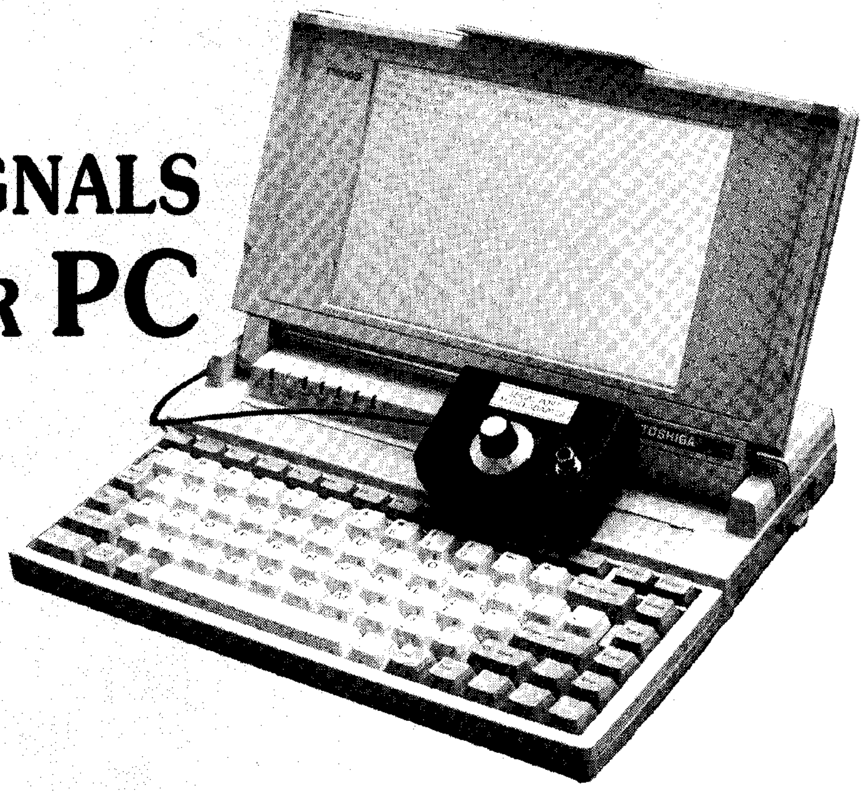
PRECISION AUDIO SIGNALS FROM YOUR PC



Build this computer-controlled audio generator—for under ten bucks.

MICHAEL A. COVINGTON

DO YOU NEED A PRECISION SQUARE-wave generator for audio testing? If so, look no further than your computer's serial port, a handful of passive components, and a very simple 30-line BASIC program.

That combination can deliver signals with frequencies as high as 4800 hertz, and with crystal-controlled accuracy of 0.1% or better. The software was developed on an IBM-compatible PC, but it should run on just about any computer.

How it works

The trick is U, the ASCII character "U," that is. The hexadecimal value of "U" is 55, which in binary is 01010101 (with eight data bits and no parity, or seven data bits and even parity).

The RS-232 protocol specifies that the bits of an ASCII character are transmitted from least to most significant, preceded by a start bit (always 0) and followed by a stop bit (always 1). So, after adding the requisite start and stop bits, the result is 1010101010.

Now suppose a string of U's is generated at the serial port at some steady rate. The result is a continuous series of alternating ones and zeroes—a squarewave.

The frequency of the signal will be half the baud rate, which by definition is the number of transitions per second. Each cycle of a squarewave comprises two transitions so, for example, a 9600-bps baud rate produces a 4800-hertz squarewave.

In practical terms, just about any computer should be able to deliver frequencies of 55, 150, 300, 600, 1200, 2400, and 4800 hertz, corresponding to the standard baud rates from 110 to 9600. In addition, the signal can appear at many other discrete frequencies, limited only by the CPU speed of the computer. But there is a catch to this scheme.

Frequency limits

The catch is that a PC cannot generate just *any* frequency. Why not? Because the UART in the RS-232 port generates its output by dividing the frequency of a 1.8432-megahertz crystal oscillator. The UART can divide only by whole numbers. So, for example, a frequency of exactly 1000 hertz can't be generated; the nearest you can get is 993.1035 hertz.

If you want to tune a guitar, you can't quite produce a standard concert-pitch "A" (440 hertz), but you can get a very clsoe 439.6947 hertz, which is off by only 1/80 of a semitone. The software will display the nearest standard value to any requested frequency.

Another limitation is that some computers might not be able to deliver high frequencies,

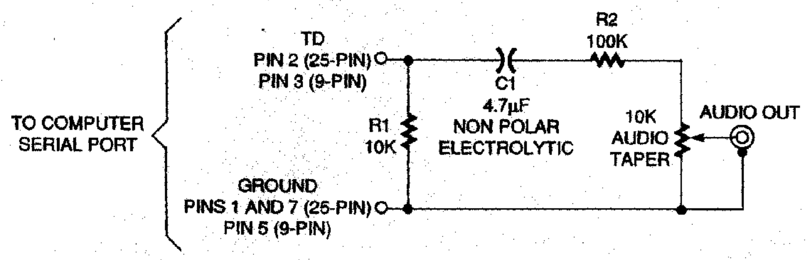


FIG. 1—COMPLETE CIRCUIT of the precision audio generator requires just a handful of passive components.

because they can't output U's fast enough. In that case, squarewave bursts, with silence in between, is generated. If you could hear that, it might sound like buzzing, flapping, or clicking superimposed on the high-pitched tone. The best way to detect this kind of problem is with an oscilloscope or frequency counter.

Even if the computer is fast, there can be breaks in the squarewave. That can happen if the computer is doing task-switching (e.g., under Windows), or if it is heavily loaded with terminate and stay resident programs (TSRs). But under DOS, the author had good results up to 4800 hertz with an old Toshiba laptop.

Hardware and software

There's not much to the circuit, which is shown in Fig. 1. The output of a serial port is nominally 24 volts peak-to-peak, which is much too high a voltage to feed to the input of an audio amplifier. The circuit attenuates the signal to a more useful level, a variable 2-volts peak-to-peak. The circuit also protects the computer from static electricity and voltage surges. Capacitor C1, a non-polarized unit, blocks DC because the serial port, when idling, outputs approximately −12 volts.

The attenuator consists of only four components, so it does not need a PC board. The circuit was built in a small plastic case by mounting the resistors and capacitors directly to the potentiometer and output jack, as shown in Fig. 2. Figure 3 shows the assembled unit.

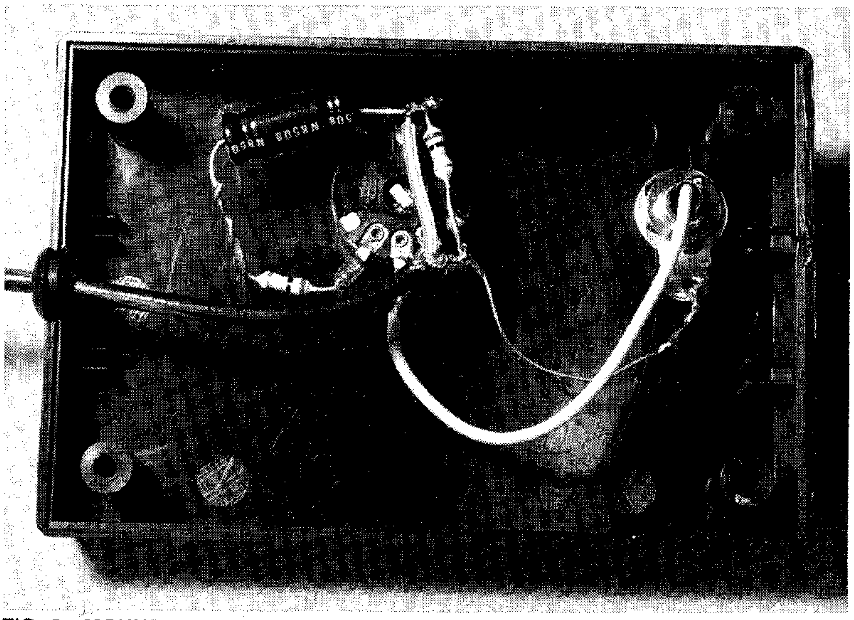


FIG. 2—MOUNT ALL COMPONENTS directly on the potentiometer, as shown here. This eliminates the need for any kind of circuit board.

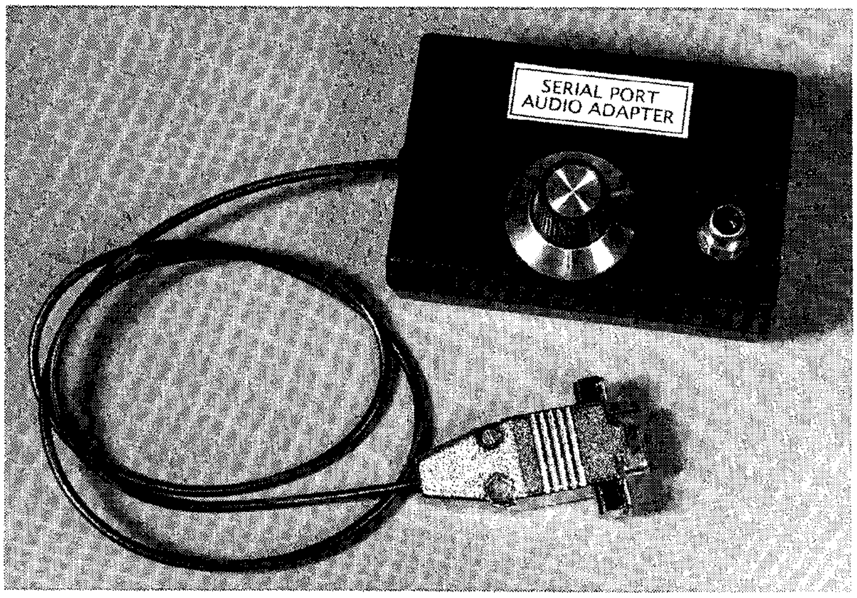


FIG. 3—THE ASSEMBLED UNIT is compact and attractive. An even smaller case than the one shown here could also be used.

PARTS LIST

R1—10,000 ohms, ⅛-watt
R2—100,000 ohms, ⅛-watt
R3—10,000 ohms, audio-taper potentiometer
C1—4.7 µF, 50-volts, non-polarized electrolytic capacitor (Radio Shack 272-998 or equivalent)
Enclosure, cables, and connectors to suit your equipment and needs.

If you want the circuit to deliver signals for testing, just run an appropriate cable from J1 to your equipment. For audio output, say for tuning musical instruments or playing audible tones, the circuit can drive a speaker directly, as shown in Fig. 4.

A word of caution before discussing applications: RS-232 ports are supposed to be tolerant of static charges, short circuits, and extraneous voltages, but be aware that some ports are not. Use extra care if your port is part of a multifunction card that also includes a disk controller.

Many of these cards include both the serial port and the disk controller in a single, fragile VLSI device. If part of the circuit fails, you're likely to lose the whole thing—including access to your disk drives. Generally speaking, when experimenting with accessories connected to a serial port, it's safer to use a card with discrete RS-232 transmitter and receiver ICs.

As for the software, Listing 1

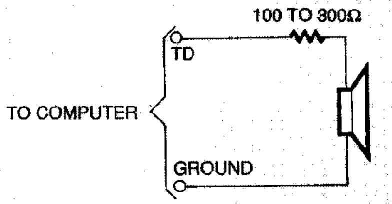


FIG. 4—TO DRIVE A SPEAKER from the generator, use this circuit.

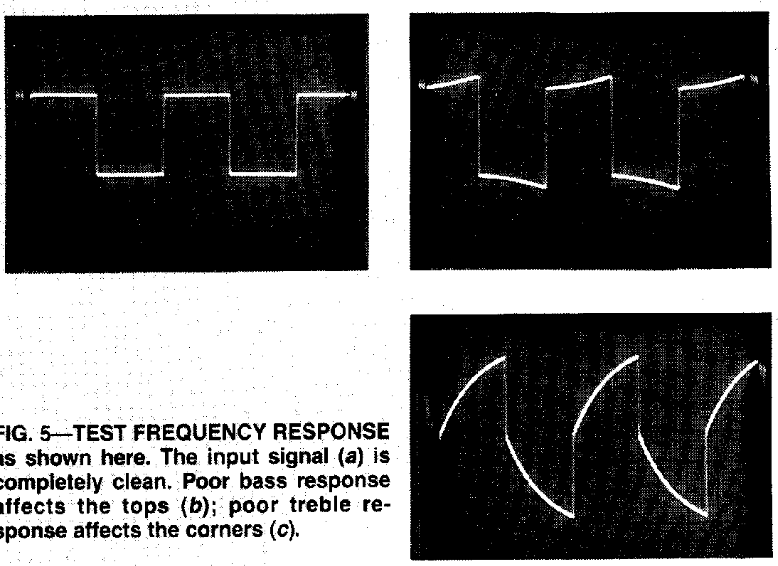


FIG. 5—TEST FREQUENCY RESPONSE as shown here. The input signal (*a*) is completely clean. Poor bass response affects the tops (*b*); poor treble response affects the corners (*c*).

shows the complete program. After setting up several constants, the program requests an output frequency, displays the nearest attainable value, and then starts pumping U's from the serial port. That continues until the user presses a key and then it stops.

The constants defined in lines 120 and 130 specify values for the PC's standard COM1 serial port. To use a different port, make the appropriate modifications. Also note line 190, which specifies the dividend in the frequency calculation. To run the program on another computer, that value might have to be adjusted.

Putting it to use

A precision squarewave generator has many applications. For example, you can use it to test the frequency response of an amplifier, as shown in Fig. 5. The input signal to the amplifier appears in Fig. 5-*a*; note the signal's sharp corners and flat tops. If the amplifier circuit has poor bass response, the tops of the waveform won't be flat (Fig. 5-*b*); if the treble is weak, the corners of the waveform won't be square (Fig. 5-*c*).

In general, a clean looking squarewave at the output of an amplifier indicates a good frequency response over a 100-to-1 range. For example, an amplifier that cleanly reproduces a 1-kilohertz squarewave should provide good performance from 100 hertz to 10 kilohertz. Be aware, however, that a squarewave test won't detect clipping in the amplifier.

I have already mentioned tuning musical instruments. Another application is testing wow and flutter in tape recorders. Generate a constant frequency and record it; then play it back while comparing it to the same frequency coming directly from the computer. You can do the comparison by ear, but it's better to use an oscilloscope, with one squarewave going to the external sync input and the other to the vertical input. Then look closely to see how much the waveform jiggles from side to side. Ω

LISTING 1—BASIC PROGRAM

```
100 PRINT "PC Square Wave Generator - M. Covington 1994"
110 INPUT "Frequency (Hz)? ", FREQ
120 ADDR% = &H3F8                          ' 2F8 for COM2
130 PARM$ = "COM1:4800,N,8,1,cs0,ds0"  ' or "COM2..." etc.
140 '
150 ' Open the serial port
160 OPEN PARM$ FOR OUTPUT AS #1
170 '
180 ' Choose customized baud rate
190 DIVIDEND = 115200                      ' 111850 on PCjr
200 DIVISOR% = INT(.5 + DIVIDEND / (FREQ * 2))
210 FREQ = DIVIDEND / (DIVISOR% * 2)
220 PRINT "Actual frequency: "; FREQ; " Hz"
230 '
240 ' Set serial port to new baud rate
250 WAIT ADDR% + 5, &H20
260 OUT ADDR% + 3, INP(ADDR% + 3) OR &H80
270 OUT ADDR%, DIVISOR% MOD 256
280 OUT ADDR% + 1, DIVISOR% / 256
290 OUT ADDR% + 3, INP(ADDR% + 3) AND &H7F
300 '
310 ' Transmit square wave until told to stop
320 PRINT "Press any key to stop."
330 WHILE INKEY$ <> "": WEND               ' clear kbd buffer
340 WHILE INKEY$ = ""
350   PRINT #1, "UUUU";
360 WEND
370 PRINT "Emptying buffer..."
380 CLOSE #1
390 PRINT "All done."
400 END
```

LISTING 1-BASIC PROGRAM

```
100 PRINT "PC Square Wave Generator - M. Covington 1994"
110 INPUT "Frequency (Hz)? ", FREQ
120 ADDR% = &H3F8                          ' 2F8 for COM2
130 PARM$ = "COM1:4800,N,8,1,cs0,ds0"  ' or "COM2..." etc.
140 '
150 ' Open the serial port
160 OPEN PARM$ FOR OUTPUT AS #1
170 '
180 ' Choose customized baud rate
190 DIVIDEND = 115200                      ' 111850 on PCjr
200 DIVISOR% = INT(.5 + DIVIDEND / (FREQ * 2))
210 FREQ = DIVIDEND / (DIVISOR% * 2)
220 PRINT "Actual frequency: "; FREQ; " Hz"
230 '
240 ' Set serial port to new baud rate
250 WAIT ADDR% + 5, &H20
260 OUT ADDR% + 3, INP(ADDR% + 3) OR &H80
270 OUT ADDR%, DIVISOR% MOD 256
280 OUT ADDR% + 1, DIVISOR% / 256
290 OUT ADDR% + 3, INP(ADDR% + 3) AND &H7F
300 '
310 ' Transmit square wave until told to stop
320 PRINT "Press any key to stop."
330 WHILE INKEY$ <> "": WEND               ' clear kbd buffer
340 WHILE INKEY$ = ""
350   PRINT #1,"UUUU";
360 WEND
370 PRINT "Emptying buffer..."
380 CLOSE
390 PRINT "All done."
400 END
```