
590 CURSOR 10,3:INPUT "INSTRUCTIE " :S$
595 CURSOR 46,1:PRINT SPC(13):CURSOR 46,1:PRINT S$
600 REM SERIAL CASES
605 IF S$="CLEAR" THEN CURSOR 0,1:PRINT SPC(57):FOR X=#300 TO
610 #308:POKE X,0:NEXT:CALL #312:GOSUB 5000:GOTO 545
615 IF S$="INIT" THEN GOSUB 3000:GOTO 535
620 REM DECODE STRING
625 REM FORMAT : MOV A,B MVI A /:FF MVI A /255
630 L=LEN(S$)
635 IF L=0 THEN 1115
640 REM LOOK FOR "/"
645 FOUND=0
650 FOR X=0 TO L-1
655 M$=MID$(S$,X,1)
660 IF M$="/" THEN FOUND=X
665 NEXT
670 IF FOUND=0 THEN COM$=S$:GOTO 1070
675 COM$=LEFT$(S$,FOUND-1)
680 FOR X=0 TO 255
685 IF COM$=HEXL$(X) THEN INSTR=X:GOTO 1117
690 NEXT
695 FOR X=1 TO 5:POKE #B576,#F3:WAIT TIME 10:POKE #B576,#FB:WAIT
700 TIME 10:NEXT:GOTO 540
705 IF INSTR=7 OR INSTR=#17 THEN GOSUB 20000:REM LEFT
710 IF INSTR=#F OR INSTR=#IF THEN GOSUB 21000:REM RIGHT
715 CURSOR 46,3:PRINT HEX$(INSTR):CURSOR 37,3:PRINT HEXL(X)+1:

```

```

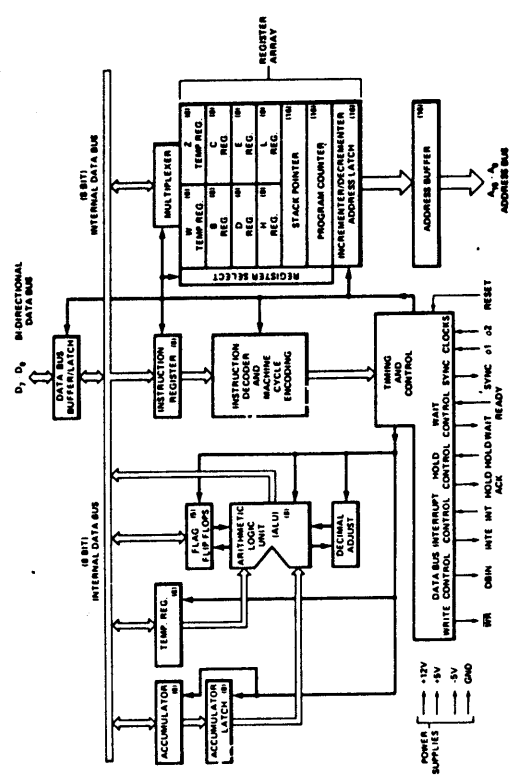
C
1130 POKE LOC, INSTR: IF HEXL(INSTR)=0 THEN EF=1:GOTO 540
1140 M$=MID$(S$, FOUND+1, 1)
1142 S$=RIGHT$(S$, L-1-FOUND)
1144 CURSOR 48, 3:PRINT SPC(7)
1145 CURSOR 48, 3:PRINT S$: IF M$<>": THEN OPE=VAL(S$)
1150 IF M$=":" THEN GOSUB 2500:REM HEX
1160 IF HEXL(INSTR)=1 THEN POKE LOC+1, OPE:EF=1:GOTO 540
1170 IF HEXL(INSTR)=2 THEN POKE LOC+1, (OPE 1AND 255):POKE LOC+2,
(OPE SHR 8):EF=1:GOTO 540
C
2500 REM SUB INPUT HEX
2510 OPE=0:HM=0
2565 FOR I=LEN(S$)-1 TO 1 STEP -1
2570 H=ASC(MID$(S$, I, 1))
2575 IF H>47 AND H<58 THEN D=H-48:GOTO 2600
2580 IF H>64 AND H<71 THEN D=H-55:GOTO 2600
2590 1 PRINT " ERROR HEX NUMBER":EFH=1:RETURN
2600 1 OPE=OPE+DX*(#10^HM)+0.5
2605 1 HM=HM+1
2610 1 NEXT
2630 1 RETURN
C
3000 REM INIT
3010 XP=0:AD=#300:REG$="C":GOSUB 3500
3020 XP=7:AD=AD+1:REG$="B":GOSUB 3500
3030 XP=14:AD=AD+1:REG$="E":GOSUB 3500
3040 XP=21:AD=AD+1:REG$="D":GOSUB 3500
3050 XP=28:AD=AD+1:REG$="A":GOSUB 3500
3060 XP=35:AD=AD+2:REG$="L":GOSUB 3500
3070 XP=42:AD=AD+1:REG$="H":GOSUB 3500
3075 CURSOR 0, 1:PRINT SPC(59)
3080 RETURN
3500 CURSOR XP, 1:PRINT " ";REG$: INPUT S$
3510 IF LEFT$(S$, 1)="" THEN GOSUB 2500
3515 IF EFH=1 THEN EFH=0:RETURN
3520 IF LEFT$(S$, 1)<>": THEN OPE=VAL(S$)
3530 IF OPE>255 THEN RETURN
3540 POKE AD, OPE
3545 CALLM #312
3547 GOSUB 5000
3550 RETURN
5000 FOR X=0 TO 8:CURSOR 28, 22-X*2:V=PEEK(#300+X):PRINT SPC(10)
5010 SPC(3-LEN(HEX$(V))):HEX$(V); SPC(6-LEN(STR$(V))):V:NEXT
C
5020 RETURN
10000 FOR X1=0.0 TO 8.0
10005 L11=48974.0-268.0*X1
10006 L21=L11-60.0
10010 FOR Y1=0.0 TO 17.0
10020 POKE L11-Y1*2, #FF
10030 POKE L21-Y1*2, #FF
10040 NEXT:PRINT:RETURN
20000 FOR ARROW=24 TO 9 STEP -1
20010 1 CURSOR ARROW, 15:PRINT CHR$(136):WAIT TIME 5:CURSOR ARROW, 15:
PRINT CHR$(32)
C

```

```

20020 NEXT:RETURN
21000 FOR ARROW=9 TO 24
21010 1 CURSOR ARROW, 15:PRINT CHR$(137):WAIT TIME 5:CURSOR ARROW, 15:
C PRINT CHR$(32)
21020 NEXT:RETURN
0000 00
0310 00 00 E5 C5 D5 F5 21 13 BF CD 5D 03 21 00 03 4E
0320 23 46 23 5E 23 56 23 7E 2A 06 03
032B 00 00 00
032E 22 06
0330 03 F5 7E 32 08 03 F1 21 00 03 71 23 70 23 73 23
0340 72 23 77 23 F5 D1 73 21 4F BF CD 5D 03 21 2B 03
0350 36 00 23 36 00 F1 D1 C1 E1 C9 3E 09 32
0360 00 00 11 00 03 22 10 03 1A CD 82 03 13 2A 10 03
0370 01 F4 FE 09 22 10 03 3A 00 00 3D 32 00 00 C2 6B
0380 03 C9 06 08 B7 17 DC 9F 03 D4 A2 03 2B 2B 2B 2B
0390 F5 3E 05 B8 C2 99 03 2B 2B F1 05 C2 84 03 C9 36
03A0 31 C9 36 30 C9

```



PAGE 01 :8080 SIMULATOR

*USER INSTRUCTIONS ON RESERVED LOCATIONS

REGC EQU :300
 REGB EQU :301
 REBE EQU :302
 REGD EQU :303
 REGA EQU :304
 REGP EQU :305
 REGL EQU :306
 REGH EQU :307
 REGM EQU :308

*
 TLEFT EQU 48975
 TRIGHT EQU 48915
 COUNT DATA 0

*
 ORG :310
 DBL 0
 PUSH H
 PUSH B
 PUSH D
 PUSH PSM
 LXI H, TRIGHT
 CALL DISPL
 LXI H, REGC
 MOV C, M
 INX H
 MOV B, M
 INX H
 MOV E, M
 INX H
 MOV D, M
 INX H
 MOV A, M
 LHLD REGL
 RES 3, 0
 SHLD REGL
 PUSH PSM
 MOV A, M
 STA REGM
 POP PSM
 LXI H, REGC
 MOV M, C
 INX H
 MOV M, B
 INX H
 MOV M, E
 INX H
 MOV M, D
 INX H
 MOV M, A
 INX H
 PUSH PSM
 POP D

PAGE 02 :8080 SIMULATOR

055 0346 73
 056 0347 214FBF
 057 034A CD5D03
 058 034D 212B03
 059 0350 3600
 060 0352 23
 061 0353 3600
 062 0355 23
 063 0356 3600
 064 0358 F1
 065 0359 D1
 066 035A C1
 067 035B E1
 068 035C C9
 069 035D 3E09
 070 035F 320000
 071 0362 110003
 072 0365 221003
 073 0368 1A
 074 0369 CDB203
 075 036C 13
 076 036D 2A1003
 077 0370 01F4FE
 078 0373 09
 079 0374 221003
 080 0377 3A0000
 081 037A 3D
 082 037B 320000
 083 037E C26B03
 084 0381 C9
 085 0382 0608
 086 0384 B7
 087 0385 17
 088 0386 DC9F03
 089 0389 DA203
 090 038C 2B
 091 038D 2B
 092 038E 2B
 093 038F 2B
 094 0390 F5
 095 0391 3E05
 096 0393 BB
 097 0394 C29903
 098 0397 2B
 099 0398 2B
 100 0399 F1
 101 039A 05
 102 039B C2B403
 103 039E C9
 104 039F 3631
 105 03A1 C9
 106 03A2 3630
 107 03A4 C9

PAGE 03 :8080 SIMULATOR

108 03A5

 * S Y M B O L T A B L E *

BNEXT 0384 COUNT 0000
 DISPL 035D NEXT 0368
 REGB 0304 REGB 0301
 REGE 0302 REGH 0307
 REGP 0305 TLEFT BF4F
 ZERO 03A2

055 0346 73
 056 0347 214FBF
 057 034A CD5D03
 058 034D 212B03
 059 0350 3600
 060 0352 23
 061 0353 3600
 062 0355 23
 063 0356 3600
 064 0358 F1
 065 0359 D1
 066 035A C1
 067 035B E1
 068 035C C9
 069 035D 3E09
 070 035F 320000
 071 0362 110003
 072 0365 221003
 073 0368 1A
 074 0369 CDB203
 075 036C 13
 076 036D 2A1003
 077 0370 01F4FE
 078 0373 09
 079 0374 221003
 080 0377 3A0000
 081 037A 3D
 082 037B 320000
 083 037E C26B03
 084 0381 C9
 085 0382 0608
 086 0384 B7
 087 0385 17
 088 0386 DC9F03
 089 0389 DA203
 090 038C 2B
 091 038D 2B
 092 038E 2B
 093 038F 2B
 094 0390 F5
 095 0391 3E05
 096 0393 BB
 097 0394 C29903
 098 0397 2B
 099 0398 2B
 100 0399 F1
 101 039A 05
 102 039B C2B403
 103 039E C9
 104 039F 3631
 105 03A1 C9
 106 03A2 3630
 107 03A4 C9

M, E
 H, TLEFT
 DISPL
 H, USER
 M, 0
 H
 M, 0
 H
 M, 0
 M, 0
 PSM
 D
 B
 H
 A, 9
 COUNT
 D, REGC
 CURSOR
 D
 D BYTE
 D
 CURSOR
 B, -268
 CURSOR
 COUNT
 COUNT
 NEXT
 B, 8
 A
 ONE
 ZERO
 H
 H
 H
 H
 H
 PSM
 A, 5
 B
 NSPACE
 H
 H
 PSM
 B
 BNEXT
 M, :31
 M, :30
 RET

MOV
 LXI
 CALL
 LXI
 MVI
 INX
 MVI
 INX
 MVI
 POP
 POP
 POP
 POP
 RET
 MVI
 STA
 JNZ
 RET
 MVI
 ORA
 RAL
 CC
 CNC
 DCX
 DCX
 DCX
 DCX
 PUSH
 MVI
 CMP
 JNZ
 DCX
 DCX
 POP
 POP
 DCR
 JNZ
 RET
 MVI
 RET
 MVI
 RET

M, E
 H, TLEFT
 DISPL
 H, USER
 M, 0
 H
 M, 0
 H
 M, 0
 M, 0
 PSM
 D
 B
 H
 A, 9
 COUNT
 D, REGC
 CURSOR
 D
 D BYTE
 D
 CURSOR
 B, -268
 CURSOR
 COUNT
 COUNT
 NEXT
 B, 8
 A
 ONE
 ZERO
 H
 H
 H
 H
 H
 PSM
 A, 5
 B
 NSPACE
 H
 H
 PSM
 B
 BNEXT
 M, :31
 M, :30
 RET

MOV
 LXI
 CALL
 LXI
 MVI
 INX
 MVI
 INX
 MVI
 POP
 POP
 POP
 POP
 RET
 MVI
 STA
 JNZ
 RET
 MVI
 ORA
 RAL
 CC
 CNC
 DCX
 DCX
 DCX
 DCX
 PUSH
 MVI
 CMP
 JNZ
 DCX
 DCX
 POP
 POP
 DCR
 JNZ
 RET
 MVI
 RET
 MVI
 RET

DISPL
 NEXT

DBYTE
 BNEXT

NSPACE
 ONE
 ZERO

COUNT 0000
 NEXT 0368
 REGB 0301
 REGH 0307
 TLEFT BF4F
 ZERO 03A2

COUNT 0000
 NSPACE 0399
 REGB 0301
 REGL 0306
 TRIGHT BF13
 USER 032B

Because the instructions are executed by 8080, some of them can have a funest effect on the program, these should not be used. These instructions are indicated by "*" in table 1. Of cause, you could use CALL if there is some ml program somewhere in memory !

INSTRUCTION			FUNCTION			HEX			INSTRUCTION			FUNCTION			HEX		
MOVE GROUP																	
MOV A, reg	(A) ← (reg)		7F	78	79	7A	7B	7C	7D	7E	JMP addr*		(PC) ← addr			C3	al ah
MOV B, reg	(B) ← (reg)		47	40	41	42	43	44	45	46	JNZ addr*		If Z=0, (PC) ← addr			C2	al ah
MOV C, reg	(C) ← (reg)		4F	48	49	4A	4B	4C	4D	4E	JZ addr*		If Z=1, (PC) ← addr			CA	al ah
MOV D, reg	(D) ← (reg)		57	50	51	52	53	54	55	56	JNC addr*		If CY=0, (PC) ← addr			D2	al ah
MOV E, reg	(E) ← (reg)		5F	58	59	5A	5B	5C	5D	5E	JC addr*		If CY=1, (PC) ← addr			DA	al ah
MOV H, reg	(H) ← (reg)		67	60	61	62	63	64	65	66	JPO addr*		If P=0, (PC) ← addr			E2	al ah
MOV L, reg	(L) ← (reg)		6F	68	69	6A	6B	6C	6D	6E	JPE addr*		If P=1, (PC) ← addr			EA	al ah
MOV M, reg	(M) ← (reg)		77	70	71	72	73	74	75	--	JP addr*		If S=0, (PC) ← addr			F2	al ah
											JM addr*		If S=1, (PC) ← addr			FA	al ah
											PCHL *		(PCh) ← (H), (PC1) ← (L)			E9	
ACCUMULATOR GROUP																	
ADD reg	(A) ← (A) + (reg)	*	87	80	81	82	83	84	85	86	CALL addr*		(TOS) ← (PC), (PC) ← addr			CD	al ah
ADC reg	(A) ← (A) + (reg) + (CY)	*	8F	88	89	8A	8B	8C	8D	8E	CNZ addr*		If Z=0, (TOS) ← (PC), (PC) ← addr			C4	al ah
SUB reg	(A) ← (A) - (reg)	*	97	90	91	92	93	94	95	96	CZ addr*		If Z=1, (TOS) ← (PC), (PC) ← addr			CC	al ah
SBB reg	(A) ← (A) - (reg) - (CY)	*	9F	98	99	9A	9B	9C	9D	9E	CNC addr*		If CY=0, (TOS) ← (PC), (PC) ← addr			D4	al ah
ANA reg	(A) ← (A) ^ (reg)	*	A7	A0	A1	A2	A3	A4	A5	A6	CCO addr*		If CY=1, (TOS) ← (PC), (PC) ← addr			DC	al ah
XRA reg	(A) ← (A) v (reg)	*	AF	A8	A9	AA	AB	AC	AD	AE	CPO addr*		If P=0, (TOS) ← (PC), (PC) ← addr			E4	al ah
ORA reg	(A) ← (A) v (reg)	*	B7	B0	B1	B2	B3	B4	B5	B6	CPE addr*		If P=1, (TOS) ← (PC), (PC) ← addr			EC	al ah
CMP reg	(A) - (reg)	*	BF	B8	B9	BA	BB	BC	BD	BE	CPF addr*		If S=0, (TOS) ← (PC), (PC) ← addr			F4	al ah
											CM addr*		If S=1, (TOS) ← (PC), (PC) ← addr			FC	al ah
INCREMENT/DECREMENT REGISTER																	
INR reg	(reg) ← (reg) + 1	**	3C	04	0C	14	1C	24	2C	34	N.B. (TOS) ← (PC) designates the following:-						
DCR reg	(reg) ← (reg) - 1	**	3D	05	0D	15	1D	25	2D	35	((SP)-1) ← (PCh), ((SP)-2) ← (PC1), (SP) ← (SP)-2						
REGISTER PAIR GROUP																	
INX rp	(rp) ← (rp) + 1		03	13	23	33	--	--	--	--	RETURN GROUP						
DCX rp	(rp) ← (rp) - 1		0B	1B	2B	3B	--	--	--	--	RET *		(PC) ← (TOS)			C9	
LDAX rp	(A) ← (rp)		0A	1A	--	--	--	--	--	--	RNZ *		If Z=0, (PC) ← (TOS)			C0	
STAX rp	(rp) ← (A)		02	12	--	--	--	--	--	--	RZ *		If Z=1, (PC) ← (TOS)			C8	
DAD rp	(H,L) ← (H,L) + (rp) ***		09	19	29	39	--	--	--	--	RNC *		If CY=0, (PC) ← (TOS)			DO	
PUSH rp *	(SP) ← (SP) - 1, ((SP)-2) ← (rp), (SP) ← (SP) - 2		C5	D5	E5	--	F5	--	--	--	RC *		If CY=1, (PC) ← (TOS)			D8	
POP rp *	(rp) ← (SP) + 1, ((SP)+1) ← (rp), (SP) ← (SP) + 2		C1	D1	E1	--	F1	--	--	--	RPO *		If P=0, (PC) ← (TOS)			EO	
											RPE *		If P=1, (PC) ← (TOS)			E8	
											RP *		If S=0, (PC) ← (TOS)			FO	
											RM *		If S=1, (PC) ← (TOS)			F8	
N.B. (PC) ← (TOS) designates the following:- (PC1) ← (SP), (PCh) ← ((SP)+1), (SP) ← (SP)+2																	
DIRECT ADDRESS GROUP																	
LDA addr	(A) ← (addr)		3A	al	ah	RESTART GROUP											
STA addr	(addr) ← (A)		32	al	ah	RST 0 *		(TOS) ← (PC), (PC) ← 016			C7						
LHLD addr	(L) ← (addr), (H) ← (addr+1)		2A	al	ah	RST 1 *		(TOS) ← (PC), (PC) ← 816			CF						
SHLD addr	(addr) ← (L), (addr+1) ← (H)		22	al	ah	RST 2 *		(TOS) ← (PC), (PC) ← 1016			D7						
						RST 3 *		(TOS) ← (PC), (PC) ← 1816			DF						
						RST 4 *		(TOS) ← (PC), (PC) ← 2016			E7						
						RST 5 *		(TOS) ← (PC), (PC) ← 2816			EF						
						RST 6 *		(TOS) ← (PC), (PC) ← 3016			F7						
						RST 7 *		(TOS) ← (PC), (PC) ← 3816			FF						
IMMEDIATE GROUP																	
MVI A, data	(A) ← data		3E	dd	ROTATE/CONTROL/SPECIAL GROUP												
MVI B, data	(B) ← data		06	dd	RLC		(An+1) ← (An), (A0) ← (A7), (CY) ← (A7) ***			07							
MVI C, data	(C) ← data		0E	dd	RRC		(An) ← (An+1), (A7) ← (A0), (CY) ← (A0) ***			0F							
MVI D, data	(D) ← data		16	dd	RAL		(An+1) ← (An), (A0) ← (CY), (CY) ← (A7) ***			17							
MVI E, data	(E) ← data		1E	dd	RAR		(An) ← (An+1), (A7) ← (CY), (CY) ← (A0) ***			1F							
MVI H, data	(H) ← data		26	dd	NOP *		No operation			00							
MVI L, data	(L) ← data		2E	dd	HLT *		Processor stopped until interrupt or reset			76							
MVI M, data	(M) ← data		36	dd	DI *		Interrupts disabled			F3							
ADI data	(A) ← (A) + data	*	C6	dd	EI *		Interrupts enabled after next instruction			FB							
ACI data	(A) ← (A) + data + (CY)	*	CE	dd	XTHL *		(L) ← (SP), (H) ← (SP)+1			E3							
SUI data	(A) ← (A) - data	*	D6	dd	SPHL *		(SPH) ← (H), (SP1) ← (L)			F9							
SBI data	(A) ← (A) - data - (CY)	*	DE	dd	XCHG		(H) ← (D), (L) ← (E)			EB							
ANI data	(A) ← (A) ^ data	*	E6	dd	DAA		Decimal adjust accumulator		*	27							
XRI data	(A) ← (A) v data	*	EE	dd	CMA		(A) ← (A)			2F							
ORI data	(A) ← (A) v data	*	F6	dd	STC		(CY) ← 1			37							
CPI data	(A) - data	*	FE	dd	CMC		(CY) ← (CY)			3F							
LXI B, addr	(B) ← ah, (C) ← al		01	al ah	OUT port }		Not used in DCE Systems			D3	port						
LXI D, addr	(D) ← ah, (E) ← al		11	al ah	IN port }					DB	port						
LXI H, addr	(H) ← ah, (L) ← al		21	al ah													
LXI SP, addr *	(SPH) ← ah, (SP1) ← al		31	al ah													

DAI

Alan Sutcliffe continues his thought-provoking series.

Since I've just purchased a musical high-resolution graphics computer — the DAI — it seems appropriate to take a close look at how I've tackled a music playing program. While I'm at it, I throw in a few comments about the DAI, then I move on to produce the micro equivalent of staring into the flames of a fire. To finish off the article this month, I suggest a competition which you may like to enter.

Prelude and chance

Bach's Prelude No. 1 from the 48 Preludes and Fugues is one of the few keyboard pieces that I can play ade-

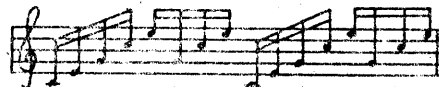


Fig 1

quately, the reason being that only one note is played at a time, except for the final chord. Figure 1 shows the first bar. The next 32 bars of the piece each follow the same pattern: an ascending arpeggio of five notes, with the last three notes repeated, and then the whole sequence of eight notes repeated.

There are then two bars each with a slightly different pattern of pitches, without the exact repetition. And then the Prelude ends with a chord of notes played together.

Such simplicity and regularity make it a natural piece to synthesise on my new computer, a DAI from Data Applications. It has a sound chip, with three oscillators which are driven by Basic statements. I am not going to describe all these facilities, but I'll give you just enough to follow the program I have written to realise the Prelude: Program A. Each note is shaped by an envelope, which gives it an attack, sustain and decay portion. The shape of the envelope can determine the quality of the note as we perceive it just as much as the tone quality, or wave-form, of the sound.

ENVELOPE a c,d,e,f,g,h,i
ENVELOPE 1 8,4;15,8;10,20;0

In this example: a is the envelope number, 0 or 1; c e and g are amplitudes, from 0 to 15; d f and h are durations, in arbitrary units of time of 3.2 milliseconds; i is also an amplitude, and since it does not have a duration the note stays at this level indefinitely, until either another note is started on the same oscillator or the SOUND OFF statement is obeyed. There can be any number of amplitude duration pairs. The envelope in the second example is shown in Figure 2.

Amplitude 0 is not silence but gives a positive audible level. A note cannot therefore be turned off simply by put-

ting the final amplitude to zero. One of the two aforementioned means must be used. This is inconvenient because the time that elapses in the program must be matched to the length of note required, rather than just leaving it to the envelope to give a note of the right length. This is done most easily by using the WAIT TIME statement with a suitable integer, which gives a delay in units of 20 milliseconds. Alternatively, the next bit of useful program can be obeyed, but this must be timed — by trial and error — so that the note can be ended at the right moment.

ENVELOPE is a declarative statement and obeying it does not cause any output. Two envelopes can be stored at any one time and either of them can be redefined at any point in the program.

A note is played by executing a SOUND statement:

SOUND a b c d e
SOUND 2 1 15 0 4000;

a is the number of the oscillator to be used, 0 1 or 2; b is the number of the envelope, 0 or 1; c is the overall amplitude for the note, 0 to 15; d is the tremolo or glissando, not used in this piece; e is the period of oscillation of the note in units of 1/2 microseconds. In the example, 4000 units of 1/2 microsecond each gives a frequency of 500 cycles per second. Because this is not a convenient way of specifying frequency, a built-in function is provided to do the conversion. Middle C happens to be a frequency of 256 cycles per second and `FREQ(256)` gives the number of units to achieve this frequency.

In the program, a table is set up in `PH(72)` with the frequencies for six octaves of notes, in order to save calling `FREQ` for each note as it is played.

Sound quality

The sound produced by the oscillators is a simple square wave. A note made

with a single oscillator is flat and uninteresting. Being used to using sound synthesisers, it is the weediness of the notes that I find least appealing about most of the tune-playing that is done on small computers, even when several notes are being played at once. In this program, since only one note is played at a time, I use all three oscillators for each note. They are coupled together in octaves in this case and, together with the envelope, this gives a reasonably keyboard-like sound. It is obviously equally easy to set up other harmonic structures, including fifths and thirds above the fundamental note.

It is one of the mysteries of perception that we hear some such sounds as a single note, while some other very similar sounds are heard as chords of separate notes. Even the same sound can be heard in both ways, depending on whether the component sounds all start at the same time or come in one by one.

Even these notes seem somewhat dull and the ear quickly tires of the quality. Since the DAI has stereo sound output, as well as putting the sound out to the TV with the video signal, I can connect this output to my small EMS synthesiser. This is essentially a collection of analogue devices, so that filtering and a little bit of reverberation can be added. The two halves of the sound signal can also be ring-modulated together, which adds further to the harmonic spectrum of each note. A ring modulator takes two input frequencies, A and B, and outputs their sum and difference, $A+B$ and $A-B$.

On the DAI oscillators 0 and 1 go to stereo channel 1, while oscillators 1 and 2 go to channel 2. In this piece, with the oscillators tuned in octaves, for each base frequency F, one channel has F and 2F while the other has 2F and 4F. Thus the ring-modulator adds $3F = F+2F = 4F-F$, $5F = F+4F$ and $6F = 2F+4F$, as well as repeating the frequencies already present.

Although this involves a bit of arithmetic, it does take us outside the subject of personal computing, since synthesisers are now far less common than computers. However I mention it because I believe that this approach of mixed analogue and digital synthesis is usually the most economical way of making electronic sounds. Each method has its own strengths: digital for completely reliable frequencies and analogue for processes like reverbera-

Amplitude

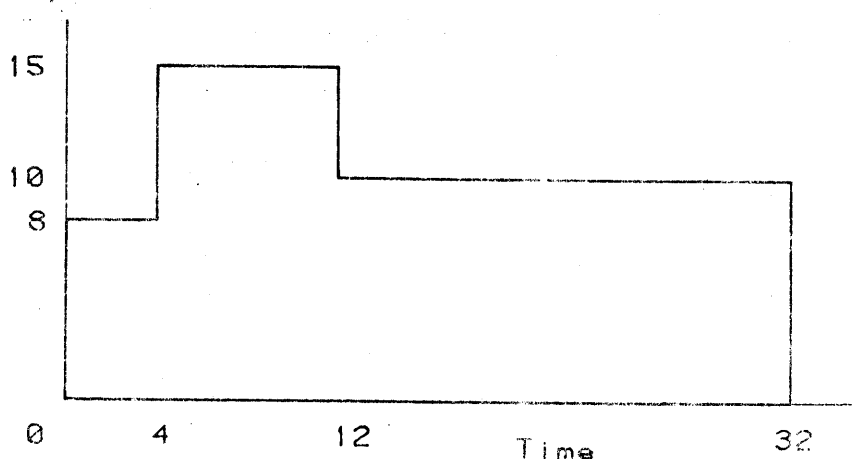


Fig 2 Example envelope

tion that are difficult to perform digitally.

Another unsatisfactory aspect of the sound form this program is the rigid timing, which the ear quickly tires of. There are two approaches to this, both of which depend of altering the parameter to the WAIT TIME statements, since these are what determine the lengths of the individual notes. The refinements to the timing tend to be of two kinds: changes to single notes, usually for emphasis, and changes to whole phrases or sections. In the first approach, an extra piece of data is added for each note to give its individual timing. In this way, both kinds of change can be made, but it is tedious to have to give another data item for every note. It also has the disadvantage of being awkward to change. Obviously this can be partly solved by using variables rather than fixed data. For example, the first note of each of each bar should be emphasised. This can be done by increasing the amplitude slightly, but as important is to lengthen the note by a tiny amount. Also, on the whole, lower notes are played more slowly than higher ones. In this piece, these two aspects work together since the first note in each bar is also the lowest.

The WAIT TIME statement is a little crude for this kind of adjustment as the unit of time is 1/50 of a second. Any such wait statement can be replaced by a dummy FOR loop:

```
FOR L=1 TO M:NEXT L
```

By setting M, the time delay can be set in units that should be less than a millisecond — unless you have a very slow computer. M can then be set either from a data item, or using a formula based on pitch and position in the bar. The other approach is to control M with an input device such as a game paddle. This is most suitable for the macro-changes and is equivalent to the function of a conductor. There is no reason why M cannot be derived as a combination of a value from within the program and another one from the performer.

Program details

Now let me make a few comments on the program as it appears, which should help in following it and in adapting it to another system.

In DAI's the distinction is made between integer variables and floating point ones, as it is in Fortran. This is in addition to the usual further data type for string variables. The default option is that all variables are floating point. The IMPLICIT statement allows, for example, all those variables beginning I to F to be declared as integer variables. This has to be done before any of the program is input as variables are given their type at the time the program line is input. IMPLICIT is therefore better thought of as a command rather than a program statement.

It is also necessary to allocate space for arrays at the start of a program using the CLEAR statement.

A peculiarity of the DAI system is that subscripts, even though they may be typed in as integers, are always listed — and presumably stored — as floating point numbers. This does not seem sensible, but I have not found a way round it yet. The program here is shown as it



Fig 3a M=0.3

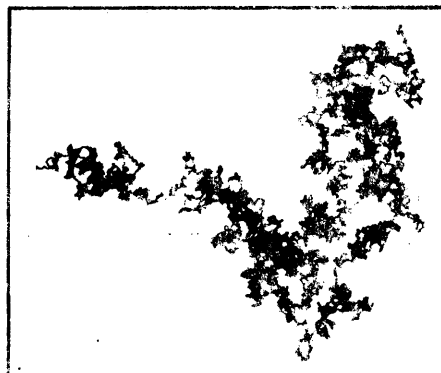


Fig 3b M=0.5

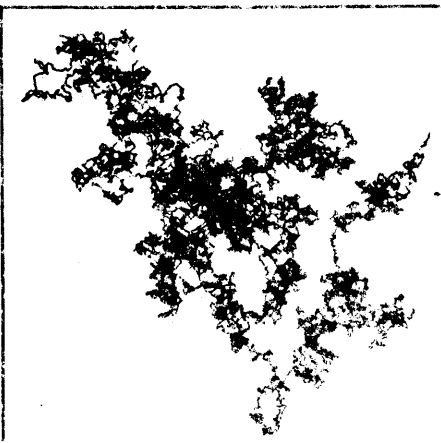


Fig 3c M=1

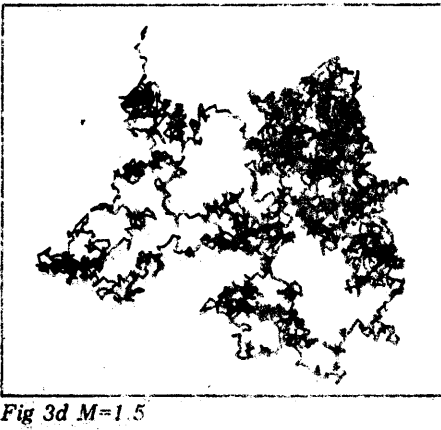


Fig 3d M=1.5

was input.

An octave is divided into 12 equal semitones. Since frequency is logarithmic — double the frequency increases the pitch by an octave — multiplying a frequency by the twelfth root of 2 increases the pitch by a semitone. The frequency of the lowest note in the piece, C two octaves below middle C, is 64, and this is set by Q. The whole piece can be transposed up or down in pitch by altering this initial

value for Q. To avoid small errors creeping in from repeated multiplication it is better to compute just one octave of semitones and derive the others from it by doubling the frequencies.

For the main part of the piece only five data items are needed for each bar and the program expands these. The notes are numbered from the lowest as 1, while middle C, for example is 25. These numbers, which are easier to transcribe from the score than the actual frequencies, are converted to the appropriate frequency values via the table PH.

The last two bars need eight values for each half bar. The same subroutine is used to play each group of eight notes. A problem arises from the fact that for each group of notes it takes a perceptible period of time to get the data for the next group and start the loop again. So it is necessary to have a slightly shorter WAIT after the last note of each group than after each of the others. I found that three units less was the right amount.

In the statements, I expected that the two higher components would need somewhat lower amplitudes than the fundamental, but on experimentation I found that having them all at the maximum level of 15 gave as good a result as any.

A characteristic of natural sounds is that the higher frequencies both start and die away more quickly. I have modelled this very roughly by using a slightly shorter envelope for the two higher components in each note. Again, this is a matter for experimentation. At one point I had the envelopes numbered the wrong way round and I did notice that the effect was slightly dulled compared to the intended way, and a tiny bit less like the harpsichord sound I was after. Not that the final result is at all like a real instrument — it is clearly synthetic, but has a pleasing little twang to it.

The final chord is not satisfactory. The notes are spread a little in time and are heard as a chord, but since there is only one oscillator per note, the effect is feeble, when it should rather end the piece with a good solid sound.

This Prelude is the one taken by Gounod as the accompaniment for his 'Ave Maria', a piece of Victorian kitsch that I cannot take seriously, but my next project is to take the top oscillator out and use it for the Gounod tune to see if something listenable can be made of it.

DAI comments

I am in the process of learning the peculiarities of my new DAI system. I chose it particularly for the high resolution colour graphics and am very well pleased with it. But some idiosyncracies of its Basic are worth recording. On the debit side the problem with floating point subscripts has already been mentioned. The most glaring omission is the absence of user defined functions. Very odd. On the plus side I particularly like the MOD operator. $1 - 7 \text{ MOD } 3$ puts I equal to 1, the remainder of 7 on division by 3. This can always be defined as a user function in a system that allows them, in the following way: $\text{DEF FNM}(I,N) = I - N * \text{INT}(I/N)$. But I prefer to have it as an operator rather than a function as this is more

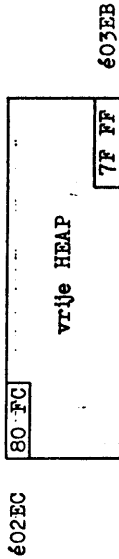
HET GEBRUIK VAN DE "HEAP".

1. De HEAP (in goed nederlands: 'hoop') is dat gedeelte van het RAM dat door een BASIC programma gebruikt wordt om strings en arrays op te slaan.

De HEAP begint op adres 602EC. Bij een reset (of power-on) worden voor de HEAP 256 bytes gereserveerd. De grootte van de HEAP staat in de HEAPsize pointer 6029D/E (60100).

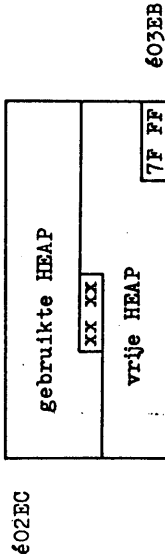
Van het aantal voor de HEAP gereserveerde bytes gebruikt de HEAP er zelf vier: twee om het einde van de HEAP aan te geven (en tevens de maximale grootte) en twee om de nog vrije HEAP ruimte te berekenen.

Na het inschakelen van de DAI ziet de HEAP er als volgt uit:



De vrije HEAP-ruimte wordt berekend uit: 7FFF + 80FC = 00FB. Er zijn dus 252 vrije bytes ter beschikking.

Als (bv. na RUN van een programma) een gedeelte van de HEAP is gebruikt, is de pointer voor het berekenen van de vrije ruimte aangepast en opgeschoven tot achter het laatste gebruikte byte:



2. CLEAR commando:

De grootte van de HEAP kan met behulp van het CLEAR commando aangepast worden aan het aantal en de grootte van de arrays en de strings die in een programma gebruikt worden.

bijv. CLEAR 1000 : 1000 bytes worden voor de HEAP gereserveerd. Er zijn dan dus 996 bytes ter beschikking.

Het getal 1000 in het CLEAR commando heeft een decimale waarde :

Na uitvoering van dit CLEAR commando is de HEAPsize pointer 6029D/E aangepast (603EB). De tekstbuffer - waarin het BASIC programma staat - en de symbol table zijn naar een hoger RAM adres opgeschoven.

De maximale grootte van de HEAP is 32767 bytes (67FFF).

Het is goed elk programma met CLEAR xxxx te beginnen. Dit hoeft slechts één keer in een programma gedaan te worden. Alleen een reset of power-on brengen de HEAP terug tot 256 bytes. Een NEW commando doet dit niet!

3. ARRAYS:

Wanneer in een programma arrays gebruikt worden, dan moet hiervoor eerst geheugenruimte gereserveerd worden. Deze ruimte wordt gezocht in de HEAP. D.m.v. een CLEAR commando moet gezorgd worden dat de HEAP voldoende ruimte heeft.

Het dimensioneren van de geheugenruimte voor arrays wordt gedaan met het DIM commando.

Aan de hand van onderstaand programma voorbeeld zal de werking daarvan toegelicht worden.

```

10 DIM A%(2)
20 DIM B%(3)
30 DIM C%(5)
40 DIM D%(1,1)
    
```

Na een RUN van dit programma ziet de HEAP er als volgt uit:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
02E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
02F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0300	00	0E	01	05	00	00	00	00	00	00	00	00	00	00	00	00
0310	00	0E	01	05	00	00	00	00	00	00	00	00	00	00	00	00
0320	00	13	02	01	00	00	00	00	00	00	00	00	00	00	00	00
0330	00	00	00	00	00	80	B3	00	00	00	00	00	00	00	00	00
.																
.																
03E0	00	00	00	00	00	00	00	00	00	00	00	00	00	7F	FF	

In de bij dit programma horende symbol table staat dan:

```

51 41 52 EE 02 41 42 42 FE 02 61 43 62 12 03
51 44 52 22 03
    
```

(Het adres waar de symbol table begint staat in de pointer op adres 602A1/2).

De symbol table geeft aan welke variabelen gebruikt zijn en waar in de HEAP ruimte voor de arrays is gereserveerd:

```

A% - 51 41 52 EE 02 - A {41}, % {51 .. 52} op adres 602EE.
B% - 41 42 42 FE 02 - B {42}, % {41 .. 42} op adres 602FE.
C$ - 61 43 62 12 03 - C {43}, $ {61 .. 62} op adres 60312.
D% - 51 44 52 22 03 - D {44}, % {51 .. 52} op adres 60322.
    
```

In de HEAP is nu de ruimte 602EC tot 60334 gebruikt. Op 6035/6 staat nu de pointer 60B3. Door dit op te tellen bij 67FFF wordt de vrije HEAP ruimte gevonden: 6B3 (179 bytes).

Binaire representatie in het-geheugen van integer- en floating point getallen

De doelstelling van dit artikel is de opbouw van deze bitrijen toe te lichten, zodanig dat de lezer zelf met papier en potlood (eenvoudige) omzettingen kan uitvoeren en deze daarna met programma 1 controleren. Een beter inzicht in de opslag van de data in het geheugen wordt hierdoor bekomen.

1. Inleiding

Programma 1 beeldt op scherm de binare representatie af van een integer- of een floating point getal. Vooreerst vraagt het programma of een integer (I) of een floating point getal (F) moet worden omgezet. Nadien wordt het om te zetten getal opgevraagd.

Programma 1

```

100 REM BITRIJGENERATOR
110 PRINT CHR$(12):GOSUB 2000
120 INPUT " INTEGER (I) OF FLOATING POINT GETAL (F) ";T$:PRINT
130 IF T$="I" THEN 170
140 IF T$="F" THEN 150
145 GOTO 120
150 INPUT "   FLOATING POINT GETAL";W$:PRINT :M%:=VARPTR(W):GOTO 200
160 INPUT "   INTEGER GETAL";W$:PRINT
180 M%:=VARPTR(W%)
190 FOR A%=M% TO M%+3
200 P%:=PEEK(A%)
300 P%:=PEEK(A%)
400 FOR X%=7 TO 0 STEP -1.0
500 B%:=P% IAND (2^(X%+1))
600 C%:=B% SHR X%:ST$="0";IF C%=1 THEN ST$="1"
700 PRINT ST$;
1000 NEXT:PRINT " ";
1100 NEXT
1150 PRINT
1200 GOTO 120
2000 CURSOR 1,22
2050 PRINT "BITRIJGENERATOR VOOR INTEGERS EN FLOATING POINT GETALLEN"
2100 PRINT :PRINT :RETURN
    
```

Indien met dit programma de binare voorstelling bepaald wordt van:

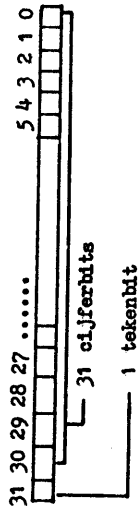
- a. het integer-getal 23 vinit men;
- b. het floating point getal 23.0 vinit men;
- c. het floating point getal 20.23 vinit men;

Vastgesteld wordt dat zowel voor de voorstelling van een integer- als voor deze van een floating point getal 4 bytes worden gebruikt.

2. Integer-getallen

2.1 2-complement notatie met 22 bits

De integer-getallen worden in 4 bytes door middel van de 2-complement notatie- vorm voorgesteld. Dit betekent dat van de 32 beschikbare bits er 31 (bit 0 tot en met bit 30) gebruikt worden voor de representatie van de numerieke waarde en 1 (bit 31) voor het teken van het getal.



2.2 Voorstelling van een positief getal in de 2-complement notatie

Voor een positief getal is de tekenbit steeds 0. De numerieke waarde wordt be- komen door het decimaal equivalent te bepalen van het binare getal voorgesteld door de 31 cijferbits.

voorbeeld:

00000000 00000000 00000100 10100000

└ de 31 cijferbits bepalen het getal 1184
└ de tekenbit is 0 het getal is positief

De gegeven bitrij stelt bijgevolg het getal (+)1184 voor. Het grootste integer-getal dat in deze notatievorm kan worden gerepresenteerd is bijgevolg:

01111111 11111111 11111111 11111111 = $2^{31} - 1$
= 2147483647

2.3 Voorstelling van een negatief getal in de 2-complement notatie

Voor een negatief getal is de tekenbit steeds 1. De numerieke waarde wordt be- komen door het decimaal equivalent te bepalen van het binare getal voorgesteld door de 31 cijferbits en dit af te trekken van $2^{31} = 2147483648$

voorbeeld:

10000000 00010110 11011000 00000111

└ de 31 cijferbits bepalen het getal 1497095
de numerieke waarde van het voorgestelde getal is dan:

2147483648 - 1497095 = 2145986553

Daar de tekenbit negatief is, stelt de opgegeven bitrij bijgevolg het getal - 2145986553 voor.

Het kleinste negatieve integer-getal dat met 4 bytes in de 2-complement methode kan worden voorgesteld is dus dit waarbij het getal bepaald door de 31 cijferbits gelijk is aan 0. In dit geval wordt slechts 0 afgetrokken van 2147483648, zodat dit kleinste getal gelijk is aan : - 2147483648, met als binaire representatie:

10000000 00000000 00000000 00000000

opmerking: indien in de 2-complement notatie gewerkt wordt met 2 bytes moet worden afgetrokken van 2^{15} . Analooz voor 8 bits.

2.4 Van decimaal naar binair in de 2-complement methode

In vorige twee nummers werden de algoritmen gegeven om, vanuit de binaire representatie het decimaal equivalent te bepalen. Nu wordt het omgekeerde probleem gesteld: welk is de binaire 2-complement notatie van een gegeven decimaal getal ? voorbeeld 1: binaire representatie van het integer-getal: (+)2543

De corresponderende bitrij kan als volgt bepaald worden bereken het gehele quotient en de rest van de deling van 2543 door 2. Het quotient is 1271 en de rest 1. Deze rest is de meest rechtse bit van de gevraagde binaire notatie; dit algoritme wordt nu herhaald op het quotient 1271 tot 0 als quotient bekomen wordt; de nieuwe resten worden telkens links van de vorige geschreven. Schematisch krijgen we volgende notatie:

gehele	quotienten	:	0	1	2	4	9	19	39	79	158	317	635	1271	2543
resten		:	1	0	1	1	1	1	0	1	1	1	1	1	1

binaire representatie

We besluiten:

2543 = 10011110111

Om tot een representatie met 4 bytes te komen worden deze cijferbits links met nul- len aangevuld tot 31 bits worden bekomen. De bijhorende tekenbit is 0, zodat de 2-complement notatie van (+)2543 is:

(+) 2543 = 00000000 00000000 00001001 11101111

voorbeeld 2: binaire representatie van -2543

Theoretisch kan dit als volgt gebeuren: de tekenbit moet 1 zijn, terwijl de cijferbits het binaire getal 2147483648 - 2543 = 2147481105 moeten vormen. Van dit getal kan de binaire representatie met 31 bits bepaald worden volgens

het algoritme uit het vorige voorbeeld. We vinden:

2147481105 = $\underbrace{11111111}_{7\text{bits}} \underbrace{11111111}_{8\text{bits}} \underbrace{11110110}_{8\text{bits}} \underbrace{00010001}_{8\text{bits}}$
 31 cijferbits

zodat:

- 2543 = 11111111 11111111 11110110 00010001

De aftrekking van 2147483648 kan vermeden worden met volgend meer praktisch algoritme om de 2-complement notatie van -2543 te bepalen:

a. bepaal de 2-complement vorm van het positieve getal (2543) volgens de methode uit voorbeeld 1

Dit geeft:

2543 = 00000000 00000000 00001001 11101111

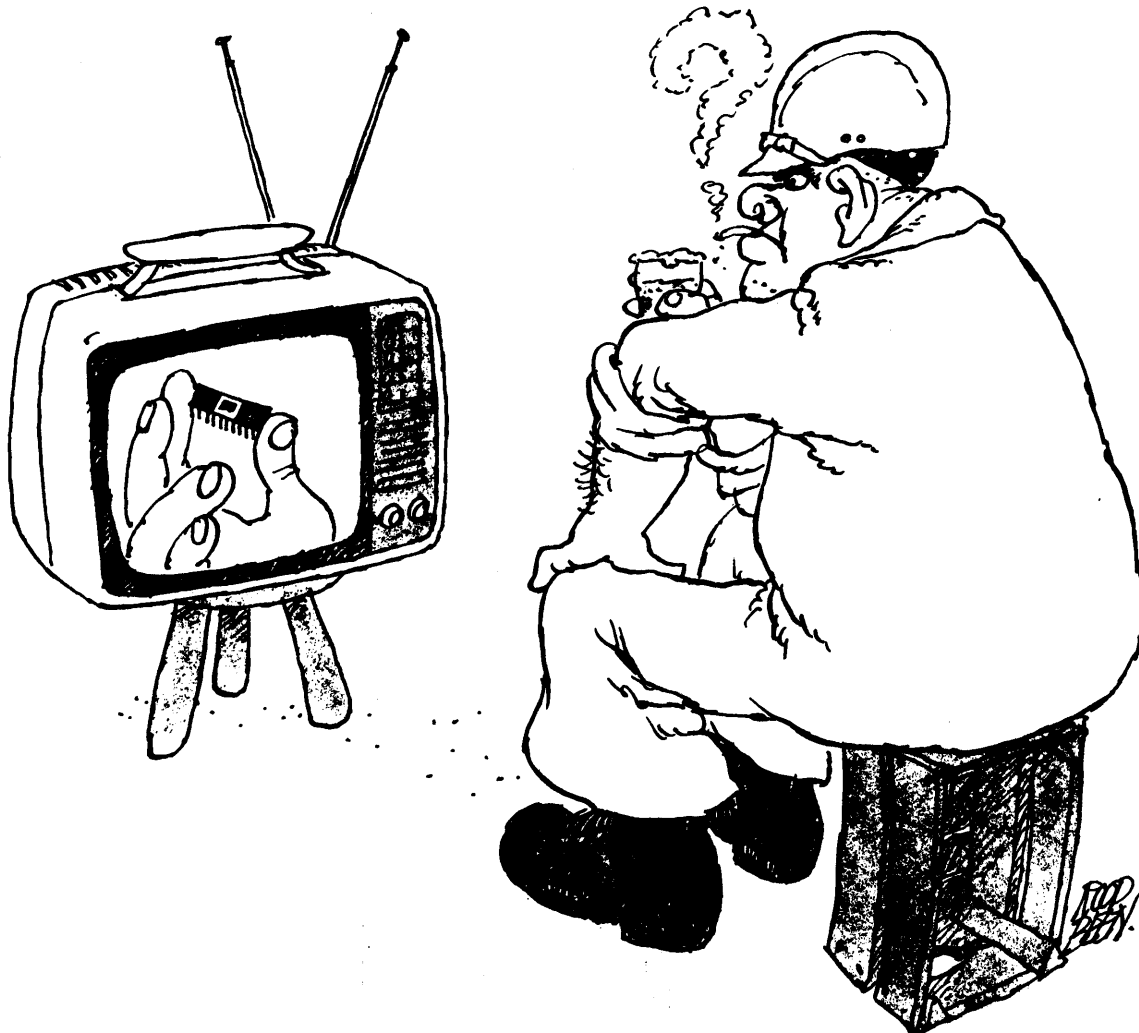
b. inverteer alle bits van deze bitrij, d.w.z. vervang een 1 door een 0 en omgekeerd. Men bekomt: 11111111 11111111 11110110 00010000

c. tel bij deze nieuwe bitrij 1 op:

11111111 11111111 11110110 00010000
 +
 11111111 11111111 11110110 00010001

Deze laatste som is de gewenste 2-complement notatie.

micro-electronica in bedrijf



De snelle ontwikkeling van moderne technische hulpmiddelen komt als een golf over ons heen. Dit uit zich onder meer in de opkomst van de micro-electronica, die zo ongeveer samenvalt met de ontwikkeling van een industrieel beleid, dat is gericht op vernieuwing (innovatie). De echte doorbraak kwam pas bij de introductie van geïntegreerde schakelingen. Sindsdien heeft de ontwikkeling van de 'chip' zich in een sneltreinvaart voortgezet. In de loop der jaren is met name het formaat van de computer enorm afgenomen. Mede door de lage prijs en de hoge werksnelheid wordt micro-electronica steeds vaker toegepast. Zo is bijvoorbeeld een snelle verwerking van de stroom

betalingen mogelijk geworden door ver-doorgevoerde automatisering bij banken. Micro-electronica is echter niet alleen een hulpmiddel voor het verrichten van rekenwerk. Micro-electronische schakelingen zijn op de meest onverwachte plaatsen te vinden, ook in het dagelijks leven. Denkt u maar aan polshorloges, televisietoestellen en foto-apparatuur, om maar eens enkele voorbeelden te noemen. Micro-electronica biedt veel mogelijkheden, óók voor uw bedrijf. Door het gebruik van micro-computers kan de kwaliteit van uw produkt(en) duidelijk worden verbeterd. In de cursus *Micro-electronica in bedrijf* leert u welke mogelijkheden

er zijn om deze techniek toe te passen bij úw specifieke produkt(en). Het gaat daarbij om de innovatie in het produkt, maar óók in het produktieproces. Micro-electronica in bedrijf, wellicht uw bedrijf?

Inhoud

De cursus is bedoeld voor het hoger- en middenkader van industrieën en bedrijven. Mensen die beslissingen nemen over het wel of niet innoveren in een werksituatie. 'Micro-electronica in bedrijf' handelt over de toepassing van micro-electronica bij de verandering en verbetering van produkten en produktieprocessen. Er wordt aandacht besteed aan de

technische mogelijkheden van micro-electronische componenten, het belang van deze techniek voor het Nederlandse bedrijfsleven en de diverse bedrijfskundige aspecten van een innovatieproces in het algemeen. De cursus geeft, populair gezegd, de nodige bedrijfskundige kennis voor technici en de nodige technische kennis voor de meer bedrijfskundig geïnteresseerden. Voor alle duidelijkheid: de cursus 'Micro-electronica in bedrijf' is géén vervolg op de cursussen 'Micro-processors' die Teleac eerder verzorgde.

In 'Micro-electronica in bedrijf' worden zowel de technische, als de sociaal/economische aspecten uitvoerig behandeld. Vooral het laatstgenoemde aspect verdient de aandacht, omdat toepassing van micro-electronica in een bedrijf nogal wat veranderingen vereist. Veranderingen van de bedrijfsstructuur, maar ook een wijziging in de arbeidsomstandigheden en het takenpakket van de werknemers. Zo is een numeriek bestuurd draaibank een goed voorbeeld van een machine, die zelf reeksen handelingen uitvoert. Deze handelingen moest de bankwerker vroeger zelf verrichten. Nu worden de handelingen vooraf ingesteld en daarna zal de machine zonder tussenkomst van de mens ze keer op keer reproduceren met groot gemak, grote nauwkeurigheid en grote snelheid. Door de introductie van de micro-electronica is er in dit voorbeeld voor de bankwerker, die vroeger routinematig werkzaamheden verrichtte, weinig plaats meer. De machine kan zijn werk sneller en vaak nauwkeuriger uitvoeren. Er zal echter een nieuw vak ontstaan, namelijk het beroep om werktekeningen om te zetten in een reeks opeenvolgende handelingen voor de machine. De man die deze werkzaamheden verricht, zal inzicht moeten hebben in de mogelijke basishandelingen van de machine. Hoe kunnen al deze veranderingen worden opgevangen? Moeten de arbeiders worden omschoold of is het beter om mankracht in te huren om een computerprogramma

te maken? Zo ja, welke mensen zijn daarvoor dan nodig? Dit zijn slechts enkele van de vele vragen die in de cursus worden beantwoord. Een groot deel van de onderwerpen zal worden behandeld aan de hand van praktijkvoorbeelden uit Nederlandse bedrijven.

Huiswerk

Tijdens voorgaande cursussen van Teleac is gebleken, dat er bij de cursisten behoefte bestaat om de kennis te toetsen aan de hand van huiswerkopgaven. De enige manier om het ingeleverde huiswerk juist en snel te kunnen nakijken, is via het systeem van multiple choice-vraagstukken. Bij de cursus 'Micro-electronica in bedrijf' horen wekelijks *diagnostische meerkeuzevragen*. Bij de correctie van de vraagstukken wordt u er op gewezen welk gedeelte van de stof u verkeerd begrepen heeft en nog eens na moet kijken. Het is voor uw studie van wezenlijk belang dat u regelmatig uw huiswerk ter correctie inzendt. Het huiswerk wordt met behulp van een computer gecorrigeerd en uw resultaten ontvangt u vervolgens schriftelijk binnen een week nadat u het materiaal heeft ingestuurd. Het is niet mogelijk de cursus met een examen af te sluiten. Wel krijgt iedereen die in voldoende mate heeft deelgenomen aan de huiscorrectie, ter afsluiting een *verklaring* van deelname. Op deze verklaring wordt het aantal lessen vermeld waarvan huiswerk is ontvangen, evenals de behaalde gemiddelde score.

Studievoorwaarden

De Stichting Teleac, afdeling schriftelijk onderwijs is erkend door de minister van Onderwijs en Wetenschappen bij beschikking van 1 juni 1978, kenmerk VO/OV/SFO-502.368.

Als zodanig is Teleac bevoegd huiswerkcorrectie te verzorgen. Bij de cursussen waarbij huiswerkcorrectie aan het cursuspakket is toegevoegd – in dit geval de cursus 'Micro-electronica in bedrijf' – zijn de studievoorwaarden van Teleac van

toepassing. Deze studievoorwaarden kunt u vinden op pagina 10 van dit cursusprogramma.

Televisie

De serie bestaat uit twintig televisielessen van dertig minuten. De uitzendingen vinden plaats op dinsdagavond van 18.28-18.58 uur via Nederland 1; de herhalingen op zaterdagmorgen van 10.50-11.20 uur via Nederland 1. De eerste uitzending is op 3 november 1981.

Radio

Naast de televisielessen zullen ook twintig radioprogramma's van dertig minuten worden verzorgd. In de radiolessen zal vooral worden ingegaan op de mens in het automatiserings- en innovatieproces. De radio-uitzendingen vinden plaats op donderdag van 21.30-22.00 uur op Hilversum 2; de herhalingen op zaterdag van 21.30-22.00 uur op Hilversum 2. De eerste uitzending is op 15 oktober 1981. In de eerste drie programma's zal worden gediscussieerd over de maatschappelijke gevolgen van de invoering van micro-electronica.

Cursuspakket

Het cursuspakket bestaat uit de volgende onderdelen:

- twintig televisielessen van dertig minuten;
- twintig radiolessen van dertig minuten;
- een losbladig cursusboek van ruim 400 bladzijden;
- wekelijkse huiswerkcorrectie;
- uitgebreide documentatie.

Extra:

- een slotmanifestatie in het voorjaar van 1982.

Cursusprijs

De prijs voor dit pakket bedraagt f 275,—/BF 4165, inclusief huiswerkcorrectie, BTW en verzendkosten. U kunt cursist worden door f 275,— over te maken op postgiro 544232 ten name van Teleac, Utrecht, onder vermelding van: *Micro-electronica in bedrijf*. Voor Belgische cursisten geldt een andere bestelwijze. Zie hiervoor pagina 27.

In de vele microcomputertijdschriften zijn de laatste tijd heelwat artikels verschenen betreffende speech synthesizers. Ook in de advertenties valt op dat het aanbod van kleine goedkope spraaksystemen voor de meeste populaire micros (APPLE, TRS-80, PET, enz.) enorm is toegenomen.

Verscheidene grote firma's brengen nu spraak-IC's uit. Om er maar enkele te vernoemen: TI, NS, GI, VOTRAX, Om te beginnen moet er een duidelijk onderscheid gemaakt worden tussen de verschillende soorten spraaksystemen.

(1) Fixed vocabulary is een eerste soort.

Hierbij bevindt een (beperkte) woordenschat zich op een of andere manier in ROM gecodeerd en wordt vervolgens teruguitgelezen en gedecodeerd. Na wat filteren enz. is opnieuw een verstaanbaar taal te horen. Meestal wordt vertrokken van hoge kwaliteitsbandopnamen die in professionele studio's worden gedigitaliseerd. Texas Instrument, National Semiconductor en General Instruments brengen dergelijke systemen op de markt. Ze zijn in hoofdzaak bestemd voor "dedicated applications" bvb. een ROM kan "ingesproken" worden met woorden die typisch zijn voor een wagen. De woordenschat hiervan zou totaal verschillend zijn van die van een winkelkassa.

Het interessante is echter dat deze firma's evaluation kits verkopen met een algemeen bruikbare woordenschat.

(2) Het tweede soort kan geen volledige woorden maar wel fonemen ten gehore brengen. Groot voordeel hierbij is dat men niet beperkt blijft tot een vaste woordenschat. Het nadeel is dat het uitspreken van een woord iets meer moeite kost.

Meestal zijn deze systemen ook duurder.

Toonaangevend op dit vlak is de firma VOTRAX. Onlangs heeft deze de SC-01 op de markt gebracht voor ongeveer \$70. Deze prijs omvat allen de chip. Een volledig systeem kost \$375 (TYPE-'N-TALK).

Na deze korte inleiding kunnen we even kijken naar de artikels die onlangs verschenen en gewijd waren aan 'spraak voor computers'

(1) Micro Systemes Mars-Avril 1981 p.97-109

Presenteert een zelfbouwproject met een 1802 processor.

Het systeem kan zowel met de hand als door de P.C. bestuurd worden. Misschien heeft een DAInamicer reeds zo een systeem gebouwd.

Graag zou ik weten hoe het klinkt.

Het is gebaseerd op het tweede systeem, fonetisch dus.

Nadeel is wel dat de fonetiek van de verschillende talen niet gelijk zijn! In het Engels of het Frans vinden we klanken die het Nederlands niet kent en omgekeerd.

(2) BYTE June 1981 p.46-68

Eveneens een zelfbouwproject volledig gebaseerd op de SPC DIGITALKER van N.S.. Dit systeem heeft een woordenschat bestaande uit een zinnetje (this is digitalker) en 143 algemeen bruikbare woorden (cijfers, eenheden, enz.)

(3) KILOBAUD MICROCOMPUTING May 1981 p.134-139

Ook hier wordt een zelfbouwproject met de SPC beschreven

Buiten nr. (1) weinig origineel. De schema's zijn bijna letterlijk overgenomen uit de datasheets van N.S.

Ook op de DAI-dag demonstreerde dhr. SIP een dergelijk systeem aangesloten op de DCE bus.

Ikzelf denk er aan om een dergelijk systeem te commercialiseren maar dan wel met enige extratjes zoals volumeregeling enz.

Wie geïnteresseerd is kan contact opnemen met DAInamic.

Om naar uit te kijken: in BYTE van September moet een systeem gebaseerd op de SC-01 verschijnen!

EC-01 Phonetic Voice Synthesizer

DESCRIPTION

The EC-01 Speech Synthesizer is a completely self contained unit. This unit phonetically synthesizes continuous speech of unlimited vocabulary from low data rate inputs.

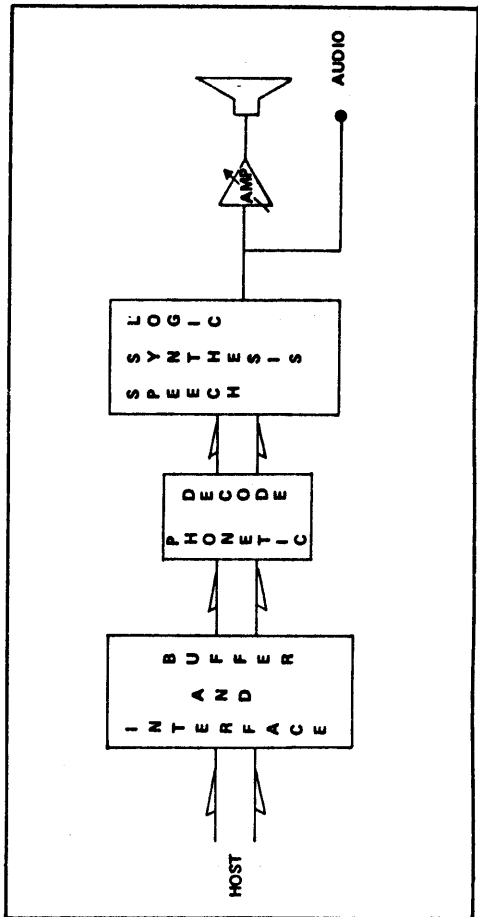
Speech is synthesized by combining phonemes (the building blocks of speech) in the appropriate sequence. The EC-01 contains 64 such phonemes which are accessed by ASCII code. It is the proper sequential combination of these phoneme codes that creates true speech.

The EC-01 has been designed to provide the computer user with a low cost, ultra flexible speech interface that requires a minimum of programmer and hardware support.

FEATURES

- UNLIMITED VOCABULARY
- CONTINUOUS SPEECH
- PROGRAMMABLE INFLECTION
- SIMPLE TO CONNECT
- DIRECT PHONETIC INPUT
- SPEAKER/AUX. OUTPUT
- ON BOARD BUFFER
- POWERED BY HOST.
- MINIMAL POWER CONSUMPTION

FUNCTIONAL BLOCK DIAGRAM



ELECTRICAL SPECIFICATIONS (Typical @ 25°C.)

- Connector type:- Edge connector or ribbon
- Decoding address:- Unused location in host
- Host clock frequency:- Preset but selectable
- Input buffer size:- 1024 Characters
- Audio output power:- Approx. 500mW 8 ohms
- Logic Inputs/Outputs:- 100% compatible with host
- Temperature Range:- Operating 0-40°C
- Power Supply:- +5 Volts 3%
+8-12 Volts reg or unreg
(Both, always from host unit)

ON BOARD BUFFER

To optimise the use of the EC-01 the unit contains an internal buffer, this will allow a message of great length to be output to the unit without 'teeing up' the host unit whilst the message is being output.

DIRECT PHONEME INPUT

To simplify the task of generating the code needed to produce the required speech, the EC-01 has been designed to accept the phonemes as a mnemonic code thus allowing phoneme strings to be sent to the unit in a simple format that can be easily understood by the programmer.

SIMPLE TO CONNECT

The EC-01 comes in various forms to suit many of the micro/computers available. Suitable connector supplied.

OUTPUTS

Two outputs are available on the EC-01. Audio level to suit an external amplifier and speaker level to suit a 500mW loud-speaker.

Price £149.95 + V.A.T.

P. & P. extra

PHONEME CHART

Phoneme Symbol	Duration (ms)	Example Word
EH3	59	jacket
EH2	71	enlat
EH1	121	heavy
PA0	47	no sound
DT	47	butler
A2	71	madge
A1	103	madge
ZH	90	azure
AH2	71	honest
13	55	inhibit
12	121	inhibit
11	103	inhibit
M	80	mat
N	71	bag
B	71	van
V	71	chip
CH*	121	shop
SH	146	zoo
Z	71	zoo
AW1	146	lawful
NG	121	thing
AH1	146	father
OO1	103	looking
OO	186	book
L	103	land
K	80	trick
J*	47	judge
H*	71	hello
G	71	fat
F	103	fast
D	55	paid
S	90	pass

Phoneme Symbol	Duration (ms)	Example Word
A	185	day
AY	65	day
Y1	80	yard
UH3	47	mission
AH	250	map
P	103	past
O	185	cold
T	186	pin
U	103	move
Y	71	any
T	71	tap
R	90	red
E	186	met
W	80	win
AE	185	dad
AE1	103	after
AW2	90	salty
UH2	71	about
UH1	103	uncle
UH	185	cup
O2	80	fur
O1	121	aboard
IU	59	you
UI	90	you
THV	80	the
TH	71	thin
ER	146	bird
EH	186	get
E1	121	big
AW	250	call
PA1	185	no sound
STOP	47	no sound

Esicomp Ltd. reserves the right to alter its product line at any time, or change product specifications or design without notice and without obligation.
Copyright Esicomp 1981.

MICRO, DO YOU HEAR ME ?

a voice recognition system

THE advertisement announced **BIG EARS** in bold capital letters making me wonder if this was a joke from Toy Town; however, a quick scan of the text showed that this was another product from William Stuart Systems. As I already own an audio oscilloscope and a colour graphics system from the same stable I felt confident that my £45 + VAT was not going to buy a toy. I confess to being somewhat relieved that when I unpacked my parcel less than a week later the name Big Ears was not on the brushed aluminium front but in its place an impressive "Speech Recognition System Interface SR1."

Big Ears or SR1 can be connected to any personal computer like NASCOM, TRS 80, PET, UK101, Superboard or to any other via a spare user input port. In my case, an 8K UK101, hooking up couldn't have been much simpler with just five connections to be made. Five core ribbon cable and standard 5 pin DIN plug are provided so plugging in the SR1 is quite straightforward. Power requirements are low so the +5 volts needed comes from the motherboard via the ribbon cable. All this is explained in a very clear User Manual containing Connection Details, Software Loading Instructions, User Instructions, Demonstration Software, Theory of Operation and BASIC Software Listings. A small part of the software is written in machine code and is provided in a form to suit the most popular 6502 or Z80 based systems.

The purpose of the SR1, if you hadn't guessed, is to provide speech input to your computer and open the door to direct man-machine communication. For anyone who has formed a close interpersonal love affair with their computer (yes Ursula Katrina 101 is a person) then giving her Big Ears could well extend a beautiful relationship. To be more exact it is more like one Big Ear in that her ear comes in the form of an electret condenser microphone plugged into the smart aluminium fascia of a black metal case 15cm x 12cm x 5cm using a standard jack plug. Relationships can blossom in that with a speech recognition system it's conceivable that one's computer could be verbally commanded to execute various options in robot control, games programs, etc. Captain Kirk could command "Fire" to zap the Klingons or a remote controlled vehicle could be requested to move "Forward", "Back", "Left" or "Right". It is even conceivable that I could flirt with Ursula Katrina via an interactional conversation program in which she would only have learned (and thus understood) my voiceprint; such faithfulness!

The minimum amount of memory required by the SR1 is 5K but since machine code real-time input routines are loaded into the top of user memory the last one K must be located in the 8th

K sockets leaving the 5th, 6th and 7th sockets empty in the UK 101/Superboard. However, now that 2114 Rams aren't so expensive one might as well upgrade to 8K unless funds are very tight. 8K will allow a larger vocabulary to be stored.

The demonstration software (BASIC) is very interesting providing correlation tables for the words learnt and a 5 x 6 two-dimensional array of numbers as the individual word's "voiceprint". During the learning stage the computer asks for the word to be spelled and then spoken. The spoken word should be repeated 4 to 8 times to achieve optimum recognition and the voiceprint is printed each time. It is naturally important to speak clearly and consistently. Having taught the computer a number of words it is then possible to test its recognition. It can have problems understanding the difference between Fine, Pine and Wine but less problems distinguishing between Apples, Pears and Raspberries. The computer compares the voice it hears with its averaged memory voiceprints and perceives you to have said the word which has the highest correlation. It signals its understanding by printing "You said Raspberries" or whatever. Both the correlation table and voiceprint printout can be deleted by removing lines and the computer's response can be tailored to individual needs.

The computer's voice perception is based on frequency analysis of the first and second formants of the speech waveform. The interface unit SR1 separates the formants and delivers digital values to the computer which then performs frequency analysis for each of sixty-four 16ms sampling periods. This is implemented in machine code. For each period the two formant counts are then compared against threshold data values to determine which frequency ranges are present. The two range indices are then used to determine which location in a two-dimensional array will be incremented. Thus the sixty-four 16ms samples must all fit in this two-dimensional histogram forming a kind of "frequency-space". To learn a word four or more such histograms are averaged and normalised to have a mean value of zero with a uniform standard deviation. The resulting averaged voiceprint-histogram is stored for future correlation.

If necessary the sensitivity of the SR1 can be adjusted by going inside the cabinet with a small screwdriver and adjusting a preset potentiometer; clockwise to increase and anti-clockwise to decrease. The systems software is listening for the first sound in order to start sampling but if the system is too sensitive it could be triggered by background noise. In noisy environments this adjustment is very useful as sensitivity can be set low and the speaker can speak up. Don't clear your voice or stutter because Ursula (or whoever) will be less than understanding about it. Moreover one should remember that with sixty-four samples of 16ms any phrase or word to be learnt should not last longer than about one second. This is quite okay in practice as long as one avoids Welsh railway station names.

If one wishes to use the SR1 in the application of data enquiry (which might need a large vocabulary) then the secret of success is to use key words to define which group of words the computer can expect you to be using next. In this way the computer need only compare the word it hears with a reasonably small group of words and the chance of misunderstandings is thus correspondingly reduced. Words which the computer might confuse can be located in different groups. Vocabulary, therefore, can grow like a tree. An example of this is a Travel Reservation System which might initially ask "Inland, European or Intercontinental?" Each of these headings might access 8 or 10 destinations with further destinations to be found by "Other" always being one of the options.

In conclusion Ursula Katrina's Big Ear might not be quite as perceptive as a human ear but then it's not such fun to nibble either. Notwithstanding that I'm still convinced that she's got the hot-heatsinks for me... "Love me Katrina?"... "I DO HONEY"... If only things could always be that easy.

TYPE-'N-TALK™ IS T.N.T.

The exciting text-to-speech synthesizer that has every computer talking.

- **Unlimited vocabulary**
- **Built-in text-to-speech algorithm**
- **70 to 100 bits-per-second speech synthesizer**

Type-'N-Talk™, an important technological advance from Votrax, enables your computer to talk to you simply and clearly — with an unlimited vocabulary. You can enjoy the many features of Type-'N-Talk™, the new text-to-speech synthesizer, for just \$345.00.

You operate Type-'N-Talk™ by simply typing English text and a talk command. Your typewritten words are automatically translated into electronic speech by the system's microprocessor-based text-to-speech algorithm.

The endless uses of speech synthesis.

Type-'N-Talk™ adds a whole new world of speaking roles to your computer. You can program verbal reminders to prompt you through a complex routine and make your computer announce events. In teaching, the computer with Type-'N-Talk™ can actually tell students when they're right or wrong — even praise a correct answer. And of course, Type-'N-Talk™ is great fun for computer games. Your games come to life with spoken threats of danger, reminders, and praise. Now all computers can speak. Make yours one of the first.

Text-to-speech is easy.

English text is automatically translated into electronically synthesized speech with Type-'N-Talk™. ASCII code from your computer's keyboard is fed to Type-'N-Talk™ through an RS 232C interface to generate synthesized speech. Just enter English text and hear the verbal

response (electronic speech) through your audio loud speaker. For example: simply type the ASCII characters representing "h-e-l-l-o" to generate the spoken word "hello."

TYPE-'N-TALK™ has its own memory.

Type-'N-Talk™ has its own built-in microprocessor and a 750 character buffer to hold the words you've typed. Even the smallest computer can execute programs and speak simultaneously. Type-'N-Talk™ doesn't have to use your host computer's memory, or tie it up with time-consuming text translation.

Data switching capability allows for ONLINE usage.

Place Type-'N-Talk™ between a computer or modem and a terminal. Type-'N-Talk™ can speak all data sent to the terminal while online with a computer. Information randomly accessed from a data base can be verbalized. Using the Type-'N-Talk™ data switching capability, the unit can be "de-selected" while data is sent to the terminal and vice-versa — permitting speech and visual data to be independently sent on a single data channel.

Selectable features make interfacing versatile.

Type-'N-Talk™ can be interfaced in several ways using special control characters. Connect it directly to a computer's serial interface. Then a terminal, line printer, or additional Type-'N-Talk™ units can be connected to the first Type-'N-Talk™, eliminating the need for additional RS-232C ports on your computer.

Using unit assignment codes, multiple Type-'N-Talk™ units can be daisy-chained. Unit addressing codes allow independent control of Type-'N-Talk™ units and your printer.

Look what you get for \$345.00. TYPE-'N-TALK™ comes with:

- Text-to-speech algorithm
- A one-watt audio amplifier
- SC-01 speech synthesizer chip (data rate: 70 to 100 bits per second)
- 750 character buffer
- Data switching capability
- Selectable data modes for versatile interfacing
- Baud rate (75-9600)
- Data echo of ASCII characters
- Phoneme access modes
- RS 232C interface
- Complete programming and installation instructions

The Votrax Type-'N-Talk™ is one of the easiest-to-program speech synthesizers on the market. It uses the least amount of memory and it gives you the most flexible vocabulary available anywhere.

Order now. Toll free.

Call the toll-free number below to order or request additional information. MasterCard or Visa accepted. Charge to your credit card or send a check for \$345.00 plus \$4.00 delivery. Add 4% sales tax in Michigan.

1-800-521-1350.

Votrax

Distributed by Vodex
A Votrax Company — Dept. RT
500 Stephenson Highway, Troy, MI 48064
(313) 568-0341

Type-'N-Talk™ is covered by a limited warranty. Write Votrax for a free copy.

Volgens ingewijden duurt het niet langer meer dan pakweg een jaar of twee, voordat we de eerste schrijfmachine hebben waarbij het toetsenbord is vervangen door een mikrofoon. Voorleesapparaten voor blinden en spreek- en leesapparatuur voor doofstommen zullen ook niet lang meer op zich laten wachten.

Een van de experts op dit gebied beeerde onlangs al schertsend dat het hoog tijd werd om maatregelen te nemen die het gebruik van sprekende instrumenten in vliegtuigcockpit verbieden – daarbij wijzend op de akoestische chaos die ontstaat als zo'n 20 meetinstrumenten door elkaar gaan zitten praten.

tong (de generator), opgehangen in een metalen houder. Terwijl hij op die tong tokkelt, drukt de speler de houder tegen zijn tanden en gebruikt dan zijn mondholte als klankkast. Door de mondompening groter en kleiner te maken, verandert de kleur van het geproduceerde geluid. Geoefende spelers kunnen zo klanken voortbrengen die sterk doen denken aan de klinkers a, e, i, o en u in verschillende toonhoogten, zulks afhankelijk van de afmetingen van hun "klankkast".

Reeds in 1936 ontwierp ene meneer K.W. Wagner een systeem waarmee het akoestische generator/klankkast-principe welhaast perfect langs elektrische

H. Baumann

spraaksyntese (1)

dialog met machine geen toekomstmuziek meer

Ook in gewone huis, tuin- en keukenzaken heeft de "chip" al lang zijn intrede gedaan. Het is ondertussen al min of meer gewoon geworden dat bij winkelkassa's, wasautomaten en naaimachines, instructies via een toetsenbordje worden doorgegeven en dat een of andere vorm van microprocessor ervoor zorgt dat de machine die wensen stipt uitvoert.

Maar een wasmachine zonder knoppen die mondelinge instructies uitvoert, of bijvoorbeeld een automatisch lasapparaat dat met de technicus een babbeltje maakt over een werkstuk, dat is weer even iets anders. Voor sommigen een wensdroom, voor anderen een schrikbeeld, maar door vrijwel iedereen voorlopig nog als science-fiction beschouwd. Toch is het dat al lang niet meer. Er wordt op het ogenblik heel hard gewerkt om machines te leren praten en luisteren.

Dat mag dan een tikje overdreven zijn, feit is wel dat men met de sprekende machines al veel verder op scheut is dan de meesten erg in hebben. Alle reden dus om eens nader op deze materie in te gaan.

Omwille van de overzichtelijkheid is dit artikel opgesplitst in twee delen; in het eerste deel zullen we ons hoofdzakelijk bezig houden met het kunstmatig opwekken van spraaksignalen, terwijl in het tweede deel aandacht wordt besteed aan de problematiek rond het "gehoor" van de machines

De eerste pogingen

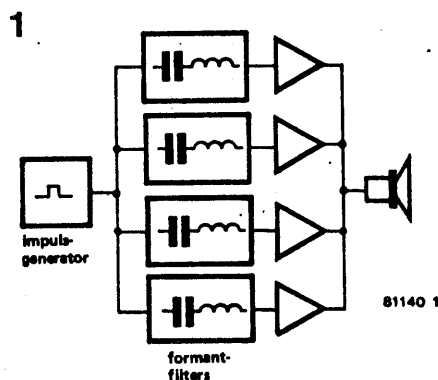
Twee elementaire bestanddelen van spraakachtige klanken vindt men terug in de zg. mondtrommel; een instrument dat in de volksmuziek van het alpengebied graag gebruikt wordt. Zo'n mondtrommel bestaat uit een blikken

weg kan worden gerealiseerd. Figuur 1 toont dat het daarbij ging om een impuls-generator, gevolgd door een aantal resonantiefilters ("formant-filters"). Met dit, naar hedendaagse begrippen gerekend zeer simpele apparaat konden alle klinkers, alsmede de stemhebbende medeklinkers l, m, n en r reeds zo goed worden gesynthetiseerd, dat zelfs individuele kleuring van het geluid mogelijk was.

Slechts luttele jaren later kwam H. Dudley in de Bell Telephone Laboratories een belangrijke stap verder: voortbordurend op Wagners vinding, bouwde hij in 1939 de "voder" (voice operation demonstrator). Dit spraakapparaat was al in staat om alle klanken op te wekken die voor normale spraak nodig zijn – al hing het resultaat wel voor een groot deel af van de bekwaamheid van de degene die de knoppen bediende. In de opzet van de "voder" zijn al zoveel spraakgenerator-principes te herkennen dat een nadere beschouwing zeker de moeite waard is. Maar eerst even een kort uitstapje naar de PTT.

Wanneer u per telefoon naar de juiste tijd informeert, dan hoort u aan de andere kant van de lijn de stem van een beklagenswaardig iemand die ooit eens een bandje met tijden voor een hele dag ingesproken moet hebben. Een monnikenwerk! In werkelijkheid loopt het echter zo'n vaart niet: elk getal, alsmede de woorden "uur", "minuut(en)", "en", "seconde(n)" zijn slechts één enkele keer opgenomen en elke afzonderlijke tijdsaankondiging wordt automatisch uit deze woorden samengesteld.

We gaan in gedachten eens een stapje verder en beperken de ontleding van een zin niet tot de afzonderlijke woorden, maar bekijken ook de klanken waaruit



Figuur 1. Het eerste elektronische apparaat waarmee stemgeluiden konden worden opgewekt werd in 1936 gebouwd door K. W. Wagner. Ondanks de simpele opzet waren de prestaties al verrassend goed te noemen.

elk gesproken woord is samengesteld. Het woord "acht" bijvoorbeeld, bestaat uit drie spraakklanken ("fonemen"): de klinker "a", de op rose ruis gelijkende "ch" en de harde klank "t". Deze klanken vormen de kleinst mogelijke elementen waarin spraak valt te ontleden. Wanneer het lukt om alle voor een bepaalde taal noodzakelijke klanken op te wekken, dan is het mogelijk om een machine te laten "praten". En dat is nu precies wat Dudley's "voder" al kon.

Figuur 2 geeft het blokschema van het door Dudley ontwikkelde apparaat. Een impulsgenerator die een signaal produceert dat rijk is aan harmonischen, levert het ruwe materiaal voor alle stemhebbende klanken, een ruisgenerator doet dienst als signaalbron voor de stemloze klanken (zoals bijv. de f, s, sch, enz). Welke klank er gevormd wordt, hangt voor een zeer groot deel af van de instelling van de filterbank die achter de generators is geschakeld en die het karakteristieke spektrum (formantbereik) van elke klank nabootst. Een voorbeeld: de klank "s" maakt de inschakeling noodzakelijk van een filter dat op een vrij hoge frekwentie is afgestemd, terwijl de "sch" een stuk donkerder klinkt waardoor het filter dus een veel lagere doorlaatfrekwentie zal moeten hebben.

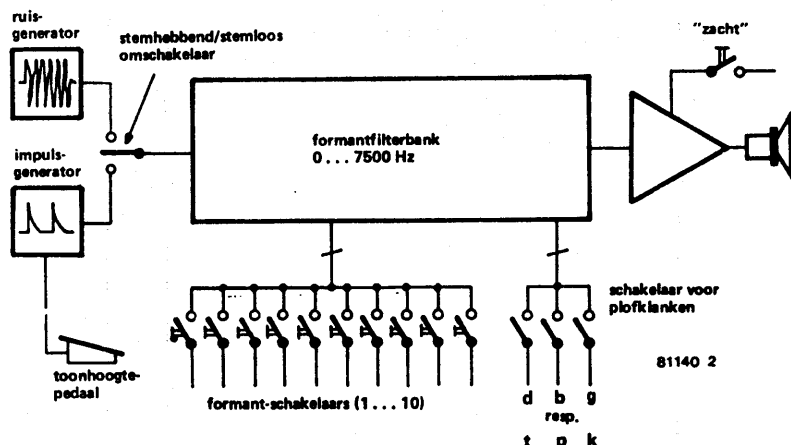
Verskillende filterschakelaars van de voder zijn onderling gekoppeld, zodat er tegelijkertijd meerdere formantbereiken in de signaalweg kunnen worden geschakeld. Met een extra schakelaar ("zacht") kan de geluidsterkte bij stemloze klanken worden verminderd. Tenslotte zijn er nog drie schakelaars voor fonemen die echt een luchtdrukstoot veroorzaken (de zogeheten plofklanken): d, b en de Duitse g in de stand "stemhebbend" en t, p en k in de stand "stemloos" van de omschakelaar. Om geen steriel robotgeluid te krijgen, moet - net als dat bij normale spraak gebeurt - bovendien onder het spreken de toonhoogte van de impulsgenerator worden gevarieerd. Daarom is voorzien in een toonhoogte-pedaal.

Al met al een behoorlijk gekompliceerd apparaat, dat een haast artistieke bekwaamheid vergde van de "knoppenist" (zie figuur 3). Vervangt men echter de "live"-bediening door een geschikte automatische sturing van de afzonderlijke functies (bijv. m.b.v. een ponsband), dan is de eigenlijke "sprekmaschine" perfect. Aan de principiële opbouw ervan is (tenminste bij de foneem-generators) sinds Dudley's voder praktisch niets veranderd. De afmetingen zijn intussen echter tot het formaat van een Eurokaartje gekrompen - recentelijk zelfs tot het formaat van een IC.

Nieuwe oplossingen

Alle tot dusver bekende methoden om tot spraaksyntese te komen, vallen in twee categorieën onder te brengen:

2



Figuur 2. Blokschema van de "voder". Dit in 1939 door Homer Dudley ontworpen apparaat kan worden beschouwd als de eerste echte "sprekmaschine". Alle voorkomende spraakklanken konden er natuurgetrouw mee worden nagebootst.

spraakopwekkers die complete woorden in een geheugen opslaan (woordgenerators) en typen die alle afzonderlijke spraakklanken kunnen vormen en daaruit willekeurige woorden samenstellen (foneemgenerators). Omdat de werking van de eerstgenoemde groep wat eenvoudiger is, zullen we die maar het eerst onder de loep nemen.

Woordgenerators

De mogelijkheid om hele woorden in halfgeleidergeheugens (ROM's) op te slaan kon pas serieus worden overwogen toen de geheugenkapaciteit hiervan belangrijk toenam en tegelijk de prijs lager werd. Dit type generator vraagt namelijk nogal wat geheugenruimte. Alle woorden worden in hun totaliteit gedigitaliseerd, en in het geheugen gestopt, waarna ze elk onder een apart adres kunnen worden afgeroepen. Dat maakt het mogelijk om op zeer eenvoudige wijze zinnen te vormen: men hoeft slechts de woordadressen na elkaar af te vragen om een complete zin te krijgen. Deze vorm van programmering kan door elke gebruiker zelf worden uitgevoerd en vereist geen bijzondere spraakteoretische kennis, daar aan het spraakmateriaal zelf niets wordt veranderd.

Het zou echter een zeer oneconomische aangelegenheid zijn om het spraaksignaal simpelweg met een A/D-omzetter te digitaliseren en dan de bits in ROM's of RAM's op te slaan. Al naar gelang de resolutie van de A/D-omzetter in kwestie krijgt men zo namelijk een hoeveelheid informatie van 50.000 tot 100.000 bits voor elke seconde dat het spraaksignaal duurt. Het is derhalve zaak om deze hoeveelheid zoveel mogelijk te reduceren, alvorens die in een geheugen op te slaan. Dat is ook heel goed mogelijk, aangezien onze spraak veel meer elementen bevat dan voor een goede verstaanbaarheid strikt noodzakelijk is.

Zonder merkbare verslechtering van de herkenbaarheid kan een aanzienlijk deel van die in wezen overvloedige informatie gewoon worden weggelaten. Dat dit zo is kan iedereen gemakkelijk zelf nagaan; zelfs wanneer men luid schreeuwt (zonder verfiende articulatie dus), met volle mond praat of bij ontzettend veel achtergrondlawaai (fabriekshal, waterval, slechte telefoonverbinding), dan nog blijft de verstaanbaarheid meestal verrassend goed. Informatie-reduktie kan op verschillende manieren worden bereikt:

a. Adaptieve deltamodulatie

Aangezien spraak, als gevolg van de mechanische traagheid van de menselijke spraakorganen, slechts relatief lang-

zame amplitudevariaties bevat, is het onnodig om voor ieder digitaliseringspunt opnieuw de absolute amplitude in het geheugen op te slaan; het volstaan om de amplitudeverandering tussen twee opeenvolgende punten vast te leggen.

Aangenomen dat een bepaald punt een amplitudewaarde van 500 bezit, dan hoeft men echt niet te verwachten dat er onmiddellijk daarna een sprong naar bijv. 18 zal optreden. Daar de digitalisering plaatsvindt met een frequentie die tenminste twee maal zo hoog is als de hoogste spraakfrequentie, zijn de tussenpozen waarmee hetingangssignaal gemonsterd wordt zó kort dat het verschil tussen twee opeenvolgende amplitudewaarden nooit groter dan + of -25 zal zijn. De volgende waarde zal dus ongetwijfeld ergens tussen 475 en 525 liggen. Het opslaan van dat verschil in het geheugen kost aanzienlijk minder bits dan het steeds opnieuw vastleggen van de absolute amplitudewaarde.

Met een klein beetje moeite kan nog meer aan geheugenruimte worden bespaard; wanneer we de amplitude steeds een zelfde (gemiddeld) stapje groter of kleiner laten worden, dan kan bij elk digitaliseringspunt worden volstaan met een richtingsaanduiding: "omhoog" (logisch 1) of "omlaag" (logisch 0). Deltamodulatie geeft dus al een behoorlijke besparing aan geheugenruimte.

b. Onderdrukking van signaalkomponenten met geringe amplitude
Frequenties die na Fourier-analyse slechts met een zeer kleine amplitude in het spraaksignaal vertegenwoordigd blijken, kunnen nagenoeg zonder enig kwaliteitsverlies worden weggelaten.

De totale hoeveelheid vast te leggen informatie kan daarmee tot ongeveer de helft worden teruggebracht.

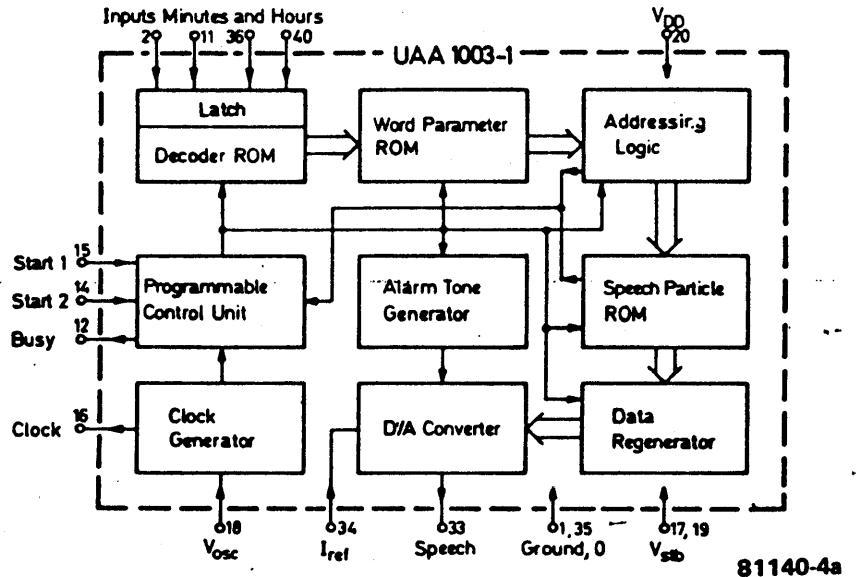
c. Herhaling van pseudostatische trillingsperioden

Dat klinkt een beetje moeilijk, maar dat is het niet. Het betreft hier een vorm van kompressie die gebaseerd is op het feit dat afzonderlijke trillingen binnen een bepaalde stemloze of stemhebbende klank onderling geen markante verschillen vertonen. Een langgerekte klinker bijvoorbeeld, hoeft daarom niet perse in zijn geheel te worden vastgelegd; het volstaat om één periode te bekijken en die samen met een herhalingsinstructie (voor een bepaald aantal perioden) in het geheugen op te slaan. Meestal is herhaling met een faktor 4 mogelijk bij stemhebbende en een faktor 10 bij stemloze klanken. Ook dit geeft een aanzienlijke besparing aan benodigde geheugenruimte.

Tot nu toe wordt de toepassing van kunstmatige spraakapparatuur volgens het woordgenerator-principe toch nog steeds beperkt door enkele wezenlijke nadelen.

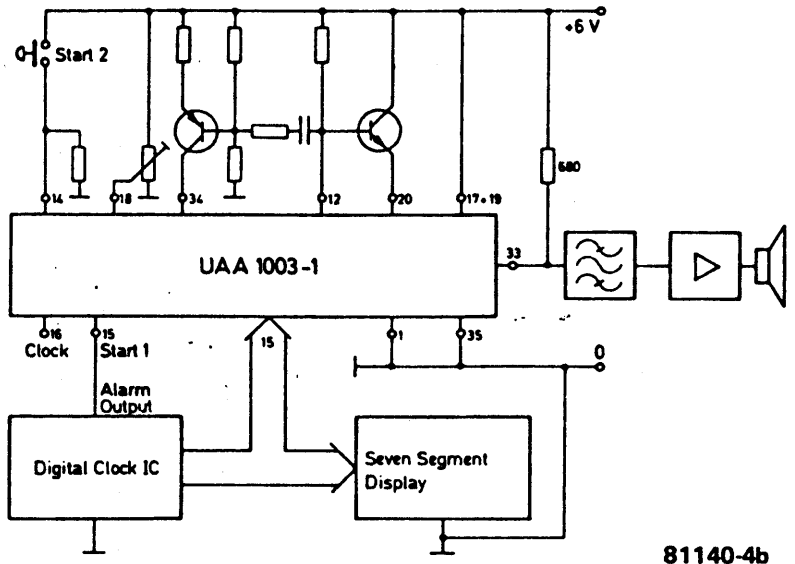
1. De vokabulaire is niet "per programma" uit te breiden. Weliswaar kunnen de ROM's tegenwoordig al door de fabrikant van een specifieke vokabulaire worden voorzien als men dat

4a



Figuur 4a. Het IC UAA1003 van ITT is een 1-chip-spraakgenerator. Behalve de eigenlijke spraakgenerator bevat het ook een geprogrammeerd geheugen (ROM) met een capaciteit van maximaal 20 woorden. Uitbreiding van het geheugen is niet mogelijk en het IC leent zich dus bij uitstek voor korte aankondigingen.

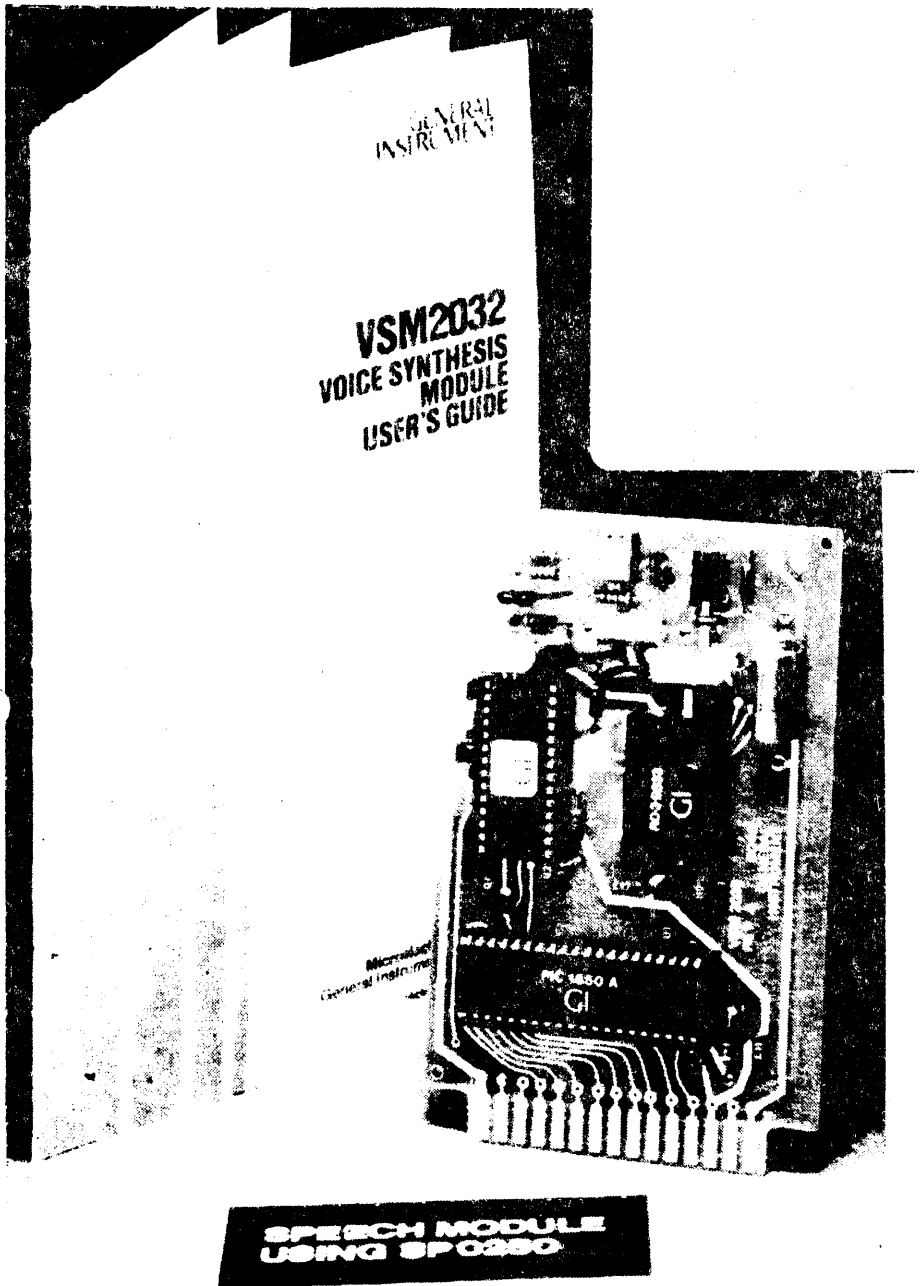
4b



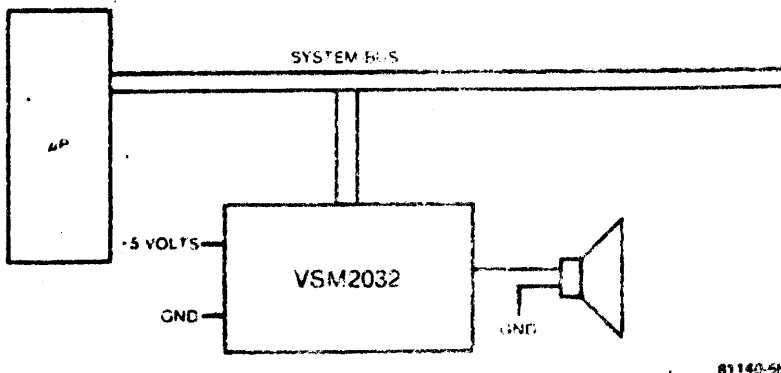
Figuur 4b. Typische toepassing van de UAA1003-1 in een "spreekende" digitale klok. De data-ingangen van de spraakgenerator zijn rechtstreeks met de anoden van de 7-segment displays van de klok verbonden. Via de "alarm"-uitgang van het klok-IC wordt een tijdsaankondiging verkregen met wektoon vooraf; drukknop "start 2" is voor tijdsaankondigingen zonder wektoon. De instelpot bepaalt de interne clock-frequentie van de UAA1003; de aan pen 34 liggende stroombron levert een referentiestroom welke het nivo van het uitgangssignaal op pen 33 bepaalt.

In rust trekt de schakeling via pen 17 en 19 een uiterst geringe "stand by"-stroom. Aan het begin van elke aankondiging wordt via pen 12 de aan pen 20 liggende transistor opengestuurd; die levert dan de voor de aankondiging benodigde stroom. Het gemiddelde stroomverbruik blijft zo heel laag.

Tussen het generator-IC en de LF-versterker kan met een simpel bandfilter worden volstaan.



5b



81140-5h

Figuur 5. De spraakmodule VSM 2032 van General Instruments is uitgevoerd als een klein steekprintje en bevat een complete spraaksynthesizer, bestaande uit interface-unit, spraakgenerator-IC SP-0250, ROM en LF-versterker. De adressering geschiedt via een microprocessor-systeem met 8 TTL-kompetibele adressignalen (zie 5b). Er kunnen 128 woorden worden geadresseerd; de capaciteit bedraagt ongeveer 30 seconden spraak. Behalve ROM's met standaardinhoud, levert G.I. ook naar wens geprogrammeerde ROM's

wenst, maar voor wijzigingen of uitbreidingen is gelijk een heel nieuwe ROM nodig. Wil het voor de gebruikers nog een beetje goedkoop en economisch blijven, dan zullen zij zich dus moeten beperken tot het gebruik van standaardvokabulaires; het feit dat ietwat afwijkende eisen het apparaat al gauw letterlijk en figuurlijk sprakeloos doen staan moet men dan maar op de koop toe nemen.

2. De totale woordenschat van woordgenerators is, hoe dan ook, zeer beperkt; het haalbare maximum ligt op het ogenblik bij ongeveer 250 woorden. Bij dit aantal heeft de "Speech Processor Set" van National Semiconductor al 128 Kbit aan geheugenruimte nodig en dat is zelfs naar huidige begrippen gerekend niet bepaald weinig. Daar komt nog bij dat deze verhouding tussen vokabulaire en geheugenruimte slechts mogelijk is als alle mogelijke informatie-reduktietruuks worden toegepast - anders zou er aanzienlijk meer geheugenruimte nodig zijn geweest.

3. Onze spraak is eigenlijk zeer gecompliceerd van samenstelling; het gaat niet alleen om de juiste afzonderlijke klanken, maar er zijn ook nog tal van overgangen, klemtoonvariaties, pauzes en toonhoogtewisselingen. Deze parameters liggen bovendien niet vast, maar veranderen voor elk woord al naar gelang de context waarin het wordt gebruikt. Zo leren kinderen bijvoorbeeld al vroeg om aan het eind van een zin de toonhoogte van hun stem te laten dalen. Dat is nu precies een van die dingen die een woordgenerator niet kan; elk woord is met een neutrale uitspraak in het ROM vastgelegd, zodat de geproduceerde spraak altijd sterk aan robotstemmen zal doen denken en geen echt menselijk karakter heeft.

Dat wil overigens niet zeggen dat er voor de woordgenerator totaal geen toekomstperspektief zou zijn - tenslotte zijn we ondertussen ook gewend geraakt aan de lalijke, maar praktische zeven-segment-symbolen.

Fonengenerators

Deze groep spraakapparatuur vertoont sterke verwantschap met Dudley's "voder"-koncept. Ook hier vinden we de eigenlijke spraakgenerator met stemhebbend/stemloos klankopwekkers, alsmede de formantfilters en de toonhoogtesturing. Nieuw ten opzichte van de voder zijn echter de gehegencellen waarin de werkvoorschriften (algoritmen) zijn opgeslagen voor de vorming van de afzonderlijke fonemen.

Wanneer er aan de ingang van het spraakblok een instructie verschijnt tot vorming van een bepaalde fonem, dan worden alle parameters zo ingesteld dat het generatorsignaal in combinatie met de filterwerking het juiste geluid oplevert. Al naar gelang de complexiteit van de te produceren klanken, kunnen daarvoor maximaal 10 parameters

tegelijk nodig zijn. Voor de vorming van een standaardvokubulaire heeft de foneemgenerator dus nog een geheugen nodig waarin de foneemcodes worden opgeslagen van de voor elk woord benodigde afzonderlijke klanken.

Het grote voordeel van foneemsynthese zal ondertussen duidelijk zijn. Als het apparaat in staat is om alle klanken weer te geven die in een taalfamilie voorkomen, dan beschikt het in principe over de komplette woordenschat van deze talen. Weliswaar bezit elke taal meer klanken dan letters, maar aangezien dat aantal in ieder geval onder de honderd blijft, zijn ze toch betrekkelijk gemakkelijk zonder een buitensporige hoeveelheid foneem-vormingscodes te reproduceren. Voor de taalgroep Duits/Engels bijvoorbeeld, volstaan 64 fonemen om alle voorkomende woorden goed verstaanbaar te syntetiseren.

De organisatie van elk foneemkommando geschiedt bij de VOTRAX VS-6.0 (om maar een voorbeeld te noemen) in de vorm van een 8-bit kodewoord. Van die 8 bit dienen er 6 voor het uitkiezen van de gewenste foneem en worden er twee gebruikt om met behulp van toonhoogtevariëaties etc. de spraak levendiger en echter te doen schijnen. In de 64 foneemcodes zijn ook verschillende variatie-stapjes inbegrepen voor klemtoon en snelheid van een bepaalde klank, zodat er sprake is van een overzichtelijke en duidelijke samenhang tussen code en klank. Tabel 1 toont als voorbeeld de foneeminstructies van de VOTRAX SC 01 speech synthesizer.

Net zo goed als de vormingsparameters van geluid tot geluid verschillen, is ook de lengte van elke complete foneem weer anders. Die kan variëren van 250 ms voor de langste, tot 47 ms voor de kortste klank. Pauze-instructies en een "stop"-kode completeren het instructiepakket.

In plaats van bij een paar (meestal drie of vier) formantfilters in een bepaalde volgorde de resonantiefrequentie te veranderen, zoals dat bij de VOTRAX gebeurt, kan men ook een stel (tot maximum 15) parallelgeschakelde filters toepassen die elk een andere vast-ingestelde resonantiefrequentie bezitten.

Als ingangssignaal dient weer het stemhebbend/stemloos signaal van de impuls-, respectievelijk ruisgenerator. Variaties in het klankarakter worden bereikt door elk afzonderlijk filter te laten volgen door een spanningsgestuurde versterker (VCA); daarmee kan het aandeel dat elk gefilterd frequentiebandje in het uiteindelijke geluidsspektrum heeft worden vastgelegd — precies zoals dat in het syntesedeel van een vocoder gebeurt.

Voor het vastleggen van de vormingsalgoritmen wordt bij de fabrikant het amplitude-aandeel van elk afzonderlijk frequentiebandje voor de fonemen onderzocht en in een geheugen opgeslagen. Een speciale variant van de foneemsynthese met formantfilters vormt

het Linear Predictive Coding (LPC). Evenals bij de vorige methode is ook hier de flexibiliteit van de geproduceerde spraak zeer hoog en de vokubulaire onbepaald.

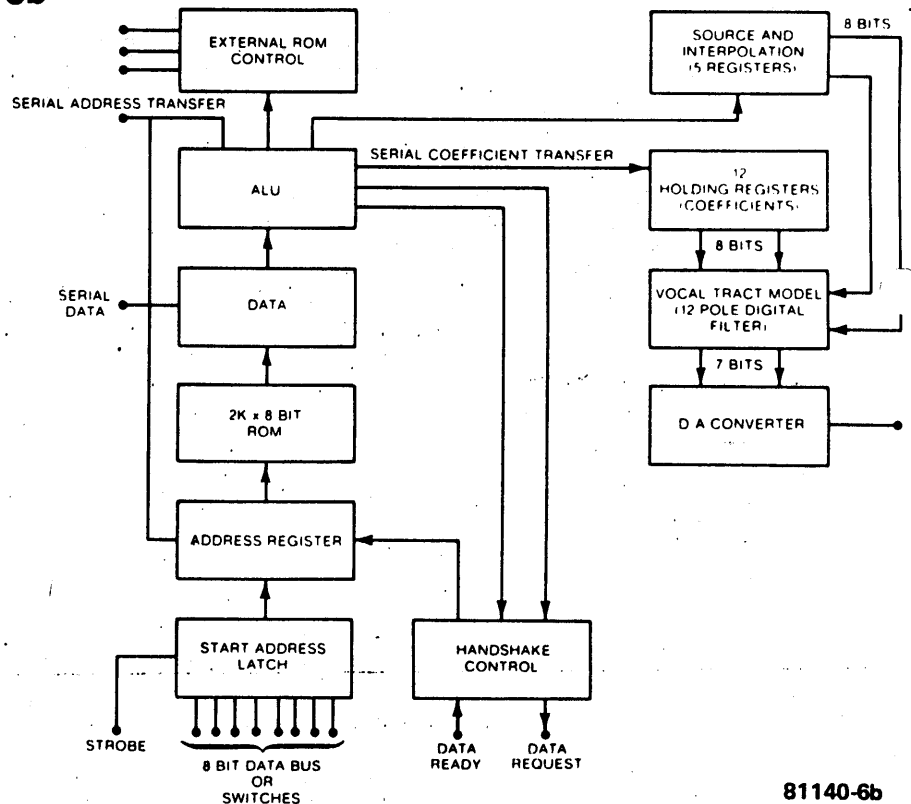
Bij foneemgenerators kan ook gebruik worden gemaakt van alle eerdergenoemde (bij de bespreking van de woordgenerator) truiks om geheugenruimte te sparen. De winst die dat oplevert kan worden gebruikt voor de regeling van de zogeheten "allofonen; dat zijn de zich tussen de afzonderlijke fonemen in bevindende overgangstoestanden, welke in hoge mate medebepalend zijn voor de natuurlijkheid van de spraak.

Samenvattend kan worden vastgesteld dat foneemgenerators voldoen aan zeer

hoge eisen wat betreft woordenschat en uitspraak. Het enige "maar" vormt eigenlijk het feit dat de vokubulaire niet in letters maar in klanken moet worden ontleend en geprogrammeerd. Schrijf- en spreektaal liggen zo goed als altijd een flink eind uiteen (met het Fins als gunstige en bijvoorbeeld het Frans als bijzonder ongunstige uitzondering). Tot dusver was het daarom niet mogelijk om de vokubulaire van een spraaksynthesizer volgens het foneemgenerator-principe met behulp van een toetsenbordje in te voeren.

Recentelijk is men er echter in geslaagd om de "spreekmachines" ook spraakregels bij te brengen; voor het Amerikaanse taalgebied bestaan al verschillende programma's die de om-

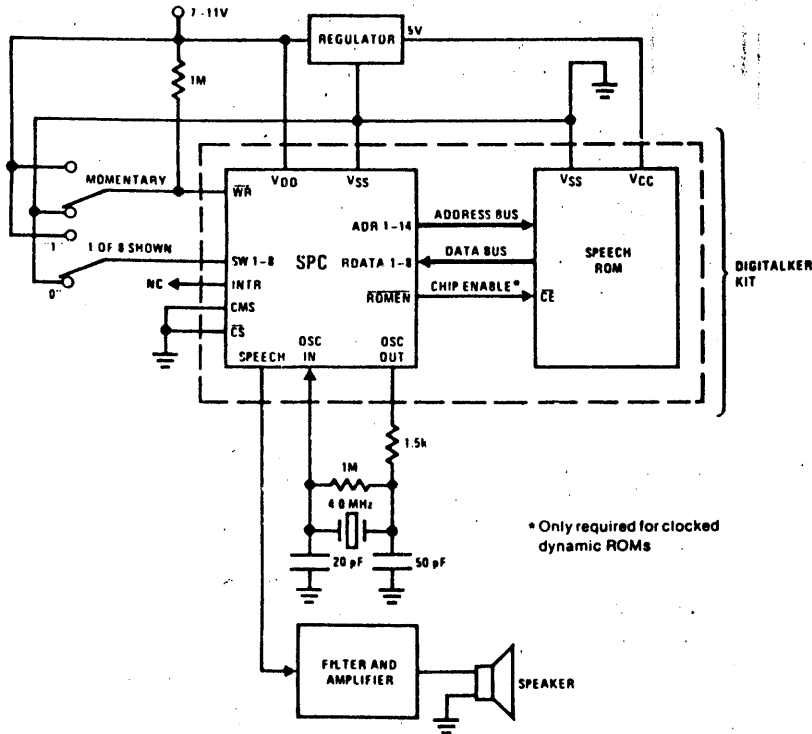
6b



81140-6b

Figuur 6. De SP-0256 van G.I. is een verdere ontwikkeling van de SP-0250. Deze nieuwe woordgenerator heeft een geheugen van 16K ROM. Bovendien kan er maximaal 491K externe ROM worden geadresseerd. Opvallend is het programmeerbare digitale filter in de spraaksynthesizer. Bij de opzet van dat filter hebben de eigenschappen van de menselijke spraakorganen als voorbeeld gediend en het stelt de gebruiker in staat om zelf de kwaliteit en dus de vereiste geheugenruimte voor de geproduceerde spraak (tussen ca. 700 en 2000 bit per seconde) te kiezen.

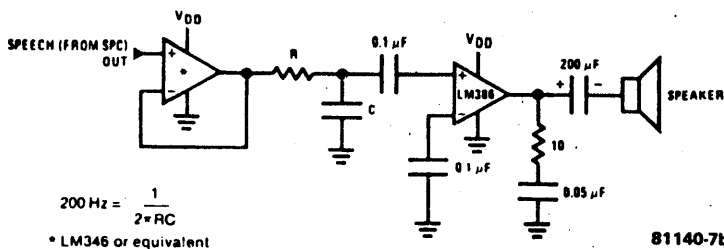
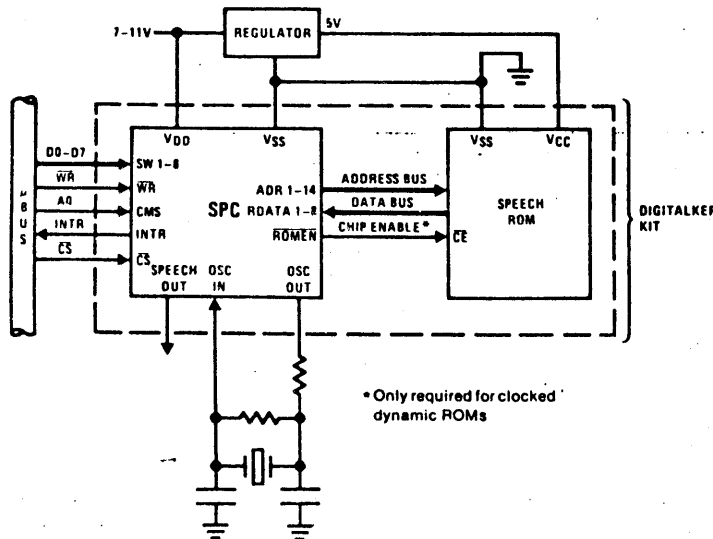
7a



Tabel 1

Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word
00	EH3	59	jacket
01	EH2	71	enlist
02	EH1	121	heavy
03	PA0	47	no sound
04	DT	47	butter
05	A2	71	made
06	A1	103	made
07	ZH	90	azure
08	AH2	71	honest
09	I3	55	inhibit
0A	I2	80	inhibit
0B	I1	121	inhibit
0C	M	103	mat
0D	N	80	sun
0E	B	71	bag
0F	V	71	van
10	CH*	71	chip
11	SH	121	shop
12	Z	71	zoo
13	AW1	146	lawful
14	NG	121	thing
15	AH1	146	father
16	OO1	103	looking
17	OO	185	book
18	L	103	land
19	K	80	trick
1A	J*	47	judge
1B	H	71	hello
1C	G	71	get
1D	F	103	fast
1E	D	55	paid
1F	S	90	pass
20	A	185	day
21	AY	65	day
22	Y1	80	yard
23	UH3	47	mission
24	AH	250	mop
25	P	103	past
26	O	185	cold
27	I	185	pin
28	U	185	move
29	Y	103	any
2A	T	71	tap
2B	R	90	red
2C	E	185	meet
2D	W	80	win
2E	AE	185	dad
2F	AE1	103	after
30	AW2	90	safty
31	UH2	71	about
32	UH1	103	uncle
33	UH	185	cup
34	O2	80	for
35	O1	121	aboard
36	IU	59	you
37	U1	90	you
38	THV	80	the
39	TH	71	thin
3A	ER	146	bird
3B	EH	185	get
3C	E1	121	be
3D	AW	250	call
3E	PA1	185	no sound
3F	STOP	47	no sound

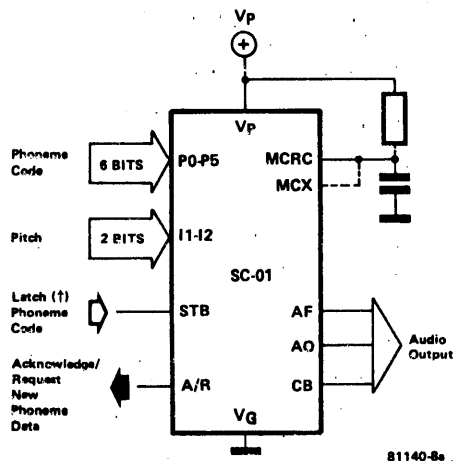
7b



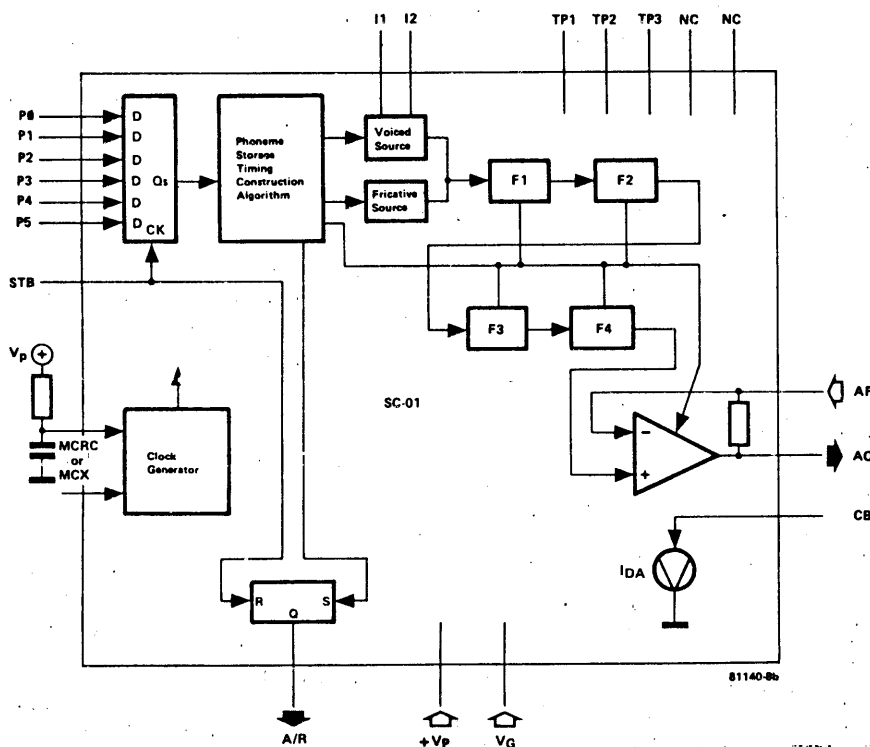
Figuur 7. "Digitaltalker" noemt National Semiconductor haar spraaksynthese-systeem. Het bestaat uit een spraakprocessor met externe ROM's. Figuur 7a toont de allereenvoudigste toepassing met een "schakelaar-interface" voor de adreskeuze. De SPC (speech processor) haalt de bij het gekozen 8-bit adres behorende digitale informatie uit de ROM en zet die in spraaksignalen om. Er kunnen 256 verschillende woorden resp. zinnen worden afgeroepen. De SPC kan 128 K-ROM rechtstreeks adresseren en eenvoudig met een nog groter geheugen worden uitgebreid. Figuur 7b toont nog een typische toepassing van het IC in combinatie met een μ p-systeem en met een zo eenvoudig mogelijk gehouden LF-filtering.

Tabel 1. Foneeminstructies van de VOTRAX SC 01 speech synthesizer.

8a



8b



Figuur 8. Het CMOIC-IC SC-01 van VOTRAX herbergt een vrijwel complete foneemgenerator. Figuur 8a toont de in- en uitgangen. M.b.v. 8-bit adressen (zie tabel 1) kunnen 64 verschillende fonemen worden afgeroepen. Twee andere bits (pitch) maken een keuze tussen vier verschillende toonhoogten mogelijk. Met "latch" wordt de laatst ingegeven foneemcode in een tussengeheugen opgeslagen. Als de geadresseerde foneem aan de uitgang is verschenen, vindt via R/A een terugmelding plaats, waarmee de generator de volgende foneemcode opvraagt. Van de drie audio-uitgangen levert er slechts één het spraaksignaal. In figuur 8b is te zien dat AO de stuuruitgang is voor de (externe) eindtransistoren; AF dient voor de tegenkoppeling en CB voor de ruststroominstelling van de eindtrap. Figuur 8b illustreert tevens de werking van de foneemgenerator. De foneemcode aan de ingang belandt eerst bij een tussengeheugen en adresseert vervolgens een ROM waarin de foneemparameters zijn opgeslagen. Die ROM stuurt het syntese-deel, dat uit een impulsgenerator voor de stemhebbende klanken (voiced source), een ruisgenerator voor de stemloze klanken (fricative source) en vier foneemfilters (F1... F4) bestaat. Voor de stemhebbende klanken wordt de toonhoogte van de generator rechtstreeks via de ingangen 11 en 12 bepaald - dus onafhankelijk van de foneemcode.

zetting verzorgen van letter-toetsenbord naar klank-code. Als we de fabrikant mogen geloven, dan zullen dergelijke microprocessor-programma's binnenkort ook in andere talen beschikbaar zijn. De door de machines te spreken tekst kan dan zo vanaf het papieren concept worden ingetikt, hetgeen vooral bij vaak wisselende teksten (beursberichten, telefoonbeantwoorder) een niet onaanzienlijk voordeel betekent.

Welk systeem wordt het?

Over de vraag welk type spraaksynthesizer de beste papieren heeft voor de toekomst, valt wellicht een voorzichtigte prognose te doen als we de benodigde geheugenruimte (in bit per seconde) van de beide systemen eens op een rijtje zetten:

woordgenerators

digitalisering met gedeeltelijke informatiereductie ... ca. 2000 bps

digitalisering met gebruikmaking van alle mogelijke reductiemethoden ... ca. 600-800 bps

foneemgenerators

linear predictive coding ... ca. 1200 bps
volgens systeem van de "voder" ... ca. 70 bps

Dat laat aan duidelijkheid niets te wensen over. Wat de benodigde geheugenruimte voor de spraakkodes betreft, biedt de "voder"-achtige foneemgenerator grote voordelen. Toch zal, zeker als geheugenruimte nog goedkoper gaat worden, ook het woordgenerator-systeem met vaste vokabulaires zich waarschijnlijk ook wel handhaven; vooral bij toepassingen waar een vokabulaire van zo'n 500 woorden (wellicht haalbaar maximum) voldoende is en een zekere steriliteit van het geluid niet storend is, vormt de eenvoudige programmering een ijzersterk punt.

Literatuur:

Pedro the Voder. A machine that talks. Bell lab. 17 (1939)
The voder Electrician, Lond. 123 (1939)

Rabiner, L.R. Computer Synthesis of Speech by Concatenation of Formant-Coded Words. Bell System Technical Journal (1971)

McIlroy, M. Douglas. Synthetic English Speech by Rule. Bell Telephone Laboratories Inc, Murray Hill, NJ
Teja, E. Voice input and output. EDN, November (1979)

Diverse publikaties van de volgende firma's:
Computalker Consultants Box 1951 Sante Monica CA 90406

Telesensory Systems 3408 Hillview Ave Palo Alto CA 94304
Votrax Div of Federal Screw Works 500 Stephenson Hwy Troy MI 48084
National Semiconductor Corp 2900 Semiconductor Dr Santa Clara CA 95051

***** b o e k e s r e k i n g *****
 # " 8080 / 8085 Software Design " #
 # uit 'The Blacksburg Continuing Education Series' #
 # door Christopher A. Titus, Peter R. Rony, #
 # David B. Larsen, Jonathan A. Titus #
 # uitg. Howard W. Sams & Co., Inc. 1978 ISBN 0 672 21541 1 #
 # *****

Er was eens een 4004 van Intel, vanaaruit de 8080 ontstond, die weer model stond voor de 8080A-microprocessor welke als internationale industriestandaard valt aan te merken. Het is een fraai stuk volwasseneducatie wat het team van Blacksburg heeft geschreven over deze microproces-sor. Meer dan twaalf delen over onderwerpen als 8080 inter-facing, op-aap-circuits, timer-toepassingen en experienten met datacommunicatie verschenen onder de vlag 'Blacksburg Continuing Education Series'. Onder andere de Bug-books over 8080A-interfacing en programmering zullen de electro-techni-ci zeker wel eens opgevallen zijn.

Het deel '8080 / 8085 Software Design' geeft nogal wat experimentele opzetten voor I / O-hardware en drukt er ook de software in assembly bij af. Het boek laat je de keus, of je octaal, dan wel hexadecimaal met de 8080 wilt werken.

Evenwel kun je op de D.A.I. zonder extra software niet octaal werken, zodat de keus wel voor de hand zal liggen. Voor de meeste software zal echter een assembler-loader-editor nodig zijn, wil men zich niet gaan bezighouden met het hercoderen van de assembly-instructies in machine-code. Toch geeft het octale stelsel je een zeker inzicht in de organisatie van de 8080.

Besteld dat je een instructie MOVLD wilt veranderen in MOVDL, dan zou je in hexadecimaal 6A in 55 moeten omzetten, terwijl in octaal de code 152 veranderd moet worden in 125.

Deze omkering van de twee rechtse getallen veroorzaakt hetzelfde, als de verwisseling van het D- en het L-regiester in de instructie. Om deze reden prefereren veel programmeurs het octale stelsel bij het programmeren van de 8080.

Na een introductie behandelt het boek de basisinstruc-ties van de 8080, zodat men daar een beetje in thuis raakt. De subroutinebehandling, de registerpaarbehandelingen, en de mathematische routines zijn een paar onderdelen die verderop aan de orde komen.

Uiteraard worden de getalconversies en de ASCII- en BCD-routines behandeld, en alles steeds weer met listings.

Het zevende hoofdstuk ten slotte gaat uitgebreid in op de I / O, welke onder andere de gemultiplixte toetsenbord-interfacing beschrijft, waarna ook memory-mapped I / O met elektronische schema's wordt behandeld.

De Blacksburg-serie kent een royaal aantal zeer goede boeken op hard- en softwaregebied.

De gebruiker zal zelf een waardeoordeel moeten geven, naargelang het deel dat hij uit serie op een bepaald moment nodig heeft voor zijn taak.

We hopen daar in DAINamics nader over te horen.

Inno broekman

```

1 REM PROGRAMMA OM TE LATEN ZIEN DAT HET OOK MOGELYK IS OM
2 REM 16 KLEUREN TE KRYGEN IN EEN 4 KLEUREN MODE
3 REM *****
4 REM *
5 REM * JAAP MOL 18-04-1981
6 REM * VELDWEG 123
7 REM * WESTZAAN
8 REM *
9 REM *****
10 MODE 2A:REM ..... ANDERE MODE ANDERE WAARDEN !!!
20 COLORG 0 10 9 5:REM ... IN KLEUR 10 KOMT DE POKE
25 RAM=#4000:REM ..... OFFSET VOOR 48K
30 P1=#31480.0+RAM:REM .... SCREENBYTE RECHTSONDER
40 P2=#32750.0+RAM:REM .... SCREENBYTE LINKSBOVEN
50 P3=#31502.0+RAM:REM .... EERSTE LINE CONTROL BYTE
55 REM ..... NEEMT STEEDS MET 24 BYTES TOE
60 REM ..... HIERMEE KAN DE ACTERGROND
65 REM ..... OF HET OBJECT ZELF VAN KLEUR
70 REM ..... VERANDERD WORDEN // + MEER
75 REM ..... ACHTERGROND 191 < POKE < 208
80 REM ..... OBJECT 207 < POKE < 224
90 PA=#31584.0+RAM:REM ..... ZOMAAR EEN SCREENBYTE
100 POKE PA,255:REM ..... 255 BINAIR OP 'T SCHERM
110 FOR X=#6.0 TO XMAX STEP 10.0
112 DRAW X,0 X,YMAX 9
114 DRAW X-2,0 X-2,YMAX 10
116 DRAW X-4,0 X-4,YMAX 5
120 NEXT
130 FILL 10,30 40,40 10
140 FILL 20,0 30,40 10
185 GOTO 400
190 WAIT TIME 100
200 REM OBJECT KLEUR WISSELING
210 PLATS=P3+10.0*24.0:REM LINE CONTROL BYTE 10
220 FOR X=208.0 TO 223.0
230 POKE PLATS,X:REM ..... OBJECT RWISSELT VAN KLEUR
240 WAIT TIME 50
250 NEXT
300 REM ACHTERGROND KLEUR WISSELING
310 PLATS=P3+10.0*24.0:REM LINE CONTROL BYTE 10
320 FOR X=192.0 TO 207.0
330 POKE PLATS,X:REM ..... ACHTERGROND WISSELT VAN KLEUR
340 WAIT TIME 50
350 NEXT
400 REM ALLES DOOR ELKAAR
405 L=207.0
410 FOR X=P3 TO P3+50.0*24.0 STEP 24.0
420 POKE X,L
430 L=L-1.0
435 IF L=191.0 THEN L=207.0
440 NEXT
450 L=223.0
460 FOR X=P3 TO P3+50.0*24.0 STEP 24.0
470 POKE X,L
475 WAIT TIME 25
480 L=L-1.0
485 IF L=207.0 THEN L=223.0
490 NEXT
999 DRAW 0,0 XMAX,YMAX 3:REM GEEFT COLOR NOT AVAILABLE

```

The IMP STR Command is faulty (by Thomas Krebs, Switzerland)

Example 1:

```
*NEW
*CLEAR 9
*IMP INT (or IMP FPT)
*10 A$ = "FF"
*UT
>D300 309
0300 21 41 22 00 00 ...
```

Example 2:

```
*NEW
*CLEAR 9
*IMP STR A-C
*10 A = "FF"
*UT
>D300 309
0300 21 41 24 00 00 00 00 ...
```

incorrect!

The IMP STR command generates a faulty symbol table. Using the UT command, the symbol table can be looked at. In the two examples it begins at the address #300. After the command IMP STR A-C (example 2) the symbol table becomes incorrect, because four bytes instead of only two have been reserved for the variable A\$ (abbreviated with A). This failure is only remarkable in special circumstances (with- in the execution of DIM, EDIT, CLEAR ...) but not in LIST. In most cases it occurs at runtime, causing a system crash.

Example 3:

```
*NEW
*IMP STR A-C
*10 A = "FF"
*20 DIM B(0)
*RUN
*RUN
```

After the first RUN in example 3 nothing happens; but try it again! The system crashes.

Conclusions:

- Avoid the use of IMP STR commands.
- You can use IMP STR commands; but bever starting the program (RUN) it must be "squeezed" with the following sequence:


```
*IMP INT (or IMP FPT)
*CLEAR xxxx
*EDIT
(break)(break)
*POKE#135,2
```

Try it with exmple 3.
I recommend to squeeze existing programs.

DON'T SLEEP IN THE SUBWAY BABY.....
DON'T CLEAR IN YOUR SUBROUTINE PROGRAMMER.....

Up to now, we had some strange experience with a few programs. This could happen in two ways:

RUN CRASH !

RUN the programs stopped for some unknown reason. Today we found the cause of this problem:

To gain speed, it is a good habitude to put your frequently used subroutines in front of the program. So some programmers start the program with a CALL to an initialisation routine, because this has to be executed only once.

In this initialisation routine we put MODE 0, COLOR XXXX or MODE X, COLORC XXXX, ? CHR\$(12), DIM(...) and so on.

Very fine, as long as you don't CLEAR in your subroutine. After your CLEAR, the BASIC TEXT + SYMBOL TABLE is moved, depending of the size of your CLEAR. When the RETURN of your subroutine is reached, BASIC supposes to find the return address on stack of 8080. And here we go ... after the CLEAR the information on stack is false, with the unwanted effects as described above !!

Of cause, what exactly happens is depending of the contents of the "false" RETURN address !

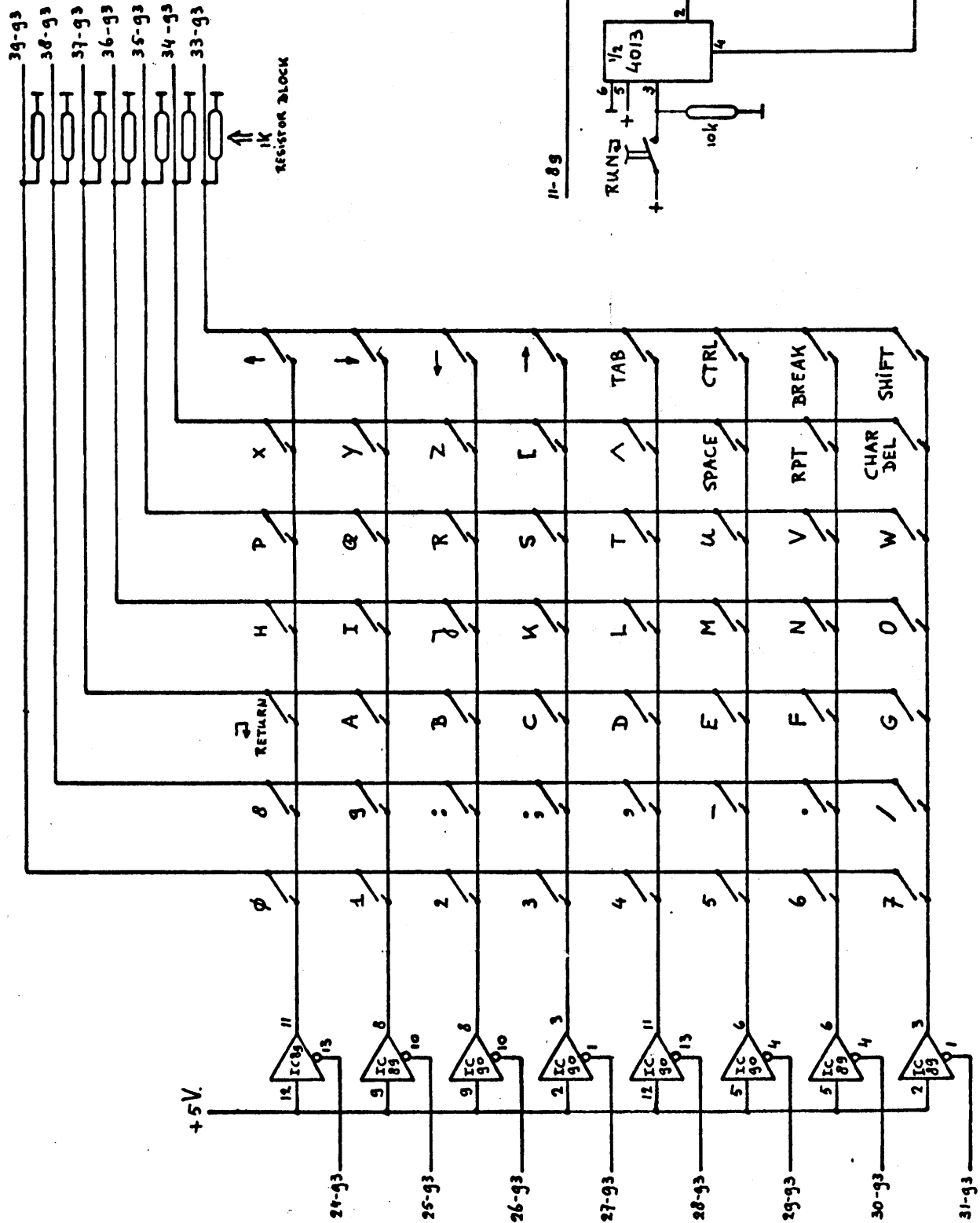


Fig. 1. DAI-PC KEYBOARD

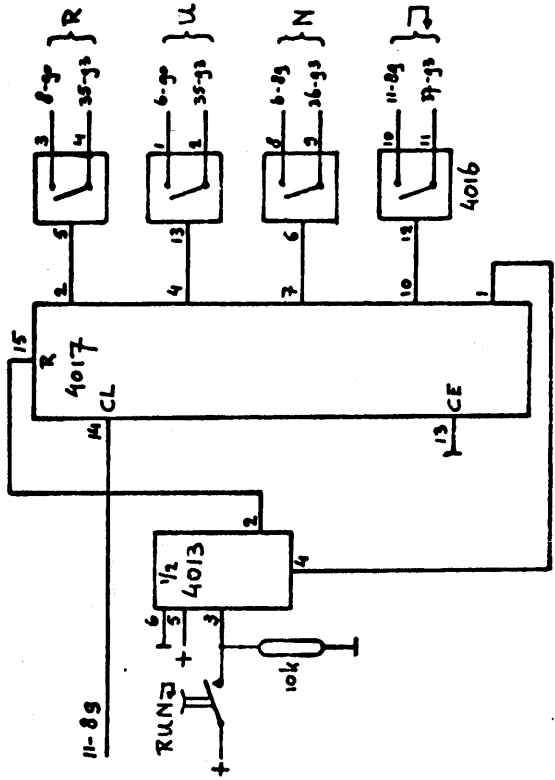


Fig. 2

?CHR\$(12) IN HARDWARE

Dat men bij de DAI-PC ?CHR\$(12) in moet tikken om het TV-scherm te clearen is een tekortkoming in het ontwerp van de DAI. Hardwarematig is dit op te lossen met de volgende schakeling in CMOS, die met één druk op een ekstra toets het TV-scherm netjes schoonveegt. Door uitbreiding van de schakeling kan men elke gewenste functie oproepen.

Eerst een kijkje naar de opbouw van het DAI-PC keyboard (zie fig. 1). U ziet hier dat het keyboard bedraad is volgens de matrix die U terugvindt in het HANDBOOK op pagina 34.

Via de software worden de horizontale lijnen vanuit de 5501 om de beurt actief gemaakt, waarbij de software telkens de verticale lijnen via de 5501 uitleest.

Stel dat de S-toets ingedrukt is. Op het moment dat pin 27 van IC93 (5501) laag wordt, wordt pin 3 van IC90 en dus ook pin 35 van IC93 hoog. (Waar IC89, 90 en 93 op het PC-board zitten ziet men in fig. 4).

Aan de hand van de kode die op dat bepaalde moment aan de pinnen 33 t/m 39 van de 5501 aangeboden wordt bepaalt de software welke toets er ingedrukt is.

Aan de keyboard-schakelaars kan men andere schakelaars parallel aansluiten waardoor de werking van het keyboard niet beïnvloed wordt. Hiervan wordt bij deze schakeling gebruik gemaakt.

In fig. 2 wordt de functie RUN met één schakelaar gerealiseerd.

In de rusttoestand is de flip-flop (4013) gereset, pin 2 is hoog, waardoor de JOHNSON-teller (4017) ook gereset gehouden wordt. Op de clock ingang (pin 14) van de 4017 staan de pulsen van IC89 pin 11, die regelmatig door de software bij het scannen van het keyboard gegenereerd worden. De uitslagen van de analoge schakelaars (4016) zijn parallel met de keyboard-switches R, U, N en \square verbonden.

Drukt men op de schakelaar 'RUN', dan wordt de flip-flop geset, dus pin 15 van de 4017 wordt laag en de 4017 kan gaan tellen.

Bij de eerstvolgende puls op pin 14 wordt pin 2 hoog en de eerste analoge schakelaar wordt gesloten. De software ziet dit via de 5501 en er verschijnt een R op het TV-scherm. Bij de volgende puls op pin 14 wordt pin 2 laag en pin 4 hoog. De volgende analoge schakelaar wordt gesloten zodat er een 'U' op het scherm verschijnt enzovoorts.

Als pin 1 hoog wordt, wordt de 4013 gereset en de 4017 weer geblokkeerd.

Een uitbreiding van deze schakeling ziet men in fig. 3, waarin de functies ?CHR\$(12), RUN, LIST en EDIT zijn gerealiseerd. Voor wat betreft de nummering in het schema een voorbeeldje: pin 3 van IC210 (AND) gaat naar pin 3 van IC215 (OR) en naar pin 5 van IC 204 (ANALOG SWITCH).

- NADELEN:
- 1) Bij het aanzetten van de PC begint de teller soms te lopen en men krijgt dan een SYNTAX ERROR.
 - 2) In verhouding veel hardware nodig (voor de schakeling van fig. 3 al 17 IC's!). Wil men nog meer functies realiseren dan wordt het interessant om een PROM te gebruiken.
 - 3) Het keyboard moet in UPPER CASE mode staan.

De schakeling heb ik via een flat-cable aangesloten. Het beste kan men de draden voorzichtig op de achterzijde van het PC board op de pinnen van IC89 en 90 en op de pinnen van het blauwe weerstandenblokje rechts naast IC90 solderen.


```

5 REM TALK EDITOR Sip/Her mans
10 REM INIT SCREEN
11 POKE #75,32
15 MODE 0: CLEAR 2000: PRINT CHR$(12);
20 INPUT "ADRESSE D'IMPLANTATION (#6000) STARTADDRESS FOR TABLE
C "; AD: START=AD: PRINT CHR$(12);
21 PRINT : PRINT TAB(27); "TALK": PRINT : PRINT
30 PRINT "COMMANDS:"
40 PRINT "0 1 2 ..... FREQUENCE CHANNELS 0 1 2"
50 PRINT "A B C ..... VOLUME CHANNELS 0 1 2"
60 PRINT "N ..... VOLUME NOISE"
70 PRINT "D #sec.....DELAY"
80 PRINT "E ..... END"
90 PRINT "G ..... 60"
110 PRINT "X ..... NEW"
115 PRINT "L ..... LIST"

120 REM INPUT
122 POKE #B7BE, #CC
124 CURSOR 0, 2: PRINT "COMMAND: ?";
125 CURSOR 50, 6: PRINT "#"; HEX$(AD);
130 A=GETC: IF A=0 THEN 130

140 REM DECODE
145 IF A=ASC("X") THEN GOTO 10
150 IF A=ASC("G") THEN POKE AD, #FF: TALK #6000: SOUND OFF : GOTO
C 120
160 IF A=ASC("0") THEN 1000
170 IF A=ASC("1") THEN 1100
180 IF A=ASC("2") THEN 1200
190 IF A=ASC("A") THEN 2000
200 IF A=ASC("B") THEN 2100
210 IF A=ASC("C") THEN 2200
220 IF A=ASC("N") THEN 2300
230 IF A=ASC("E") THEN POKE AD, #FF: GOTO 120
240 IF A=ASC("D") THEN 3000
250 IF A=ASC("L") THEN POKE AD, #FF: GOTO 5000: REM DISPLAY
260 GOTO 120

```

```

2000 REM VOLUME CHANNEL 0
2003 POKE AD, #B: AD=AD+1
2005 CURSOR 0, 4: PRINT "CH 0: ";
2010 CURSOR 6, 4: INPUT "VOLUME: "; VO
2020 IF VO<0 OR VO>#F THEN 2010
2025 GOSUB 3100
2030 POKE AD, VO: AD=AD+1: GOTO 120

2100 REM VOLUME CHANNEL 1
2105 POKE AD, #9: AD=AD+1: CURSOR 0, 4: PRINT "CH 1: "; GOTO 2010

2200 REM VOLUME CHANNEL 2
2205 POKE AD, #A: AD=AD+1: CURSOR 0, 4: PRINT "CH 2: "; GOTO 2010

2300 REM VOLUME NOISE
2305 POKE AD, #B: AD=AD+1: CURSOR 0, 4: PRINT "NOIS: "; GOTO 2010

3000 REM DELAY
3005 POKE AD, #C: AD=AD+1
3010 CURSOR 0, 5: INPUT "DELAY: "; DL
3020 IF DL<0 OR DL>#FFFF THEN 3010
3030 D1=INT(DL/256.0): D2=DL IAND #FF
3031 GOSUB 3100
3040 POKE AD, D1: AD=AD+1: POKE AD, D2: AD=AD+1: GOTO 120
3100 FOR X=0 TO 6: CURSOR 0, X: PRINT SPC(59);: NEXT: RETURN

5000 REM LIST
5010 PRINT CHR$(12);
5030 DI=START
5040 IF CURY<1 THEN IF GETC<>32 THEN 5040
5045 V=PEEK(DI)
5050 IF V<8 THEN GOSUB 6000: REM CHANNELS
5060 IF V>7 AND V<#B THEN GOSUB 7000: REM VOL CHANNELS
5070 IF V=#B THEN GOSUB 8000: REM VOL NOISE
5080 IF V=#C THEN GOSUB 9000: REM DELAY
5090 IF V=#FF THEN GOSUB 10000: REM END
5095 IF V=#FF THEN IF GETC=0 THEN 5095: PRINT CHR$(12); GOTO 21
5100 GOTO 5040

```

```

6000 REM CHANNELS
6010 PRINT TAB(10); "CHANNEL "; V/2; TAB(25); "FREQ :
C "; (2000000)/(PEEK(DI+2)*256+PEEK(DI+1))
6020 DI=DI+3: RETURN

7000 REM VOLUMES CHANNELS
7010 PRINT TAB(10); "VOLUME ~ CHANNEL "; V-8; " :
C "; TAB(30); PEEK(DI+1)
7020 DI=DI+2: RETURN

8000 REM VOL NOISE
8010 PRINT TAB(10); "VOLUME ~ NOISE : "; PEEK(DI+1)
8020 DI=DI+2: RETURN

9000 REM DELAY
9010 PRINT TAB(10); "DELAY : "; PEEK(DI+1)*256+PEEK(DI+2)
9020 DI=DI+3: RETURN

10000 REM END
10010 PRINT TAB(10); "END
C RETURN

```

MX80-DAI trough-serial interface, dipswitch setting

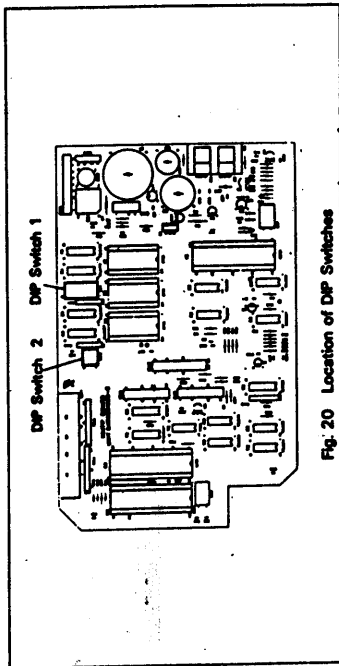


Fig. 20 Location of DIP Switches

DIP switch 2 : 1	ON	DIP switch 1 :	1	ON
2	ON	2	ON	
3	OFF	3	OFF	
4	OFF	4	OFF	
5	OFF	5	OFF	
6	OFF	6	OFF	
7	OFF	7	OFF	
8	ON	8	ON	

DIP switch on serial card:	
1	OFF
2	OFF
3	OFF
4	OFF
5	ON
6	OFF
7	OFF
8	ON

CABLE connections	
DAI	MX-80
1	1
2	3
4	20
7	7

Specifications

Print method Serial impact dot matrix
 Print Rate 80 CPS
 Print Direction Bidirectional
 Number of Pins in Head 9
 Matrix 9x9
 Line Spacing 18", 16", 7/32", plus programmable
 Logical seeking function—
 105 LPM, 20 character line;
 73 LPM, 40 character line;
 46 LPM, 80 character line.
 Throughput at 10 CPI.

PRINTING CHARACTERISTICS
 Character Set Full 96-character ASCII with descenders
 Graphics Characters 64 block characters
 Printing Modes Standard
 Double (advance paper 1/206th and repeat line).
 Emphasized (shift right and double strike).
 Double Emphasized (combination of above).

PRINTING SIZES

Characters per inch	Max. Characters per line
Normal 10	80
Normal Expanded 5	40
Compressed 16.5	132
Compressed Expanded 8.25	66

FORMS HANDLING
 Line Feed Programmable length 1 to 857/2nds
 Form Feed Programmable length to 66 lines
 Horizontal Tab To 24 positions
 Vertical Tab To 64 positions

MEDIA HANDLING
 Paper Feed Adjustable tractor-type pin feed
 Paper Width Range 4" to 10"
 Number of Parts 3
 Paper Path Rear

INTERFACES
 Standard Centronics-style 8-bit Parallel
 RS232C IEEE488
 Buffer Size 1 line

SWITCHES/LIGHTS/DETECTORS
 Indicators Power Light; Printer Ready; Paper Out; On Line
 Switches Power On/Off; On Line; Form Feed; Line Feed
 Detectors Internal buzzer (bell) responds to Paper Out and error conditions with a periodic 3-second tone for 30 seconds.

RELIABILITY
 Print Head Life 50 to 100 x 10⁶ characters
 Expectancy 5 million lines
 MCBF (Excluding Print Head) 5 million lines

INKED RIBBON
 Color Black
 Type Cartridge
 Life Expectancy 3 Million characters

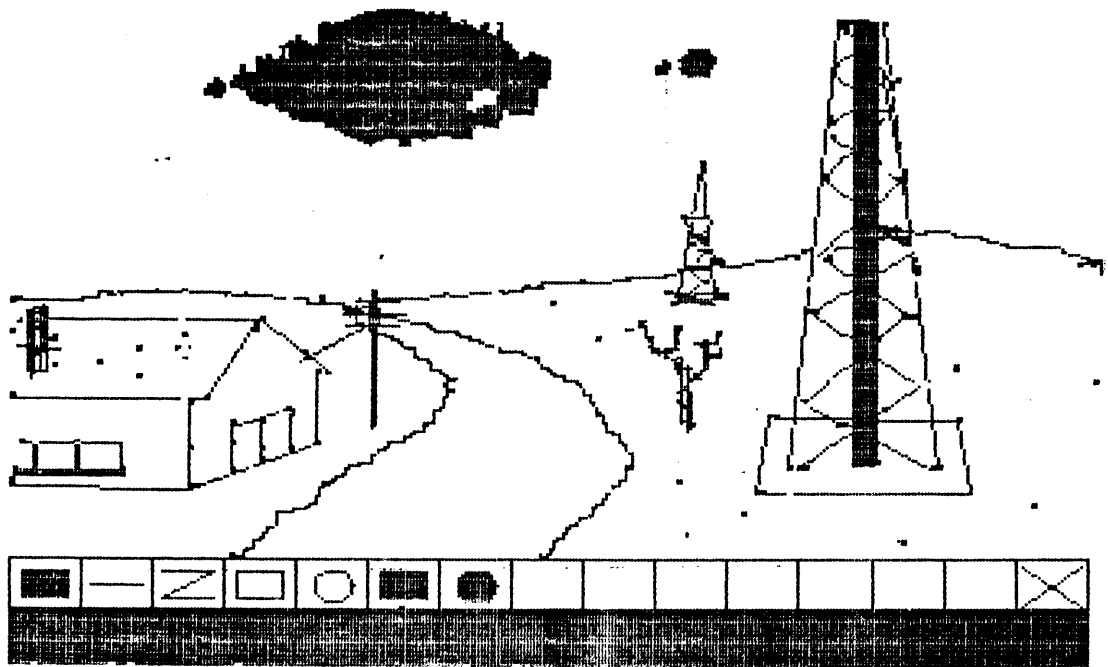
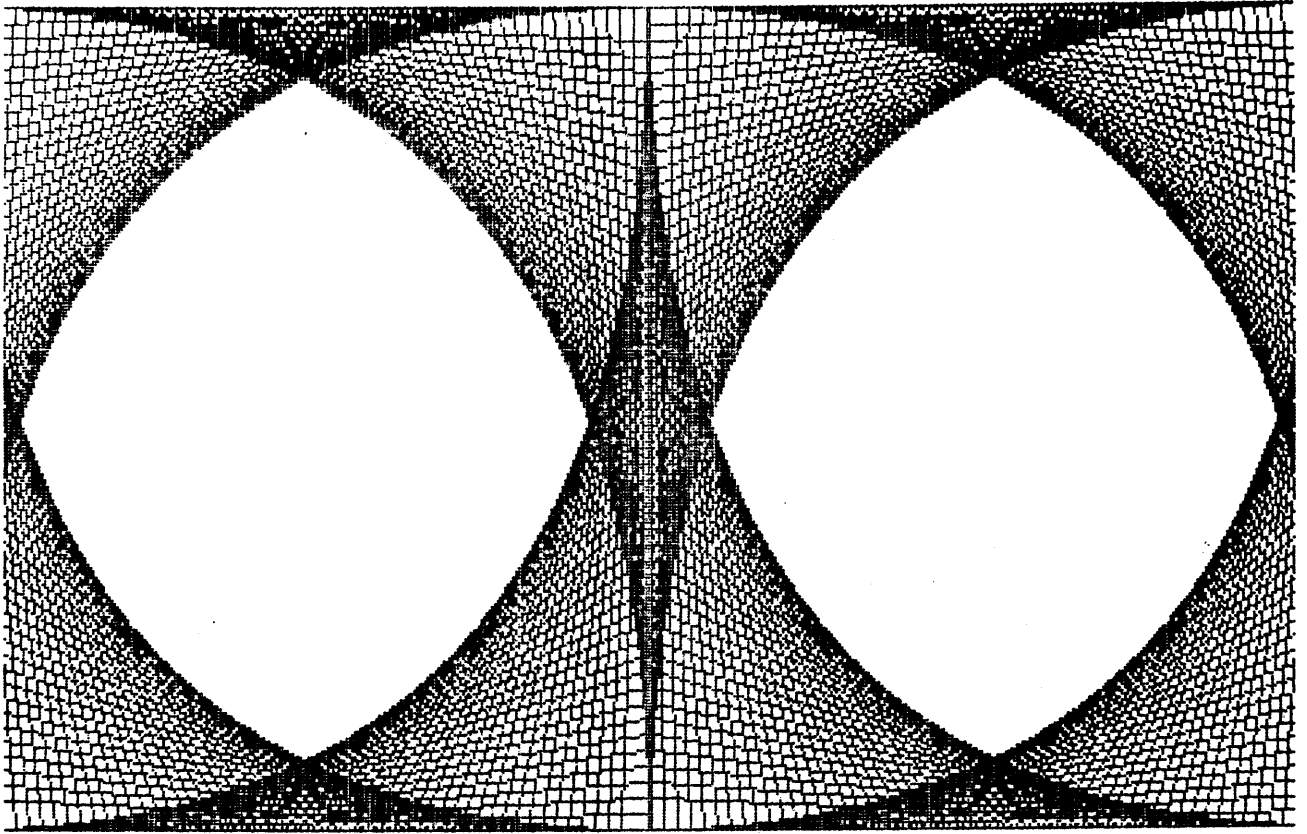
ENVIRONMENTAL CONDITIONS
 Operating Temperature Range 41°F to 95°F
 Operating Humidity 10 to 80% non-condensing

POWER REQUIREMENT
 Voltage 115V, 60Hz or 220-240V, 50 Hz
 Current <1 amp
 Power Consumption 100 VA maximum

SELF TEST MODE
 Depressing Line Feed Switch while turning power ON engages self-test which prints all characters in ROM.

PHYSICAL CHARACTERISTICS
 Height 4.2"
 Width 14.7"
 Depth 12.0"
 Weight 12 lbs.

Specifications subject to change without notice.



```

1 REM CIJFERTABEL - PROGRAMMA VOOR ONDERWIJSGEVENDEN
2 REM C.W.A.VAN DIJK KAMPEN
3 PRINT CHR$(12);
4 COLOR B 0 7 0
10 PRINT :PRINT "
12 CURSOR 13,0:PRINT "
13 REM *****VARIABLEN INVOEREN: ST/MAX%/H/AFR%
15 PRINT :PRINT :PRINT "
20 G!:=GETC:WAIT TIME 5:IF G!:=0 THEN 22
22 IF G!:=18 THEN 20
25 IF G!:=70.0 THEN A$="" fouten":ST:=1.0:PRINT "F":GOTO 30
27 IF G!:=80.0 THEN A$="" punten":ST:=-1.0:PRINT "P":GOTO 30
28 PRINT :PRINT "
30 PRINT :PRINT "
32 PRINT :PRINT "
33 PRINT :PRINT "
37 H:=1.0:IF MAX>99 OR MAX<1 THEN 30
40 PRINT "
42 G!:=GETC:WAIT TIME 5:IF G!:=0 THEN 42
45 IF G!:=18.0 THEN 30
46 IF G!:=53.0 THEN P$="" hele/halve":H:=0.5:PRINT "5":GOTO 50
47 IF G!:=48.0 THEN P$="" hele":PRINT "0":GOTO 50
48 PRINT :PRINT "
49 PRINT "
50 PRINT "
51 PRINT "
52 G!:=GETC:WAIT TIME 5:IF G!:=0 THEN 52
55 IF G!:=18.0 THEN PRINT :GOTO 37
56 IF G!:=48.0 THEN AFR$="" helen":AFR:=1.0:PRINT "0":GOTO 68
57 IF G!:=53.0 THEN AFR$="" halven":AFR:=2.0:PRINT "5":GOTO 68
58 IF G!:=49.0 THEN AFR$="" tienden":AFR:=10.0:PRINT "1":GOTO 68
60 PRINT :PRINT "
62 PRINT "
64 PRINT "
66 GOTO 50
67 REM *****SAMENVATTING VARIABLEN
68 GOSUB 380
70 PRINT CHR$(10); "
71 PRINT CHR$(10); "
80 GOSUB 380
90 CURSOR 13,1:PRINT "
92 G!:=GETC:WAIT TIME 5:IF G!:=0 THEN 92
95 IF G!:=74.0 THEN 100
96 IF G!:=78.0 THEN 3
98 PRINT "
100 REM *****AANTAL KOLOMMEN EN SPATIE
110 KOL:=INT(MAX/20.0/H!):SP:=(48.0-12.0*KOL)/(KOL+2.0)+5.0
120 PRINT CHR$(12)
129 REM *****KOPREGEL
130 PRINT "
: NEXT:PRINT
140 GOSUB 380
150 ST:=H!*ST!
160 E=0.0
170 IF ST!<0.0 THEN E=MAX

```

```

179 REM *****TABEL PRINTEN
180 CD=46044:COL=134
190 REM AD%=#BFEA-(#86*(23-CURY))-(2*CURX+4)
200 FOR RI=0.0 TO 19.0:PRINT "
210 FOR K=0.0 TO KOL:SC:=E+(RI+20.0*K)*ST!:GOSUB 360
220 IF ST!>0.0 THEN CF:=11.0-CF!
230 IF SC!<0.0 OR SC!>MAX THEN 280
240 PRINT SPC(SP-LEN(STR$(SC!))):SC!:SPC(5-LEN(STR$(CF!))):CF!:GOSUB 300:PRINT CHR$(10);
280 NEXT:PRINT :NEXT
285 G!:=GETC
290 G!:=GETC
291 G!:=GETC:IF G!:=0.0 THEN WAIT TIME 5:GOTO 291
295 GOTO 3
297 STOP
299 REM *****ONVOLDOENDES KLEUR SEVEN
300 IF CF!>5.5 THEN 350
310 AD=CD+COL*CURY-2*CURX
330 FOR M=0.0 TO 4.0 STEP 2.0:POKE AD+M,#FF:NEXT
350 RETURN
359 REM *****CIJFER BEREKENEN UIT SCORE
360 CF!:=INT(AFR*SC!*9.0/MAX+0.5)/AFR+1.0
370 RETURN
379 REM *****HOR.LIJN
380 FOR N=1.0 TO 60.0:PRINT CHR$(11):NEXT:PRINT :RETURN

```

STOP PRESS +++++ LAATSTE BERICHTEN

levering DCR's in week van 21 september.
 DATABUS van 11 sept : artikel over
 communicatiestandaard voor personal computers.
 what's wrong with the cover of this issue ?
 vermeedelijke prijs van de bandjes voor DCR:
 10 Gld + BTW.

Er zijn nog een paar plaatsen vrij in onze
 autobus naar UTRECHT op 28 nov.
 Gelieve contributie voor 82 nog niet te storten:
 info over reorganisatie in NEWSLETTER 8.

+++++

```

0 zwart      alle adressen in HEXvorm!
1 blauw
2 d.rood    29B-29C   start heap           131,0   output scrn+
3 rood      29D-29E   size heap           RS232
4 paars     29F-2A0   start text buffer   131,1   screen only
5 groen     2A1-2A2   start symbol table  131,2   edit buffer
6 d.bruin   2A3-2A4   end of symbol table 135,2   read from
7 l.bruin   2A5-2A6   bottom screen ram   edit buffer
8 grijs
9 blauw
10 oranje
11 rose     75         cursor symbol       MODE    XMAX    YMAX
12 l.blauw  74         cursor mode         1/2     71      64
13 l.groen  72-73     cursor position     3/4     159    129
14 geel
15 wit      40,28     cass motor 1 ON
           40,18     cass motor 2 ON
           40,30     1 and 2 OFF
           MERGE
           °CLEAR XXX
           °LOAD"A"
           °EDIT BREAK/BREAK
           °LOAD"B"
           °POKE 135,2

COLORG R1 R2 R3 R4
        20 21 22 23
16 :R2*R1 R4*R3
17 :R1*R2 R3*R4      32K 7XXX
18 :R3*R1 R4*R2      12K 2XXX
19 :R1*R3 R2*R4      8K 1XXX

           IMP INT *** IMP FPT
           °IMP FPT
           °CLEAR XXXX
           °EDIT BREAK/BREAK
           °IMP INT
           °POKE 135,2

LIJN   CTRL  COLOR      LIJN  CTRL  COLOR
23     BFEF  BFEE      11    B9A7  B9A6
22     BF69  BF68      10    B921  B920
21     BEE3  BEE2      9     B89B  B89A
20     BE5D  BE5C      8     B815  B814
19     BDD7  BDD6      7     B78F  B78E
18     BD51  BD50      6     B709  B708
17     BCCB  BCCA      5     B683  B682
16     BC45  BC44      4     B5FD  B5FC
15     BBBF  BBBE      3     B577  B576
14     BB39  BB38      2     B4F1  B4F0
13     BAB3  BAB2      1     B46B  B46A
12     BA2D  BA2C      0     B3E5  B3E4

           CTRL&COLOR BYTES IN A-MODE
           MODE   CTRL   COLOR  LIJN
           1A/2A  BAE7   BAE6   3
           BA61   BA60   2
           B9DB   B9DA   1
           B955   B954   0
           3A/4A  ACD3   ACD2   3
           AC4D   AC4C   2
           ABC7   ABC6   1
           AB41   AB40   0
           5A/6A  7557   7556   3
           74D1   74D0   2
           744B   744A   1
           73C5   73C4   0

FD00  b2 page signal      FF00  ser.inp.buff
      b3 serial out rdy  FF01  b0-6 keyb.inp.
      b4 right paddle    b7 in7 DCE
      b5 left paddle     FF02  Interr.reg.
      b6 random data     FF03  b1 frame error
      b7 cass. input     b2 overrun error
FD01  Trigger paddle     b3 rec.buf.loaded
FD04  0-3 volume ch.1(0) b4 trans.buf.empty
      4-7 volume ch.2(1)
FD05  0-3 volume ch.3(2) FF04  COMMAND REGISTER
      4-7 volume noise  FF05  BAUD RATE REGISTER
FD06  b0 cass.out        FF06  ser.out buf.
      b1/2 paddle select FF07  keyb.output
      b3 paddle enable   FF08  interr.mask reg.
      b4 cass motor 1
      b5 cass motor 2
      b6/7 ROM BANK SWITCH
           TEST EVENT
           PEEK(éFD00) IAND 32
           PEEK(éFD00) IAND 16
           PEEK(éFD00) IAND 48
           CH 0 FC00/FC01
           CH 1 FC02/FC03
           CH 2 FC04/FC05
           STATUS FC06/FC07

```