Interfacing COMPUKIT

Part 5 D. E. Graham

THIS month we will be looking at the Programmable Sound Generator on the Analogue Board. A separate page is also included which elaborates on the use of SK4 and SK5 of the Decoding Module.

THE AY-3-8910 PROGRAMMABLE SOUND GENERATOR

Fig. 5.1 gives the circuit of the audio section of the Analogue Board. This consists of a General Instrument AY-3-8910 Programmable Sound Generator feeding an LM386 audio amplifier which produces 200mW into an 8 Ohm load. The audio circuitry is based on a design from G.I.'s data manual.

The 8910 is a sixteen register device containing three independent tone generators, a variable pitch white noise generator, mixers and an envelope controller. Details of the operation of the very similar 8912 were given in the *September 1980* issue of *P.E.*, and for this reason a full device description will not be given here. For additional information on the device the reader is referred both to that article, to G.I.'s data manual on the chip, and to the article *Micro Sounds* in *Personal Computer World, October 1980.*

Although the 8910 is a sixteen register device, it is intended for CPUs which use a shared address and data bus, and for this reason cannot be used directly with 6502 based machines. This difficulty may, however, be circumvented by using the data bus to carry data which the 8910 can use either as true data or as an address, depending on the state of its control lines (which are designed to inform it of such things). One has then simply to set up the correct configuration on its control lines before furnishing it with appropriate data on its shared bus.

This task is performed by IC8c and IC8d on the Decoding Module. The required configuration of the two control lines (coded BDIR and BC1) is given in Table 5.1. From this it may be seen that to select any one of the P.S.G.'s 16 internal registers, both BDIR and BC1 must go high, and at the same time the number of the register required must be placed on the data bus by the CPU for the P.S.G. to read. Once the operation is completed, both BDIR and BC1 must be taken low. If it is then required to write data into the particular register just called up, BDIR must be taken high. Any data on the data bus will then be written into the given P.S.G. register.

The two signals BDIR and BC1 are produced by the Decoding Module in response to the R6, W6 and W7 lines, as may be seen from the circuit of Fig 5.2. This is organised so that reading and writing to the P.S.G. is performed in the following way. To place the value 100 (decimal) into P.S.G. register 4, one simply executes:

POKE 61317, 4
POKE 61318, 100

The first command calls up register 4. The second places the data into it. To read the contents of register 7, execute:

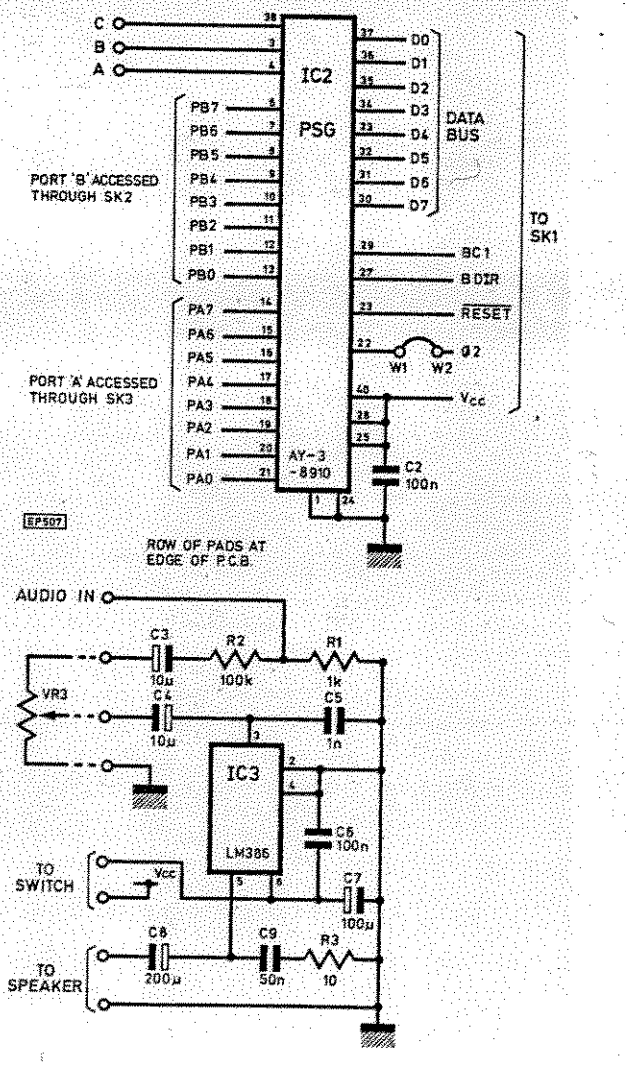
POKE 61317, 7
PRINT PEEK (61318)

This calls up register 7, then reads from it.

TESTING THE PSG

When setting up the PSG for the first time, connect the pads as shown in Fig 5.3, and switch on S1. Check that the audio amplifier is working by setting the volume control VR3

Fig. 5.1. PSG section of Analogue Board. W1 and W2 allow alternative clock drive (1–2MHz) if desired.



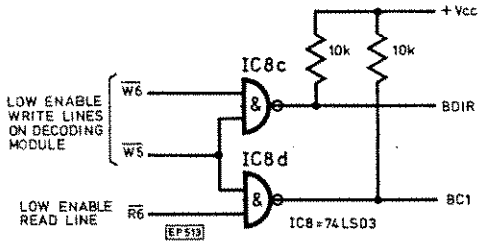


Fig. 5.2. PSG decoding circuitry on Decoding Module Board.

Table 5.1. PSG Control line functions

Function	BDIR	BC1
Inactive	Ø	Ø
CPU read from PSG	Ø	1
CPU write to PSG	1	Ø
Latch address of PSG register	1	1

Fig. 5.3. Connection of PSG pads for normal operation.

Fig. 5.4. Connection of PSG pads for normal operation on channel C, with analogue output on channels A and B.

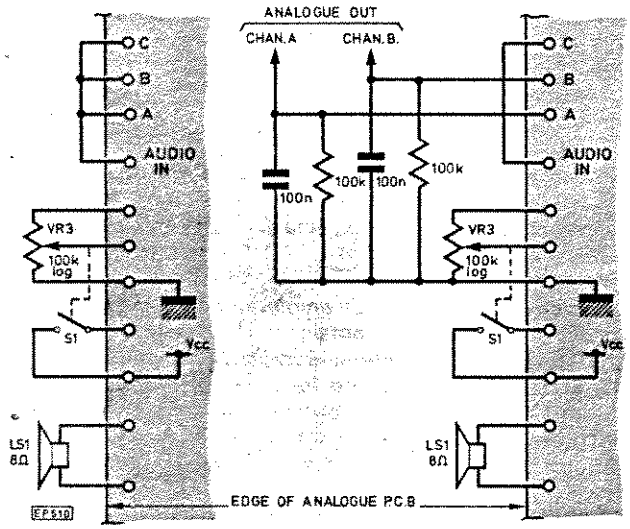


Table 5.2. Manual operation of Programmable Sound Generator.

```
50 REM INTERFACING UK101 PROGRAM 9
80 REM AY-3-8910 MANUAL ENTRY PROGRAM
100 FORS=ITO16:PRINT:NEXT
110 PRINT,"AY-3-8910 POKE/PEEK ROUTINE"
120 PRINT,"USING ADDRESSES 61317 & 61318
124 PRINT,"(NOTE DATA > 255 CAUSES A READ)"
125 PRINT:PRINT
130 A=61317:D=61318
135 REM*********************************
140 REM ZERO ALL REGISTERS
150 FORX=0TO15
160 POKEA,X
170 POKED,0
180 NEXT
190 REM*********************************
200 REM POKE/PEEK ROUTINE
210 INPUT"    REGISTER";A1
220 POKEA,A1
230 INPUT"    DATA";D1
240 IFD1>255THEN300
250 POKED,D1
260 GOTO210
300 PRINT,"CONTENTS OF REG";A1;" :   ";PEEK(D)
310 GOTO210
```

to full volume, and touching the wiper pin. If this produces hum, then run the program shown in Table 5.2. This has been written for manually controlling the contents of the P.S.G.'s registers. It first zeroes all registers, and then requests a register number. When this is entered, it requests the data that is to be put into it (any integer form 0 to 255). If a number greater than 255 is entered, the program will read from the register instead of writing to it, and print out the results.

Run the program and use it to examine the contents of the P.S.G.'s registers: all should be at zero. Data may then be placed in registers 0 to 13, and a check made to verify that the data has been properly stored. Remember here however, that certain registers (1, 3, 5, 6, 8, 9, 10 and 13) are only 4 or 5 bits wide, and attempting to place the value 255 say, into register 1 will cause the value 15 to be stored in that 4-bit register.

If registers 0 to 13 do not read or write correctly, first check the status of all lines to the P.S.G. (supply, Ø2, etc.). Next check the functioning of the control lines BDIR and BC1 with a 74LS75 latch (or similar) in the manner indicated in part 2 of the series. Their states should be checked against those given in Table 5.1. If these are incorrect, then the decoding circuitry should be checked. If they are functioning correctly, then a way should be devised of testing the P.S.G. chip itself. This could be done using the PIA on the Decoding Module, with one port connected to the 8 data/address lines of the P.S.G. and two lines of the other port supplying the BDIR and BC1 signals.

Once the registers read and write correctly, it should be possible to produce some sounds. The fastest way to make noise is to enter 15 into register 8 after resetting the P.S.G. This should produce white noise whose colour may be altered by placing data into the lowest 5 bits of register 6. To stop the noise, place 254 into register 7. If this is followed by placing 100 into register 0, a pure tone should be produced whose note may be varied by changing the data in registers 0 and 1.

If audio is not forthcoming, but the registers read and write correctly, then the audio circuitry should be checked, and pins 4, 3 and 38 of the P.S.G. examined for audio output.

THE P.S.G.'s REGISTERS

In order to produce more subtle sounds, it is necessary to be acquainted with the functions of the P.S.G.'s full set of registers. These are represented diagrammatically in Table 5.3.

Perhaps the most important register is number 7, the master enable. This is organised, somewhat inconveniently, to be active-low, so that placing a zero into it enables all tone and noise channels, 255 silences them all, while 254 enables tone on channel A only, etc. Note frequencies must then be set using the first six registers (two for each channel); and then amplitudes (registers 8, 9 and 10 for channels A, B and C respectively). Data from 0 to 15 in these registers produces differing amplitudes, while data of 16 in any of the three registers puts the associated channel over to envelope control.

Registers 11 and 12 control the time period of the envelope, while 13 controls, its shape, and whether it is repeating or not. For more precise details of this register the reader is referred to one of the three works cited earlier, but in Table 5.4 we give a selection of useful data for this register.

Registers 14 and 15 are two 8 bit ports, whose function is controlled by the top two bits of register 7 (0 for input, 1 for output). The connections to these two ports have been taken

out to the 16 pin d.i.l. sockets SK2 and 3. See Table 5.5 for connections. It will be seen that they have been made similar to those for the PIA on the Decoding Module, although it should be noted that the two sockets face the opposite direction to those on the Decoding Module.

Register	Function	BIT							
		7	6	5	4	3	2	1	0
R0	Channel A tone period	8-bit fine tune A							
R1					4-bit coarse tune A				
R2	Channel B tone period	8-bit fine tune B							
R3					4-bit coarse tune B				
R4	Channel C tone period	8-bit fine tune C							
R5					4-bit coarse tune C				
R6	Noise period				5-bit period control				
R7	Enable	In/out		Noise			Tone		
		I/O B	I/O A	C	B	A	C	B	A
R8	Channel A amplitude			Env en	4-bit amplitude				
R9	Channel B amplitude			Env en	4-bit amplitude				
R10	Channel C amplitude			Env en	4-bit amplitude				
R11	Envelope period	8-bit fine tune							
R12		8-bit coarse tune							
R13	Envelope profile					4-bit control			
R14	I/O port A	8-bit parallel port							
R15	I/O port B	8-bit parallel port							

Table 5.3. Programmable Sound Generator registers.

Table 5.4. Selected functions of register 13 of PSG —envelope control

Data	Function
Ø	Non repeating, fast attack, slow decay
4	Non repeating, slow attack, fast decay
8	Repeating, fast attack, slow decay
12	Repeating, slow attack, fast decay
14	Repeating, slow attack, slow decay

Table 5.5. Connections to SK2 and SK3
SK3 = Port A of PSG
SK2 = Port B of PSG

Pin	Function	Pin	Function
1	GND	9	P7
2	NC	10	P6
3	NC	11	P5
4	GND	12	P4
5	GND	13	P3
6	GND	14	P2
7	NC	15	P1
8	VCC	16	P0

MAKING SOUNDS

We now give one or two sequences that may be entered using the program in Table 5.2 to produce some simple sounds. The short sequence below will produce a continuous note:

7—248
8—15
0—100

The number to the left is the register number, that to the right, the data to be placed in it. The first line sets register 7 to give tone output on all three channels, the second sets maximum volume on channel A, while the third selects a note. Entering alternative values into register 0 will change the note, and placing data in register 1 will considerably lower its pitch. If this sequence is followed by:

9—15
2—102

the note will change in timbre, becoming richer as the results of the beating of notes from channels A and B.

It is also possible to place this composite tone under envelope control. The following sequence achieves this:

8—16
9—16
12—30

The first two entries place channels A and B under envelope control, while the third determines the envelope period. Subsequently placing zero into register 13 (even if it already contains zero) will produce a one-shot decaying chime, which may be repeated by placing further zeros into 13. Alternatively, placing an 8 into 13 will produce a repeating chime or electronic piano note. This note may be further enriched by adding sounds from channel C.

For a different effect, if the following data is entered after the P.S.G. has been reset (either by rerunning the program, or by pressing the Reset button on the Decoding Module), the sound of an explosion will be produced.

6—31
7—7
8—16
9—16
10—16
12—20
13—0

Further entries of 13—0 will repeat the effect, while entering 13—8 will cause continuous repetition. Register 12 determines the decay rate, and 6 the colour of the white noise.

The program in Table 5.2 is useful for testing the P.S.G. and for experimenting with simple sound effects, but a richer variety of effects can be obtained with a little practice when the P.S.G.'s registers are altered under program control using FOR loops to vary the frequencies, timbres, and amplitudes of notes. To give an example of this, the program listed in Table 5.6 produces a two part effect by varying the frequency of channel A in different ways.

KEYBOARD ORGAN

As an example of a somewhat different usage, the program listing in Table 5.7, which should just squeeze into a 4K machine, is for a 14 note organ operated from the Compukit keyboard. When the program is run, a four part menu appears on the screen:

KEYBOARD ORGAN PROGRAM
CONTENTS
1. To play press P (or GOTO 2000).
2. To hear press H (or GOTO 4000).
3. To record press R (or GOTO 5000).
4. To load press L (or GOTO 6000).
NOTE: Space Bar Exits Routines

If P is pressed, the program requests a note period (try around 30 for this) and then a further integer which determines the difference between the two frequencies used for the organ note. The organ may then be played using keyboard letters W–I and S–K. Once 100 notes have been played, or if the space bar is pressed, the program will exit this routine, and return to the menu.

Table 5.6. Dynamic sound effects.

```
70 REM INTERFACING UK101 PROGRAM 10
80 REM SOUND EFFECTS PRODUCED BY
85 REM VARYING DATA IN PSG REGISTER A
90 REM INITIALISATION********************
100 J=61317
110 K=J+1
120 POKEJ,7:POKEK,248
130 POKEJ,8:POKEK,15
140 POKEJ,0
200 REM FIRST EFFECT*********************
220 FORA=0TO128
230 POKEK,A-64*INT(A/64)
240 POKEK,128-A
260 POKEK,A
270 NEXT
290 POKEK,0
300 FORZ=1TO100:NEXT
380 REM SECOND EFFECT********************
390 FORB=1TO10
400 FORA=0TO255STEP10
420 POKEK,A
440 NEXT
450 NEXT
500 POKEK,0
600 INPUT"  AGAIN";YY$
610 IFLEFT$(YY$,1)="Y"THEN200
```

Table 5.7. Keyboard organ using the UK101 and its QWERTY keyboard.

```
20 REM INTERFACING UK101 PROGRAM 11
30 REM AY-3-8910 KEYBOARD ORGAN
50 QA=61808
55 QD=QA+1
60 K=57088
65 DIML(100)
100 PRINT:PRINT:PRINT:PRINT:PRINT,"KEYBOARD ORGAN PROGRAM"
110 PRINT:PRINT," CONTENTS"
120 PRINT:PRINT:PRINT"  1. To play press P (or GOTO 2000)."
130 PRINT:PRINT"  2. To hear press H (or GOTO 4000)."
140 PRINT:PRINT"  3. To record press R (or GOTO 5000)."
150 PRINT:PRINT"  4. To load press L (or GOTO 6000)."
155 PRINT:PRINT"     NOTE  Space Bar Exits Routines"
160 INPUTY$
170 IFY$="P"THEN2000
180 IFY$="H"THEN4000
190 IFY$="R"THEN5000
200 IFY$="L"THEN6000
220 PRINT,"NOT RECOGNISED"
225 PRINT,"ENTER AGAIN PLEASE"
230 GOTO160
2000 REM ORGAN
2003 GOSUB8000
2007 PRINT:PRINT:PRINT
2020 Z=0
2021 INPUT"  TONE QUALITY 0,1,2 ETC";XX
2022 PRINT:PRINT,"KEYBOARD READY":PRINT:PRINT:PRINT
2030 POKE530,0
2040 POKE530,1
2050 POKEK,247
2060 P=PEEK(K)
2080 FORA=9TO15
2090 READB
2100 IFB=PTHENL=A:A=20
2110 NEXT
2120 RESTORE
2122 IFA>19THEN2220
2125 POKEK,239
2130 P=PEEK(K)
2135 FORA=2TO8
2140 READB
2145 IFB=PTHENL=A:A=20
2150 NEXT
2155 RESTORE
2220 POKEK,253
2230 IFPEEK(K)=239THEN3600
2240 IFA<19THEN2030
3000 POKEQA,2
3005 POKEQD,L*5+40
3010 POKEQA,0
3015 POKEQD,L*10+80+XX
3020 POKEQA,13
3030 POKEQD,0
3040 IFL=L1THEN2050
3045 L(Z)=L:L1=L
3050 Z=Z+1
3200 IFZ<101THEN2030
3500 DATA127,191,223,239,247,251,253
3600 POKEQD,0
3602 S=Z-1
3605 POKE530,0
3610 GOTO100
3990 REM AUTO REPLAY ROUTINE
4000 PRINT:PRINT:PRINT:PRINT:PRINT
4001 GOSUB8000
4003 POKEQA,2
4005 PRINT,"AUTO REPLAY"
4010 PRINT,"SEQUENCE LENGTH ";S
4012 PRINT,"FILE NAME  ";FL$
4015 PRINT:PRINT:PRINT
4020 INPUT"REPEATING? Y OR N";R$
4030 INPUT"NOTE LENGTH? 0 - 2000";NL
4100 FORZ=0TOS
4105 POKEQA,2:POKEQD,L(Z)*5+40
4110 POKEQA,0:POKEQD,L(Z)*10+80+XX
4115 POKEQA,13:POKEQD,0
4120 FORA=1TONL:NEXT
4123 POKE530,1
4125 POKE57088,253:IFPEEK(57088)=239THEN4140
4127 POKE530,0
4130 NEXT
4132 POKE57088,253:IFPEEK(57088)=239THEN4140
4135 IFR$="Y"THEN4100
4140 POKEQD,0
4145 POKE530,0
4150 GOTO100
4990 REM SAVE ROUTINE
5000 PRINT:PRINT:PRINT,"FILE CREATION"
5002 INPUT"    ENTER FILE NAME";FL$
5004 PRINT:PRINT:PRINT
5010 PRINT,"SET RECORDER, AND PRESS"
5020 PRINT,"ANY KEY"
5030 INPUTX$
5040 SAVE
5044 PRINT"ZZZ"
5045 PRINTFL$
5050 PRINTS
5055 PRINTXX
5060 FORZ=0TOS
5070 PRINTL(Z)
5080 NEXT
5090 POKE517,0
5095 PRINT:PRINT:PRINT,"RECORDING COMPLETE"
5100 GOTO100
5990 REM LOAD ROUTINE
6000 PRINT,"NOTE SEQUENCE LOADER"
6002 INPUT"    NAME OF FILE REQUIRED";FL$
6004 PRINT:PRINT:PRINT
6010 PRINT,"START TAPE, & PRESS ANY KEY"
6020 INPUTX$
6030 LOAD
6032 INPUTFF$
6034 IFFF$<>FL$THEN6032
6036 PRINT,"FILE   ";FF$;"  FOUND"
6040 INPUTS
6045 INPUTXX
6050 FORZ=0TOS
6060 INPUTL(Z)
6070 NEXT
6075 POKE515,0
6080 PRINT:PRINT:PRINT,"LOAD COMPLETE"
6090 PRINT,"PLAY?"
6130 GOTO100
8000 REM AY-3 INITIALISE
8110 POKEQA,0:POKEQD,0
8120 POKEQA,1:POKEQD,0
8130 POKEQA,3:POKEQD,0
8140 POKEQA,7:POKEQD,248
8150 POKEQA,8:POKEQD,16
8160 POKEQA,9:POKEQD,16
8170 POKEQA,11:POKEQD,0
8180 POKEQA,12:INPUT" PERIOD";PE:POKEQD,PE
8200 RETURN
```

If H is then entered, a replay routine is initiated providing an opportunity to hear again the sequence just played, and to alter note length and replay speed. The replay may be put in a repeating mode if desired.

There is also a facility to save on tape the sequence of digits representing the notes played, and, using a load routine, to reload them on a subsequent occasion.

The save routine uses a file name system, and when a tape is reloaded, the program asks for the name of the file required, and will ignore all other files that it comes across. A vital part of the SAVE/LOAD routines is the PRINT "ZZZ" statement in line 5044. This data (ie, ZZZ) is subsequently read and ignored by the load routine, and ensures that the loading of data is not confused by random noise or unwanted data preceeding the file name. The sequence of data used by the routines is as follows: File Name, Length of Note Sequence, and Timbre Indicator. This data is followed by a sequence of numbers representing the notes played.

SEPARATE USE OF THE 8910's D/A CONVERTERS

The 8910 P.S.G. contains three internal D/A converters, one of which is used for each channel. These may be employed separately for 4 bit D/A conversion, to supplement the ZN425 converter on the Analogue Board. One could, for example, use channels A and B each as 4 bit converters, while retaining channel C for audio output. Fig. 5.4 shows the wiring of the 8910 pads on the Analogue Board for this contingency. Channels A and B are then controlled simply by registers 8 and 9 which, under normal circumstances, would have controlled audio output level. Now, data from 0 to 15 in these registers will determine the d.c. voltage on pads A and B, while data of 16 will place the d.c. voltage under

envelope control. The envelope facility might prove to be particularly useful in a number of different applications. Each of the 3 channels of the P.S.G. could be used to control the level of banks of lights for example, in which case each or all of them could be brought under envelope control to give a slow fade, etc. In such an application each channel output could be made to drive a 741 op. amp. followed by an opto isolator and thyristor or triac controller. It should be noted, however, that the PSG's 4-bit converters may not provide sufficient resolution for some applications.

When using the D/A converter in this way the state of register 7 should be borne in mind. The converter will function whether the associated channel's noise or tone outputs are enabled or not. But with both disabled a non-linearity appears in the transition from 0–1. This is minimised by enabling tone and noise, although the latter places some noise on the d.c. output. The effect can be reduced by increasing the capacitor taken from the pad outputs to ground.

Constructor's Note

A C60 cassette tape containing all the numbered programs of this series is available from *Technomatic Ltd.*, (see advertiser's index), at £3·50 + VAT and p & p.

Next month we shall explore the facilities by the 6522 Versatile Interface Adaptor on the Analogue Board, and will discuss such applications as frequency counters and real time clocks.

PIN CONNECTIONS TO SK3 AND SK4 OF THE DECODING MODULE

One or two readers have asked for specific pin connections for interfacing the devices described in part 2 of the series.

Interfaces such as the light sensor, sound detector or joystick control box described in part 2 of the series, or the l.e.d. indicator or relay output described in part 3 are accessed directly through SK3 or SK4 of the Decoding Module.

The pin connections of these two sockets (SK3 for port A and SK4 for port B) are identical, and were given in Table 1.6 of part 1 of the series.

Devices which require a single port line, a ground and +5V line (such as the l.d.r. circuit of Fig. 2.10 in part 2 of the series) should be connected as indicated in Fig. 5.5.

Pin 8 of SK3 and 4 carries Vcc (+5 volts), pin 1 ground, and pin 16 data line DØ of the port (sometimes labelled PAØ). If the header wired as in Fig. 5.5 is plugged into SK3 of the Decoding Module, and the Logic Tester program of Table 2.5 (part 2 of the series) is run, DØ of the screen display should register a zero for dark conditions, and a 1 in daylight.

The joystick control box of Fig. 2.9 of part 2 connects directly to a 16-pin header that may be inserted into SK3 or SK4 of the Decoding Module. The wiring for this is given in Fig. 5.6. If the header is plugged into SK3 it may be used in conjunction with the screen drawing program listed in Table 2.4 of part 2.

Similar pin connections are made to SK3 or SK4 when using the PIA for digital output. As an example, Fig. 5.7 gives the pin connections for simple audio output from the PIA. Again use is made of the 5 volt line supplied by the Decoding Module. In this instance it is used to power the 2N2926 audio amplifier. This circuit should be used in conjunction with the programs in Table 3.5 and 3.6 of part 3 of the series.

Other input and output applications use similar connections to SK3 and SK4.

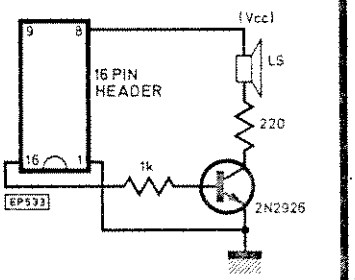


Fig. 5.4. Pin connections to 16-pin header for simple audio output from PIA

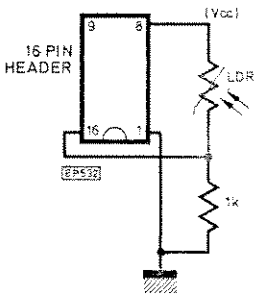


Fig. 5.5. Pin connections to 16-pin header for use with LDR. The header plugs into SK3 or SK4 of the Decoding Module

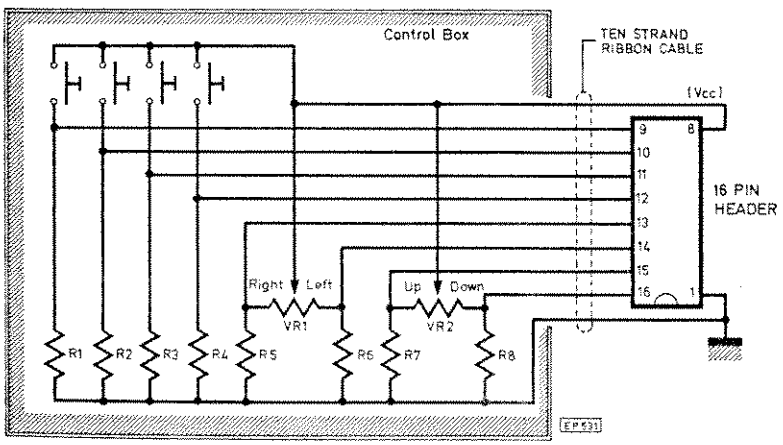


Fig. 5.6. Joystick control box giving connections to 16-pin header to be plugged into SK3 of the Decoding Module