Make music with your computer

Our Compumuse project is easy to build and simple to use. It connects to any computer with a Centronics-style parallel port and provides music over a five octave range and a wide variety of sound effects. Your computer programs may never be the same again!

by PETER VERNON

Programmed performance of music by computers has many practical uses apart from the satisfaction of coding the score and hearing the results. Music education using computer-assisted programs is an effective way of learning musical notation, for example.

Sound effects are also a big aspect of computer games, as any "Space Invaders" addict will tell you. Those who have game programs that run without sound will also tell you that it's just not the same in silence.

Audible feedback can also be used effectively in other programs, to signal the acceptance of data or indicate an error, or to alert the operator that some task has been completed.

Many small computers have some form of audio output, even if it is only a single bit driving an amplifier. This approach leaves much to be desired, however. Using this system the processor is fully occupied producing sound and simultaneous sound and movement of graphics images, for example, is impossible.

What is needed is a low-cost sound generator capable of music and a wide variety of sound effects for games. It should be as musically accurate as possible, and preferably be able to be used with any computer system. We think our "Compumuse" unit fits the bill.

How it works

Our circuit is based on the Texas Instruments SN76489AN sound generator IC, which contains three independently programmable frequency generators, a noise generator and a programmable attenuator for each sound source. The outputs of the four attenuators are summed to provide a single audio output (see Fig.1).

Each of the three tone generators of the chip contains a 10-bit programmable counter used as a frequency divider. The output frequency of each generator is determined by the value loaded into the 10-bit register and the overall clock frequency of the sound chip.

Once loaded the register is decremented at a rate of N/16, where N is the chip clock frequency. When the register value reaches zero the frequency flipflop is toggled and the counter reloaded with the original value. Since the output flipflop must be toggled twice to produce a square wave pulse the counter value specifies half the period of the waveform.

The frequency produced for any counter value, R0, can thus be calculated with the formula:

F0 = N/(32*R0).....equation 1

where N is the clock frequency.

The maximum number that can be repesented in 10 bits is 1023, but the SN76489AN will interpret a zero value as 1024, using "one's complement" arithmetic. The first count decrements 0 to give a result of 1023. Effectively then, 0 represents 1024 and the maximum amount we can divide the clock frequency by is 32 × 1024 or 32,768.

With a 2MHz clock frequency the lowest tone that can be produced is therefore:

2,000,000/(32*1024) = 61.03Hz

and the highest tone available is;

2,000,000/(32*1) = 62,500Hz

Since the register value must be an integer, some frequencies cannot be produced exactly. An exact 440Hz note (A above middle C) for example, requires a divider value of R0, where;

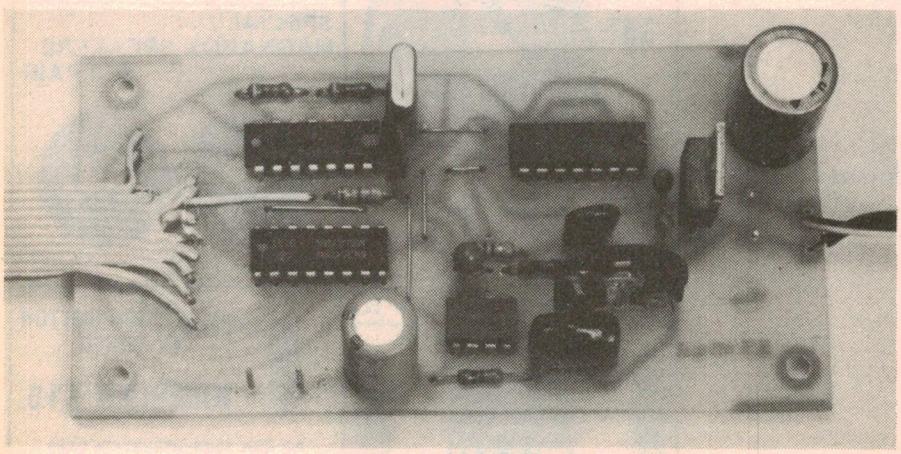
R0 = 2MHz/(32*440).....equation 2

which is equal to 142.045

Taking the integer value of R0 and substituting it back into equation 1 gives us the actual frequency which will be produced;

F0 = 2,000,000/(32*142) = 440.14Hz

This represents an error of about .035%.



The circuit has three programmable tone sources and a noise generator.

Each of the tone output flipflops is fed into a four stage attenuator. Depending on the setting of the control byte the attentuation stages provide attenuation from 0dB to 30dB (off) in 2dB steps, allowing control of the envelope of each tone produced.

The noise source consists of a shift register with an exclusive-OR feedback network. If the feedback network is enabled the shift register produces white noise. A repeating pattern, sounding like a low tone, is produced if the feedback network is disabled.

The output of the noise source is also fed to a programmable attenuator, allowing the volume of the noise to be controlled in the same way as the volume of the tone sources.

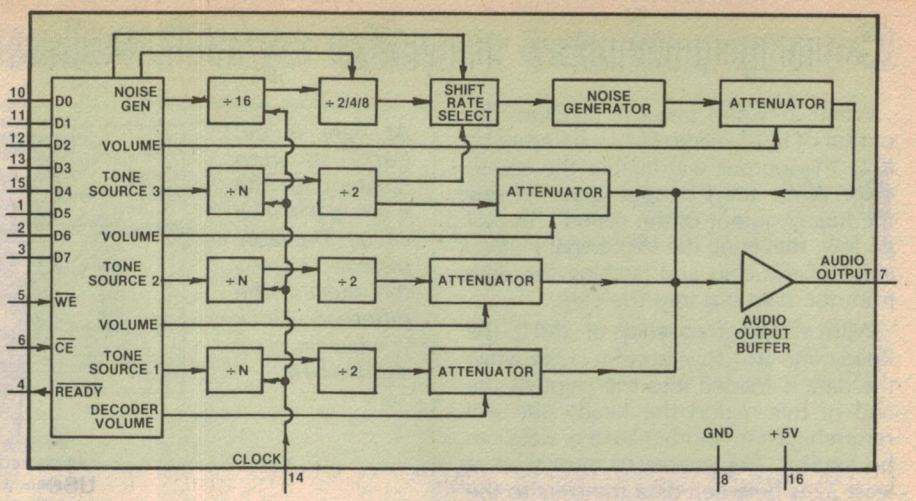


Fig. 1

This block diagram of the Texas Instruments SN76489AN sound generator chip shows the separate divider circuits and attentuators for each of three tone sources and noise. Full circuit diagram is shown below.

The circuit

While it is possible to connect the SN76489AN directly to the microprocessor data bus, such a circuit would be dependent on the actual computer system used with the sound generator. The circuit shown in Fig. 2 is designed to be connected to a Centronics-style parallel interface, and can be used with any computer that has a latched parallel output port with a strobe output and a Ready signal input.

These include the System-80 and TRS–80 with expansion units, the Super-80 fitted with a parallel interface board and the MicroBee computer, among others.

The maximum clock frequency of the SN76489AN is 4MHz, but since the range (and to a lesser extent, the accuracy) of the sound generator is determined by the clock frequency, we have made provision for users to select one of three clock frequencies, as summarised in Fig. 3.

The 4MHz clock frequency limits the range in the lower octaves while providing an extended range in the (useless) ultrasonic regions. In some cases though, particular frequencies can be more closely approximated with a 4MHz clock.

Our prototype used a 2MHz clock, which provides a usable range of five octaves with acceptable accuracy and data load time. A 1MHz clock provides a six octave range, but accuracy tends to fall off.

All clock frequencies are generated by a 4MHz crystal in conjunction with the two NAND gates of IC1. Depending on which of the links shown on the component overlay are installed, the 4MHz clock will be fed directly to the SN76489AN or passed through either one or both flipflops of IC2.

Connection to host computer

The computer loads a byte of data to the sound chip by putting it on the data lines and pulsing the Strobe input line low. The input data must remain stable for 32 clock cycles (32μs at 2MHz) to allow time for the chip to latch it.

Centronics ports such as those used by the Super-80, System-80 and TRS-80 produce the strobe pulse automatically when the address of the printer port is selected. Users of other computers will need to make other arrangements.

On the Sorcerer for instance, a "Data Available" signal appears on pin 3 of the parallel port connector when data is written to the output port FF. This signal will remain low until reset by a low-going signal on pin 2 of the parallel connector. The Ready output of the sound generator board can be used as a "data received" signal in this case.

The low-going Strobe signal is input to NAND gate IC1d. Before the Strobe line is activated both inputs to IC1d (pins 12 and 13) will be high, and the output (pin 11) will be low. This output is inverted by IC1c, placing a "1" on the CE input and disabling the sound chip.

Compumuse music synthesiser

When a Strobe pulse is received the output of IC1d will go to "1". Inverted by IC1c this output will enable the sound chip. When the CE input is pulled low the Ready output of the sound chip will go low, triggering the Write Enable (WE) input of the chip and latching the data from the data bus into the chip.

With a clock frequency of 2MHz the Ready line will stay low for 32μs while the data is loaded into the chip. At the end of this period the Ready line will return high and another byte of data can be loaded. The processor must wait at least 32μs between data transfers to the sound generator.

Using Basic this timing does not pose a problem. Most Basic interpreters take several milliseconds to execute an OUT statement. For some sound routines, however, Basic may be too slow in updating the sound generator registers and a machine language program must be used.

For use with machine language driver routines a Ready signal is brought off the board for connection to the processor through the Busy input of a Centronics port. By monitoring the Ready signal and transferring data when this line goes high, correct data input timing will be automatic.

The sound generator's output buffer produces a signal with a voltage swing of about 2V. A simple audio amplifier based on the LM386 provides more than enough power to drive a loudspeaker. A 10kΩ trimpot is used as an overall volume control, and a Zobel network (.033μF and 22Ω) on the output of the sound generator is used to prevent oscillation of the sound chip output buffer.

The circuit requires a single +5V supply, provided by a 9V DC plugpack feeding a +5V regulator on the sound generator board. Since 9V is a "normal" rating only, the 5V regulator may be called upon to dissipate considerable power. A flag type heatsink can be added if the regulator tends to run hot. Current consumption of the circuit is less than 200mA.

Construction

Start construction with the power supply, ensuring that the regulator and tantalum and electrolytic capacitors are installed with the correct orientation, as shown on the component overlay diagram.

Before proceeding connect the plugpack and test the output of the regulator. If all is well disconnect the power and proceed to the next stage of construction.

The 4MHz crystal, resistors and capacitors can be installed, followed by the ICs. None of the integrated circuits used in this project are particularly sensitive, but be careful not to subject them to too much heat when soldering. 300°C for 10 seconds is the normal maximum soldering specification, but a good joint can be made in much less time.

Depending on which clock frequency is required, install the link options.

The sound generator is connected to the printer port of the System-80 or Super-80 by a 34-way edge connector and a length of ribbon cable. Strictly speaking it is not necessary to connect the Ready output to the computer if you will only be using Basic to control the synthesiser, but it's just as easy to make a complete job of it the first time.

Connect the loudspeaker and set the volume control to about half-way. Without connecting the sound generator to the computer, connect the power and switch on. You should be rewarded by a tone from the speaker. As no register values have been set, the tone will be a random mixture of the outputs of the three frequency generators and noise. Turn off the sound generator after this test.

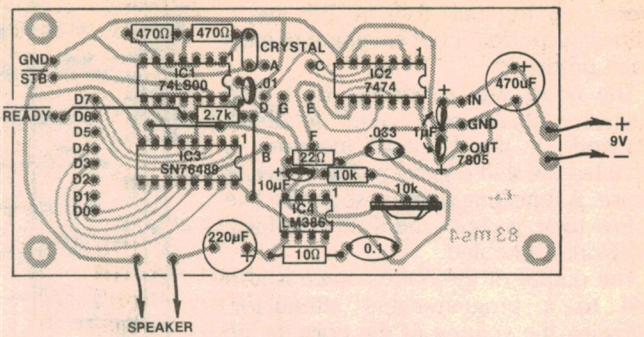
With the computer and all peripherals off connect the sound generator to the computer. As with all computer systems, peripherals (including the sound generator) should be turned on before the computer itself, and turned off after the computer has been switched off.

Data formats

The SN76489AN contains eight 8-bit registers to control the tone and noise generators and attenuators. Noise and attenuation values are single bytes, while the frequency produced by the tone source is specified as two bytes.

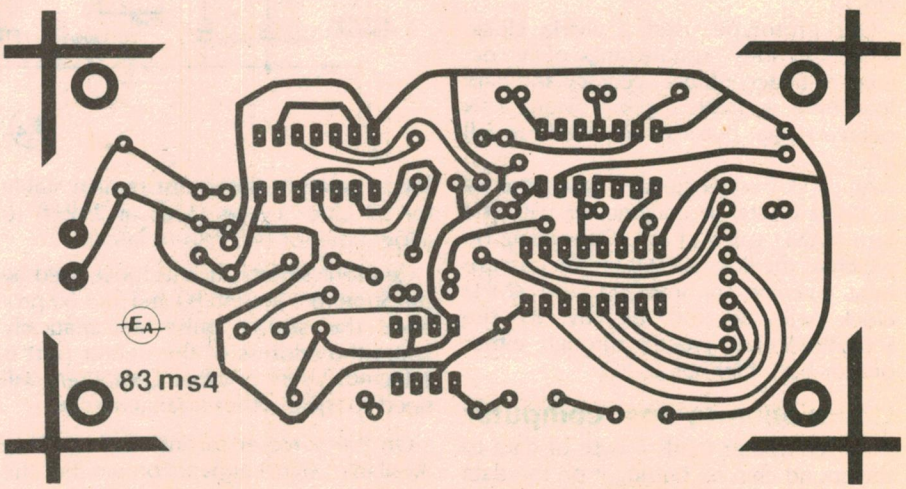
All first byte or single byte data has the most significant bit set to "1". The se-

At right is the parts location diagram for the Compumuse circuit. The table below shows the effect of the different clock frequencies available.



FOR 4MHz CLOCK LINK A-B AND DO NOT INSTALL IC2
FOR 2MHz CLOCK LINK A-C, B-D AND E-G INSTALL IC2
FOR 1MHz CLOCK LINK A-C, B-F AND D-E INSTALL IC2

Clock Frequency	Highest tone	Lowest tone
4MHz	125kHz	122.07Hz
2MHz	62.5kHz	61.03Hz
1MHz	31.25kHz	30.58Hz



Above is the full-size pattern for the Compumuse printed circuit board.

cond byte of a frequency value has the most significant bit "0".

Understanding how the SN76489AN is programmed is complicated by the fact that Texas Instruments uses a different method of numbering binary values. TI designates the most significant bit of a binary value d0, and the least significant as d7, the exact opposite of the usual scheme. We will ignore TI's "standard" and present all binary values with the most significant bit (the leftmost bit) as d7 and the least significant bit as d0.

Confusion can arise when comparing TI publications with the information given here unless this point is kept in mind.

The first byte of a two byte frequency value has a MSB (most significant bit) of "1", followed by a 3-bit register address field which determines which register receives the information which follows. The four least significant bits of the 10-bit register value are stored in bits 3 to 0 of the first byte, with the six most significant bits stored in bits 5 to 0 of the second byte. Bit 7 of the second byte is "0", and bit 6 is "don't care" (either "1" or "0").

Programming noise requires only a single byte parameter. The MSB is "1", again followed by a 3-bit address field. Bit 3 is "don't care", and bit 2 is the feedback bit. When this bit is "1" the noise generator will produce random white noise. If bit 3 is "0" the feedback will be disabled and a periodic tone will be produced.

The two least significant bits of the noise control byte set the rate at which the shift register is clocked. Four options are available (N is the overall clock frequency);

NF1	NF0	Shift rate
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Shift rate set by tone source 3

The fourth option allows the shift rate of the register to be controlled by the output of tone source 3. High shift rates produce high-pitched noise and low shift rates produce low-pitched noise. By varying the output of tone generator 3 the dominant pitch of the noise source can be accurately controlled.

The use of frequency sweeps to control the noise generator is demonstrated by the section of the program in listing 2 which produces a phaser sound using white noise, with both volume and frequency reducing over time.

Attenuation parameters are also specified by a single byte. Bit 7 is "1", and the next three bits are an address field which control which attenuation control register (tone source 1, 2, 3 or noise) will be updated. The four least significant bits are the actual value of the attenuation.

Compumuse: listing 1

```
10 CLS
20 PRINT "DO YOU WANT TO"
22 PRINT TAB(15);"DEFINE FREQUENCY (1)"
24 PRINT TAB(15);"DEFINE NOISE     (2)"
26 PRINT TAB(15);"QUIT THE PROGRAM (3)"
30 PRINT TAB(15);
40 INPUT "YOUR SELECTION  ";A
50 IF A<1 OR A>3 THEN 10
60 PRINT
70 ON A GOTO 100,500,800
90 REM THIS SECTION TAKES A FREQUENCY IN HERTZ AND
92 REM RETURNS THE REGISTER VALUE REQUIRED BY THE
94 REM             SN76489A
96 REM
100 CL=2000000:REM CHANGE THIS FOR DIFFERENT CLOCK FREQUENCIES
106 CLS
110 INPUT "WHAT FREQUENCY DO YOU WANT";F0
130 R0=INT(CL/(32*F0))
140 IF R0<1 OR R0>1023 THEN GOTO 1000
150 F1=CL/(32*R0)
160 F2=CL/(32*(R0+1))
170 IF ABS(F0-F1)<ABS(F0-F2) THEN GOTO 190
180 PRINT "CLOSEST ACTUAL FREQUENCY IS";F2 ;"HZ"
182 R0=R0+1:PRINT"WITH REGISTER VALUE=";R0
184 GOTO 200
190 PRINT "CLOSEST ACTUAL FREQUENCY IS";F1;"HZ"
192 PRINT "WITH REGISTER VALUE=";R0
200 PRINT
210 REM THIS SECTION TAKES A REGISTER VALUE AND
220 REM A VOICE NUMBER AND RETURNS THE TWO BYTES
230 REM WHICH MUST BE SENT TO THE OUTPUT PORT TO
240 REM LOAD THAT VALUE INTO THE APPROPRIATE REGISTER
260 INPUT "VOICE TO BE USED (1 TO 3)";V
270 PRINT
280 B=INT(R0/16)
290 A=R0-B*16+128
294 A=A+32*(V-1)
300 PRINT"FOR VOICE";V
310 PRINT"FIRST  BYTE=";A
320 PRINT"SECOND BYTE=";B
350 PRINT
360 PRINT"WHAT VOLUME DO YOU WANT (15 IS LOUDEST, 0 IS OFF)";
370 INPUT VL
380 VL=15-VL
390 VB=144+VL+32*(V-1)
400 PRINT
410 PRINT"VOLUME CONTROL BYTE=";VB
420 PRINT:PRINT "ENTER 'H' TO HEAR SOUND"
430 INPUT "OR 'R' TO RETURN TO MENU";A$
440 IF A$="R" THEN GOTO 10
450 N=253
460 OUT N,A:OUT N,B
470 OUT N,VB
480 FOR D=1 TO 100:NEXT D:GOSUB 900
490 INPUT "AGAIN (Y/N)";A$
492 IF A$="Y" THEN GOTO 460 ELSE GOTO 10
500 REM THIS SECTION FORMS THE NOISE CONTROL BYTE
504 CLS
510 PRINT"SELECT WHITE NOISE (1) OR PERIODIC NOISE (0)";
520 INPUT FB
530 NB=224+FB*4
534 PRINT
540 PRINT TAB(20);"SELECT HIGH PITCH (0)"
550 PRINT TAB(20);"      MEDIUM PITCH (1)"
560 PRINT TAB(20);"         LOW PITCH (2)"
570 PRINT"PITCH UNDER CONTROL OF TONE SOURCE 3  (3)"
580 PRINT:PRINT TAB(22);:INPUT"PITCH SELECTION";NF
584 IF NF<0 OR NF>3 THEN 534
590 NB=NB+NF
600 PRINT
610 PRINT"WHAT VOLUME DO YOU WANT (15 IS LOUDEST, 0 IS OFF)";
620 INPUT V
630 PRINT
640 V=15-V
650 NV=240+V
```

Listing continued ►

```
660 PRINT"NOISE CONTROL BYTE   (SENT FIRST)=";NB
670 PRINT"VOLUME CONTROL BYTE (SENT SECOND)=";NV
680 PRINT:PRINT"ENTER 'H' TO HEAR SOUND"
690 INPUT "OR 'R' TO RETURN TO MENU";A$
700 IF A$="R" THEN GOTO 10
710 N=253
720 OUT N,NB:OUT N,NV
730 FOR D=1 TO 100:NEXT D:GOSUB 900
740 INPUT "AGAIN (Y/N)";A$
750 IF A$="Y" THEN GOTO 720 ELSE GOTO 10
800 END
900 REM THIS SECTION SHUTS UP ALL SOUND
904 N=253
910 N=253
920 OUT N,159:OUT N,191
930 OUT N,223:OUT N,255
940 RETURN
1000 PRINT "SORRY, FREQUENCY IS OUT OF RANGE"
1010 INPUT "PRESS 'ENTER' TO RETURN TO MENU";A
1020 GOTO 10
```

Compumuse: listing 2

```
10 CLS
20 GOSUB 1000:REM TURN OFF ALL SOUND
30 N=253
40 PRINT "BELL (1), PHASER (2), BIRDS (3), OR EXPLOSION (4)"
50 INPUT "SELECT 1, 2, 3 OR 4";A
60 ON A GOSUB 90,220,320,440
70 GOTO 10
80 :
90 PRINT "BELL SOUND"
100 OUT N,140:REM SET VOICE 1 TO 679HZ
110 OUT N,5
120 OUT N,170:REM SET VOICE 2 TO 694HZ
130 OUT N,5
140 FOR B=0 TO 3:REM NUMBER OF CHIMES
150 FOR I=145 TO 159:REM LOOP THROUGH ATTENTUATION STEPS
160 OUT N,I:OUT N,I+32
170 FOR D=0 TO 40:NEXT D:REM LENGTH OF A CHIME
180 NEXT I
190 NEXT B
200 RETURN
210 :
220 PRINT "PHASER SOUND"
230 OUT N,231:OUT N,240
240 FOR L=0 TO 15
250 FOR A=192 TO 207
260 OUT N,A:OUT N,L
270 NEXT A
280 OUT N,(240+L)
290 NEXT L
300 RETURN
310 :
320 PRINT "BIRDSONG":T=0
330 S=INT(RND(10)):REM RANDOM CHIRP LENGTH
340 OUT N,144
350 FOR I=0 TO 10
360 OUT N,(128+I)
370 OUT N,1
380 FOR D=0 TO S:NEXT D
390 NEXT I
400 T=T+S
410 IF T=>200 THEN RETURN
420 GOTO 330
430 :
440 PRINT "EXPLOSION SOUND"
450 OUT N,230:REM SET HIGH PITCHED WHITE NOISE
460 FOR I=240 TO 255:REM LOOP THROUGH ATTENUATION VALUES
470 OUT N,I
480 FOR D=0 TO 75:NEXT D:REM LENGTH OF SOUND LEVEL
490 NEXT I
500 RETURN
```

Listing continued ►

With all four bits set, attenuation will be maximum (no sound). With these four bits "0" there will be no attenuation (maximum volume).

Programming sound effects

Perhaps the best way to understand how to program the sound generator is to analyse the program of listing 1, a utility program intended to aid in the development of music and sound effects routines.

There are two main sections of the program. The first, lines 100 to 500, produces tones and the second section, from line 500 to 800, produces noise. Lines 10 to 70 ask for the user's choice and jump to the appropriate routine.

As previously explained, the SN76489AN produces tones by dividing a clock frequency. The correct value for this frequency must be inserted in line 100. Our prototype uses a 2MHz clock frequency.

Line 110 asks for the frequency of the tone required. Given the frequency, F0, the divider value is calculated in line 130, using equation 2.

The maximum number that can be stored as a 10-bit binary value is 1023. If the required register value is greater than this figure, the frequency requested is below the lower limit of the sound generator's range. If it is less than 1 the required frequency is too high. In either of these cases the program returns to the start-up menu via lines 1000-1020 which inform the user of the problem.

Since the register value increases in discrete steps from 1 to 1023, not all frequencies can be produced accurately. Lines 150 and 160 calculate the frequency that will be produced with the calculated integer register value, and with a value one larger than the calculated value. Sometimes this frequency will be closer to that required than the frequency produced by the original register value.

Line 170 tests which of the two frequencies is closest to the value required and directs the interpreter to print either the first or the second frequency and its associated register value.

We are not yet finished. Given a 10-bit register value we must convert it into binary for storage as two bytes in the sound generator's internal registers.

Dividing by 16 shifts the binary value of R0 four places to the right, giving the correct value for the six least significant digits of the second control byte. These are the six most significant bits of the register value.

The first control byte, containing the four least significant bits of the register value, can be calculated by subtracting the value of the second byte multiplied

Compumuse music synthesiser

PARTS LIST

1 loudspeaker
1 9V DC plugpack
1 printed circuit board, 106mm x 48mm, code 83ms4
1 4MHz crystal
1 30cm length of 11-way ribbon cable
1 36-way edge connector or other connector to suit.
1 heatsink (see text)

SEMICONDUCTORS
1 74LS00 quad NAND gate
1 7474 dual D-type flipflop
1 SN76489AN sound generator
1 LM386 audio amplifier
1 7805 +5V voltage regulator

RESISTORS (¼W, 5% unless stated)
1 x 10kΩ, 1 x 2.7kΩ, 2 x 470Ω, 1 x 22Ω, 1 × 10Ω, 1 × 10kΩ vertical mounting trimpot.

CAPACITORS
1 470µF/16VW electrolytic
1 220µF/16W electrolytic
1 10µF tantalum
2 1µF tantalum
1 0.1µF polystrene
1 .003µF greencap
1 .01µF ceramic

MISCELLANEOUS
Tinned and insulated copper wire
Solder
Case and PCB stand-offs to suit

We estimate that the current cost of parts for this project is approximately

$20.00

This includes sales tax but does not include a plugpack, case or cabling.

Listing 2 continued:

```
1000 REM TURN OFF ALL SOUND
1010 N=253
1020 OUT N,159
1030 OUT N,191
1040 OUT N,223
1050 OUT N,255
1060 RETURN
```

by 16 from the original register value. Bit 7 of the first byte must be 1, and so we set this bit by adding 128 to the value of the first byte.

The first control byte also contains a 3-bit register address which determines which register the control bytes will be sent to. So far we have calculated the first control byte with these bits (b6, b5 and b4) zero, so the first byte addresses the frequency control register for tone 1.

The control register for tone source 2 is addressed by setting b5 to 1 (equivalent to adding 32 to the value for tone 1). The frequency control register for tone source 3 can be accessed by setting b6 (adding 64 to the value calculated for tone 1).

Line 294 takes care of this adjustment. Given a voice number, V we need to add 0 for voice 1, 32 for voice 2 and 64 for voice 3. This can be achieved by adding 32*(V-1) to the value for the tone 1 frequency control register to give us the control value for the same frequency produced by tone source 2 or 3.

Attenuation registers

We now have two bytes which when sent in the correct order to the sound generator will program the desired frequency from the tone source specified. Sending these bytes will not produce sound, however until the attenuation register for the appropriate tone source is programmed.

Line 360 asks for the volume level required. Volume is specified as a number between 0 and 15, with 15 being the loudest. This is the opposite of the Texas Instruments convention, which specifies attenuation, with maximum attenuation (no sound) represented by 15 and minimum attenuation (full volume) represented by 0.

Line 380 makes this adjustment, subtracting the volume number we type from 15 to give us the attenuation code.

Attenuation control bytes have the most significant bit (b7) set to 1. The next three bytes are the register addresses. Since we are only specifying attenuation control registers here, b4 will always be "1". This gives us a base value of 144, which would address the attenuation control register of tone source 1.

By adding 32 multiplied by the voice number minus 1 we can arrive at the correct address for the attenuation register of any tone source. When the lower 4 bits calculated by line 380 is added to this base value we have the specified attenuation control byte for the required voice. This addition is done in line 390 and the result printed by line 410.

Our frequency control bytes are now complete, with two bytes specifying the tone to be produced by the chosen source and a third byte specifying volume. The next section of the program gives us the choice of hearing the tone created or returning to the initial menu to specify further notes. (Lines 420 to 492.)

Tones are played by outputting the first tone control byte then the second and then the volume control byte. On the System-80 the address of the parallel printer port of the expansion interface is FD, or decimal 253. Lines 450-470 output the control values. Line 480 determines the duration of the sound produced. To use this routine on the TRS-80 Model 1, change line 450 to N = 14312 and change all OUT N, xxx statements to POKE N, xxx

After sounding the tone, the program gives the user the choice between hearing it again or returning to the main menu. Take particular note of the subroutine beginning at line 900. This routine disables all sound sources by setting the attenuation control bytes for maximum attenuation (silence). It's best to type this part of the program first, as when the computer is switched on with the sound generator attached . . .

The second section of the program calculates noise control bytes. The noise control byte is made up a 1 in the most significant bit position, a 3-bit register address field, a feedback bit for selecting random or periodic noise and two pitch control bits. Bit 3 of the byte can be either 1 or 0.

The "base value" of the noise control byte is 224, calculated as follows:

Bit 7 set to 1 gives 128. The address of the noise control register is 110, in bit positions b6 to b4, giving 96. Adding these two values gives us 224.

The first choice is between random and periodic noise. One bit of the noise control register determines whether the feedback network of the shift register is enabled (for white noise) or disabled (for periodic noise).

The feedback bit is in bit position b3, which has a decimal value of 4 when set to 1. If we specify the feedback bit as a 1 for random noise and multiply it by 4 before adding it to the base value, the feedback bit will be correctly set.

Bits 1 and 0 control the pitch of the noise source. With both 0, we get high pitch, while bit 0 set will give medium pitch and bit 1 set will give low pitch. Both bits set will place the noise pitch under the control of tone source 3.

Conveniently, the fact that the pitch control bits are in the two least significant bit positions means that if we specify high pitch as 0, medium as 1 etc, we can simply add the value of the pitch selection number to the calculated base value. Adding 1, for instance, will reset b1 and set b0, for medium pitch. Adding 3 will set both b1 and b0, placing the noise pitch under the control of tone source 3.

continued on p.141

Teleconferencing meets resistance

"Serious and surprising obstacles" are preventing the rapid growth of "teleconferencing", according to a new research report from International Resource Development Inc.

According to IRD, a persistent theme in teleconferencing experiments is that users are not comfortable with the technology, or find that it does not really convey the "presence" of remote conference participants. Perhaps, say IRD researchers, the major problem is the cultural expectations generated by the use of TV-like screens.

Contrasting the professional production of television with "teleconferencing", what comes across is an appearance of incompetence or slovenliness. "To their dismay, some executives find that teleconferencing portrays them as nerds," says Ledecky. Not surprisingly, they become reluctant to use the medium.

Compumuse . . . *From p99*

Lines 510 to 600 of our program calculate the noise control byte from the options selected. The only remaining step is to calculate the volume control byte in the same way as for the tone generators.

The noise source volume control register has a base address of 240, made up of a most significant bit set to 1, and a three bit register address field with all bits set to 1. The value of 240 corresponds to maximum volume of the noise source (minimum attenuation). Specifying volume as a number between 0 and 15 and then subtracting it from 15 before adding it to the base value gives us the required volume control byte in line 650.

Again the user has the option of hearing the sound produced. Producing the sound is a matter of first sending the control byte, specifying the type of noise and its pitch and then sending the volume control byte to enable the noise source. You could of course do it the other way around, but if the noise source is enabled first you will hear noise based on the previous contents of the control register during the short time before the noise control byte is updated.

Note that the program in listing 1 is an aid only, not a demonstration of the capabilities of the sound generator. By noting down the register values and control bytes required to produce a particular tone or noise and combining these values in your own programs a wide range of sound effects can be created. Listing 2 demonstrates some of the possibilities but there's no need to stop there.

The Preh Commander keyboard

Mayer Krieg & Co now has stocks of a new keyboard design, the Preh Commander, advertised as an "intelligent, programmable" unit. The 68-key alphanumeric keyboard is built around a microprocessor and allows the user to select parallel or serial output, with a positive or negative key pressed strobe pulse, serial transmission rates from 150 to 9600 baud, auto-repeat and caps lock, and over 200 different internal characteristics.

According to Mayer Krieg the keyboard is designed to the latest ergonomic standards, with dished keys and sloping from 35m at the rear to 10mm at the front of the keyboard. A metal frame gives rigidity while each key has a separate guide and a double spring, providing a bounce-free contact and reliable switching.

For further information contact Mayer Krieg & Co, GPO Box 1803, Adelaide, SA, 5001. Phone (08) 223 6766. Sheridan Electronics, of 164-166 Redfern St, Redfern, NSW also advertise the Commander keyboard. Phone (02) 699 5922.