Computer Bits

By Hal Chamberlain

COMPUTER MUSIC—PART II

LAST MONTH we discussed computer music techniques in general and simple timed-loop techniques specifically. Some short, illustrative programs for an 8080-based microcomputer were given. Also a circuit for a simple 8-bit digital-to-analog converter was shown. Now, we will delve a little deeper into computer music techniques that have the potential for producing complex, serious musical results.

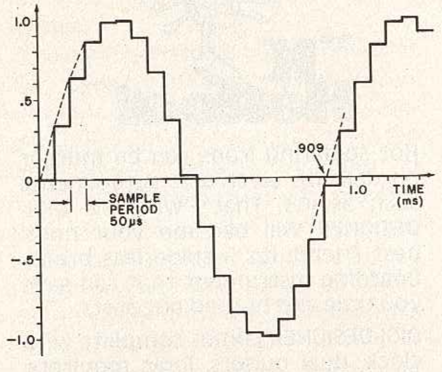


Fig. 1. Sampled sine wave.

The Sampling Theorem. Any waveform, no matter how simple or complex, can be represented as a series of discrete voltage values such as might come from a digital-to-analog converter (DAC). Figure 1 shows a sine wave as it might appear at the DAC output. This is termed a "sampled representation" because the sine-wave voltage is sampled at discrete points in time and held until the following sample point. Obviously this is a very poor sine wave, a fact that is easily demonstrated with a distortion analyzer.

Before giving up, let us look at the frequency spectrum of this staircase-like wave on a spectrum analyzer. To be specific, the sine-wave frequency is approximately 1.1 kHz, corresponding to 0.909 milliseconds per cycle. The sampling frequency, often called sampling rate, is 20 kHz or one sample every 50 microseconds. The spectral plot shows a strong frequency component at 1.1 kHz, which is the desired sine wave we are trying to synthesize. Also shown are the distortion product frequencies caused by the sampling process. Since all of the distortion components are much higher in frequency than the desired signal, they may be attenuated or removed with a sharp low-pass filter. After filtering,the distortion analyzer will confirm that a smooth, pure, sine wave is all that remains.

What will happen if the sine-wave frequency is increased but the sampling frequency remains the same? With fewer samples on each sine-wave cycle, the waveform from the DAC will appear to be even more distorted. Close examination of the first pair of distortion components in Fig. 2 will reveal that they are very much like sidebands of a 20-kHz suppressed carrier "modulated" by a 1.1-kHz "signal." The lower sideband frequency is the carrier (20 kHz) minus the signal (1.1 kHz) or 18.9 kHz. The upper sideband frequency is the sum, or 21.1 kHz. There are also sideband pairs at harmonics of the sampling frequency. If the sine-wave frequency is increased, the lowest distortion component will move downward toward it, leaving less room for the low-pass filter to do its work. The limit occurs when the desired frequency and the lowest distortion frequency actually meet each other at 10 kHz and can no longer be separated with the filter. The rule is that the highest frequency that can be reproduced with a sampled waveform is one-half of the sampling frequency. Actually achieving this requires an infinitely sharp filter; a more practical figure is ¼ or ⅓.

Of course a real digital-to-analog converter cannot generate voltages that are exact samples of the sine wave. An 8-bit converter, for example, has only 256 possible output voltage values. When a particular voltage is needed, the nearest available value will have to be used. This "round-off" error gives rise to another type of distortion in sampled waveforms called *quantization noise* which is spread throughout the frequency spectrum. The theoretical signal-to-quantization noise ratio is easily computed as 6N+4 dB, where N is the number of bits in the DAC. Actually this assumes an ideal DAC; a realistic figure is about 5 dB less. Still, an 8-bit DAC yields nearly 50 dB, as good as many tape recorders. With a 12-bit DAC, the quantization noise is negligible.

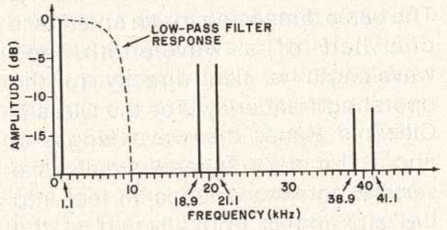


Fig. 2. Spectrum of Fig. 1.

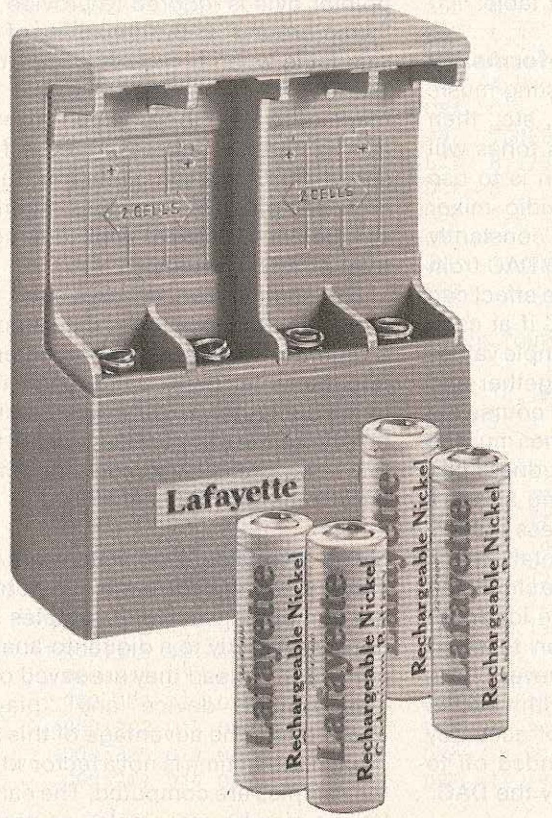
Waveforms From Tables. With this background, it is apparent that a sampled representation of a waveform may be stored in the memory of a computer. A simple way to do this is to store one cycle of the waveform in a small block of memory as a "table." Now the waveform can be reproduced by having a program scan through the table in memory and send the samples to a DAC. The frequency of the waveform is a function of the sample rate (time between sending out successive samples) and the number of points tabu-

FIG. 3

```
*       TOUCH-TONE TRANSMIT SUBROUTINE
*       ENTER WITH DIGIT TO SEND IN A, 4 BIT BINARY CODED DECIMAL
*       EXITS WITH ALL REGISTERS DESTROYED
*       ASSUMES 2 MHZ CLOCK AND NO MEMORY WAITS
*       SENDS TONES FOR 200 MILLISECONDS, SILENCE FOR 100 MS

000:200                ORG  200Q
000:200 346 017 TTXMIT ANI  017Q           MASK OFF EXCESS BITS IN A
```

(Subroutine continued on page 91)



lated for one cycle. If the sample rate is varied, care must be taken to prevent it from becoming too low.

A better way is to alter the apparent table length by scanning every second, third, etc. entry during the output process. This restricts us to a few specific frequencies. Any frequency may be generated by allowing "fractional" scan increments. When fetching a table entry, the nearest one would be used but the fractional part would be retained when computing the location of the next table entry. A prime advantage of waveform tables is that the tone color can be changed simply by using a different table.

Mixing Computed Waveforms. If a computer is to play interesting music with chords, counterpoint, etc., then two or more simultaneous tones will be required. One approach is to use multiple DAC's and an audio mixer along with a program to constantly feed sample values to each DAC from tables in memory. The same effect can be achieved with one DAC if at each sample time the current sample values for all tones are added together and the sum sent to the DAC. Of course the sample rates of all of the tones must be the same. Even relative loudness can be controlled by multiplying a tone's sample value by a "loudness factor" before it is added to the total. Attack and decay envelopes can be simulated by constantly changing the loudness factor. Care must be taken to avoid arithmetic overflows, however. Typically, the intermediate arithmetic is done to 16 or more bits of accuracy and the final result is rounded off to the number of bits used by the DAC.

Touch-Tone Program. Many of the concepts just discussed are illustrated by the Touch-Tone ^{R} transmit subroutine in fig. 3. A Touch-Tone digit consists of two simultaneous sine-wave tones, one from a low-frequency group (697, 770, 852, 941 Hz) and one from a high-frequency group (1209, 1336, 1477, 1633 Hz). The routine actually synthesizes the two tones already mixed together through an output port equipped with the simple DAC mentioned earlier.

The routine is called with register A containing the digit to be transmitted in binary. Using the binary code, the routine accesses a frequency table to determine what the "table increment" value for each tone should be. For example if a "4" is to be sent, every tenth sine-table entry would be used for the 770-Hz tone and every fifteenth for the 1209-Hz tone. Note that this gives only approximate frequencies; the 1209-Hz tone is off 3%, but the others are off less than 1.5%.

For the actual tone-generation task, the routine maintains two sine-table pointers, one for each tone. To compute a mixed-tone sample, the sine-table entry pointed to by each pointer is fetched, they are added together, and the sum is divided by 2 to produce an 8-bit result. For the next sample, the corresponding increment is added to each pointer. Overflow of the lower pointer byte is ignored to provide for "wrap-around" to the beginning of the sine table when a cycle is completed. The loop for computing samples is 100 machine cycles long, which gives a 20-kHz sample rate assuming a full-speed 8080 system. With this fairly high sample rate, low-pass filtering can be accomplished with tone controls or a scratch filter.

The routine can be expanded for more simultaneous tones if desired or a different waveform can be entered into the table. There is a definite limit to the computation between samples, otherwise the sample rate may fall too low to be useful. Also, the loop times have to be carefully controlled.

Mass Storage And Playback. In music programs for large computers, the computed waveform samples are not sent directly to a digital-to-analog converter. Instead they are saved on a mass-storage device and "played back" later. The advantage of this approach is that time is not a factor when the samples are computed. The calculations may be as complex as necessary or a high-level language can be used with no effect on the sample rate during playback. This method has perfect generality; any possible sound or combination of sounds can be synthesized, subject only to frequency-response limitations imposed by the playback sample rate.

The problem for hobbyists of course is standing the expense of a suitable high-speed, large-capacity mass-storage device. Interesting experiments can be performed however with systems having 16k or more of memory. At a 10-kHz sample rate, which gives AM radio quality, a 24k machine can hold over 2 seconds of sound. For the industrious, two-second segments can be recorded on audio tape and spliced together for the final result◆

```
000:202 041 271 000          LXI  H,TTTABL        GET ADDRESS OF FREQUENCY TABLE
000:205 207                  ADD  A               DOUBLE CONTENTS OF A
000:206 205                  ADD  L               ADD RESULT TO TABLE ADDRESS
000:207 157                  MOV  L,A             MOVE FIRST TABLE ENTRY
000:210 176                  MOV  A,M             TO TONE A INCREMENT
000:211 062 331 000          STA  TONAIN
000:214 043                  INX  H               MOVE SECOND TABLE ENTRY
000:215 176                  MOV  A,M             TO TONE B INCREMENT
000:216 062 332 000          STA  TONBIN
000:221 001 240 017          LXI  B,7640Q         SET COUNT FOR 200 MILLISECONDS IN BC
000:224 021 000 001          LXI  D,SINET         SET DE TO POINT TO SINE TABLE
000:227 041 000 001          LXI  H,SINET         SET HL TO POINT TO SINE TABLE
000:232 032           TONES  LDAX D               GET TONE A SAMPLE FROM SINE TABLE
000:233 206                  ADD  M               ADD TO IT TONE B SAMPLE
000:234 037                  RAR                  DIVIDE SUM BY 2
000:235 323 XXX              OUT  (port address)  SEND TO OUTPUT PORT WITH 8 BIT DAC
000:237 072 331 000          LDA  TONAIN          GET TONE A INCREMENT
000:242 203                  ADD  E               ADD TO IT TONE A SINE TABLE POINTER
000:243 137                  MOV  E,A             UPDATE TONE A POINTER
000:244 072 332 000          LDA  TONBIN          GET TONE B INCREMENT
000:247 205                  ADD  L               ADD TO IT TONE B SINE TABLE POINTER
000:250 157                  MOV  L,A             UPDATE TONE B POINTER
000:251 013                  DCX  B               DECREMENT AND CHECK TONE DURATION COUNT
000:252 170                  MOV  A,B
000:253 261                  ORA  C
000:254 302 232 000          JNZ  TONES           LOOP FOR ADDITIONAL SAMPLES UNTIL DONE
000:257 001 100 037          LXI  B,17500Q        SET COUNT FOR 100 MILLISECONDS OF SILENCE
000:262 013           SILENC DCX  B               DECREMENT AND CHECK SILENCE DURATION COUNT
000:263 170                  MOV  A,B
000:264 261                  ORA  C
000:265 302 262 000          JNZ  SILENC          LOOP UNTIL COUNT RUNS OUT
000:270 311                  RET                  RETURN

                      *      TOUCH-TONE FREQUENCY TABLE, TWO VALUES PER ENTRY

000:271 014 021       TTTABL DEF  12,17           0   941  1336
000:273 011 017              DEF  9,15            1   697  1209
000:275 011 021              DEF  9,17            2   697  1336
000:277 011 023              DEF  9,19            3   697  1477
000:301 012 017              DEF  10,15           4   770  1209
000:303 012 021              DEF  10,17           5   770  1336
000:305 012 023              DEF  10,19           6   770  1477
000:307 013 017              DEF  11,15           7   852  1209
000:311 013 021              DEF  11,17           8   852  1336
000:313 013 023              DEF  11,19           9   852  1477
000:315 011 025              DEF  9,21            A   697  1633
000:317 012 025              DEF  10,21           B   770  1633
000:321 013 025              DEF  11,21           C   852  1633
000:323 014 025              DEF  12,21           D   941  1633
000:325 014 017              DEF  12,15           *   941  1209
000:327 014 023              DEF  12,19           #   941  1477

000:331               TONAIN DST  1               STORAGE FOR TONE A INCREMENT
000:332               TONBIN DST  1               STORAGE FOR TONE B INCREMENT

*    SINE TABLE FOR USE WITH TOUCH TONE SUBROUTINE
*    MUST BE AT A PAGE BOUNDARY
*    LISTED IN MEMORY DUMP FORMAT TO CONSERVE SPACE

001:000  200 203 206 211 214 217 222 226 231 234 237 242 245 250 253 256
001:020  261 263 266 271 274 277 301 304 307 311 314 316 321 323 326 330
001:040  332 334 336 341 343 345 346 350 352 354 355 357 361 362 363 365
001:060  366 367 370 371 372 373 374 375 375 376 376 377 377 377 377 377
001:100  377 377 377 377 377 377 376 376 375 375 374 373 372 371 370 367
001:120  366 365 363 362 361 357 355 354 352 350 346 345 343 341 336 334
001:140  332 330 326 323 321 316 314 311 307 304 301 277 274 271 266 263
001:160  261 256 253 250 245 242 237 234 231 226 222 217 214 211 206 203
001:200  200 175 172 167 164 161 156 152 147 144 141 136 133 130 125 122
001:220  117 115 112 107 104 101 077 074 071 067 064 062 057 055 052 050
001:240  046 044 042 037 035 033 032 030 026 024 023 021 017 016 015 013
001:260  012 011 010 007 006 005 004 003 003 002 002 001 001 001 001 001
001:300  001 001 001 001 001 001 002 002 003 003 004 005 006 007 010 011
001:320  012 013 015 016 017 021 023 024 026 030 032 033 035 037 042 044
001:340  046 050 052 055 057 062 064 067 071 074 077 101 104 107 112 115
001:360  117 122 125 130 133 136 141 144 147 152 156 161 164 167 172 175
```