```
106   PRINT "Voulez-vous les instructions (O/N)?";
107   RE=GETC:IF RE=79.0 THEN GOTO 10000
108   IF RE<>78.0 THEN 107
109   PRINT CHR$(12)
110   POKE #FF05,255:DIM R$(8.0),F(9.0,5.0),A(5.0,5.0),P$(5.0,5.0),N(5.0)
120   FOR I%=1 TO 8:READ R$(I%):NEXT
130   DATA score excellent,tres bon score,bon score,score au dessus de la m
oyenne
140   DATA score moyen,score en dessous de la moyenne,score passable,mauvai
s score
150   REM
199   RETURN
200   REM
210   FOR I%=1 TO 5:FOR J%=1 TO 5:A(I%,J%)=-1.0:NEXT J%:NEXT I%
220   FOR I%=0 TO 9:FOR J%=1.0 TO 5.0:F(I%,J%)=0.0:NEXT J%:NEXT I%
230   A(1.0,1.0)=INT(RND(10.0)):A(2.0,1.0)=INT(RND(10.0)):A(3.0,1.0)=INT(RN
D(9.0)+1.0)
240   A(1.0,2.0)=INT(RND(10.0)):A(2.0,2.0)=INT(RND(9.0)+1.0)
250   N(1.0)=100.0*A(3.0,1.0)+10.0*A(2.0,1.0)+A(1.0,1.0):N(2.0)=10.0*A(2.0,
2.0)+A(1.0,2.0)
260   N(3.0)=A(1.0,2.0)*N(1.0):N(4.0)=A(2.0,2.0)*N(1.0)*10.0:N(5.0)=N(1.0)*
N(2.0)
270   M=10.0:FOR I%=1 TO 5
280   FOR J%=3 TO 5:N1%=INT(N(J%)/M+1E-2):A(I%,J%)=INT(N(J%)-N1%*M+1E-2):N(
J%)=N1%:NEXT J%
290   NEXT I%:A(1.0,4.0)=-1.0:A(5.0,3.0)=-1.0:N1%=0.0:N2%=0:N5%=0.0
292   FOR I%=1 TO 5:FOR J%=1 TO 5:IF A(I%,J%)=(-1.0) THEN P$(I%,J%)="  "
293   IF A(I%,J%)<>(-1.0) THEN P$(I%,J%)=" *"
294   NEXT J%:NEXT I%
299   RETURN
300   GOSUB 11000
305   POKE #BCCA,#D0:POKE #BFEE,#CB
306   POKE #BD50,#CC:POKE #B920,#CB:POKE #B89A,#D2:POKE #B708,#CC:POKE #B68
2,#D1:POKE #B576,#D3
310   GOSUB 1000
320   IF N5%=18.0 THEN 340
330   GOSUB 2000:GOTO 310
340   G=G+1.0:T=T+N1%:V=T/G
350   FOR I=1.0 TO 8.0:CURSOR 0,I:PRINT "
                          ":NEXT:CURSOR 0,9:G%=G
355   POKE #FF05,4:PRINT "Nombre moyen de coups rates apres";G%;" partie";:
IF G>1.0 THEN PRINT "s";
360   V%=V:PRINT " :";V%;:PRINT ".":Q=INT(V/2.0):IF Q>=1.0 THEN 367
365   Q=1.0:IF Q>8.0 THEN Q=8.0:GOTO 370
367   IF Q>8.0 THEN Q=8.0
370   PRINT "Voila un ";R$(Q);"."
899   RETURN
1000  REM
1010  CURSOR 0,17
1020  FOR I%=1 TO 5
1025  CURSOR 5,CURY
1030  FOR J%=5 TO 1 STEP -1:PRINT P$(J%,I%);:NEXT J%
1040  PRINT :IF I%<>2.0 AND I%<>4.0 THEN 1050:PRINT TAB(5);" ";:FOR I=1.0 T
O 11.0:PRINT CHR$(95);:NEXT:PRINT
1050  NEXT I%
1060  PRINT :PRINT "Nombre de coups joues :";N2%:PRINT "Nombre de coups man
ques :";N1%
1070  PRINT "Nombre de chiffres trouves :";N5%
1999  RETURN
```

```
2000  N2%=N2%+1.0
2010  CURSOR 0,5:PRINT "ESSAI No.";N2%
2020  PRINT :PRINT "Quel est votre chiffre?";
2021  RE=GETC:IF RE<48.0 OR RE>57.0 THEN 2021
2022  D=RE-48.0:PRINT CHR$(RE)
2023  PRINT "Quelle est la colonne?";
2024  RE=GETC:IF RE<49.0 OR RE>53.0 THEN 2024
2025  C=RE-48.0:PRINT CHR$(RE)
2026  REM
2030  REM
2040  IF F(D,C)=1.0 THEN 2010
2050  F(D,C)=1.0
2060  N9%=N5%
2070  FOR I%=1 TO 5:IF A(C,I%)<>D THEN 2080:P$(C,I%)=" "+MID$(STR$(D),1,1):
N5%=N5%+1.0
2080  NEXT:IF N9%=N5% THEN N1%=N1%+1.0
2090  GOSUB 3000
2999  RETURN
3000  CURSOR 25,17
3010  PRINT "     ENTREES PRECEDENTES :":PRINT
3020  FOR I%=1 TO 5:CURSOR 25,CURY:PRINT "Colonne";I%;" :";:FOR J%=0 TO 9:I
F F(J%,I%)=1.0 THEN PRINT J%;
3030  NEXT:PRINT :NEXT:PRINT
3999  RETURN
9000  INPUT R$:R$=LEFT$(R$,1):IF R$<>"O" AND R$<>"N" THEN PRINT "O OU N";:G
OTO 9000
9010  RETURN
10000 PRINT CHR$(12):POKE #BDD6,#D5:POKE #BAB2,#D2:POKE #B78E,#D0:POKE #B4F
0,#D3:POKE #FF05,3:PRINT "INSTRUCTIONS :"
10005 PRINT "==============":PRINT
10010 PRINT "     Multipuzzle est un jeu d'adresse, de chance et de"
10020 PRINT "deduction dont le but est de retrouver les chiffres d'une"
10030 PRINT "multiplication (d'un nombre de 3 chiffres par un nombre de 2"
10040 PRINT "chiffres), calculee secretement et au hasard par"
10050 PRINT "l'ordinateur.":PRINT
10060 PRINT "     Une fois lance, le programme affiche sur l'ecran le"
10070 PRINT "squelette bien connu d'une multiplication, dont la"
10080 PRINT "particularite est que tous ses chiffres sont remplaces par"
10090 PRINT "des asterisques (*). Vous demandez alors, ";CHR$(34);"Y a-t-il
 tel"
10095 PRINT "chiffre dans telle colonne?";CHR$(34);"."
10100 PRINT :PRINT "     Si le chiffre existe, il vient remplacer le ou les
"
10110 PRINT "asterisques correspondants. Dans le cas contraire, le"
10120 PRINT "programme comptabilise imperturbablement le nombre de coups"
10130 PRINT "rates.":PRINT :PRINT "BONNE CHANCE!":POKE #FF05,255:WAIT TIME
200:GOTO 110
11000 PRINT CHR$(12):POKE #BDD6,#D0:POKE #BCCA,#D3:PRINT TAB(21);"== MULTIP
UZZLE ==":PRINT TAB(21);"================":PRINT
11010 PRINT "Christian POELS - 27/5/1981":PRINT :RETURN
```

FGT-DISK-PEEK-POKE

Programma 1

```
100   REM : FGT IN MATRIX
110   CLEAR 20000
120   DIM A%(#90,#F)
130   FOR I%=0 TO #90
140   FOR J%=0 TO #F
150   S%=#1C00+I%*16+J%
160   A%(I%,J%)=PEEK(S%)
170   NEXT:NEXT
180   N$="FGT-MATH:0"
190   SAVEA A% N$
```

Programma 2

```
1000  REM : FGT INLADEN ALS MATRIX
1010  CLEAR 20000
1020  DIM A%(#90,#F)
1030  N$="FGT-MATH:0"
1040  LOADA A% N$
1050  FOR I%=0 TO #90
1060  FOR J%=0 TO #F
1070  S%=#1C00+I%*16+J%
1080  POKE S%,A%(I%,J%)
1090  NEXT
1100  NEXT
1110  CLEAR 1000:STOP
```

Programma 3

```
10    REM : POINTERS
20    POKE #29B,#0:POKE #29C,#25
30    N$="FGT-LOAD:0"
40    CALLM #300,N$
```

SPEED UP THE BITMANIPULATION PROGRAM FROM DAInamic 12

```
2050 B=P IAND (2^(X+1)) should become :
2050 B=P IAND((2 SHL X)-1)
```

- Henk Stokhorst

'n FGT - compatibel met de Diskette-driver

In het DAInamic-nummer van 4 april 1981 - blz.67 tot 77 - vinden we 'n zeer interessant artikel over het Fast Graf Text (FGT) programma. De daar beschreven FGT-routine is echter incompatibel met 'n diskette-driver. Waarom?
Dit programma wordt gepookt op #300 tot +/- #900.
Om niet gestoord te worden door het inladen van 'n Basic-programma gaat men de adressen voor Start of Heap wijzigen, zodat Heap en Basic-programma na de FGT komt . Het FGT-programma wordt dan gestart met 'n CALLM#300,A$.(Wel bekend aan DOS-gebruikers, maar dan voor het inladen van 'n nieuw programma!).
Wanneer we echter over 'n diskette-driver beschikken, wordt de DOS (Diskette Operating System) automatisch ingeladen daar waar normaal de Heap begint, en worden de adressen aangepast.
Vergelijken we nu de begintoestand:

Zonder diskette		Met diskette	
	*UT		
	>D29B 2A6 (R)		
029B EC 02 00 01 EC		029B CC 19 00 01 CC	
02A0 03 ED 03 EE 03 50 B3		02A0 1A CD 1A CE 1A 50 B3	

Wat we interpreteren als:

029B 02EC	Start of Heap	19CC	
029D 0100	Lenght of Heap	0100	(256 bytes)
029F 03EC	Start of Text	1ACC	(natuurlijk: tel op!)
02A1 03ED	Start of Symbol	1ACD	(geen programma!)
02A3 03EE	End of Symbol	1ACE	(geen variabelen!)
02A5 B350	Bottom of Screen	B350	

De FGT mag dus niet langer ingeladen worden op #300-#900 , wat dan zou onze DOS vernietigd worden, wat fataal zou zijn.
De FGT moet dus van nieuwe adressen voorzien worden , aangepast aan de plaats van de DOS . Dat werk is nu gebeurd, en er zijn aangepaste FGT-routines te verkrijgen.
Deze worden gepookt vanaf 1C00 tot'n adres afhankelijk van de gebruikte FGT . Als richtsnoer zullen we de Math-FGT gebruiken , die eindigt rond 2430. Het begin, 1C00, is ruim veilig achter de DOS, aangezien de computer zelf de Start of Heap normaal voorziet op 19CC:'n reserve van 'n halve K-byte . Hopelijk zal deze veiligheidsmarge volstaan voor de "nieuwe DOS".
Hoe gaan we nu praktisch te werk?
Bij het aanzetten wordt automatisch de DOS ingeladen.
We plaatsen onze diskette met onze FGT in Driver 0.
Desnoods gaan we met 'n DIR0 zien naar de juiste naam van de FGT.

```
*UT (R)
(>F1C00 3000 0 (R): veiligheidshalve sporen van 'n vroegere  programma
                    wissen: facultatief.)
>R FGT:0 (R)      : inlezen van het FGT-machinetaal programma.
>D1C00 2500 (R)   : om het einde van de FGT vast te stellen.
                    desnoods het ladr en hadr in D aanpassen.
                    (begin en eindadres van het display.)
                    Men noteert  het einde  van de FGT , b.v. 2439, en
                    rond af naar omhoog: b.v. 2500.
>S29B (Space Bar) CC- 00 (SP) 19- 25 (R) :  CC- en 19-  worden door de
                    computer zelf gemeld.
                    Daarmee is de Start of Heap  (op adres 29B en 29C)
                    ingesteld van 19CC naar 2500.
>B                : om terug naar Basic te gaan.
```

Door 'n NEW of 'n CLEAR 4 kan men de andere pointers aanpassen.
Natuurlijk moet uw Basic - programma voorzien zijn van 'n subroutine voor het aanroepen van de FGT . De instrukties daarvoor kunt U overnemen uit het meestal bijgeleverde demo-programma van uw FGT.

Automatisch inladen van FGT, en instellen van de pointers.

Toch blijft dit nog 'n omslachtig werkje , vooral als het dient uitgevoerd gedurende, of zelfs bij de aanvang van 'n les.
Er bestaat echter 'n methode om dit volledig automatisch te laten gebeuren. Daartoe gebruiken we drie kleine programma's,die elders afgedrukt staan.
Programma 1: hoeft maar eens gebruikt, en mag daarna vernietigd worden.
========== We laden onze FGT in als boven , en passen daarna de pointers aan (want we gaan 'n Basic-programma inladen of intypen).
Dan laden we (of typen) het eerste programma in, en doen 'n RUN.
De FGT wordt nu in 'n matrix (array) geplaatst. De dimensie (#90,#F) is ruim voor onze Math-FGT, en kan voor andere aangepast worden(desnoods) (vergroot), dit om plaatsruimte op de schijf te besparen.(73 files van) (de 580 beschikbare!).Vergeet niet bij wijziging dezelfde wijziging aan te brengen in programma 2. De matrix wordt door het programma ook automatisch weggeladen. Door 'n COPY FGT-MATH.INT:0 FGT-MATH.INT:1 kan de matrix op andere schijven gereproduceerd worden , waar programma's op staan die er gebruik van gaan maken.
Programma 2: dient om de matrix terug in te laden , en weg te poken.
========== CLEAR 20000 is nodig om de matrix in ontvangst te nemen.
Het programma is kort genoeg om toe te laten de matrix in het programma zelf op te nemen (wat'n bekend nadeel is van matrix-inlezen.)
Maar: de pointers moeten vooraf aangepast worden!
Immers: de matrix wordt weggeladen in de Heap . Door CLEAR 20000 is de Size of Heap 4E20 , en gaat dan van 19CC tot 67EC . Wanneer wij op 1C00 beginnen te poken , vernietigen we zelf onze gegevens , wat blokkering veroorzaakt.(invalid number in line 1080).
'n Aanpassen van de pointers in het begin van het programma zou al even erg zijn. Dus schakelen we programma 3 in.
(Is dat nu eenvoudiger? - Nog even geduld!).
Programma 3: Dit uiterst kort programma doet niets anders dan de Start
========== of Heap bepalen. De uitvoer van het programmaatje dat zich bevindt op 'n slechts klein aantal adressen na 19CC,wordt daardoor niet beinvloed. Met 'n CALLM #300, N$ wordt dan programma 2 ingeladen, op 'n veilige plaats(want de andere pointers worden door de CALLM aangepast).
De uitvoer van programma 2 vergt wel even tijd (matrix inladen) , en de computer verwittigt dat hij klaar is met 'n STOPPED IN LINE 1120.De DAI is nu klaar om gelijk welk programma in te laden , met gebruik van onze FGT. De CLEAR 2000 op het einde van programma 2 is nodig om de Heap,die voldoende groot moest zijn om de matrix te kunnen bevatten,terug te reduceren, en zo het adres van Start of Text aan te passen . In onze grafische programma's gebruiken wij vaak MODE 5/6, wat de Bottom of Screen naar voren brengt, zodat botsing met het BASIC-programma(out of memory) niet uitgesloten is.
Eindelijk eenvoudiger: aan ons programma 3 geven we de naam "$USER". Daartoe hebben we door 'n DELETE het DAI-$USER gewist (of liefst bij het vormen van onze schijf niet overgenomen). Bij het inschakelen van de computer wordt programma 3 automatisch ingeladen en daarna door de CALLM #300,N$ de FGT. Tussen inschakelen en bedrijfsklaarheid verloopt minder dan een minuut.(1 op 10 keer is'n Hart-Reset) (vereist: CALLM#300 hapert soms.)
Allereenvoudigst: Voor de les wijzigt men Programma 2 als volgt:

```
1110 CLEAR 1000
1120 N$="Programmatitel": CALLM #300, N$
```

Met 'n SAVE wordt dit gewijzigd programma terug (onder dezelfde naam:) (LOAD-FGT) weggeladen. Bij aanzetten van de computer in de les wordt nu niet alleen de FGT, maar ook uw lesprogramma ingeladen en gestart.

Veel succes!
J.Gesp - Asse.

Een eenvoudige goedkope en veelzijdige EPROM-Programmer

De DAI personal computer heeft standaard vele uitgebreide I/O mogelijkheden. Hiermee zijn vele applicaties te realiseren. Dit artikel behandelt een van de toepassingen.

Wat is een EPROM?

Een Erasable Programmable Read Only Memory is een geheugenelement dat, zoals de naam al zegt, alleen door de computer kan worden uitgelezen. Het kenmerkende van Read Only Memories is dat ze hun inhoud bij spanningsuitval niet verliezen. Ze zijn dus bij uitstek geschikt om in microprocessor-systemen toe te passen die direkt bij inschakelen moeten werken.
Het vervelende van ROM's is dat de inhoud al in de fabriek moet worden aangebracht, dit is alleen aantrekkelijk voor grote aantallen.
Een EPROM biedt de mogelijkheid om herhaald, door de gebruiker, geprogrammeerd te worden. Na programmeren kan de inhoud nl. met Ultra-violet licht gewist worden. Een ideale bouwsteen dus voor gebruik in de ontwikkeling, in kleine series of door de hobbyist.

Het programmeren van een EPROM

De huidige meest gangbare EPROM's zijn opgebouwd als een matrix van 1024x8, 2048x8 of 4096x8 bits. Respectievelijk de 1K, 2K en 4K EPROM. Deze kunnen heel gemakkelijk in een systeem met een woordbreedte van 8 bits worden opgenomen.
In fig. 1 zijn de EPROM's met typenummer en aansluitgegevens opgenomen.
De diverse aansluitingen (fig. 1a):

Ao..An	:	adreslijnen
Oo..O7	:	data- of outputlijnen
$\overline{CE}$	:	chip enable (active low)
$\overline{OE}$	:	output enable (active low)
Gnd,V_{cc},V_{pp}	:	spanningsaansluitingen

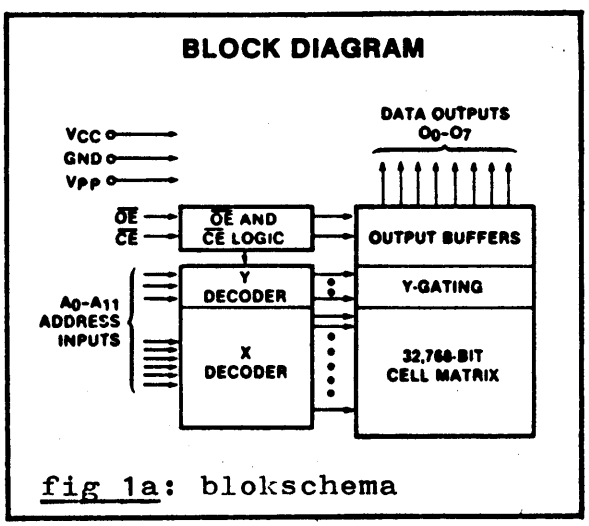
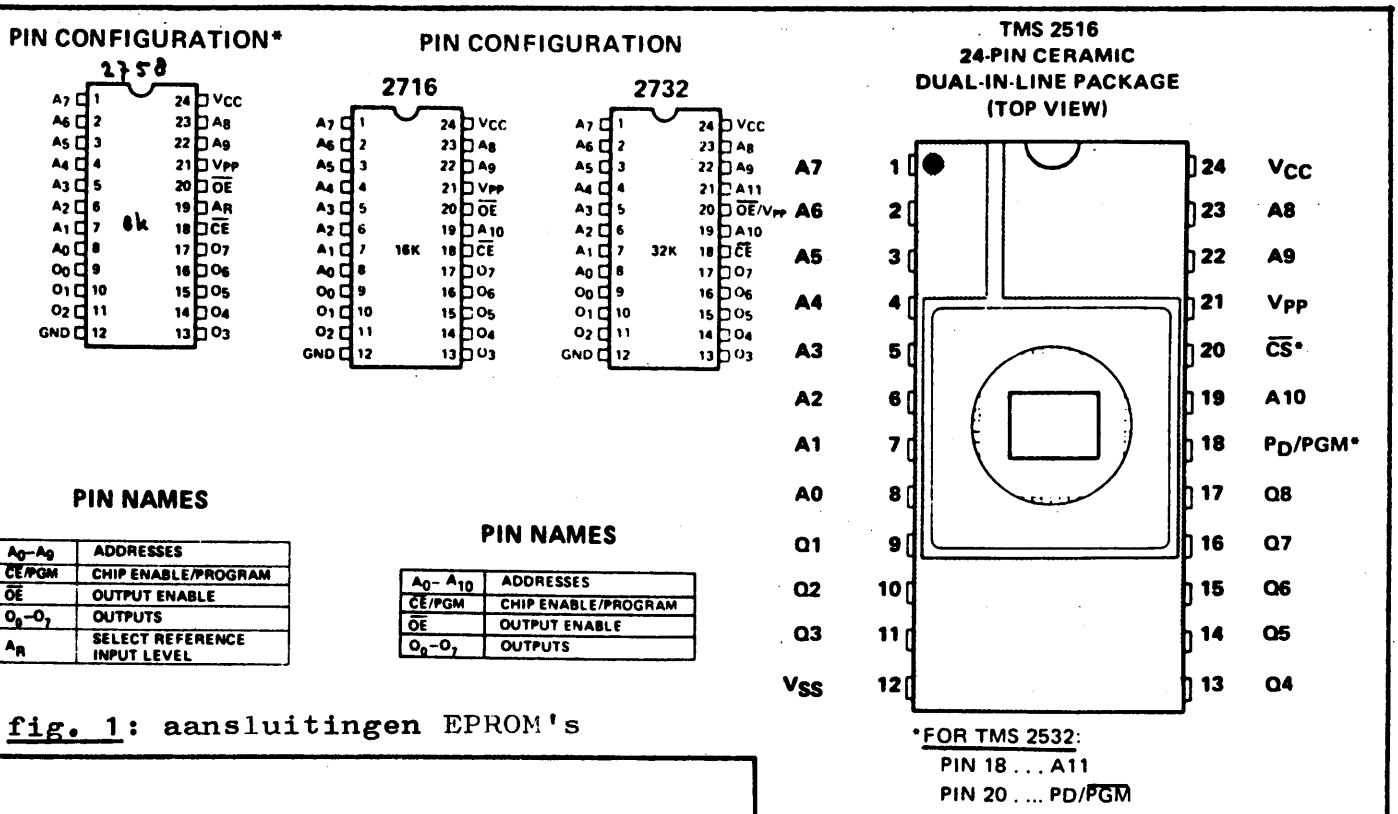


fig 1a: blokschema

Onder normale omstandigheden is de EPROM opgenomen in een µP-systeem. Biedt de microprocessor dan een adres, chipselect en leessignaal aan, dan wordt een geheugenelement geselecteerd en via de uitgangen wordt de waarde naar de µP geschreven. Deze data kan bijvoorbeeld de volgende instructie voor het systeem zijn.
De EPROM moet echter wel geprogrammeerd worden d.w.z. de inhoud moet aangebracht worden.
Uitgangspunt hierbij is een gewist ic, hierin zijn alle bits "1". Door het selectief "0" maken van een aantal bits wordt de EPROM geprogrammeerd. Dit "0" maken gebeurt door op de V_{pp} aansluiting de zogenaamde programmeerspanning van +25V aan te bieden. Vervolgens wordt een adres en datawoord aangeboden, door dan een 50 msec. TTL-puls op de PROG-ingang te zetten wordt de EPROM geprogrammeerd. De data wordt als het ware in de geselecteerde geheugenlocatie "gebrand".
Door deze handeling herhaald uit te voeren kan de gehele EPROM geprogrammeerd worden. Meestal zal men dit doen door iedere cyclus het adres met 1 op te hogen en een nieuw datawoord aan te bieden.



PIN NAMES

A_0–A_9	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS
A_R	SELECT REFERENCE INPUT LEVEL

PIN NAMES

A_0– A_{10}	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS

PIN NOMENCLATURE

A(N)	Address inputs
$\overline{CS}$	Chip Select
PD/PGM, PD/$\overline{PGM}$	Power Down/Program
Q(N)	Input/Output
V_{CC}	+5 V Power Supply
V_{PP}	+25 V Power Supply
V_{SS}	0 V Ground

fig. 1: aansluitingen EPROM's

In tabel 1 zijn de preciese gegevens betreffende het gebruik van de EPROM's opgenomen.
Doordat het programmeren van iedere plaats 50 msec. in beslag neemt, duurt het volledig programmeren van de 2758 1000x0,05=50 sec.
Voor de 2716 en 2516 is dit 100 sec., voor de 2732 en 2532 200 sec.
Na geprogrammeerd te zijn kan de EPROM gewist worden. Hiervoor zijn professionele apparaten, deze zijn echter duur. Het wissen gaat ook goed met een speciale UV-lamp van Philips, TUV 6W-E, deze kost ongeveer 50 gulden. De EPROM's kunnen hier met een elastiekje op bevestigd worden, na 15 à 20 min. zijn ze dan "schoon".

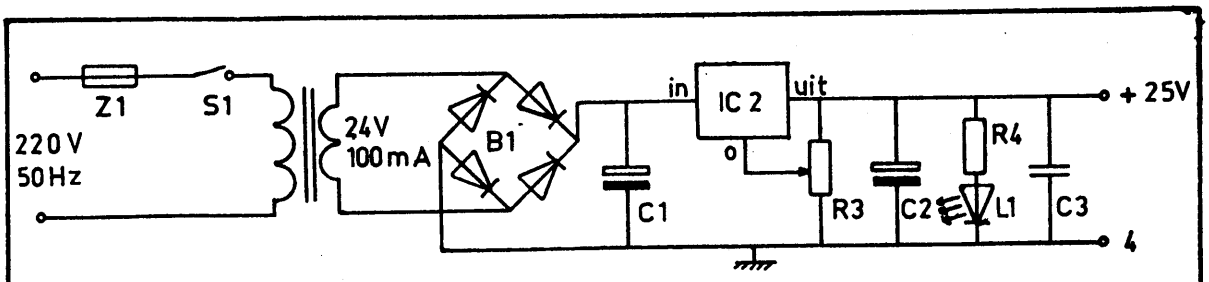
Opbouw Programmer

We hebben nu gezien dat, willen we een EPROM programmeren, we een adres, datawoord plus de nodige controlesignalen aan het ic moeten aanbieden. Deze signalen kunnen geheel door de DAI gegenereerd en via de DCE-bus naar de programmer gestuurd worden. Verder is het nog nodig om de EPROM van spanning te voorzien. De voedingsspanning van +5V is beschikbaar van de DAI, de +25V programmeerspanning moet echter opgewekt worden.

De Hardware

De hardware van de programmer komt neer op de verbinding met de DAI en het maken van de +25V spanning (fig. 2 en fig. 3). De programmeerspanning wordt gemaakt door een 24V wisselspanning gelijk te richten en af te vlakken. Daarna wordt de spanning gestabiliseerd met een 7812 (+12V) spanningsstabilisator die op het juiste uitgangsniveau wordt gebracht door een instelpotmeter in de nulleiding.

Het beste kan hiervoor een tiental uitvoering worden genomen. De potmeter wordt zodanig ingesteld dat het uitgangsniveau precies +25V bedraagt.



nul verbinden met Gnd. DAI pc

fig. 3: programmeerspanning

In fig. 2 is opgenomen welke draden van de DCE-bus moeten worden doorverbonden met de aansluitpennen van de EPROM. In tabel 2 is daarnaast een volledig overzicht van het gebruik van de DCE-bus te vinden.
Verder zijn in het schema 2 relais te zien. Deze worden via de DCE-bus door de DAI bestuurd. De 2 lijnen zijn met een transistor gebufferd. In het ontwerp is gekozen voor 12V relais die vanuit de DAI gevoed worden. Dit kan zonder problemen omdat de DAI een 12V 1A voeding heeft en hier slechts ongeveer een 0.5A van gebruikt. Het is echter ook mogelijk om 24V relais te gebruiken die dan door de programmeerspanning gevoed worden. Het is dan wel nodig een zwaardere trafo te nemen die in de grotere stroombehoefte kan voorzien.
De diodes die over de relais zijn geschakeld dienen om de inductiespanning, die bij uitschakelen ontstaat, op te vangen. Deze spanning kan nl. de transistors beschadigen.
Het ene relais verzorgt de keuze tussen de 2732 en de andere EPROM's. Dit omdat de 2732 iets andere aansluitingen heeft. In totaal moeten 3 aansluitingen verwisseld worden. Bij de meeste programmers gebeurt dit met een handbediende schakelaar waarbij vergissen nogal eens voorkomt.
Het tweede relais zorgt ervoor dat de programmeerspanning wordt aangesloten bij het programmeren van de ic's.

Het geheel kan in een kastje gemonteerd worden. Als voetje voor de EPROM's is een gewoon ic-voetje bruikbaar mits men over wat handigheid beschikt bij het aanbrengen en eruit halen van ic's. Het is daarom aan te bevelen een zero insertion force socket te gebruiken.

Poort 0	bit 0-7:	datalijnen (Oo t/m 07)
Poort 1	bit 0-7:	adreslijnen (Ao t/m A7)
Poort 2	bit 0-2:	adreslijnen (A8 t/m A10)
	bit 3 :	$\overline{CE}$/PROG (2716,2516,2758)
	:	A11 (2732,2532)
	bit 4 :	$\overline{OE}$ (2716,2516,2758)
	:	$\overline{CE}$/$\overline{PROG}$ (2732)
	:	$\overline{OE}$/$\overline{PROG}$ (2532)
	bit 5 :	"1" (2716,2516,2758,2532)
	:	$\overline{OE}$/V_{pp} (2732)
	bit 6 :	programmeerrelais
	bit 7 :	keuzerelais

TABEL 2: gebruik van de DCE-bus

MODE SELECTION 2716/2758

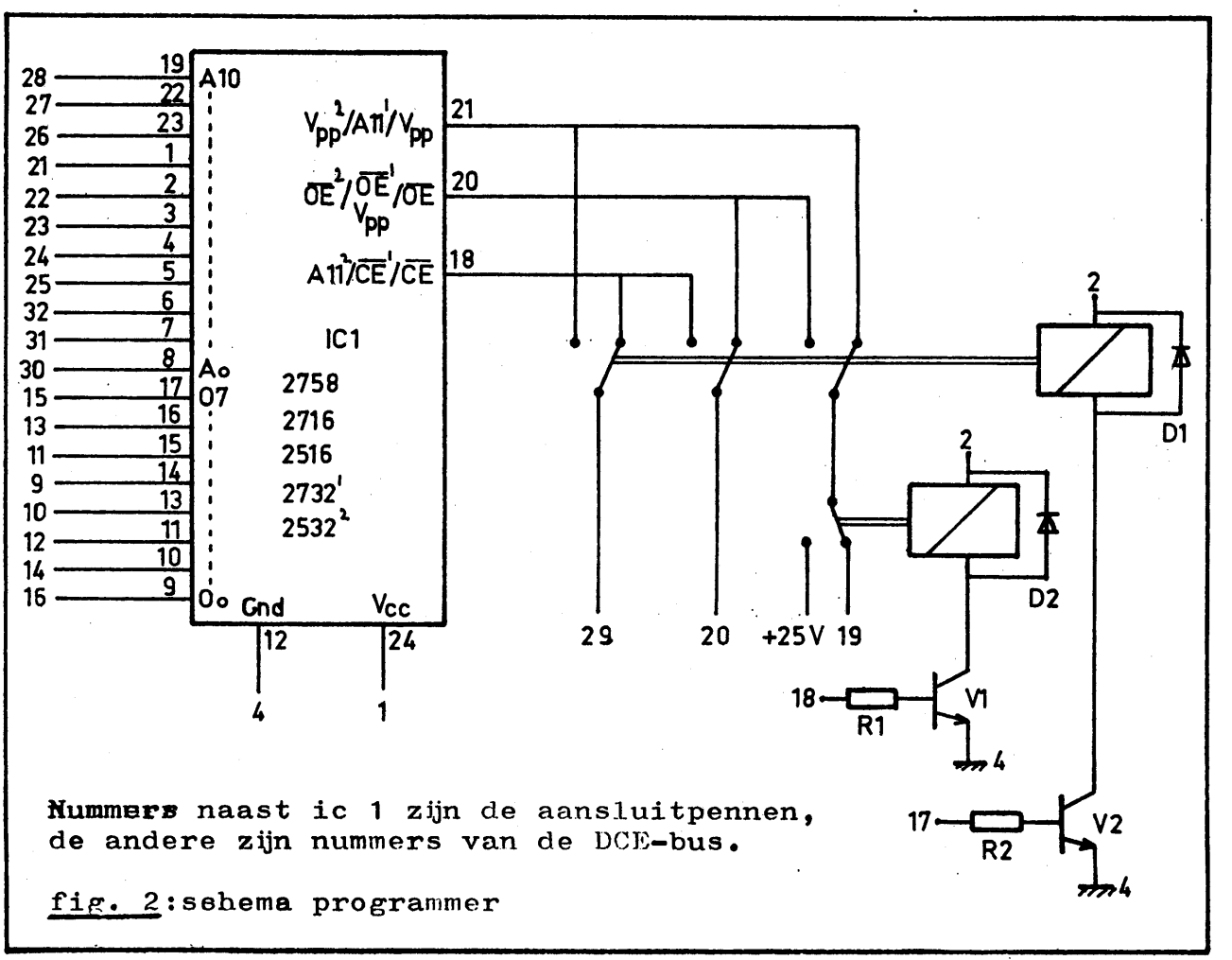
MODE \ PINS	$\overline{CE}/PGM$ (18)	A_R (19)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	V_{IL}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IL}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IL}	V_{IH}	+25	+5	High Z

MODE SELECTION 2732

MODE \ PINS	$\overline{CE}$ (18)	$\overline{OE}/V_{PP}$ (20)	V_{CC} (24)	OUTPUTS (9-11,13-17)
Read	V_{IL}	V_{IL}	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	V_{PP}	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+5	D_{OUT}
Program Inhibit	V_{IH}	V_{PP}	+5	High Z

DEVICE		MODE										
FUNCTION (PINS)		Read		Output Disable		Power Down		Start Programming		Inhibit Programming		Program Verification
TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516
PD/PGM (18)	PD/$\overline{PGM}$ (20)	V_{IL}	V_{IL}	Don't Care	V_{IH}	V_{IH}	V_{IH}	Pulsed V_{IL} to V_{IH}	Pulsed V_{IH} to V_{IL}	V_{IL}	V_{IH}	V_{IL}
$\overline{CS}$ (20)	Use PD/$\overline{PGM}$ as chip select	V_{IL}	N/A	V_{IH}	N/A	Don't Care	N/A	V_{IH}	N/A	V_{IH}	N/A	V_{IL}
V_{PP} (21)	V_{PP} (21)	+5	+5	+5	+5	+5	+ 5	+25	+25	+25	+25	+25 (or + 5)
V_{CC} (24)	V_{CC} (24)	+5	+5	+5	+5	+5	+5	+5	+5	+5	+5	+5
Q (9 to 11, 13 to 17)	Q (9 to 11, 13 to 17)	Q	Q	HI-Z	HI-Z	HI-Z	HI-Z	D	D	HI-Z	HI-Z	Q

TABEL 1: gegevens over gebruik EPROM's Vil="0" Vih="1"



Nummers naast ic 1 zijn de aansluitpennen, de andere zijn nummers van de DCE-bus.

fig. 2: schema programmer

De Software

Het programma zorgt ervoor dat de 8255 (DCE-bus) in de juiste mode staat en dat De juiste signalen verzonden worden. De software bestaat uit een BASIC en een machinetaal gedeelte. Het machinetaal gedeelte is ondergebracht in de Symbol Table en wordt bij de eerste RUN door het BASIC-programma naar de work-area geladen. Het programma voorziet in 8 eentoets-commando's, voor een overzicht zie tabel 3.

C	: Check
L	: Load
N	: Next load
P	: Program
R	: Read
T	: Type
U	: Utility
V	: Verify

TABEL 3: commando's

Check: controleert of de EPROM gewist is. De computer leest alle geheugenlocaties uit en verifieert of deze FF_H bevatten. D.w.z. of alle bits "1" zijn. Al naar gelang komt de melding "OK" of "BAD".

Load/Next load: deze 2 commando's zijn speciaal bedoeld voor gebruik met DNA. (DAInamic Assembler). Het load commando laadt een objectfile van adres 3000_H naar het werkgeheugen. Het werkgeheugen is een geheugenblok van 4K in de RAM van de DAI waar de data, die in de EPROM moet komen, wordt opgeslagen, tabel 4.
Het load commando plaatst de objectfile steeds vanaf het beginadres $A000_H$ in het werkgeheugen.
Met het next-load commando is het mogelijk verschillende files achter elkaar in het werkgeheugen te laden. In een pointer wordt bijgehouden wat het beginadres is voor de volgende te laden file. Dit commando is handig in een tweetal situaties.
In de eerste plaats als een aantal kleine programma's in dezelfde EPROM moeten komen te staan. De objectfiles kunnen dan na elkaar in het werkgeheugen geladen worden, waarna de EPROM geprogrammeerd wordt.

2EC- E75	: BASICprogramma
2EC-100B	: BASICprogramma & mlp in Symbol Table
A000-A3FF	: werkgeheugen 1K
A000-A7FF	: werkgeheugen 2K
A000-AFFF	: werkgeheugen 4K
B000-B195	: mlp
B1F5-B1FF	: variabelen & pointers zie sourceprogramma

TABEL 4: geheugen indeling

Verder kan het next-load commando gebruikt worden om een zeer groot assembler programma te laden. Het kan nl. voorkomen dat een assembler programma de maximale buffergrootte van 16K overschrijdt. De buffer is dan te klein om het gehele programma in één keer te bevatten. Door het programma dan op te delen in een aantal stukken en deze afzonderlijk te assembleren kan toch een groot programma gemaakt worden.
Er is echter één maar, indien er vanuit het ene stuk verwijzingen naar het andere stuk voorkomen, door bijvoorbeeld JMP of CALL instructies, dan moet een gedeelte van de symbol table van het ene stuk d.m.v. EQU-statements ook in het andere gedeelte worden opgenomen. Er kan dan zonder problemen van het ene stuk naar het andere verwezen worden.
De beide commando's kennen 3 foutmeldingen. En wel GAP ERROR en CHECKSUM ERROR, die geheel indentiek zijn aan die bij de BASIC LOADER V1.0, en als extra Memory range exceeded. Het kan immers voorkomen, dat bij het laden van een objectfile de geheugengrootte van de geselecteerde EPROM wordt overschreden. Er wordt dan direkt gestopt en de resterende bytes worden dus niet meer overgebracht naar het werkgeheugen. Verder kan het REServe statement (DNA pseudo-op) niet gebruikt worden, load negeert dit commando.

Resumerend is het gebruik van DNA met de programmer als volgt samen te vatten:
Er wordt een assembler programma ingetikt dat op een willekeurige plaats kan beginnen d.m.v. het ORG-statement. Het programma wordt dan geassembleerd en weggeschreven naar tape als objectfile (#O). Dan kan de EPROM-Programmer geladen worden en in utility wordt vervolgens de objectfile ingelezen (>R). Met het Load of Next-load commando kan dan de file naar het werkgeheugen overgebracht worden waarna de laatste stap het programmeren van de EPROM is.

Program: programmeert een EPROM met de data die in het werkgeheugen opgenomen is.

Read: leest een EPROM uit en copieert de inhoud in het werkgeheugen.

Type: wordt gebruikt om één van de 5 types EPROM te selecteren.

Utility: vanuit het programma wordt naar utility gesprongen. Gemakkelijk om het werkgeheugen te bekijken of om een objectfile in te lezen.

Verify: leest de inhoud van een EPROM en vergelijkt deze met de data in het werkgeheugen. Verify dient ter controle van het programmeren. Melding "OK" of "BAD".

De gehele software bestaat uit een sourceprogramma, bijbehorend machinetaal programma, BASIC-programma en de EPROM-programmer. Dit is het machinetaal en BASIC-gedeelte, samengevoegd tot een geheel, in DAInamic nummer 6 blz. 135 is beschreven hoe dit moet gebeuren.

Toepassingen:

De programmer kan gebruikt worden als hulpmiddel bij de ontwikkeling van een microprocessor systeem. Vooral in combinatie met DNA zijn er leuke mogelijkheden om op de DAI software ontwikkeling voor een 8080/8085 µP-systeem te doen. Voor mij was dat de direkte aanleiding tot het ontwerp van de programmer.
Het gebruik van de programmer hoeft niet beperkt te blijven tot de 5 genoemde types. Ombouw naar bijv. de 2708 is eenvoudig mogelijk, terwijl het met een aanpassings-printje mogelijk is een 1 chip processor zoals de 8748 van Intel te programmeren. Hierover in de toekomst, bij voldoende belangstelling, meer.

IC 1	EPROM
IC 2	7812
V1,V2	BC 548
D1,D2	1N4148
B1	brugcel B40/C1000
R1,R2	10k
R3	potmeter 5k
R4	1k5
C1	2200 µF 40V
C2	4,7 µF 30V tantaal
C3	100 nF
L1	goene LED
Z1	zekering 100 mA
S1	miniswitch

Verder: 2x 12V relais, zekeringhouder, trafo 24V 100 mA, flatcable met 34 polige connector, zero insertion force socket, kastje,netsnoer, stukje montageprint.

Vragen en opmerkingen
Questions and remarks
Fragen und bemerkungen

Guus Knoops
Siebenstraat 2B
6035 bd Ospel
Nederland
04951-31286

De programmer kan ook bij mij besteld worden.

TABEL 5: benodigdheden

program identification

title	:	EPROM-PROGRAMMER-DRIVER
author	:	G.KNOOPS
purpose	:	
comment	:	machine language part should be in RAM, hardware should be connected.

```
3     REM ******************************************************
4     REM *  Program for controling the EPROM-PROGRAMMER       *
5     REM *          Copyright GUKNO Software 1981             *
6     REM *                  21 april 1982          V3.0      *
7     REM ******************************************************
8     REM GUKNO Technics, Siebenstr. 2B, 6035 BD Ospel
9     REM Nederland; tel:04951-31286
10    MODE 0:PRINT CHR$(12):PRINT
20    PRINT TAB(40);"EPROM-Programmer"
30    DIM T$(5.0),MEMR%(5.0):RESTORE:FOR X%=1.0 TO 5.0:READ T$(X%),MEMR%(X%):NEXT
40    IF PEEK(#B000)=245 THEN 70:ES%=PEEK(#2A3)+PEEK(#2A4)*256-#B195-1
50    FOR I%=#B000 TO #B195:POKE I%,PEEK(ES%+I%):NEXT
51    REM Move MachineLanguageProgram to workarea
60    POKE #B1F4,1:POKE #B1FD,#A8:POKE #B1FA,0:POKE #B1FB,#A0
61    REM Set: TYPE Number,MEMoryRange
62    REM .    DestinationStartAddresObjectFile
70    TYPE$=T$(PEEK(#B1F4))
80    POKE #FE02,(PEEK(#FE02) IAND #BF):WAIT TIME 5:POKE #FE03,#9B:POKE #FE03,#80:PRINT TYPE$;"
";
81    REM 1:Programmingrelais off, programmingvoltage down
82    REM 2:GIC is cleared and all ports are set to output
90    A%=GETC:WAIT TIME 5
100   REM Commands: Check,Load,Next load,Program,Read,Type
101   REM ========  Utility,Verify
110   A%=GETC:IF A%=ASC("C") THEN 200:IF A%=ASC("L") THEN 300:IF A%=ASC("N") THEN 400:IF A%=ASC
"P") THEN 500:IF A%=ASC("R") THEN 600:IF A%=ASC("T") THEN 700:IF A%=ASC("U") THEN 800:IF A%=ASC
"V") THEN 900
120   GOTO 110
200   REM Check if the EPROM is erased
210   PRINT "Check"
220   GOSUB 1000
230   CALLM #B000
240   A%=PEEK(#B1FF):IF A%=0 THEN CURSOR 16,CURY:PRINT "OK":GOTO 80
250   CURSOR 15,CURY:PRINT "Not erased !":GOTO 80
300   REM Loading DNA objectfile
310   PRINT "Load"
320   POKE #B1FA,0:POKE #B1FB,#A0
321   REM Set DSAOF to #A000
330   GOTO 420
400   REM Loading Next DNA objectfile
401   REM DSAOF is end addres of previous file
410   PRINT "Next load"
420   CALLM #B037
430   A%=PEEK(#B1FF):IF A%=15 THEN CURSOR 15,CURY:PRINT "GAP ERROR":GOTO 80
440   IF A%=240.0 THEN CURSOR 15,CURY:PRINT "Memory Range exceeded":GOTO 80
450   IF A%=255.0 THEN CURSOR 15,CURY:PRINT "CHECKSUM ERROR":GOTO 80
470   GOTO 80
```

```
500   REM Programming the EPROM
510   PRINT "Program":CURSOR 15,CURY:PRINT "Busy !!"
520   GOSUB 1050
530   CALLM #B0E5
540   GOTO 80
600   REM Reading the EPROM
610   PRINT "Read"
620   GOSUB 1000
630   CALLM #B138
640   GOTO 80
700   REM Type EPROM can be changed
710   PRINT "Type:":FOR X%=1.0 TO 5.0:CURSOR 4*X%+6,CURY:PRINT T$(X%):NEXT
720   A%=GETC:WAIT TIME 5:INPUT "Select a type";IN$
730   Y%=0.0:FOR X%=1.0 TO 5.0:IF IN$=T$(X%) THEN Y%=X%
740   NEXT
750   IF Y%=0.0 THEN PRINT :PRINT TAB(14);"Not known type!!":GOTO 80
760   TYPE$=IN$:POKE #B1F4,Y%:POKE #B1FD,(MEMR%(Y%) IOR #A000)/255.0
761   REM Set new TYPE Number and MEMoryRange
770   PRINT :GOTO 80
800   CALLM #B194
900   REM Verifying if the EPROM is programmed correctly
910   PRINT "Verify"
920   GOSUB 1000
930   CALLM #B15E
940   A%=PEEK(#B1FF):IF A%=0 THEN CURSOR 16,CURY:PRINT "OK":GOTO 80
950   CURSOR 15,CURY:PRINT "BAD !":GOTO 80
1000  REM Initiate 8255 (GIC) for Check,Read,Verify
1001  REM GICB=GIC ControlByte;ECB=EPROM ControlByte
1002  REM RB=RelaisByte
1010  GICB%=#90
1020  IF TYPE$="2732" THEN ECB%=#80:RB%=#B0:GOTO 1100
1030  ECB%=#20:RB%=#38:GOTO 1100
1050  REM Initiate 8255 for Programming
1060  GICB%=#80
1070  IF TYPE$="2732" THEN ECB%=#C0:RB%=#D0:GOTO 1100
1080  IF TYPE$="2532" THEN ECB%=#40:RB%=#50:GOTO 1100
1090  ECB%=#58:RB%=#50
1100  POKE #FE03,GICB%:POKE #FE02,RB%:POKE #B1FE,ECB%:WAIT TIME 25:RETURN
10000 DATA 2716,2048,2516,2048,2732,4096,2532,4096,2758,1024
10001 REM Type,MEMoryRange
```

mlp

```
B000 F5 C5 D5 E5 21 01 FE 3A FE B1 57 1E 00 01 00 A0
B010 73 23 72 2B 2B 7E FE FF C2 2D B0 23 13 03 3A FD
B020 B1 B8 C2 10 B0 3E 00 32 FF B1 C3 32 B0 3E FF 32
B030 FF B1 E1 D1 C1 F1 C9 F5 C5 D5 E5 01 02 30 3A 00
B040 30 67 3A 01 30 6F 09 2B 2B 22 F8 B1 2A FA B1 EB
B050 0A FE 00 C2 CB B0 03 0A FE 00 C2 CB B0 03 0A FE
B060 FF C2 C3 B0 2A F8 B1 7C B8 DA C3 B0 C2 74 B0 79
B070 BD D2 C3 B0 21 F5 B1 36 FF 03 0A 86 77 03 0A 86
B080 77 03 0A FE 00 CA 91 B0 03 03 03 03 03 03 C3 50
B090 B0 03 0A 86 77 23 36 00 0A 23 77 2B 2B 03 0A 12
B0A0 13 86 77 3A FD B1 BA CA D3 B0 23 34 7E 23 BE C2
B0B0 9B B0 03 0A 2B 2B BE C2 DB B0 6B 62 22 FA B1 03
B0C0 C3 50 B0 3E 00 32 FF B1 C3 E0 B0 3E 0F 32 FF B1
B0D0 C3 E0 B0 3E F0 32 FF B1 C3 E0 B0 3E FF 32 FF B1
B0E0 E1 D1 C1 F1 C9 F5 C5 D5 E5 21 00 FE 3A FE B1 57
B0F0 1E 00 FE 58 CA FF B0 3E 10 32 FC B1 C3 04 B1 3E
B100 08 32 FC B1 01 00 A0 0A FE FF CA 2A B1 77 23 73
B110 23 72 C5 06 07 0E EB 00 00 00 0D C2 17 B1 05 C2
B120 15 B1 C1 3A FC B1 AA 77 2B 2B 03 13 3A FD B1 B8
B130 C2 07 B1 E1 D1 C1 F1 C9 F5 C5 D5 E5 21 01 FE 3A
B140 FE B1 57 1E 00 01 00 A0 73 23 72 2B 2B 7E 02 03
B150 13 23 3A FD B1 B8 C2 48 B1 E1 D1 C1 F1 C9 F5 C5
B160 D5 E5 21 01 FE 3A FE B1 57 1E 00 01 00 A0 0A 73
B170 23 72 2B 2B BE C2 8A B1 03 13 23 3A FD B1 B8 C2
B180 6E B1 3E 00 32 FF B1 C3 8F B1 3E FF 32 FF B1 E1
B190 D1 C1 F1 C9 CF 04
```

```
PAGE 01    GUKNO Software    21 april 1982  V3.0

002                          ORG   :B000     *** EPROM PROGRAMMER ***
003                PORT1     EQU   :FE01     PORT 1 OF 8255 (GIC)
004                *8255 PORT 0: USED AS DATALINES
005                *     PORT 1: USED AS LOW ADDRESLINES
006                *     PORT 2: USED AS ADDRES AND CONTROL-SIGNALS
007                SAWA      EQU   :A000     STARTADDRES WORK AREA
008                *WORK AREA: A000-A3FF FOR 1K EPROM
009                *           A000-A7FF FOR 2K EPROM
010                *           A000-AFFF FOR 4K EPROM
011                FADR      EQU   :3002     STARTADDRES OBJECTFILE
012                CHB       EQU   :B1FF     CHECKBYTE
013                *TESTED BY BASIC PROGRAM
014                ECB       EQU   :B1FE     EPROM CONTROL BYTE
015                *SET BY BASIC PROGRAM
016                MEMR      EQU   :B1FD     MEMORY RANGE
017                *SET BY BASIC PROGRAM
018                *INDICATES END OF WORK AREA
019                EXOR      EQU   :B1FC     EXOR BYTE USED IN PROGRAM
020                DSAOF     EQU   :B1FA     DSAOF
021                *DESTINATION START ADDRES OBJECT FILE
022                *OBJECT FILE IS LOADED FROM THIS ADDRES IN WORK AREA
023                FLEA      EQU   :B1F8     FILE LENGTH ADDRES
024                EXREG     EQU   :B1F5     EXTRA REGISTERS
025                *
026 B000 F5        CHECK     PUSH  PSW       CHECK IF EPROM ERASED
027 B001 C5                  PUSH  B
028 B002 D5                  PUSH  D
029 B003 E5                  PUSH  H
030 B004 2101FE              LXI   H,PORT1
031 B007 3AFEB1              LDA   ECB
032 B00A 57                  MOV   D,A
033 B00B 1E00                MVI   E,0
034 B00D 0100A0              LXI   B,SAWA
035 B010 73        NEXTC     MOV   M,E       LOW ADDRES IS APPLIED
036 B011 23                  INX   H
037 B012 72                  MOV   M,D       ADDRES & CONTROL IS APPLIED
038 B013 2B                  DCX   H
039 B014 2B                  DCX   H
040 B015 7E                  MOV   A,M       DATA IS READ FROM EPROM
041 B016 FEFF                CPI   :FF
042 B018 C22DB0              JNZ   NOTER     NOT ERASED
043 B01B 23                  INX   H
044 B01C 13                  INX   D
045 B01D 03                  INX   B
046 B01E 3AFDB1              LDA   MEMR
047 B021 B8                  CMP   B
048 B022 C210B0              JNZ   NEXTC
049 B025 3E00                MVI   A,0
050 B027 32FFB1              STA   CHB       ERASED!
051 B02A C332B0              JMP   ERASED
052 B02D 3EFF      NOTER     MVI   A,:FF
053 B02F 32FFB1              STA   CHB
054 B032 E1        ERASED    POP   H
```

```
055 B033 D1                 POP   D
056 B034 C1                 POP   B
057 B035 F1                 POP   PSW
058 B036 C9                 RET
059 B037 F5          LOAD   PUSH  PSW       LOAD OBJECTFILE IN WORK AREA
060 B038 C5                 PUSH  B
061 B039 D5                 PUSH  D
062 B03A E5                 PUSH  H
063 B03B 010230             LXI   B,FADR
064 B03E 3A0030             LDA   FADR-2    HIGH ORDER FILE LENGTH
065 B041 67                 MOV   H,A
066 B042 3A0130             LDA   FADR-1    LOW ORDER FILE LENGTH
067 B045 6F                 MOV   L,A
068 B046 09                 DAD   B
069 B047 2B                 DCX   H
070 B048 2B                 DCX   H
071 B049 22F8B1             SHLD  FLEA
072 B04C 2AFAB1             LHLD  DSAOF
073 B04F EB                 XCHG
074 B050 0A          NEXTL  LDAX  B
075 B051 FE00               CPI   :0
076 B053 C2CBB0             JNZ   GAPE      GAP ERROR
077 B056 03                 INX   B
078 B057 0A                 LDAX  B
079 B058 FE00               CPI   :0
080 B05A C2CBB0             JNZ   GAPE
081 B05D 03                 INX   B
082 B05E 0A                 LDAX  B
083 B05F FEFF               CPI   :FF
084 B061 C2C3B0             JNZ   FL        FILE LOADED
085                  *CONTROL IF END FILE IS REACHED
086 B064 2AF8B1             LHLD  FLEA
087 B067 7C                 MOV   A,H
088 B068 B8                 CMP   B
089 B069 DAC3B0             JC    FL        FILE LOADED
090 B06C C274B0             JNZ   NOTJET
091 B06F 79                 MOV   A,C
092 B070 BD                 CMP   L
093 B071 D2C3B0             JNC   FL
094 B074 21F5B1      NOTJET LXI   H,EXREG
095                  *EXREG CHECKSUM/EXR.+1 COUNTER/EXR.+2 LENGTH RECORD
096 B077 36FF               MVI   M,:FF     CHECKSUM BYTE
097 B079 03                 INX   B
098 B07A 0A                 LDAX  B
099 B07B 86                 ADD   M
100 B07C 77                 MOV   M,A
101 B07D 03                 INX   B
102 B07E 0A                 LDAX  B
103 B07F 86                 ADD   M
104 B080 77                 MOV   M,A
105 B081 03                 INX   B
106 B082 0A                 LDAX  B
107 B083 FE00               CPI   0
```

```
PAGE 03    GUKNO Software    21 april 1982  V3.0

108 B085 CA91B0             JZ    NORES     RESERVE TYPE IS NOT USED
109 B088 03                 INX   B         RESERVE TYPE IS IGNORED
110 B089 03                 INX   B
111 B08A 03                 INX   B
112 B08B 03                 INX   B
113 B08C 03                 INX   B
114 B08D 03                 INX   B
115 B08E C350B0             JMP   NEXTL
116 B091 03         NORES   INX   B
117 B092 0A                 LDAX  B
118 B093 86                 ADD   M
119 B094 77                 MOV   M,A
120 B095 23                 INX   H
121 B096 3600               MVI   M,0         CLEAR COUNTER
122 B098 0A                 LDAX  B           LENGTH RECORD
123 B099 23                 INX   H
124 B09A 77                 MOV   M,A
125 B09B 2B         NEXTB   DCX   H
126 B09C 2B                 DCX   H
127 B09D 03                 INX   B
128 B09E 0A                 LDAX  B
129 B09F 12                 STAX  D
130 B0A0 13                 INX   D
131 B0A1 86                 ADD   M
132 B0A2 77                 MOV   M,A
133 B0A3 3AFDB1             LDA   MEMR
134 B0A6 BA                 CMP   D
135 B0A7 CAD3B0             JZ    MEMREX    MEMORY RANGE EXCEEDED
136 B0AA 23                 INX   H
137 B0AB 34                 INR   M
138 B0AC 7E                 MOV   A,M
139 B0AD 23                 INX   H
140 B0AE BE                 CMP   M
141 B0AF C29BB0             JNZ   NEXTB     NEXT BYTE
142 B0B2 03                 INX   B           CONTROL OF CHECKSUM
143 B0B3 0A                 LDAX  B
144 B0B4 2B                 DCX   H
145 B0B5 2B                 DCX   H
146 B0B6 BE                 CMP   M
147 B0B7 C2DBB0             JNZ   CHECKE    CHECKSUM ERROR
148 B0BA 6B                 MOV   L,E
149 B0BB 62                 MOV   H,D
150 B0BC 22FAB1             SHLD  DSAOF     ADJUST DSAOF FOR NEXT LOAD
151 B0BF 03                 INX   B
152 B0C0 C350B0             JMP   NEXTL
153 B0C3 3E00       FL      MVI   A,0
154 B0C5 32FFB1             STA   CHB
155 B0C8 C3E0B0             JMP   END
156 B0CB 3E0F       GAPE    MVI   A,:F
157 B0CD 32FFB1             STA   CHB
158 B0D0 C3E0B0             JMP   END
159 B0D3 3EF0       MEMREX  MVI   A,:F0
160 B0D5 32FFB1             STA   CHB
```

```
PAGE 04    GUKNO Software    21 april 1982  V3.0

161 B0D8 C3E0B0             JMP   END
162 B0DB 3EFF       CHECKE  MVI   A,:FF
163 B0DD 32FFB1             STA   CHB
164 B0E0 E1         END     POP   H
165 B0E1 D1                 POP   D
166 B0E2 C1                 POP   B
167 B0E3 F1                 POP   PSW
168 B0E4 C9                 RET
169 B0E5 F5         PROG    PUSH  PSW       PROGRAMMING OF THE EPROM
170 B0E6 C5                 PUSH  B
171 B0E7 D5                 PUSH  D
172 B0E8 E5                 PUSH  H
173 B0E9 2100FE             LXI   H,PORT1-1
174 B0EC 3AFEB1             LDA   ECB
175 B0EF 57                 MOV   D,A
176 B0F0 1E00               MVI   E,0
177 B0F2 FE58               CPI   :58
178 B0F4 CAFFB0             JZ    E16
179 B0F7 3E10               MVI   A,:10
180 B0F9 32FCB1             STA   EXOR
181 B0FC C304B1             JMP   E32
182 B0FF 3E08       E16     MVI   A,:8
183 B101 32FCB1             STA   EXOR
184 B104 0100A0     E32     LXI   B,SAWA
185 B107 0A         NEXTP   LDAX  B
186 B108 FEFF               CPI   :FF
187 B10A CA2AB1             JZ    NOPROG    NOT NECESSARY TO PROGRAM
188 B10D 77                 MOV   M,A       DATA IS APPLIED
189 B10E 23                 INX   H
190 B10F 73                 MOV   M,E       LOW ADDRES IS APPLIED
191 B110 23                 INX   H
192 B111 72                 MOV   M,D       PROG-SIGNAL IS APPLIED
193 B112 C5                 PUSH  B
194 B113 0607               MVI   B,7       LOOP OF 50 MSEC.
195 B115 0EEB       LOOP2   MVI   C,235
196 B117 00         LOOP1   NOP
197 B118 00                 NOP
198 B119 00                 NOP
199 B11A 0D                 DCR   C
200 B11B C217B1             JNZ   LOOP1
201 B11E 05                 DCR   B
202 B11F C215B1             JNZ   LOOP2
203 B122 C1                 POP   B
204 B123 3AFCB1             LDA   EXOR
205 B126 AA                 XRA   D         STOP PROGRAMMING
206 B127 77                 MOV   M,A       PROG-SIGNAL IS INVERTED
207 B128 2B                 DCX   H
208 B129 2B                 DCX   H
209 B12A 03         NOPROG  INX   B
210 B12B 13                 INX   D
211 B12C 3AFDB1             LDA   MEMR
212 B12F B8                 CMP   B
213 B130 C207B1             JNZ   NEXTP
```

```
PAGE 05    GUKNO Software    21 april 1982  V3.0

214 B133 E1                  POP   H
215 B134 D1                  POP   D
216 B135 C1                  POP   B
217 B136 F1                  POP   PSW
218 B137 C9                  RET
219 B138 F5          READ    PUSH  PSW        READING THE EPROM
220 B139 C5                  PUSH  B
221 B13A D5                  PUSH  D
222 B13B E5                  PUSH  H
223 B13C 2101FE              LXI   H,PORT1
224 B13F 3AFEB1              LDA   ECB
225 B142 57                  MOV   D,A
226 B143 1E00                MVI   E,0
227 B145 0100A0              LXI   B,SAWA
228 B148 73          NEXTR   MOV   M,E
229 B149 23                  INX   H
230 B14A 72                  MOV   M,D
231 B14B 2B                  DCX   H
232 B14C 2B                  DCX   H
233 B14D 7E                  MOV   A,M
234 B14E 02                  STAX  B
235 B14F 03                  INX   B
236 B150 13                  INX   D
237 B151 23                  INX   H
238 B152 3AFDB1              LDA   MEMR
239 B155 B8                  CMP   B
240 B156 C248B1              JNZ   NEXTR
241 B159 E1                  POP   H
242 B15A D1                  POP   D
243 B15B C1                  POP   B
244 B15C F1                  POP   PSW
245 B15D C9                  RET
246 B15E F5          VERIFY  PUSH  PSW        VERIFY OF PROGRAMMING
247 B15F C5                  PUSH  B
248 B160 D5                  PUSH  D
249 B161 E5                  PUSH  H
250 B162 2101FE              LXI   H,PORT1
251 B165 3AFEB1              LDA   ECB
252 B168 57                  MOV   D,A
253 B169 1E00                MVI   E,0
254 B16B 0100A0              LXI   B,SAWA
255 B16E 0A          NEXTV   LDAX  B
256 B16F 73                  MOV   M,E
257 B170 23                  INX   H
258 B171 72                  MOV   M,D
259 B172 2B                  DCX   H
260 B173 2B                  DCX   H
261 B174 BE                  CMP   M
262 B175 C28AB1              JNZ   BAD
263 B178 03                  INX   B
264 B179 13                  INX   D
265 B17A 23                  INX   H
266 B17B 3AFDB1              LDA   MEMR
```

```
PAGE 06    GUKNO Software    21 april 1982  V3.0

267 B17E B8                  CMP   B
268 B17F C26EB1              JNZ   NEXTV
269 B182 3E00                MVI   A,0        VERIFY OK
270 B184 32FFB1              STA   CHB
271 B187 C38FB1              JMP   OK
272 B18A 3EFF        BAD     MVI   A,:FF      VERIFY BAD
273 B18C 32FFB1              STA   CHB
274 B18F E1          OK      POP   H
275 B190 D1                  POP   D
276 B191 C1                  POP   B
277 B192 F1                  POP   PSW
278 B193 C9                  RET
279 B194 CF          UT      RST   1          JUMP TO UTILITY
280 B195 04                  DATA  4
281 B196                     END

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

BAD     B18A   CHB     B1FF   CHECK   B000   CHECKE B0DB
DSAOF   B1FA   E16     B0FF   E32     B104   ECB     B1FE
END     B0E0   ERASED  B032   EXOR    B1FC   EXREG   B1F5
FADR    3002   FL      B0C3   FLEA    B1F8   GAPE    B0CB
LOAD    B037   LOOP1   B117   LOOP2   B115   MEMR    B1FD
MEMREX  B0D3   NEXTB   B09B   NEXTC   B010   NEXTL   B050
NEXTP   B107   NEXTR   B148   NEXTV   B16E   NOPROG  B12A
NORES   B091   NOTER   B02D   NOTJET  B074   OK      B18F
PORT1   FE01   PROG    B0E5   READ    B138   SAWA    A000
UT      B194   VERIFY  B15E
```

This EPROM-programmer is available from Mr. KNOOPS at the price of 258 Gld. This includes : - assembled circuit
- housing
- flatcable + DCE-connector
- zero force insertion socket

comments

We ordered one unit , but at this time there are not yet any test results. As Mr. KNOOPS can programm EPROM's on his DAIpc, I think anyone should succeed with this unit.
While the programmer is not cheap, one should consider that there are a lot of expensive items in the circuit.(zero force,flatcable..)

disadvantages : - the programmer is not interfaced following the DAI-DCE-concept.
- the programmer software can not be loaded from DCR or DISC.
- in the near future, the unit will not be available from stock.

advantages : - the unit will be delivered totally finished, including software on cassette.
- there is no need to connect any hardware inside the housing of the computer.

****

```
10    REM ********************************
11    REM EXEMPLE DE PROGRAMME PRINCIPAL
12    REM ********************************
20    PRINT CHR$(12):MODE 0:POKE #FF05,255:POKE #74,1:POKE #75,95:COLORG 12 0 0
0:COLORT 12 0 0 0
30    GOSUB 60000:REM CHARGEMENT ROUTINE MACHINE
40    PRINT CHR$(12):MODE 0:PRINT "DEFINITION DE LA FENETRE DE TRAVAIL PAR LES C
OORDONNEES DE"
50    PRINT "2 COINS EXTREMES (X1,Y1) ET (X2,Y2):"
60    PRINT :PRINT "NB : X1 doit etre un multiple de 8 et X2 un multiple"
70    PRINT "de 8, moins 1.":PRINT
80    INPUT "Fenetre (X1,Y1,X2,Y2)";X1%,Y1%,X2%,Y2%:PRINT
90    X1%=8*INT(X1%/8):X2%=8*INT((X2%+1)/8)-1
100   INPUT "Nombre de rotations";NROT%:PRINT
110   INPUT "Mode (1/2/3/4/5/6)";M%
120   GOSUB 1000:REM CALCUL DES PARAMETRES
500   REM ****************************************
501   REM FAITES VOTRE DESSIN : LIGNES 5001000
502   REM ****************************************
510   DRAW 0,YMAX/2 XMAX,YMAX/2 0
520   FOR O%=1 TO 200:DOT RND(XMAX),RND(YMAX) 0:NEXT
530   DRAW X1%,Y2% X1%,Y1% 21:DRAW X2%,Y1% X2%,Y2% 21
540   FOR O%=1 TO 50:DOT X1%+RND(X2%-X1%),Y2%+RND(Y1%-Y2%) 21:NEXT
550   CALLM #3009
560   IF GETC=0 THEN 560
570   GOTO 40
999   END
1000  REM **********************
1001  REM CALCUL DES PARAMETRES
1002  REM **********************
1003  REM DONNEES : FENETRE: X1,Y1,X2,Y2
1004  REM ..........MODE: M
1005  REM ..........NOMBRE DE ROTATIONS: NROT
1010  IF M%=1 THEN MODE 1:DE%=#B9EB:LL%=24
1020  IF M%=2 THEN MODE 2:DE%=#B9EB:LL%=24
1030  IF M%=3 THEN MODE 3:DE%=#A8BD:LL%=46
1040  IF M%=4 THEN MODE 4:DE%=#A8BD:LL%=46
1050  IF M%=5 THEN MODE 5:DE%=#6645:LL%=90
1060  IF M%=6 THEN MODE 6:DE%=#6645:LL%=90
1070  IF X1%>X2% THEN T%=X2%:X2%=X1%:X1%=T%
1080  IF Y2%>Y1% THEN T%=Y2%:Y2%=Y1%:Y1%=T%
1090  DEB%=DE%+Y1%*LL%-X1%/4:POKE #3000,DEB% MOD 256:POKE #3001,DEB%/256
1100  DEB2%=DEB%-LL%:POKE #3002,DEB2% MOD 256:POKE #3003,DEB2%/256
1200  NBYT%=(X2%-X1%+1)/4:POKE #3005,NBYT%
1210  NLIGN%=Y1%-Y2%:POKE #3004,NLIGN%
1220  FIN%=DE%+Y2%*LL%-X1%/4:POKE #3006,FIN% MOD 256:POKE #3007,FIN%/256
1230  SAUT%=LL%-NBYT%:POKE #3008,SAUT%
1240  POKE #30A0,NROT% MOD 256:POKE #30A1,NROT%/256
1999  RETURN
60000 DATA #C5,#D5,#E5,#F5,#3A,#5,#30,#47,#11,#0,#31,#2A,#0,#30,#7E,#12
60001 DATA #2B,#13,#5,#C2,#17,#30,#2A,#0,#30,#EB,#2A,#2,#30,#3A,#4,#30
60002 DATA #4F,#3A,#5,#30,#47,#7E,#12,#2B,#1B,#5,#C2,#2E,#30,#3A,#8,#30
60003 DATA #47,#7B,#90,#D2,#40,#30,#15,#5F,#7D,#90,#D2,#47,#30,#25,#6F,#D
60004 DATA #C2,#2A,#30,#2A,#6,#30,#EB,#21,#0,#31,#3A,#5,#30,#47,#7E,#12
60005 DATA #1B,#23,#5,#C2,#57,#30,#2A,#A0,#30,#2B,#22,#A0,#30,#7C,#B7,#C2
60006 DATA #0D,#30,#7D,#B7,#C2,#0D,#30,#F1,#E1,#D1,#C1,#C9
60010 FOR M%=#3009 TO #3074
60020 READ B%:POKE M%,B%
60030 NEXT:RETURN
61001 REM
```

ROUTINE MACHINE ROUTINE MACHINE ROUTINE MACHINE ROUTINE MACHINE ROUTINE MACHINE

TITRE :.....ROTATION D'UNE ZONE D'ECRAN (EN PAGE GRAPHIQUE)
AUTEUR :.....C. POELS
INTERET :.....Peut servir comme sous-routine dans un programme graphique faisant appel a l'animation.

COMMENTAIRES : Le programme suivant fait effectuer a toute zone graphique d'ecran, une rotation vers le haut. La vitesse de cette rotation varie beaucoup suivant la taille de la fenetre definie.

Le programme peut se decomposer comme suit :
Lignes 10 a 110 : Presentation et entree des parametres :
- Coordonnees de deux coins opposes de la fenetre. (X1,Y1,X2,Y2)
- Nombre de rotations vers le haut. (NROT)
- MODE graphique desire. (M)

La ligne 30 fait appel au sous-programme de chargement en memoire de la sous-routine machine.
La ligne 1000 appelle le traitement des parametres et leur stockage en memoire.
Lignes 500 a 1000 : Dessin se trouvant sur l'ecran au moment de la rotation.
Lignes 1000 a 1999 : Traitement des parametres.
Lignes 60000 a 60999 : Routine machine (occupant la zone #3009 a #3074).

---

super noise generator

BLADWIJZER

Pagina	Inhoud
1	- Bestukken van de print.
	- Inbouwen van de print.
2	- Aanbrengen van de boutjes.
3	- Aansluiten van de print.
	- Testen van de super noise generator.
	- Afregelen van trimmer P1.
4	- Een korte test.
	- Volledige test van de super noise generator.
	- Software besturing van de uitbreiding.
	- Extra noise commando's.
5	- Cassette sound sturing.
	- Principe werking.
	- Werking van de noise uitbreiding.
6	- Cassette sturing.
7	- Voorbeelden.
	- De commando's.
	- Programma voorbeelden
8	- Programma voorbeelden (vervolg).
I	- FIGURE 1 (Principe schema SNG).
II	- FIGURE 2A/2B (Componenten layout SNG).
III	- FIGURE 3 (Verbindingen met de DAI, principe).
IV	- FIGURE 4 (verbindingen met de DAI, vervolg).
	- TABLE 1 (color-code for wires).
V	- FIGURE 5 (interne conecties DAI-print).
VI	- TABLE 2 (super noise generator prom data).
VII	- TABLE 3 (result of envelope /noise commands).
VIII	- DATA-SHEETS SN 76477 N
XII	- TABLE 4 (color-code for resistors).

--- with many thanks to R.Rens---

W.De Winter R.Corswandt

```
**********************************************************
***                                                    ***
***     S U P E R   N O I S E   G E N E R A T O R      ***
***                                                    ***
***                copyright DAInamic                  ***
**********************************************************
```

OPMERKINGEN.

ALGEMEEN ADVIES : KONTROLEER STEEDS UW WERK VOORDAT U
EEN VOLGENDE STAP DOET.

Noch DAInamic, noch de heren De Winter & Corswandt kunnen verantwoordelijk gesteld worden voor beschadigingen die zouden voorkomen tijdens het installeren of het gebruiken van de Super Noise Generator.

1/ PRINT BESTUKKEN.

Kontroleer de print op eventuele onderbroken of kortgesloten banen (vergelijk met koper-layout, figure 2B)
Bestuk de print volgens het komponentenschema (zie figure 2A)
De onderdelen plaats je aan de niet-verkoperde kant
Gebruik een soldeerbout met fijne punt en een vermogen kleiner dan 40 watt.
Indien nodig plooi dan de komponenten met een tangetje zorgvuldig op maat.

Plaats en soldeer de komponenten volgens grootte :

eerst de vier draad bruggen (W1 tot W4), ontmantel het stijve draadje, knip en plooi op maat.
alle weerstanden (R1-R17) insteken (voor de waarde, gebruik TABLE 4).
print omdraaien en solderen
-hierdoor zitten deze mooi op gelijke hoogte
dan de IC-voetjes,soldeer eerst twee pootjes en controleer of ze volledig tegen de print aanzitten, eventueel corrigeren
-let op of hun inkeping aan de juiste kant zit (zoals de inkeping op het ic).
en als volgende de kondensatoren (C1-C9)
-bij de elektrolytische geeft een zwarte band de minzijde aan
-bij de druppel-tantaalkondensator C9 geeft een +teken de pluskant aan
-de andere kondensatoren hebben geen richtingsvoorkeur, voorzichtig aanbrengen, zodat de aansluitdraden niet loskomen .
soldeer als laatste de trimmer P1.

Knip de uitstekende draadjes af (lengte maximum 3mm).

DE IC's NOG NIET AANBRENGEN !!!!!!

Kontroleer of dat alle komponenten op de juiste plaats zitten.
Kijk het soldeerwerk na op eventuele fouten.

2/ INBOUWEN VAN DE PRINT.

Verbreek eerst alle verbindingen met de DAI-pc.
Trek de vier zwarte pluggen uit de zijkant van de kast.
Hef het bovendeel aan de voorkant op,en duw het naar achter, verwijder voorzichtig een eventueel aardingsdraadje tussen de metalen plaat rond het toetsenbord en de 0V.

Het inbouwen doen we aan de voorzijde van de voeding (zie FIGUUR 6).
Indien u een Digitale Cassette-Recorder bezit trek dan eerst het eprom-kaartje uit de X-bus.

Schema 2

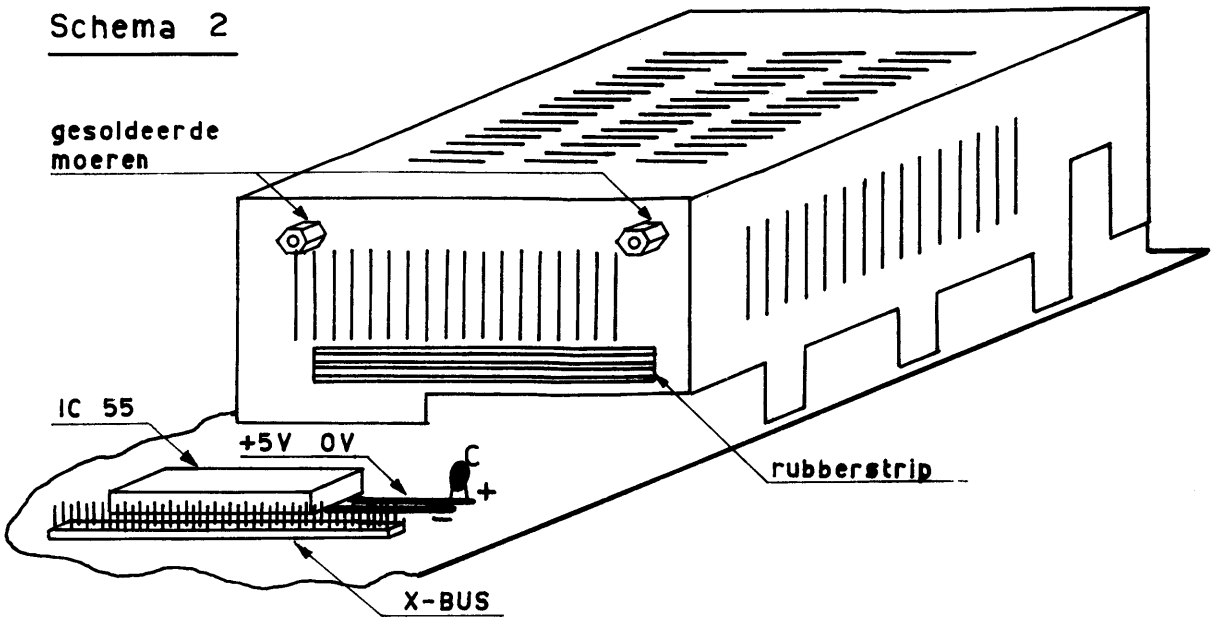


FIGURE 6

2.1/ Aanbrengen van de boutjes.

Lijm de rubber strip (zelfklevend) onderaan de voorzijde van de voeding.
Dit dient als isolerend afstandsstuk.

Verwijder het deksel van de voeding (dit om gemakkelijk te werken).

Schroef de moeren in de hiervoor voorzienen gaten op de SNG-print.
Alzo krijgen we de juiste maat om de voorzijde van de voeding te vertinnen.

Gebruik hiervoor liefst een zwaardere soldeerbout dan deze waarmee u
de komponenten soldeert (vermogen >40 watt).

Plaats de DAI-print vertikaal, met de achterzijde op de tafel,
plaats de SNG-print horizontaal (met de komponenten boven), op de
voorzijde van de voeding (VGL. FIGUUR 6).
Soldeer de moeren vast (zorg voor een goede vloeiing van het soldeer).
Kontroleer of er geen uitstekende draadjes kontakt maken met de voeding.

Schroef nu de print terug los.

NOTA !!

Wanneer er een koelplaatje gemonteerd is op de voorzijde van de voeding,
dan zijn er volgende oplossingen mogelijk:

- verwijder het koelplaatje (bij de meeste versies niet gebruikt).
- soldeer een metalen stripje aan de rechterkant van de voeding,
zodat het SNG-printje juist naast het koelplaatje komt en iets
voorbij de DAI-print.

2.2/ Aansluiten van de print.

a) verbinden van de kabel aan de S.N.G.-print.

Ontmantel nu de meegeleverde kabel zodat een 10 cm buitenmantel overblijft, dit stuk kan gepast heen en weer geschoven worden.

Trek een rode en een zwarte draad uit de buitenmantel en soldeer deze respektievelijk aan de +5 volt en de 0 volt van de print (zie FIGURE 2A).

Er zijn twee rijen aansluitpunten,deze zijn aangeduid door de letters A tot en met K (zie FIGURE 2A).
Soldeer de overige draadjes in de kabel, aan deze punten van de SNG-print. Gebruik hiervoor TABLE 1 als standaard.
Verwijder de eventueel overtollige draadjes nog niet uit de mantel, ze kunnen nog als reserve dienen bij een eventuele verkeerde handeling.

b) verbinden van de kabel aan de DAI-print.

Gebruik hiervoor FIGURE 5 om de punten terug te vinden.
Eerst verbindt je de rode en de zwarte draad. Dit kan je best doen rechts van het IC (IC 55 op FIGURE 5), tussen de voeding en de X-bus. Hier bevindt zich een condensatortje met aanduiding + en - (dwz resp. +5V en 0V).Krab zorgvuldig de lak van de banen en soldeer de rode (+5V) en de zwarte (0V) draad erop.

Schroef het printje op de moertjes en soldeer de overige draadjes op de aangeduide poten van de komponenten (zie FIGURE 3,4,5).
Daar de DAI-firma niet altijd dezelfde componenten gebruikt is soms een tweede plaats aangeduid als alternatief. U soldeert daar waar u het beste bijkan.
Om punt J te verbinden dient u de TV-kaart (print op steuntjes) weg te nemen.Let er bij het terug monteren van deze kaart op dat er geen kontakten verbogen zijn in de konnektor .

* CONTROLEER NOGMAALS OF ALLES JUIST AANGESLOTEN IS, EN OF ER GEEN KORTSLUITINGEN GEMAAKT ZIJN !!!

* PLAATS NU DE IC's IN HUN RESPECTIVELIJKE VOETJES, LET OP DE POLARITEIT !! OPPASSEN VOOR STATISCHE LADINGEN !!!

Als er geen fouten gemaakt zijn moet de SUPER NOISE GENERATOR nu volledig in orde zijn, tijd nu voor een korte test.

3/ TESTEN VAN DE SUPER NOISE GENERATOR.

Plaats het deksel van de voeding terug op zijn plaats. Zet het eventuele DCR eprom-kaartje op de X-bus. Maak de overige verbindingen in orde (KAST NIET SLUITEN !). Schakel de TV en de DAI-PC aan.

3.1/ AFREGELEN VAN TRIMMER P1 (cassette-geluid).

Sluit een cassette-recorder op CAS1 aan, plaats P1 in de middenstand.
Start een normale muziek-cassette, en tik in POKE#40,0 : de cassette start en er moet muziek hoorbaar zijn op de TV.
Regel nu trimmer P1 af op een normaal geluidsniveau.

* NOTA: als er veel vervorming optreed, regel dan het geluidsvolume op de recorder.

3.2/ EEN KORTE TEST (new noise).

geef NEW en tik in :
```
* 10 ENVELOPE 0 16
* 20 NOISE OFF:WAIT TIME 1
* 30 NOISE 0 7:WAIT TIME 200
* 100 GOTO 20
* RUN
```

En nu hoort u een wegstervende lage ruisexplosie

3.3/ VOLLEDIGE TEST VAN DE SUPER NOISE GENERATOR.

Start de meegeleverde demonstratie-cassette.
Gebruik bij voorkeur een recorder met aangesloten remote-control,
alle programma's laden automatisch het volgende.
De band bevat volgende programma's :

```
0 SNG START
0 SNG DEMO+TEST                        : test alle bijkomende noise commando's
0 VIERTAKT MOTOR BOOTSTRAP LOADER
1ML VIERTAKT MOTOR 300 350
0 BASIC VIERTAKT MOTOR MET SNG         : programma voorbeeld noise
0 CASSETTE SOUND DEMO                  : programma voorbeeld cassette sound
  MUZIEK
```

4/ SOFTWARE STURING VAN DE UITBREIDING.

De uitbreiding met de S.N.G. is praktisch altijd kompatibel met vroegere muziekprogramma 's.
Dit doordat de bijkomende hardware gebruik maakt van de onderste volumeniveaus (1-7) van de normale DAI-noise.

4.1/ EXTRA NOISE COMMANDO'S

Het uiteindelijke DAI-volume is een kombinatie van NOISE-volume en ENVELOPE-volume (zie tabel 1).
Deze kombinatie kunt u te weten komen via (enkel voor noise commando):

```
ENVELOPE X Y:NOISE X Z:WAIT TIME 10:?HEX$(PEEK(#295) SHR 4.
```

Voor de besturing van de SNG is het eenvoudigste het envelope-volume op 16 te zetten en het noise-volume te gebruiken van 0 tot 7
(zie programma voorbeelden).

Bijkomende effecten verkregen door de S.N.G.

```
NOISE X 0 : S.N.G. reset, NOISE OFF; dit kommando of NOISE OFF dient u
            steeds te gebruiken voor u de niveaus 4, 5, 6 of 7 gebruikt.
NOISE X 1 : kontinu lage ruis.
NOISE X 2 : kontinue ruis met regelbare frequentie(F) dit gebeurt via
            SOUND 2 X 0 K FREQ(F) (X=ENVELOPE-NR;K=Kies 0,1,2 of 3)
NOISE X 3 : kontinu hoge ruis.
NOISE X 4 : hoge, snel wegstervende ruis
NOISE X 5 : hoge, traag uitstervende ruis
NOISE X 6 : lage, snel wegstervende ruis
NOISE X 7 : lage, traag uitstervende ruis
```

NOISE X 8 tot NOISE X 15 :normale DAI-ruis.

4.2/ CASSETTE SOUND STURING.

Een extra mogelijkheid van SNG bestaat erin om het geluidssignaal,komende van CAS1 door te schakelen naar de T.V. en stereo output.Dit onder software controle, doormiddel van POKE's.Dit zowel vanuit command-mode als uit BASIC.

-1) POKE#40,0 : start cassette 1 + 2 ,
+ doorgeven van cassettegeluid naar TV en stereo output.

POKE#40,#10 : stop cassette 2, cassette 1 op ; geen doorgeven.

POKE#40,#20 : stop cassette 1, cassette 2 op ; geen doorgeven.

POKE#40,#30 : stop cassette 1 + 2 ; geen doorgeven.

-2) Wanneer een van volgende POKE's gegeven is , wordt er, bij een LOAD opdracht, de aangeduide cassette geselecteerd.

POKE#13D,#10 : selecteer cassette 1

POKE#13D,#20 : selecteer cassette 2

POKE#13D,#30 : selecteer beide cassettes; doorgeven van geluid.

-3) DCR gebruikers kunnen, inplaats van bovenstaande POKE's, gebruik maken van de volgende commando's:

CAS / CAS1 /CASSETTE / CASSETTE 1 :selecteer cassette 1

CAS2 / CASSETTE 2 :selecteer cassette 2

CAS3 / CASSETTE 3 :selecteer cassette 1 +2 ;
doorgeven geluid.

5/PRINCIEPE WERKING.

Gebruik hiervoor FIGURE 1,3,4 en TABLE 4 .

5.1/ WERKING VAN DE NOISE UITBREIDING.

De eigenlijke schakeling bestaat uit 3 delen :

- de adres-decodering + besturing ; gevormd door de prom IC1.
- de wisselschakeling (voor digitale niveau's) en signaal oppoetser IC3 (N1...N4).
- en het eigenlijke 'hart' van de schakeling, de 'COMPLEX SOUND GENERATOR' IC2 met toebehoren.

A) adres-decodering :

Op de adress select ingangen van IC1 komen de vier data-bits toe die het volume van de normale DAI-noise regelen.Elk adres levert een bepaald bitpatroon aan de acht data uitgangen, die elk een bepaalde functie sturen van het sound-IC (IC2).Bestudeer hiervoor TABLE 2.
Het adres valid signaal (pin 15) word steeds op grond potentiaal gehouden, zodat elke verandering aan de adres ingangen een ander datawoord doet verschijnen aan de data uitgangen.

B) de schakeling rond IC3.

De schakeling selecteert een signaal,ofwel opgewekt door SOUND 2,ofwel van de VCO oscillator gevormd door C5,R17 + een deel van IC2 en de stuurspanning gekreerd door R10,R11.De selectie gebeurd door data-bit D1 van IC1. Het uitgangssignaal wordt gebruik om de noise oscillator in IC2 te sturen. Om de signalen terug op TTL niveau te krijgen is er gebruik gemaakt van smith-trigger inputs.

C) de 'COMPLEX SOUND GENERATOR'.

Gebruik hiervoor ook de DATA-sheets! (vooral het blokschema PAGE IX)

Via pin 3 van IC2 wordt het clock-signaal (verkregen via IC3) toegevoerd aan de noise generator.

Dit pseudo random signaal wordt vervolgens door de noise fiter gevoerd.De kantel-frequentie word bepaald door R12,C4 ,waarvan C4 m.b.v. D4 (IC1) wel of niet aangesloten wordt, (de datauitgangen van IC1 zijn open collectoruitgangen), wat resulteerd in een hoge of lage ruis.

Vervolgens wordt dit signaal aangeboden aan de mixer, waarvan enkel het noise kanaal gebruikt is (vast geselecteerd via pin 25,26,27).

Nu wordt in de envelope generator de stijg- en daaltijd van het noisesignaal bepaald, de volledige duurtijd gebeurd door het one shot circuit (zie hieronder).Door middel van de envelope select logic (pin 1,28 IC2) kan men verschillende vormen selecteren,de selectie gebeurd door D5,D7 IC1

Als men de one shot logic selecteerd, is de totale duurtijd van het ruissignaal vast bepaalt door R13,C8, evenals de stijgtijd ,omdat R15 een zeer lage weerstands waarde heeft, is de verandering van de capaciteit aan pin 8 van weinig invloed.Anders is dit bij de afvaltijd,dan wordt R16 gebruikt en dan is de verandering van de capaciteit aan pin 8 wel van belang. De waarde van deze capaciteit is te veranderen door D3 (IC1),als D3 laag is wordt C6 parallel geplaatst met C7, wat resulteerd in een langdurige ruis.

Ook moet de one shot na elke cyclus opnieuw gestart worden,dit gebeurd door pin 9 'even' hoog te maken (gestuurd door D6 IC1).

Dit 'behandelde' ruissignaal wordt nu op niveau gebracht door de amplifier en zijn instel weerstanden R8,R9,R14.

Om de originele DAI-ruis zomin mogelijk te storen is er S1 toegevoegd. Deze 'solid state' schakelaar word enkel gesloten als er een van de nieuwe noise mogelijkheden aangevraagd is (gestuurd door D2 IC1).

5.2/ CASSETTE STURING.

De schakeling wordt gevormd door de volgende onderdelen:

C1,C2 ;ontkoppel condensators
R1,R2 ;instelling
P1 ;volume regeling
S2,S3,S4;solid state schakelaars

Als beide cassette motors afstaan, zijn de de punten H en I laag. Hierdoor is de stuurspanning van S4 ook laag (pin 12). S4 wordt pas geleidend als beide motors gestuurd worden, in dit geval wordt het geluid komende van de cassette recorder toegevoerd aan IC14 (zie hiervoor FIGURE 3),en meegemoduleerd op het tv kanaal (ook naar de stereo plug).

6/ VOORBEELDEN.

In dit hoofdstuk worden alle besturingen nog eens op een rijtje gezet,en tevens enkele voorbeelden gegeven.

6.1/ DE COMMANDO'S.

A) super noise generator.

```
NOISE X 0 : SNG reset,noise off.
NOISE X 1 : continuous, low noise.
NOISE X 2 : continuous, variable noise (sweep).
NOISE X 3 : continuous, high noise.
NOISE X 4 : high, short decay noise.
NOISE X 5 : high, long  decay noise.
NOISE X 6 : low , short decay noise.
NOISE X 7 : low , long  decay noise.

NOISE X 8 to NOISE X 15 : normal DAI noise.
```

RESET : NOISE OFF of NOISE 0 0 :WAIT TIME 1

B) cassette sturing.

```
-direct command: POKE#40,0     ; CAS1 + CAS2 + SOUND
                 POKE#40,#10   ; CAS2 OFF; CAS1 ON ; SOUND OFF
                 POKE#40,#20   ; CAS2 ON ; CAS1 OFF; SOUND OFF
                 POKE#40,#30   ; CAS2 OFF; CAS1 OFF; SOUND OFF
-cassette mask : POKE#13D,#10 ; CAS1 SELECTED ; SOUND OFF
                 POKE#13D,#20 ; CAS2 SELECTED ; SOUND OFF
                 POKE#13D,#30 ; CAS1 + CAS2 SELECTED ; SOUND ON
```

6.2/ PROGRAMMA VOORBEELDEN.

- STENGUN 1.

```
10   ENVELOPE 0 6,10;0,2;3,24;0,2;
20   NOISE 0 15
```

*** met de 'envelope' selecteerd men hier een lage, korte ruis gedurende 32mS gevold door 6,4mS reset.Dan een doorlopende, hoge ruis gedurende 76,8mS terug gevolgd door 6,4mS reset, waarna men terug begint.

- STENGUN 2.

```
10   0 3,13;0,6;
20   NOISE 0 15
```

- EXPLOSION.

```
10   ENVELOPE 0 7,200;0
20   NOISE 0 15
```

*** daar de 'envelope' hier niet afgesloten is met ';' zal de explosin niet herhaald worden.

- PISTOL AND RIFLE

```
10   ENVELOPE 0 4,120:0,3;6,20;0,20;6,20;0,25;
20   NOISE 0 15
```

*** merk op dat elke nieuwe noise aanvraag voorafgegaan is van een reset gedurende minstens 6,4mS.Dit is noodzakelijk daar de interpreter geen volume verandering toelaat groter dan de helft van het bestaande volume.

- FLAK-KANON.

```
10    ENVELOPE 0 6,23;0,2;5,10;0,4;
20    NOISE 0 15
```

- ????????.

```
10    ENVELOPE 0 6,3;0,2;3,9;0,2;
20    NOISE 0 15:WAIT TIME 40:NOISE OFF:WAIT TIME 1
30    ENVELOPE 0 3,2;0,2;6,6;0,2;
40    NOISE 0 15:WAIT TIME 80:NOISE OFF:WAIT TIME 1
50    GOTO 10
```

*** Het is ook mogelijk om de 'envelope' te veranderen.

- DRUM'S.

```
10    ENVELOPE 0 3,6;0,7;3,6;0,9;3,5;0,8;3,6;0,30;
20    NOISE 0 15
```

- ROCKET.

```
10    ENVELOPE 0 16:NOISE 0 2
20    FOR X=100 TO 25000 STEP 100:SOUND 2 0 0 0 FREQ(X):NEXT
30    SOUND 2 OFF:WAIT TIME 15:GOTO 20
```

*** Met behulp van het commando NOISE X 2 selecteerd men de mogelijkheid om met SOUND 2 de clock frequentie te regelen, wat resulteerd in een variabele noise.(SOUND 2 wordt dan volledig onderdrukt door de SNG)

- THE OCEAN.

```
10    ENVELOPE 0 16:NOISE 0 2
20    FOR X=5 TO 200:SOUND 2 0 0 0 X:WAIT TIME 1:NEXT
30    FOR X=200 TO 5 STEP -1:SOUND 2 0 0 0 X:WAIT TIME 1:NEXT
40    GOTO 20
```

*** Als men de functie 'FREQ' niet gebruikt dan wordt de waarde van de variabele rechtstreeks gebruikt om het sound ic te sturen.
De waarde van de variabele komt dan wel niet mee overeen met de frequentie (voor de juiste berekening zie het handboek)

!!!!!!! OOK IS HET MOGELIJK OM BIJ GEBRUIK VAN DE CASSETTE UITBREIDING TE 'KIJKEN' OF ER NOG GELUID OP DE BAND STAAT, EN DIT DOOR DE STATUS VAN DE CAS.BIT TE CHECKEN, DIE GEBRUIKT WORDT OM EEN PROGRAM TE LADEN.

Een manier is de hier onder vermelde:

```
10    B%=PEEK(#FD00) IAND #A0     ; READ BEGIN STATUS OF CAS.BIT
20    A%=PEEK(#FD00) IAND #A0     ; READ STATUS OF CAS.BIT
30    C%=C%+1                     ; INCREMENT COUNTER
40    IF A%<>B% THEN C%=0         ; WANNEER VERANDERING COUNTER=0
50    IF C%>300 THEN 70           ; EXIT PROG. WANNEER GEDURENDE ONGEVEER
                                  ;  5 sec.GEEN VERANDERING OPGETREDEN IS
60    B%=A%:GOTO 20               ; ONTHOU DE OUDE TOESTAND EN CHECK NOGMAALS
70    ????                        ; EXIT
```

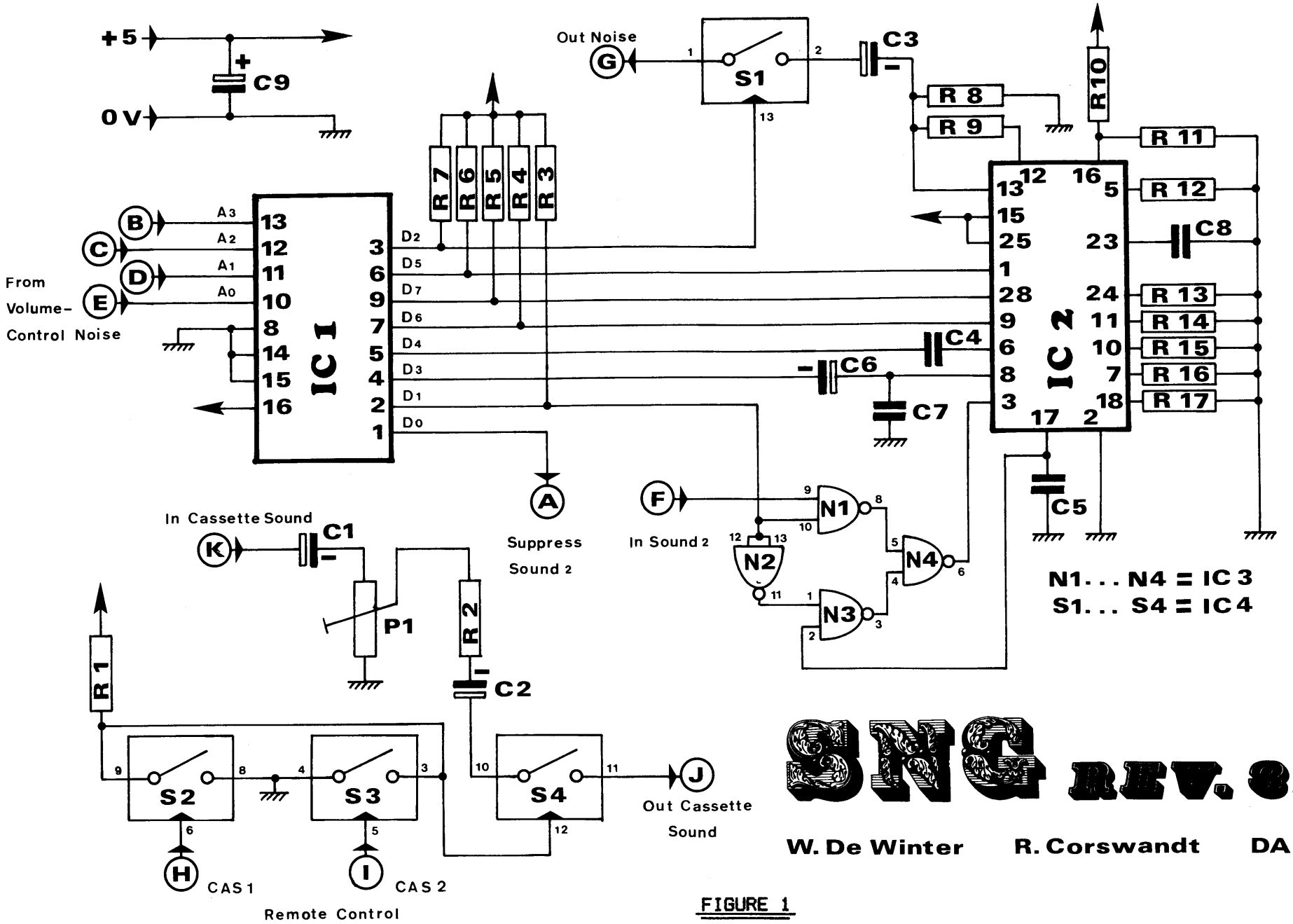
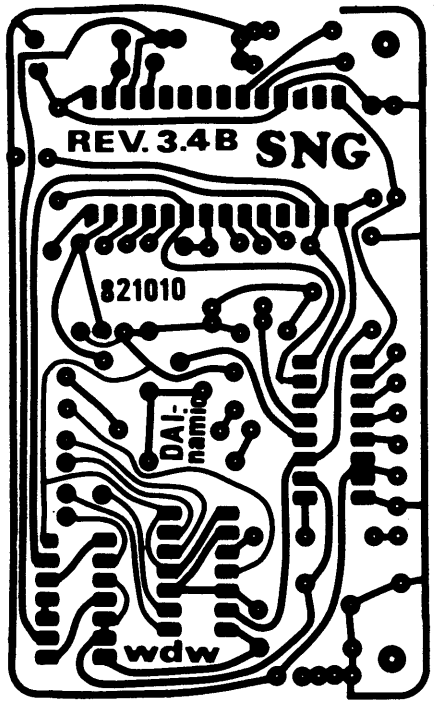


FIGURE 1

SNG REV. 8.4

W. De Winter R. Corswandt DAInamic



COPERSIDE

FIGURE 2B

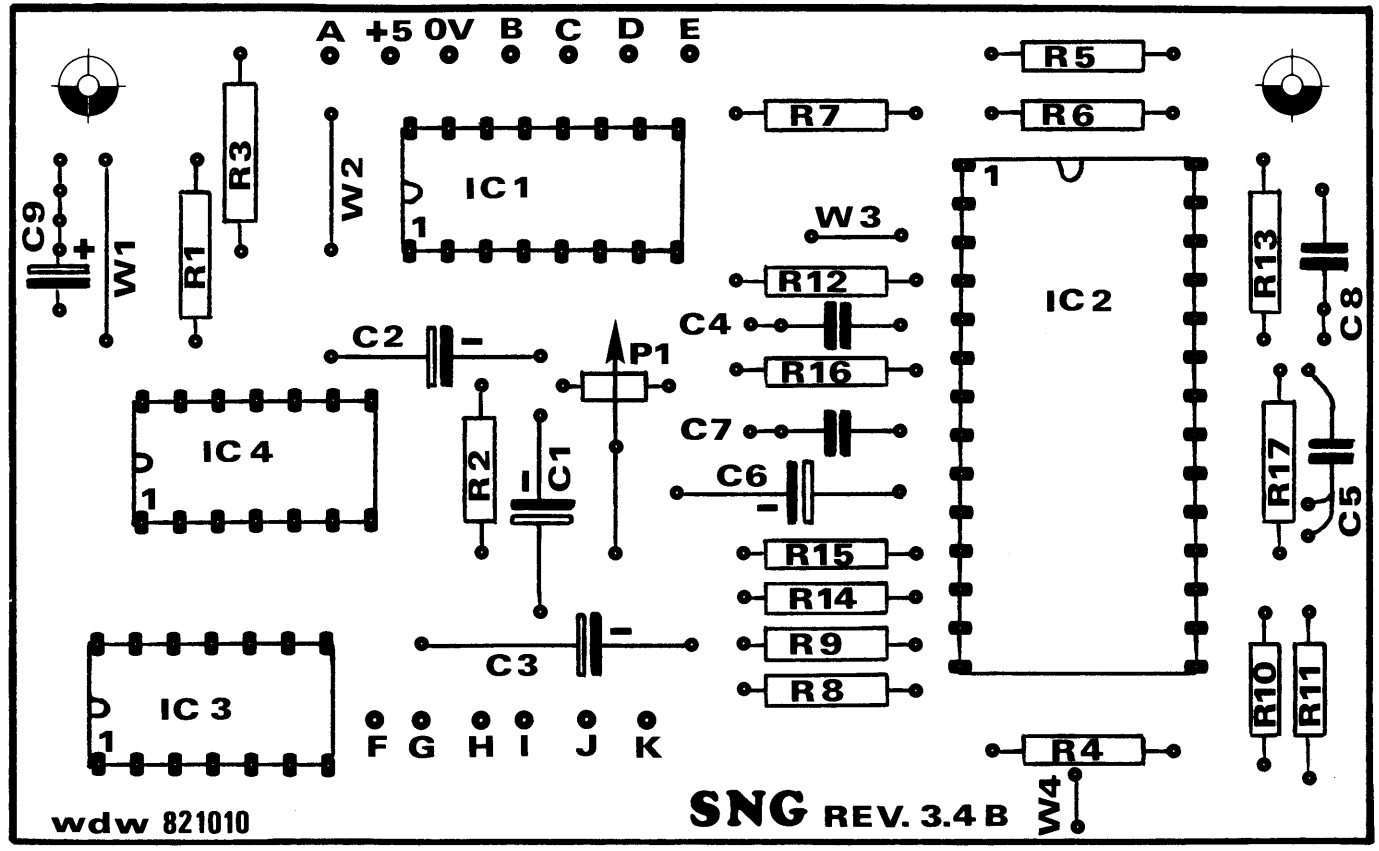
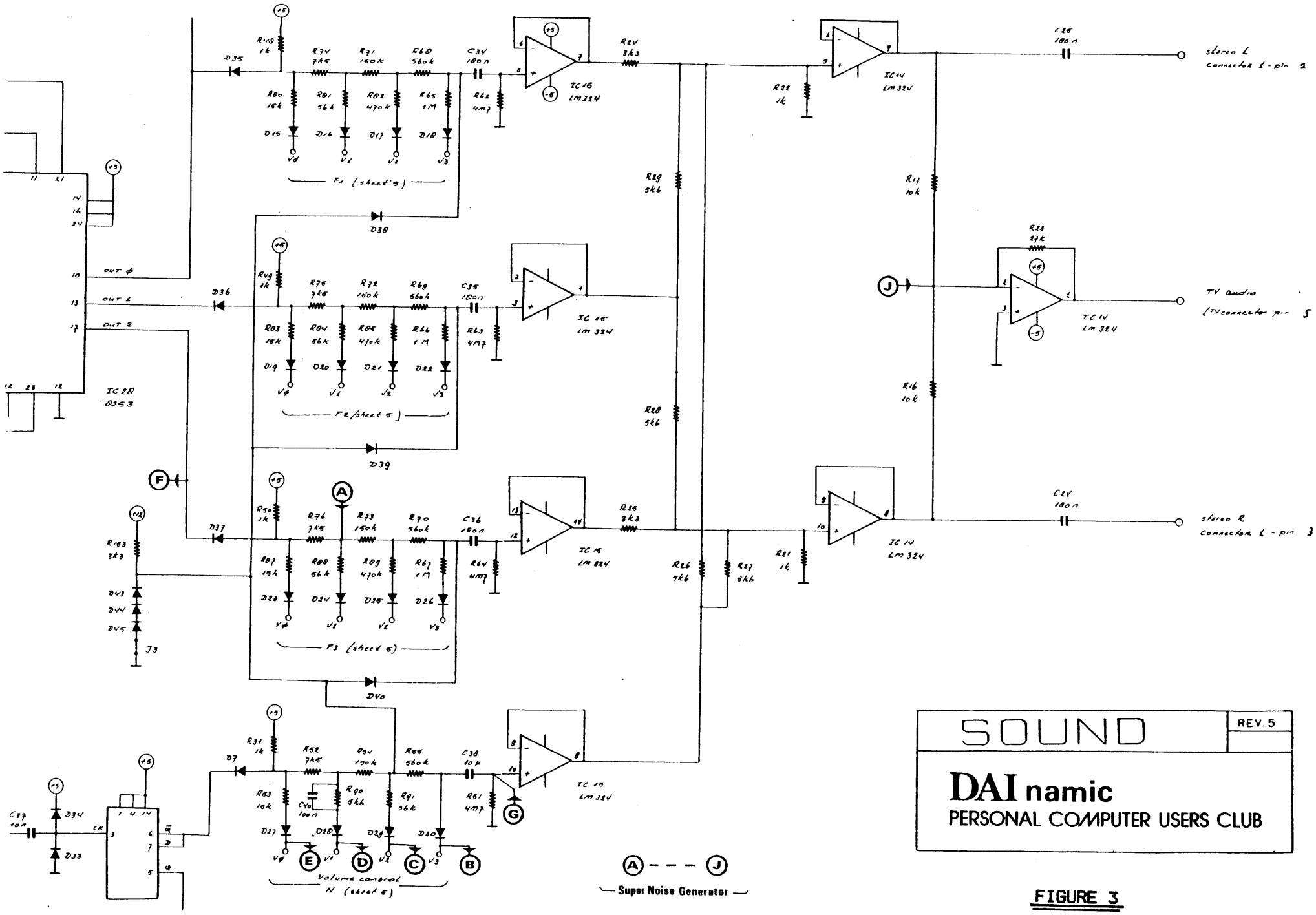


FIGURE 2A

R1/R3/R4/R5/R6/R7/R17	=	4K7	1/4 W	C4	= 1n2 MKM
R2/R8	=	10K	1/4 W	C5	= 4n7 MKM
R9	=	27K	1/4 W	C6	= 4.7u AXIAL 25V
R10	=	47K	1/4 W	C7/C8	= 470n MKM
R11	=	33K	1/4 W	C9	= 10u TANTAAL 25 V
R12	=	100K	1/4 W		
R13	=	330K	1/4 W	IC1 (PROGRAMMED FOR SNG)	= MMI-6330, 74S188 OR EQUI. (LABELED WITH 'SNG')
R14	=	150K	1/4 W		
R15	=	3K3	1/4 W	IC2	= SN76477 N
R16	=	270K	1/4 W	IC3	= 74 LS 132
P1	=	47K VERT. TRIMMER		IC4	= 4066 (MOS)
C1/C2/C3	=	4.7u A 10 u AXIAL 25V			

R.CORSWANDT / W.DE WINTER



SOUND

REV. 5

DAInamic

PERSONAL COMPUTER USERS CLUB

FIGURE 3

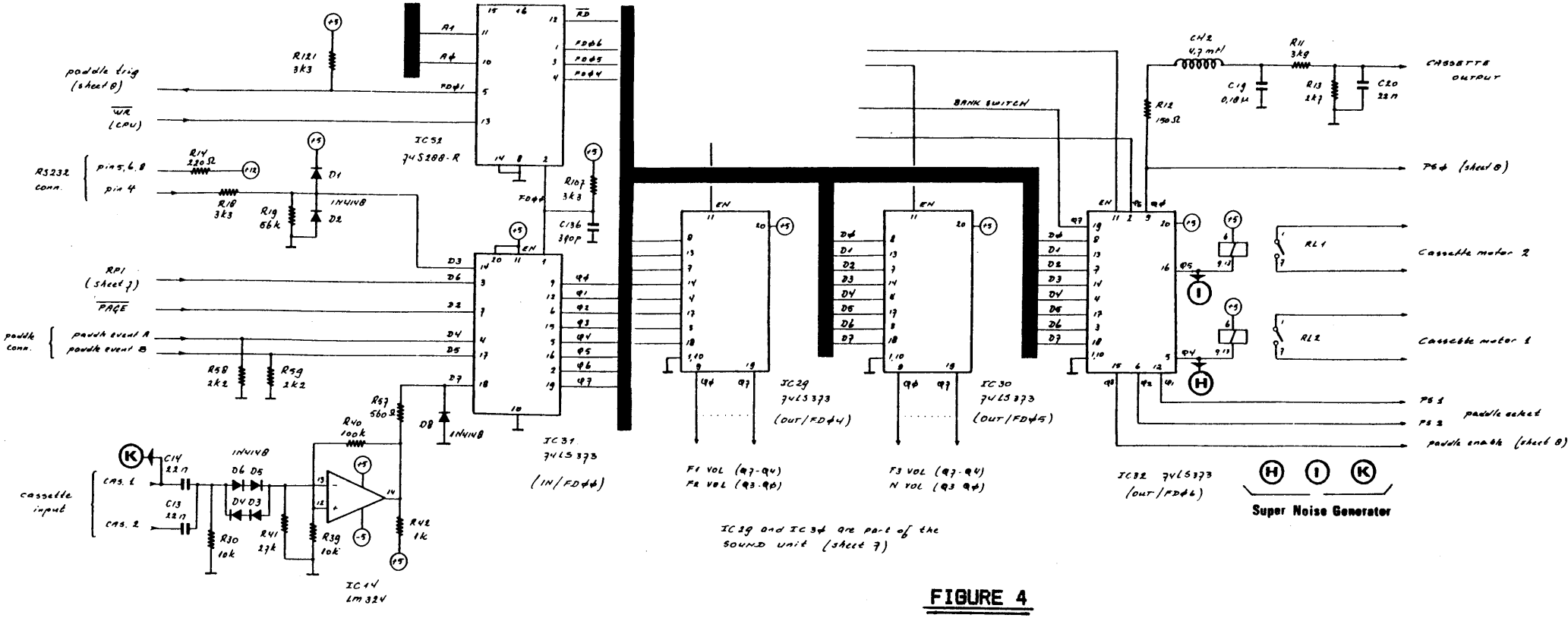
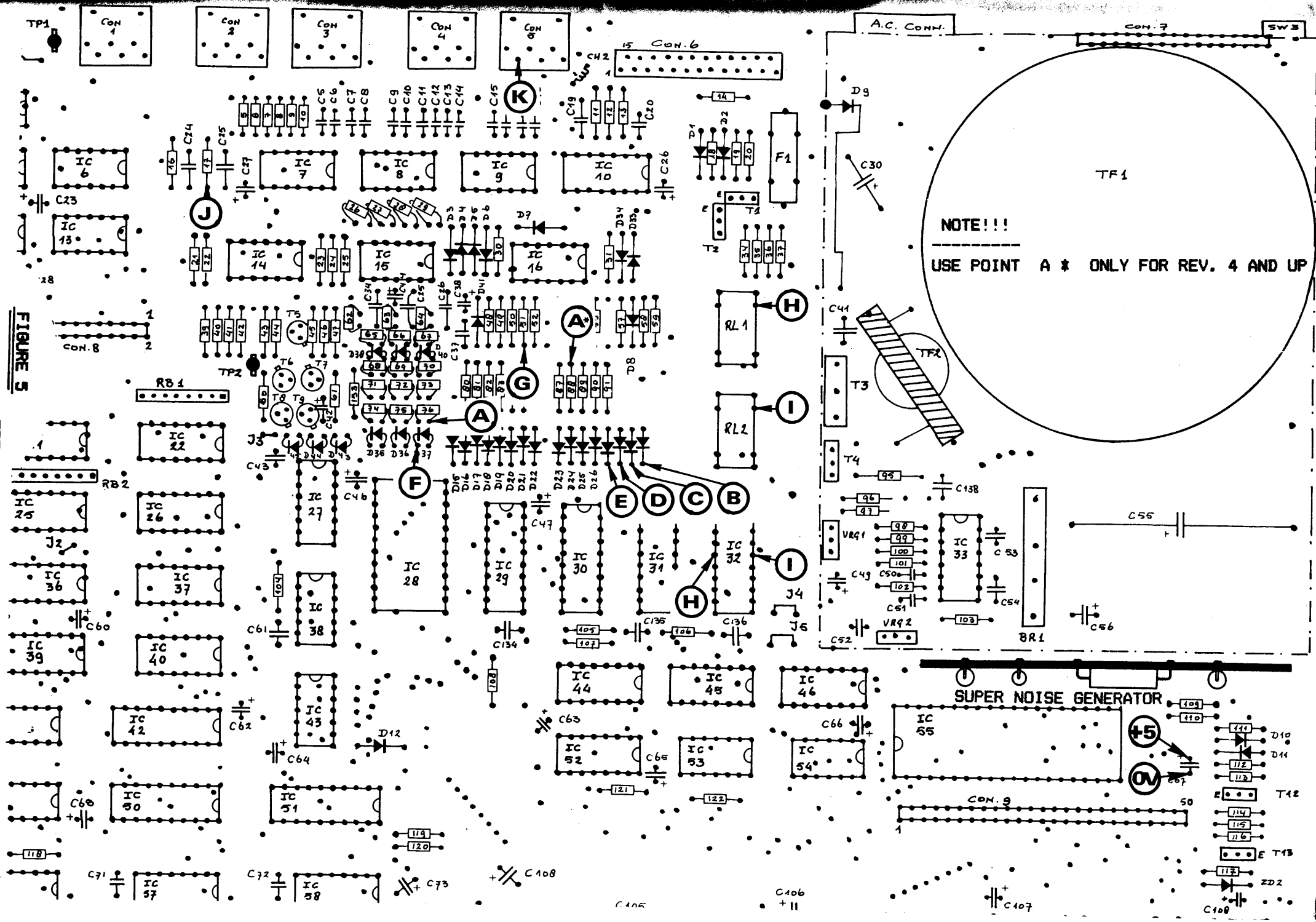


FIGURE 4

COLOR-CODE FOR WIRES

POINT	WIRE COLOR	POINT	WIRE COLOR
[A]	BROWN	[H]	GREY
[B]	RED/BLUE	[I]	WHITE
[C]	PINK	[J]	WHITE/GREEN
[D]	YELLOW	[K]	WHITE/YELLOW
[E]	GREEN	+5	RED
[F]	BLUE	0V	BLACK
[G]	VIOLET		

TABLE 1


A.C. CONN.
CON.7
SW3
TF1
NOTE!!!
USE POINT A * ONLY FOR REV. 4 AND UP
SUPER NOISE GENERATOR
CON.6
CON.8
CON.9
TP1
TP2
RB1
RB2
RL1
RL2
F1
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
ZD2
BR1
VR91
VR92
+5
0V
A
B
C
D
E
F
G
H
I
J
K
FIGURE 5
373
SNG PAGE V

SUPER NOISE GENERATOR PROM-DATA

ADRESS						DATA								
HEX	BINAIR					BINAIR								HEX
	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	0	0	0	1	1	1	1	0	0	1	79
1	0	0	0	0	1	1	0	0	0	1	1	0	1	8D
2	0	0	0	1	0	1	0	0	1	1	1	1	0	9E
3	0	0	0	1	1	1	0	0	1	1	1	0	1	9D
4	0	0	1	0	0	0	0	1	1	1	1	0	1	3D
5	0	0	1	0	1	0	0	1	1	0	1	0	1	35
6	0	0	1	1	0	0	0	1	0	1	1	0	1	2D
7	0	0	1	1	1	0	0	1	0	0	1	0	1	25
8	0	1	0	0	0	0	1	1	1	1	0	0	1	79
9	0	1	0	0	1	0	1	1	1	1	0	0	1	79
A	0	1	0	1	0	0	1	1	1	1	0	0	1	79
B	0	1	0	1	1	0	1	1	1	1	0	0	1	79
C	0	1	1	0	0	0	1	1	1	1	0	0	1	79
D	0	1	1	0	1	0	1	1	1	1	0	0	1	79
E	0	1	1	1	0	0	1	1	1	1	0	0	1	79
F	0	1	1	1	1	0	1	1	1	1	0	0	1	79

```
                                          I    I    I    I    I    I    I    I
      COMANDO'S                           I    I    I    I    I    I    I    SUPPRESS SOUND 2
     ===========                          I    I    I    I    I    I    I
NOISE X 0  = RESET, NOISE OFF             I         I    I    I    I    D1=0
NOISE X 1  = CONT. LOW NOISE              I    T    I    I    I    I    SELECT INTERN CLOCK
NOISE X 2  = VARIABLE NOISE (SWEEP)       I    R    I    I    I    I    D1=1
NOISE X 3  = CONT. HIGH NOISE             I    I    I    I    I    I    SELECT SOUND 2
NOISE X 4  = HIGH, SHORT DECAY NOISE      I    G    I    I    I    I
NOISE X 5  = HIGH, LONG  DECAY NOISE      I    G    I    I    I    I--> 1=ON NOISE
NOISE X 6  = LOW,  SHORT DECAY NOISE      I    E    I    I    I
NOISE X 7  = LOW,  LONG  DECAY NOISE      I    R    I    I    I-------> D3=1 SHORT NOISE
                                          I         I    I              D3=0 LONG  NOISE
NOISE X 8  = NORMAL DAI NOISE             I         I    I
  "    "      "     "    "                I         I    I------------> D4=1 LOW   NOISE
  "    "      "     "    "                I         I                   D4=0 HIGH  NOISE
  "    "      "     "    "                I         I
NOISE X 15 = NORMAL DAI NOISE             I-------->I-----------------> ENVELOPE SELECT
                                                                        0/0  VCO/AM
                                                                        1/0  NO MODULATION
WILLY DE WINTER   SEPTEMBER 1982          SNG REV 3.4                   0/1  ONE-SCHOT
                                                                        1/1  VCO/PRODUCT
```

TABLE 2

THIS TABLE GIVES THE RESULT OF ALL THE COMBINATIONS OF VOLUME SELECTS, WITH ENVELOPE AND THE SOUND OF NOISE STATEMENTS

RESULT	ENV	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NOISE	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NOISE	X	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
NOISE	X	2	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2
NOISE	X	3	0	0	0	1	1	1	1	2	2	2	2	2	2	3	3	3
NOISE	X	4	0	0	1	1	1	1	2	2	2	3	3	3	3	4	4	4
NOISE	X	5	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5
NOISE	X	6	0	0	1	1	2	2	2	3	3	4	4	5	5	5	6	6
NOISE	X	7	0	0	1	1	2	2	3	3	4	4	4	5	6	6	7	7
NOISE	X	8	0	1	1	2	2	3	3	4	4	5	6	6	7	7	8	8
NOISE	X	9	0	1	1	2	2	3	4	5	5	6	6	7	8	8	9	9
NOISE	X	10	0	1	1	2	3	3	4	5	6	6	7	8	8	9	10	10
NOISE	X	11	0	1	1	2	3	3	4	5	6	7	8	8	9	10	11	11
NOISE	X	12	0	1	2	2	3	4	5	5	6	7	8	9	10	11	11	12
NOISE	X	13	0	1	2	2	3	4	5	6	7	7	8	9	10	11	12	13
NOISE	X	14	0	1	2	3	4	4	5	6	7	8	9	10	11	11	12	13
NOISE	X	15	0	1	2	3	4	5	6	7	8	8	9	10	11	12	13	14

NOTE :

- Read Your handbook, chapter 'PROGRAMMABLE SOUND FACILITY' (page 89...). Attention !!!, the maximum number for volume in the ENVELOPE statement is not 15 but 16 (see table above).

- X is one of the two posible ENVELOPE numbers (0 or 1).

TABLE 3

TECHNICAL DATA

AN EXCLUSIVE RADIO SHACK SERVICE TO THE EXPERIMENTER

SN76477N COMPLEX SOUND GENERATOR

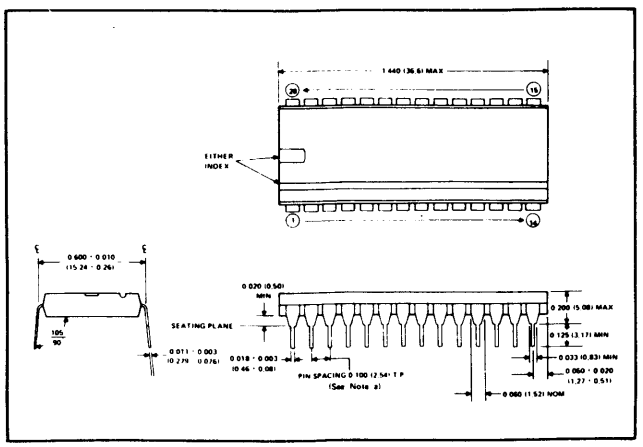
DESCRIPTION:

The SN76477N Complex Sound Generator is a linear/I^2L device which provides noise-, tone- or low-frequency - (or a combination of these) based complex sounds. Programming is via external components, (user-selected), which allows a wide variety of sounds to be created. The SN76477N is designed for ultimate flexibility in user-defined sounds, and may be used in any application requiring audio feedback to the operator (i.e. arcade/home video games, pinball games, toys, etc.; consumer oriented equipment, such as timers, alarms, controls, etc.; industrial equipment for indicators, alarms, feedback controls, etc.).

FEATURES

- Generates Noise, Tone or Low-Frequency-Based Sounds, or Combination of These
- Allows Custom Sounds to be Created Easily
- Low Power Requirements
- Allows Multiple-Sound Systems
- Compatible With Microprocessor Systems

OUTLINE DIMENSIONS



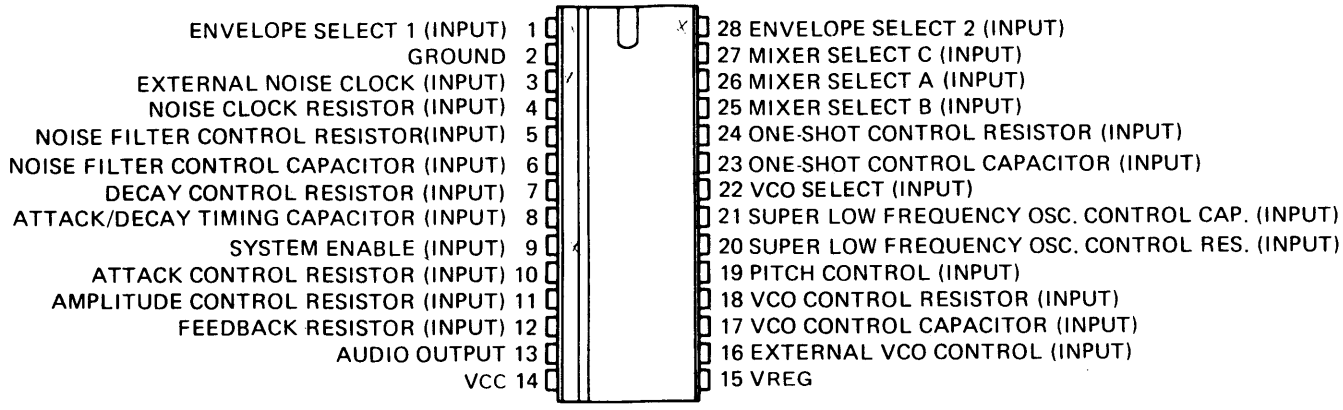
ABSOLUTE MAXIMUM RATINGS AT T_A = 25°C (UNLESS OTHERWISE SPECIFIED)

Supply Voltage, V_{REG}, Pin 15 . . . . . 6.0V
Supply Voltage, V_{CC}, Pin 14 . . . . . 12.0V
Input Voltage Applied to any Device Terminal 6.0V
Operating Temperature Range . . . . . −55°C to +120°C
Lead Temperature 1/16 Inch From Case For 10 Seconds . . . +260°C

RECOMMENDED OPERATING CONDITIONS

	MIN.	TYP	MAX	UNITS
Supply Voltage, V_{REG}, Pin 15	4.5	5.0	5.5	V
Supply Voltage, V_{CC}, Pin 14	7.5		9.0	V
Operating Free-Air Temperature	0	25	70	°C

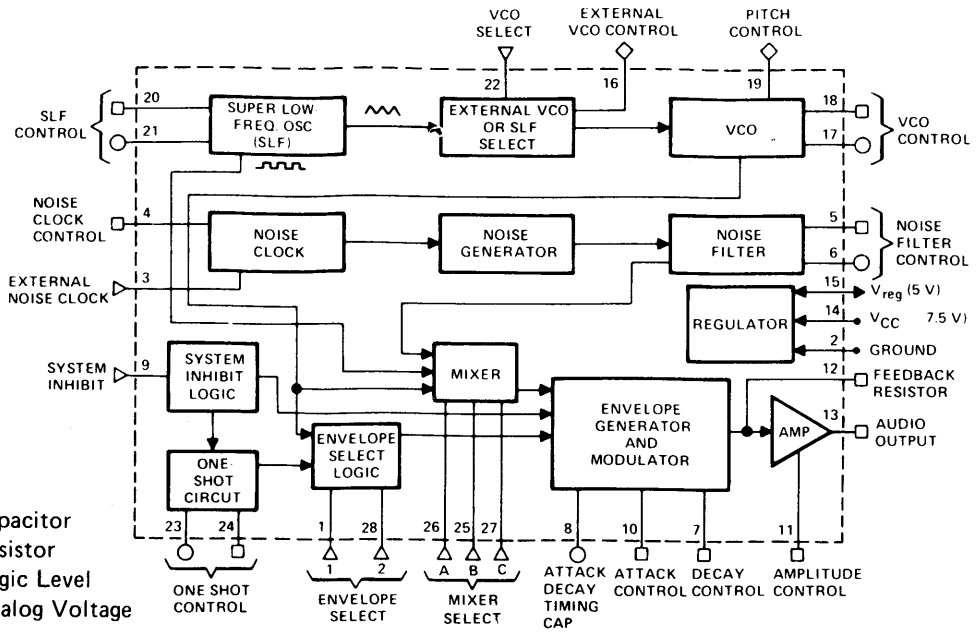
DUAL-IN-LINE PACKAGE (TOP VIEW)



OPERATING CHARACTERISTICS AT T_A = 25°C AND V_{REG} = 5.0V

PARAMETER	PIN	CONDITIONS	MIN	TYP	MAX	UNITS
ICC	14	VREG = 5.0V; NO EXT. LOAD		15	40	MA
VREG	15	VCC = 8.25V; I_{LOAD} = 10MA	4.5		5.5	V
INPUT REGULATION	15	I_{LOAD} = 10MA VCC = 7.5V TO 9.0V		150		MV
CONTROL INPUT CURRENT RANGE			1		200	uA
NOISE CLOCK	4					
NOISE FILTER	5					
DECAY	7					
ATTACK	10					
AMPLITUDE	11					
VCO	18					
ONE SHOT	24					
LOGICAL "1" INPUT CURRENT						
ENVELOPE SELECT 1 & 2	1, 28	@ 2.0V		40	52	uA
MIXER SELECT A, B, C	25, 26, 27	@ 2.0V		40	52	uA
VCO SELECT	22	@ 2.0V		40	52	uA
EXTERNAL NOISE	3	@ 2.0V		40	52	uA
SYSTEM ENABLE	9	@ 2.0V			100	uA
LOGICAL "1" INPUT VOLTAGE			2.0			V
ENVELOPE SELECT 1 & 2	1, 28					
MIXER SELECT A, B, C	25, 26 27					
VCO SELECT	22					
EXTERNAL NOISE	3					
SYSTEM ENABLE	9					
LOGICAL "0" INPUT VOLTAGE					0.8	V
ENVELOPE SELECT 1 & 2	1, 28					
MIXER SELECT A, B, C	25, 26 27					
VCO SELECT	22					
EXTERNAL NOISE	3					
SYSTEM ENABLE	9					
EXTERNAL VCO CUTOFF	16		2.5			V
TRIP POINTS				2.5		V
ONE-SHOT CAP	23					
VCO CAP	17					
NOISE FILTER CAP	6					
SLF CAP	21					
MAXIMUM PEAK-TO-PEAK OUTPUT VOLTAGE SWING	13	R_{LOAD} = 10K R_{FDBK} = 10K I_{11} = 200uA	2.5			V
DYNAMIC OUTPUT IMPEDANCE	13			100		OHMS

BLOCK DIAGRAM:



○Denotes Programming Via Capacitor
□Denotes Programming Via Resistor
△Denotes Programming Via Logic Level
◇Denotes Programming Via Analog Voltage

OPERATION

1. SLF (SUPER LOW FREQUENCY OSCILLATOR)

The SLF is normally operated in the range of 0.1 – 30 Hz, but will operate up to 20 kHz. The frequency is determined by the SLF control resistor (Pin 20) and capacitor (pin 21) according to the following equation:

$$\text{SLF Frequency (Hz)} \approx \frac{0.64}{R_{SLF}\, C_{SLF}} \qquad @\ V_{REG} = 5.0V$$

Equation 1: SLF Frequency Equation

The SLF feeds a 50% duty cycle square wave to the "mixer"; it also feeds a triangular wave to the "ext. VCO/SLF Select" logic, which is fed through to control the VCO when Pin 22 is high (see further explanation below).

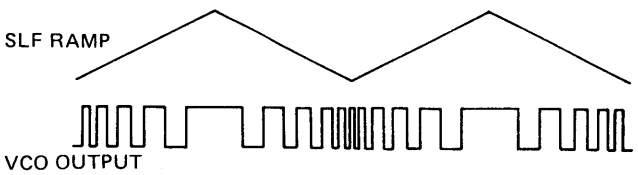
2. VCO (VOLTAGE CONTROLLED OSCILLATOR)

The VCO circuitry produces a tone output whose frequency is dependent upon the voltage at the input of the VCO. The higher Pin 16 voltage is, the lower the frequency. The controlling voltage may be either the SLF output, or it may be an externally applied signal on Pin 16. The selection of control modes (external – Pin 16; internal – SLF) is via the binary logic level on Pin 22, VCO Select, according to the following table:

Pin 22	Control Mode
0	External (Pin 16)
1	Internal (SLF)

Table 1: VCO Control Mode Selection

The input at the External VCO Control, Pin 16, may be a DC voltage, (producing a constant tone at the output of the VCO), or any waveform, producing a frequency modulated output from the VCO. A frequency modulated waveform also results when the SLF ramp controls the VCO (Pin 22 = high), as shown below:



VCO FM output

An alternate method to apply an external voltage to the VCO input is to place the controlling voltage on the SLF Control Capacitor Pin (Pin 21). In some applications this may be more convenient than using the Pin 16 input. The frequency Range of the VCO is internally determined at an approximate ratio of 10:1. The minimum frequency of the VCO may be determined by adjusting the RC time-contant of the VCO Control Resistor (Pin 18) and the VCO Control Capacitor (Pin 17), according to the following equation:

$$\text{Min VCO Freq. (Hz)} \approx \frac{0.64}{R_{VCO}\, C_{VCO}} \qquad @\ V_{REG} = 5.0V$$

Equation 2: Minimum VCO Frequency

The Pitch Control (Pin 19) varies the duty cycle of the VCO output according to the following equation:

$$\text{VCO Duty Cycle} \approx 50 \times \frac{\text{Voltage at Pin 16}}{\text{Voltage at Pin 19}} \%$$

Equation 3: Pitch Control of VCO Duty Cycle

By leaving Pin 19 high, a constant 50% duty cycle may be achieved. The specific % duty cycle, applies to constant tones produced by applying a constant DC voltage at the External VCO Control Pin (Pin 16). However, the Pitch Control may still be used to aesthetically alter the pitch of any frequency-modulated VCO output signals.

3. NOISE CLOCK

The Noise Clock clocks the Noise Generator. This circuit requires a 43K resistor to ground at Pin 4 to set an internal current level. An external noise clock may be supplied at Pin 3 to allow generation of lower frequency noise. This external clock should be a maximum 5 volt peak-to-peak square wave.

4. NOISE GENERATOR/FILTER

The Noise Generator is a binary psuedo random white noise generator whose output passes through the Noise Filter before being inputed to the mixer. The filter is a variable band width low-pass filter whose 3dB point is defined by the following equation:

$$\text{3dB Frequency (Hz)} \approx \frac{1.28}{R_{NF}\, C_{NF}} \qquad @\ V_{REG} = 5.0V$$

Equation 4: Noise Waveform 3 dB Frequency

5. The Mixer Logic selects one, (or a combination), of the inputs from the generators and feeds the output to the Envelope Generator and Modulator.

Mixer Select			Mixer Output
C (Pin 27)	B (Pin 25)	A (Pin 26)	
0	0	0	VCO
0	0	1	SLF
0	1	0	Noise
0	1	1	VCO/Noise
1	0	0	SLF/Noise
1	0	1	SLF/VCO/Noise
1	1	0	SLF/VCO
1	1	1	Inhibit

Table 2: Mixer Select Logic

6. SYSTEM ENABLE LOGIC

The System Enable Logic provides an enable/inhibit for the system output. The sound output is controlled according to the following table:

Pin 9	Output
0	Enabled
1	Inhibited

Table 3: System Enable Logic

This input also triggers the "one-shot" logic for momentary sounds, such as gunshots, bells and/or explosions. The "one-shot" logic is triggered by the negative-going edge. This may be accomplished by a momentary switch, or by a square wave input at Pin 9. Pin 9 must be held low for the entire duration of the one-shot sound (including attack and decay period). The one-shot logic is operable only when the proper Envelope Select Logic selection is made. (see Envelope Select Logic).

7. "ONE-SHOT" LOGIC

The duration of the "one-shot" is defined by the following equation:

$$\text{Duration (seconds)} \approx 0.8\, R_{OS}\, C_{OS} \qquad @\ V_{REG} = 5.0V$$

Equation 5: One-Shot Duration

cont. on page 324

SOUVENIR MIKROBEL 82

PRETTIGE FEESTEN !