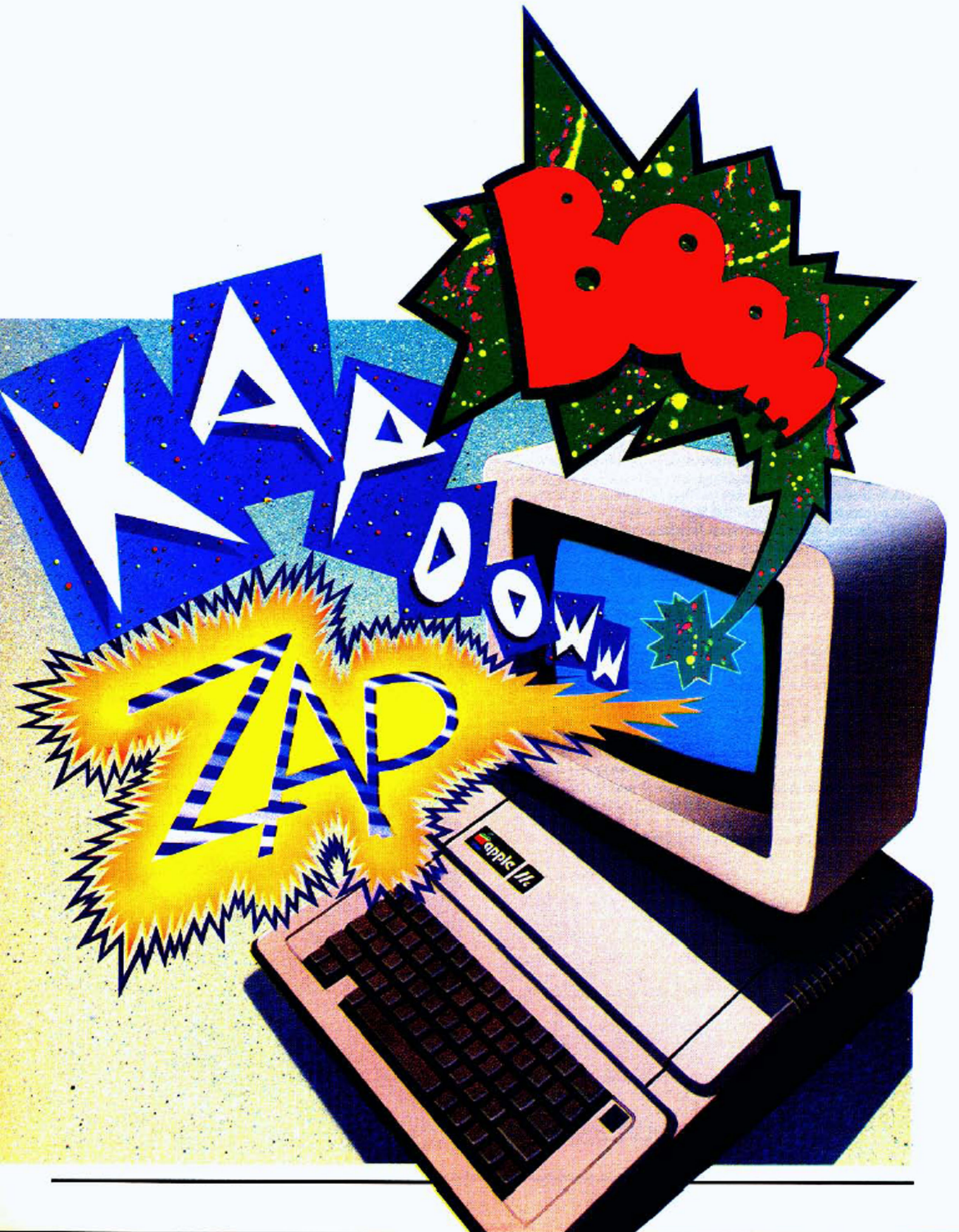
BOOM
KAPOOW
ZAP
apple II

ARCADE SOUND EDITOR

FEATURE ARTICLE

Create your own *two-pitch sound sequences and save them in convenient sound tables. Then use ampersand commands in your own program to control them.*

by Stephen H. Zimmerman and S. Scott Zimmerman

Have you ever wanted your Apple to go ZAP!, BOOM! or KAPOWEE!, but could only get click, buzz or beep? Have you ever dreamed of dual-pitched synthesizer sounds in your Applesoft program, but thought it was impossible? If so, then Arcade Sound Editor (ASE) is for you.

Arcade Sound Editor is a utility for creating, testing and editing sound tables. A sound table, similar to a shape table, is a binary file that can be saved to disk and loaded into your Applesoft programs. A sound table contains individual sounds and sound sequences that can be accessed by simple ampersand (&) commands.

The unique feature of ASE is that each sound can consist of two pitches played simultaneously to produce a musical chord or a sound effect. Moreover, ASE allows either or both pitches to vary as the sound is played to produce a wide range of interesting and dramatic sound effects.

ASE allows a sound table to have up to 255 sound sequences, each with up to 255 sounds. Of course, memory limitations in Apple II series computers restrict the actual size of your sound table. Typically, games and educational programs require three to ten sound sequences with one to ten sounds in each; therefore, your sound table will rarely contain more than 100 different sounds.

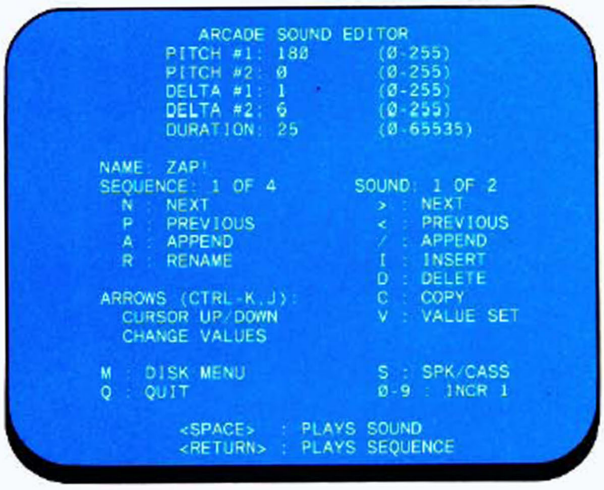

FIGURE 1: ASE's Edit Screen

ASE TUTORIAL

To start Arcade Sound Editor (**Listing 1**), type **RUN ASE** from 40-column Applesoft BASIC. A title page will appear; two binary files, **DUO** (**Listing 2**) and Sound Editor Utility (**Listing 3**), are loaded into memory; and the message "Press Return to Start" is printed at the bottom of the screen. After you press Return, the edit screen is displayed (**Figure 1**).

ASE assumes that you want to start by inputting a sound sequence; therefore, it puts the cursor next to the "NAME:" label, so you can name it. For now, press Return. We will come back to the Name feature later.

Now, the zero after pitch #1 is shown in inverse, indicating the location of the edit cursor. You can move the edit cursor by pressing the Up-Arrow key (or Control-K on an Apple II Plus) to move it up, and the Down-Arrow key (or Control-J) to move it down. Press these keys several times to see the cursor move through the values of the pitch, delta and duration. All edit commands and the current status of the edit features are shown.

In addition, the edit screen displays the sound number currently being edited, the number of sounds in the current sound sequence, the current sound sequence, and the number of sound sequences in the sound table. For example:

SEQUENCE: 2 OF 3 SOUND: 1 OF 5

would indicate that the sound table has three sound sequences, you are currently editing sequence 2, which has five sounds, and you are currently editing sound 1. Of course, when you first start ASE, the sequence and sound numbers are all ones, indicating that you are editing sound 1 of sound sequence 1.

Each sound that you create requires four parameters to specify the tones and one to specify the duration (see **Table 1**).

Changing Pitch Parameters

When you first run ASE, the program automatically initializes one sound with all of the tone parameters set to zero. To change a parameter, move the edit cursor to the parameter by pressing the Up- or Down-Arrow key (or by pressing Control-K or Control-J), then press the Right- or Left-Arrow key to increase or decrease that parameter's value.

Another way to adjust the values is to position the cursor at the parameter and press V (for value set). The input (flashing) cursor then appears in place of the edit cursor. Type a number within the allowable range (use the Right- and Left-Arrow keys to correct typing errors) and press Return. If you accidentally press the Up- or Down-Arrow, you can restore the screen by pressing Return.

Let's do some examples. Move the edit cursor to pitch #1 and press the Right-Arrow key enough times to increase the value to 25. Press the Down-Arrow key (or Control-J) to move the edit cursor to duration. Then press the Right-Arrow key until the value is 10. Having set at least one of the pitches and the duration to a nonzero value, you are now ready to play a note. Press the Space bar. You should hear a moderately low pitch with a short duration.

Now move the cursor back up to pitch #1. Press the Right-Arrow key once to increment the pitch to 26, then press the Space bar again. Repeat this process to hear the pitch slowly increase in frequency.

Changing the Duration Parameter

To demonstrate how the duration value affects the length of the tone, move the cursor to the duration parameter and use the Right- and Left-Arrow keys to change the value several times, pressing the Space bar after each change. You can also set the duration value by pressing V, then typing the number of the value. Press the Space bar to hear the sound with the new duration.

If you set the duration to a very large value and press the Space bar to play the sound, you may have to wait a long time to regain control of your computer. Instead of waiting, you can press Control-Reset to break out of the program. To re-enter ASE for this or any other reason without erasing the sound table in memory, type **GOTO 260**.

Dual Pitches

Because of the way the sound routine (DUO) works, the two pitches in a dual-pitch tone are *not* independent. They do not behave like two notes that are played on a piano. Rather, they interact in such a way that the overall pitch and timbre (tone quality) depend upon the relative values of pitch #1 and pitch #2 in a somewhat unpredictable way. Therefore, ASE makes a poor music editor. (If you want computer music, see our T.U.N.E.S. program, published in *Nibble* Vol. 4/No. 7.) However, it is because of the interaction between the two pitches that ASE can produce a wide variety of sound effects.

Let's look at a few examples of dual-pitch sounds. Set pitch #1 to 25 and its duration to 200. Then move the edit cursor to pitch #2 and adjust this parameter to 26. Now press the Space bar to make the sound. Instead of a single, clear tone, you now hear a synthesizer-like dual tone.

TABLE 1: Tone Parameters

Parameter	Description
Pitch #1	Is the first pitch of a two-pitch tone. Its value must be in the range 0-255. A zero indicates a rest (no pitch). A one produces a tone of very low frequency. The highest value, 255, produces a tone of very high frequency.
Pitch #2	Is the second pitch of a two-pitch tone. Pitches #1 and #2 are played simultaneously, not sequentially. Pitch #2 has the same range as pitch #1 and functions in exactly the same way.
Delta #1	Is the change in pitch #1. Each of the two pitches can change in value while playing. This change is called the delta value, which must be in the range 0-255. A delta value of 0 means there is no change in pitch. A value of 1 increases the frequency of the pitch as it plays. A value of 255 decreases the frequency of the pitch as it plays (think of 255 as −1 because it decreases the pitch by 1). The more you go in either direction, the more the pitch changes. The best way to understand the effect of a delta value is to experiment. Some specific examples are given below.
Delta #2	Is the change in pitch #2. Delta #2 has the same function and range of values as delta #1, except that it applies to pitch #2.
Duration	Is the duration (length) of the sound. Its range starts at 0 (no duration, and hence no sound). A value of 1 will usually make a little click, depending on the pitches. The highest number, 65535, will create a six-minute tone. Unlike in some routines, the duration value in ASE is independent of the pitch; that is, all tones with the same duration value make a sound for the same length of time.

Now press the Right-Arrow key once to increase pitch #2 to 27. Press the Space bar to hear the new tone. The new sound has a more hollow or echo-like tone. If both pitches are set to 25, on the other hand, the sound is very weak (or, because of the way the Apple speaker works, totally silent). When pitch #2 is set to 28, the tone is still different: you hear a scratchy sound which decreases and then increases in volume. Because of the way the Apple's speaker works, if both pitches are set to the same number, the sound is very weak or no sound at all is played. Unless you want dual-pitch sound, leave pitch #2 at zero.

Try various pitch combinations to explore the sound effects capabilities of ASE. Some other examples are included later on.

Changing the Delta Parameters

To see how the delta function works, start with a new sound by pressing the slash character (/) key (append sound). The display indicates that the current sound is now sound 2 of 2 and that all of its parameters are zero. Set pitch #1 to 10 and its duration to 200. Move the cursor to delta #1 and set its value to 1. When you press the Space bar, the sound's pitch progressively increases in frequency.

Now press the Left-Arrow key twice to change the delta #1 value from 1 to 255, and press the Space bar to make the new sound. This time, the frequency decreases as the sound is played. You hear two separate phases of the sound because the pitch starts at 10, decreases to 0, starts again at 255 and decreases from there throughout the duration of the tone. To demonstrate this, change pitch #1 to 255 (leaving the delta value still at 255). In this case, the tone frequency starts high enough that the pitch value never reaches zero.

Creating a Sound Table

If you have been doing the examples in the ASE tutorial, you should have several sounds in your current sound table. Clear these

from memory by typing Q for quit, but when you are asked "DO YOU WISH TO QUIT (Y/N)?" press N for no. At the "CLEAR THE TABLE (Y/N)?" prompt, press Y to clear the current sounds from memory. ASE will automatically return you to the edit screen.

If you are not currently in ASE, type **RUN ASE** to start the program.

We will create a sound table with four sound sequences. Sequences 1 and 2 will have two sounds each, sequence 3 will have three sounds, and sequence 4 will have four sounds. Press Return at the "NAME:" prompt, then perform the following steps:

1. Set sound 1 of 1 to:

   Pitch #1: 180
   Pitch #2: 0
   Delta #1: 1
   Delta #2: 6
   Duration: 25

   Move the edit cursor with the Up- and Down-Arrow keys (or Control-K and Control-J) to the parameter, and use the Right- and Left-Arrow keys or the value set (V) method to adjust the values of the parameters.

2. Since the current sequence has only one sound, append the second sound by pressing the slash character (/) key. The edit screen will display "Sound 2 of 2," with all of the parameters set to zero. Change the parameters as follows:

   Pitch #1: 210
   Pitch #2: 209
   Delta #1: 1
   Delta #2: 0
   Duration: 10

   You have completed typing in sequence 1. Press the Space bar to play each sound individually, and press Return to play the sequence. This sequence is one way of producing a "ZAP!" sound.

3. Name this sequence by pressing R (for rename). When the flashing cursor appears next to the "NAME:" prompt, type **ZAP!** and press Return. This completes the ZAP! sound sequence.

4. To create sequence 2, press A to append a new sequence. ASE will show a flashing cursor next to the "NAME:" prompt. Enter the name **BOOM!** The display now indicates that you are editing sequence 2 of 2, sound 1 of 1. Adjust the sound parameters to 252, 250, 1, 2 and 60, respectively. Press the Space bar to play the sound.

FIGURE 2: ASE's Disk Menu Screen

```
                DISK MENU

C : CATALOG

D : CHANGE CURRENT DRIVE (1)

S : SAVE SOUND TABLE

L : LOAD SOUND TABLE

    FILENAME: DEMO

          (80 BYTES LONG)




     <ESCAPE>   BACK TO EDIT
```

5. The next sound in the sequence is similar to the first, so press C to copy the current sound to the end of the sequence. When you press C, the screen shows that you are now editing sound 2 of 2, but all of the parameters are the same as in sound 1. In sound 2, we want to change pitch #1 from the current 252 to 250. With the edit cursor at pitch #1, press the Left-Arrow key twice. Now move the edit cursor down two places (to delta #1) and press the Left-Arrow key twice to make the value 255. Move the cursor down another place and set the delta #2 value to 254 by pressing the Left-Arrow key four times. Finally, set the duration to 130. Press the Space bar to hear sound #2, then press Return to hear the sequence. This completes the sequence "BOOM!"

6. Sequence 3 contains three sounds. Press A to append a sequence, type the name **KAPOWEE!**, and then set the first sound parameters to 200, 199, 1, 6 and 30. Press '/' to append the next sound and set the parameters to 250, 250, 255, 254 and 20. Press '/' again, and set the sound parameters to 214, 213, 255, 255 and 200. You have just finished the sequence "KAPOWEE!"

7. To do the final sequence, press A to append a new (blank) sequence. Enter the name **SIREN**. Adjust the sound parameters of the first sound to 170, 0, 0, 0 and 130. Press C three times, since the other three sounds in the sequence are very similar to the first. Now change sound 2, pitch #1 from 170 to 190. Do the same with sound 4. When you press Return, you should hear four clear tones that sound like a European siren.

8. To save this sound table to disk so that we can use it in a later program, press M to display the Disk menu. You will see a screen that looks like **Figure 2**. Press S to save the sound table. At the file name prompt, type DEMO. The ASE program automatically appends ".SNDS" to the end of your file name. This not only helps you to recognize sound files on your disk but also ensures that ASE loads the right type of file. Press S to save the table. ASE will access the disk to see if there is already a file named DEMO.SNDS. If so, you are warned that the file will be overwritten. Press the Space bar to save the file or press Escape to cancel the save.

9. Press C for catalog to make sure the file was saved. The catalog should contain two new files, DEMO.SNDS and DEMO .NMS. The first is the actual sound table, and the second is a text file containing the list of sequence names. After cataloging the disk, ASE prompts for a disk command. To execute an additional disk command such as DELETE, LOCK or UNLOCK, type in the command at this point. If the disk command contains a comma (e.g., the RENAME command), enclose the entire command in double quotation marks. If you don't want to execute an additional disk command, just press Return.

10. Press Escape to get out of the disk menu and return to the edit screen. Press Q to quit ASE, and Y when you are asked if you really wish to quit.

This completes the ASE tutorial. Spend some time trying each of the commands to understand how they work. With practice you will be creating your own sound tables.

ASE Commands

Let's look at the ASE commands listed on the edit screen in more detail.

Up-Arrow or Control-K — Moves the edit cursor up. If the cursor is at pitch #1, the first sound parameter, pressing the Up-Arrow key causes the cursor to wrap around to duration, the last sound parameter.

Down-Arrow or Control-J — Moves the edit cursor down. If the cursor is at duration, the last sound parameter, pressing the Down-

Arrow key causes the cursor to wrap around to pitch #1, the first sound parameter.

> or comma (,) — Goes to the next sound in the sequence. If there is only one sound in the current sequence, pressing the right angle bracket (>) or the comma (,) has no effect. Otherwise, the next sound in the sequence becomes the current sound.

< or period (.) — Goes to the previous sound in the sequence. This works similarly to >, except in the opposite direction.

/ — Appends a new sound to the current sequence. This adds a sound whose parameters are all zero to the end of the current sequence.

I — Inserts a new sound in the current sequence. This pushes all the sounds in the current sequence back one, and opens up space for a new sound at the current sound position.

D — Deletes the current sound from the current sequence. If there is only one sound in the sequence, the entire sequence is deleted from the edit sound table.

C — Copies the current sound to the end of the current sequence. This works the same as the append command (/), except that the new sound has the same parameters as the current sound.

V — Sets the value of the parameter at the edit cursor. This is an alternative to using the Right- and Left-Arrow keys to adjust the parameter values. When you press V, a flashing input cursor appears. Type in the value number and press Return. Do not press the Up- and Down-Arrow keys until you have pressed Return.

S — Toggles between speaker mode and cassette mode. Speaker mode plays the sounds through the Apple's speaker. Cassette mode sends the sound signal out through the Apple's cassette port. For an Apple II Plus or IIe, connect the cassette port to the tape-in port of the amplifier of your stereo system and select tape output from the amplifier. Do not try to connect the cassette output port directly to your stereo speakers! To use an Apple IIc with external speakers, connect the earphone output directly to your speakers and keep DUO in normal mode (do *not* use & POP in this case).

Right-Arrow — Increases the value of a sound parameter by the increment amount. The default increment is 1. You can set the increment to a different value by pressing a number key (see below). If pressing the Right-Arrow key would increment the parameter past its maximum allowable value (255 or 65535), the value wraps around to zero.

Left-Arrow — Decreases the value of the sound parameter by the increment amount. If pressing the Left-Arrow key would decrease the parameter below zero, the value wraps around to 250 if you are editing the pitches or duration, or to 255 if you are editing the deltas. (We chose to go back to 250 for some of the parameters in order to preserve round numbers. Higher values of those parameters must be set with the Right-Arrow key or the V command).

0-9 — Sets the increment value. If you press a number from 1 through 9, the increment value is set to that number. If you press 0, the increment value is set to 10.

N — Goes to the next sound sequence within a sound table. If there is only one sequence in the edit sound table, this command has no effect. Its function is analogous to >, which goes to the next sound within a sequence.

P — Goes to the previous sound sequence. This command is similar to N, but moves you in the opposite direction.

A — Appends a new sound sequence. This command allows you to start a new sound sequence.

R — Renames the sequence. This command lets you rename the sequences in your table. It allows a maximum of 15 characters in each name. Any keyboard character, except a control key, is allowed in the sequence name.

M — Selects DOS menu mode. This lets you load or save a sound table and perform other disk operations.

Q — Quits ASE with the option of clearing the current sound table. After pressing Q, type Y or N after the prompts to quit ASE or to clear the sound table.

Disk Menu Commands

Figure 2 shows the Disk menu commands. To display the Disk menu, press M from edit mode. The functions of the Disk commands are described in the ASE Tutorial section.

USING DUO

The heart of the Arcade Sound Editor system is the assembly language program DUO (**Listing 2**), which lets you play back your sounds with five powerful ampersand commands. Using DUO in your Applesoft BASIC program requires five steps:

1. Using ASE, create a sound table containing the sequences that you want in your program. This will produce two files, the actual sound file with a '.SNDS' suffix and a text file containing the names (.NMS suffix).
2. BLOAD the sound table into any place in memory that does not interfere with your program. A system for selecting the BLOAD address is explained below. You do not need to read the text file containing the sound sequence names, and you will usually *not* use it. The list of sound sequence names was created only for reference in using ASE.
3. BRUN DUO from within your Applesoft program. For example, in **line 200** of **Listing 1**:

   ```
   200 PRINT CHR$ (4);"BRUN DUO,A$8400"
   ```

   we include the address to which DUO is BRUN. In most of your programs, you will want the DUO address to be higher in memory; for example, at $9400 (37888). DUO can run at any location in memory as long as it and your program do not interfere with one another. Furthermore, DUO uses memory locations $3C8-$3CF (968-975) in the page 3 user space. If you want to use a program or file that BLOADs into page 3 (e.g., many short, machine language programs use a CALL 768 to page 3), be sure that it doesn't require those eight bytes used by DUO.
4. Include a HIMEM and/or LOMEM command to protect DUO and your sound table from being overwritten by Applesoft variables and strings. Consult your *Applesoft BASIC Programmer's Reference Manual* to see how HIMEM and LOMEM are used. ProDOS requires that you set HIMEM at a memory page boundary; i.e., the HIMEM value must be an even multiple of 256. If you BLOAD your sound table and BRUN DUO high in memory near DOS or ProDOS, HIMEM should be set below the BRUN address of DUO. This will be explained later.
5. POKE the address of your sound table into memory locations 206 and 207. This is analogous to POKEing the address of a shape table into memory locations 232 and 233. The POKEs to 206 and 207 let DUO know where your sound data is located.

Now your program is ready to accept ampersand (&) commands for making sound effects.

Selecting Addresses

The following is a simple system for selecting the addresses for BLOADing your sound table, BRUNning DUO, setting HIMEM, and POKEing the sound table address.

TABLE 2: Ampersand Commands Recognized by DUO

Command	Function
& *n*	Plays sequence *n*. For example, if you have a sound table with three sound sequences and you want to play sequence 2, the command would be & 2. You can also use a variable or Applesoft expression instead of the actual number. For example, A = 2: & A also plays sequence 2.
& *n,m*	Plays sound *m* of sequence *n*, where *n* and *m* can be constants, variables or expressions. This command allows you to select and play individual sounds within a sequence.
& STOP	Stops (turns off) the sound. All ampersand commands following & STOP will be silent and have a duration of zero. This is used, for example, to select and deselect sound in a game.
& POP	Sends the sound to the cassette output port. This turns off the Apple's speaker and allows you to use external speakers by connecting the cassette output port to an audio amplifier. Another use of &POP is to turn off the sound but maintain the timing of the notes. An example would be in a game where the speed of animation is determined by the duration of the tones. In such a case, you may want to use & POP instead of & STOP, since & STOP sets the duration to zero.
&NORMAL	Returns sound output to normal. If the sound was turned off with & STOP or redirected to the cassette output port with & POP, you can restore the sound to normal with & NORMAL.

1. Set the initial HIMEM to 38400 ($9600). This is where ProDOS begins; it is also where DOS 3.3 begins if MAXFILES=3. For example, include in your program the statement:

```
120 HI = 38400: HIMEM: HI
```

   This is not critical in DOS 3.3, but in ProDOS it ensures that the BLOAD region is available; otherwise you may get a NO BUFFERS AVAILABLE error message. If you are using an Applesoft editor, such as MicroSPARC's GALE, you may want to set HIMEM to the highest memory address below the editor (check the manual that came with your editor).
2. Calculate the address for BLOADing your sound table, using the formula AS = HI − L − 1, where AS is the address for the sound table, HI is the initial HIMEM address, and L is the length of the sound table. The length of a sound table is shown in ASE's Disk menu. For example, include in your program the statement:

```
200 L = 80:AS = HI - L - 1: PRINT CHR$ (4);
    "BLOAD filename.SNDS,A";AS
```

   where filename is the name you supplied in ASE.
3. Calculate the address for BRUNning DUO by subtracting the length of DUO (445 bytes) from the address (AS) calculated in step 2. Your program will have, for example, the statement:

```
220 AD = AS - 445: PRINT CHR$ (4);"BRUN DUO,A";AD
```

4. If your program includes a shape table or other binary file, BLOAD it just below DUO at the address AX, where AX = AD − LB. (LB is the length of the binary file.)
5. Set the final HIMEM to the memory page boundary below the binary file with the lowest BLOAD address. If your only two binary files are the sound table and DUO, set HIMEM to the page boundary below AD. If your program runs under ProDOS and includes a command such as CAT or OPEN, make HIMEM 1024 bytes *below* the lowest BLOAD address. For example, include in your program a statement like this:

```
240 AD = 256 * INT (AD/256) - 1024: HIMEM: AD
```

TABLE 3: Beginning of Sound Table

Byte Offset	Description
0	The number of sound sequences in the table
1	Not used (usually set to zero)
2	Low-order byte of offset for sequence #1
3	High-order byte of offset for sequence #1
4	Low-order byte of offset for sequence #2
5	High-order byte of offset for sequence #2
.	Etc.
.	

   If you have an additional binary file in your program (step 4), use AX rather than AD to set HIMEM.
6. Tell DUO where in memory your sound table starts. POKE the low-order byte of AS (calculated in step 2) into memory location 206 and the high-order byte of AS into memory location 207. The easiest method for doing this is to define two functions that calculate high-order and low-order bytes of a number. For example, use the statement:

```
280 DEF FN HB(A) = INT (A / 256): DEF FN LB(A) =
A - FN HB(A) * 256
```

   and follow that statement with the actual POKEs:

```
300 POKE 206, FN LB(AS): POKE 207, FN HB(AS)
```

   Now the sound table and DUO are in memory and ready for use within your program.

Ampersand Commands

Once the sound table and DUO are properly installed in memory, your program can include appropriate ampersand (&) commands recognized by DUO. Five types of ampersand commands are available, as shown in **Table 2.**

USING DUO.DEMO

DUO.DEMO (**Listing 4**) demonstrates the use of sound tables and DUO in an Applesoft program. After you type **RUN DUO .DEMO**, the program title appears on the screen. The program then BLOADs the sound table DEMO.SNDS (which you created previously), reads the name file DEMO.NMS, and BRUNs the sound routine DUO. If you haven't created these files, the program will supply the data.

You will then see a menu of the four sounds in the table — ZAP!, BOOM!, KAPOWEE! and SIREN — and a list of other commands: Q to quit the program; S to stop the sound (by executing the & STOP command); P to execute the & POP command; and N to execute the & NORMAL command of DUO.

You can use DUO.DEMO to demonstrate other sound sequences. If your table has more than 14 sequences, you will have to revise DUO.DEMO to allow their names to fit on the screen. To use a table other than DEMO.SNDS, modify **lines 160 and 170 of Listing 4** to use the proper file length (L) and to BLOAD the proper sound file; modify **lines 160 and 190** to open the proper file of sequence names.

The main purpose of DUO.DEMO, of course, is to show you how to use DUO with your sound tables. The program Starlaser and its accompanying article in this issue also demonstrate the programs and methods explained here.

ENTERING THE PROGRAMS

To enter the programs, start by keying in **Listing 1** and saving it with the command:

SAVE ASE

TABLE 4: Data Structure of Sound Sequences

Byte Offset	Duration
0	Number of sounds in the sequence
1	Pitch #1 of sound 1
2	Pitch #2 of sound 1
3	Delta #1 of sound 1
4	Delta #2 of sound 1
5	Low-order byte of the duration of sound 1
6	High-order byte of the duration of sound 1
7	Pitch #1 of sound 2
8	Pitch #2 of sound 2
9	Delta #1 of sound 2
A	Delta #2 of sound 2
B	Low-order byte of the duration of sound 2
C	High-order byte of the duration of sound 2
.	Etc.
.	
.	

If you have an assembler, enter the source code from **Listing 2** and assemble it using the object file name DUO. If your assembler does not support macros, skip the macro definitions in lines 71-92. Macro source lines, which should be skipped if you don't have a macro assembler, are indicated with ">>>" in the mnemonic field. The following lines are the macro expansion and should be entered as shown if you aren't using a macro assembler. The lines with "<<<" indicate the end of a macro expansion and should not be entered. Note that the source line numbers will not match the listing if you aren't using a macro assembler. If you are using a macro assembler, you should enter only the macro source line, using the format appropriate for your assembler.

If you don't have an assembler, enter the Monitor with CALL −151 and key in the hex code. Save the program with the command:

BSAVE DUO,A$9400,L$1BD

If you have an assembler, see the comments on macros above and enter the source code from **Listing 3**. Assemble it and save the object file using the name SEU. If you don't have an assembler, enter the Monitor with CALL −151 and key in the hex code. Save the program with the command:

BSAVE SEU,A$8000,L$3C8

If you are using Key Perfect on an assembled object file, BLOAD the file, rename the file on disk, and BSAVE another copy using the command shown above. (This will shorten the file and remove any extraneous values stored in variable space by your assembler.) Run Key Perfect using this copy.

Finally, enter the Applesoft program shown in **Listing 4** and save it with the command:

SAVE DUO.DEMO

For help with entering *Nibble* listings, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

TECHNICAL NOTES

The source code is carefully annotated so that assembly language programmers can follow the program logic. We have made liberal use of Monitor and Applesoft ROM routines, as documented in the books *Apple II Monitors Peeled*, published by Apple Computer, and *All About Applesoft*, published by the Apple PugetSound Program Library Exchange (A.P.P.L.E.).

The data structure of a sound table is loosely based on the data structure of Applesoft shape tables. A sound table consists of two major parts, the index and the sound data. In **Table 3**, which shows the index structure, the offset refers to the number of bytes from the beginning of the file or the beginning of the file section.

Table 3 shows that the index portion of the table requires $2n+2$ bytes, where n is the number of sound sequences in the sound table. The two-byte offset for each sequence is the number of bytes from the beginning of the file to the start of the particular sound sequence data.

Each sound sequence has the data structure shown in **Table 4**, where the offset values are relative to the start of that set of sound sequence data. This means that each sound sequence requires $6m+1$ bytes of memory, where m is the number of sounds within the sound sequence.

LISTING 1: ASE

```
10  REM ************************
20  REM *          ASE           *
30  REM * ARCADE SOUND EDITOR    *
40  REM * BY S & S ZIMMERMAN     *
50  REM * COPYRIGHT (C) 1987     *
60  REM * BY MICROSPARC, INC.    *
70  REM * CONCORD, MA. 01742     *
80  REM *                        *
90  REM ************************
100  REM *-----------------
110  REM * INTRODUCTION:
120  REM *-----------------
130  HIMEM: 16384 - 512 * ( PEEK (48896) = 76
     ): TEXT : HOME : DIM N$(255)
140  VTAB 2:SP$ = " ARCADE SOUND EDITOR ": INVERSE
     : GOSUB 1040: NORMAL
150  VTAB 5:SP$ = "BY ": GOSUB 1040: VTAB 7:S
     P$ = "STEPHEN H. ZIMMERMAN": GOSUB 1040
160  VTAB 9:SP$ = "AND": GOSUB 1040: VTAB 11:
     SP$ = "S. SCOTT ZIMMERMAN": GOSUB 1040
170  VTAB 15:SP$ = "COPYRIGHT (C) 1987": GOSUB
     1040
180  VTAB 16:SP$ = "BY MICROSPARC, INC": GOSUB
     1040
190  VTAB 17:SP$ = "CONCORD, MA  01742": GOSUB
     1040: ONERR  GOTO 2860
200 EF = 1: PRINT  CHR$ (4);"BRUN DUO,A$8400"
210 EF = 2: PRINT  CHR$ (4);"BLOAD SEU"
220  POKE 216,0: VTAB 23: PRINT "  PRESS <RET
     URN> TO START -> ";: GET OP$: PRINT OP$:
      CALL 32768
230  REM *-----------------
240  REM * SET UP:
250  REM *-----------------
260  HOME : FOR J = 1 TO 5:VP(J) = J + 2: NEXT
     : POKE 206,0: POKE 207,64
270 IN = 1:PA = 1:PN = 1:PS = 1:CA$ = "CATALO
     G": IF  PEEK (48896) = 76 THEN CA$ = "CA
     T"
280 CS = 1: GOSUB 1820: POKE 800,1: POKE 801,
     1: CALL 32771
290 CN = 1: GOSUB 1310:SQ = 32786:CL = 32789:
     D = 1:SD$ = ".SNDS":NS$ = ".NMS"
300 IT = 32768:T2B = 32771:B2T = 32774:IS = 3
     2777:DE = 32780:SS = 32783
310  HOME : GOSUB 380: &  NORMAL : IF N$(CS) =
     "" THEN  GOSUB 1140
320  GOTO 570
330 OV =  ASC (OP$): IF OV > 95 THEN OV = OV -
     32:OP$ =  CHR$ (OV)
340  RETURN
350  REM *-----------------
360  REM * MAIN SCREEN:
370  REM *-----------------
380 SP$ = " ARCADE SOUND EDITOR ": VTAB 1: INVERSE
     : GOSUB 1040: NORMAL
390  VTAB 3: HTAB 7: PRINT "PITCH #1: ";PV(1)
     ;"    ": HTAB 7: PRINT "PITCH #2: ";PV(2
     );"    "
400  HTAB 7: PRINT "DELTA #1: ";PV(3);"    ":
      HTAB 7: PRINT "DELTA #2: ";PV(4);"    "
     : HTAB 7: PRINT "DURATION: ";PV(5);"
     "
410  FOR J = 1 TO 4: VTAB 2 + J: HTAB 26: PRINT
     "(0-255)": NEXT J: HTAB 26: PRINT "(0-65
     535)"
420  VTAB 9: PRINT "NAME: ";N$(CS);: CALL  -
     868: PRINT : PRINT "SEQUENCE: ";CS;" OF
     ";SE;"  "
```

```
430  PRINT "  N : NEXT": PRINT "  P : PREVIOU
     S": PRINT "  A : APPEND": PRINT "  R : R
     ENAME"
440  VTAB 16: PRINT "ARROWS (CTRL-K,J):": PRINT
     "  CURSOR UP/DOWN": PRINT "  CHANGE VALU
     ES"
450  VTAB 10: HTAB 24: PRINT "SOUND: ";CN;" O
     F ";NS;"  ";
460  VTAB 11: HTAB 26: PRINT "> : NEXT": VTAB
     12: HTAB 26: PRINT "< : PREVIOUS"
470  VTAB 13: HTAB 26: PRINT "/ : APPEND";: VTAB
     14: HTAB 26: PRINT "I : INSERT";: VTAB 1
     5: HTAB 26: PRINT "D : DELETE";: VTAB 16
     : HTAB 26: PRINT "C : COPY";
480  VTAB 17: HTAB 26: PRINT "V : VALUE SET";
     : VTAB 20: HTAB 26: PRINT "S : ";: IF  NOT
     MF THEN  INVERSE
490  PRINT "SPK";: NORMAL : PRINT "/";: IF MF
      THEN  INVERSE
500  PRINT "CASS": VTAB 20: NORMAL : PRINT "M
     : DISK MENU": PRINT "Q : QUIT";
510  VTAB 21: HTAB 26: PRINT "0-9 : INCR ";: INVERSE
     : PRINT IN: NORMAL
520  VTAB 23: HTAB 8: PRINT "<SPACE>  : PLAYS
      SOUND": HTAB 8: PRINT "<RETURN> : PLAYS
      SEQUENCE";
530  RETURN
540  REM *----------------
550  REM * MAIN LOOP:
560  REM *----------------
570  VTAB VP(PN): HTAB 17: INVERSE : PRINT PV
     (PN);: NORMAL : PRINT "    "
580  IF  PEEK ( - 16384) < 128 THEN 580
590  GET OP$: GOSUB 330
600  IF OP$ = " " OR OP$ =  CHR$ (13) THEN  GOSUB
     940: GOTO 570
610  IF OP$ =  CHR$ (8) THEN PV(PN) = PV(PN) -
     IN: GOSUB 810: GOTO 570
620  IF OP$ =  CHR$ (21) THEN PV(PN) = PV(PN)
      + IN: GOSUB 810: GOTO 570
630  IF  ASC (OP$) > 11 OR  ASC (OP$) < 10 THEN
     680
640  IF OP$ =  CHR$ (10) THEN PS = PN:PN = PN
      + 1: IF PN > 5 THEN PN = 1
650  IF OP$ =  CHR$ (11) THEN PS = PN:PN = PN
      - 1: IF PN < 1 THEN PN = 5
660  VTAB VP(PS): HTAB 17: NORMAL : PRINT PV(
     PS);"    "
670  GOTO 570
680  IF OP$ = "0" THEN IN = 10: GOTO 710
690  IF  VAL (OP$) < 1 THEN 720
700 IN =  VAL (OP$)
710  VTAB 21: HTAB 37: INVERSE : PRINT IN;: NORMAL
     : PRINT " ": GOTO 570
720  GOSUB 1370: POKE 800,CS: POKE 801,CN: CALL
     B2T
730 OP = OV - 43: IF OP < 0 THEN OP = 0
740  ON OP GOTO 1650,580,1650,1080,580,580,58
     0,580,580,580,580,580,580,580,580,580,16
     50,580,1650
750 OP = OV - 64: IF OP < 0 THEN OP = 0
760  ON OP GOTO 1080,580,1890,1510,580,580,58
     0,580,1440,580,580,1650,1970,1650,580,16
     50,2740,1270,1590,580,580,880,580,1650
770  GOTO 570
780  REM *----------------
790  REM * CHECK VALUES:
800  REM *----------------
810 MAX = 255:MX = 250: IF PN = 5 THEN MAX =
     65535
820  IF PV(PN) > MAX THEN PV(PN) = 0
830  IF PV(PN) < 0 THEN PV(PN) = MX: IF PN =
     3 OR PN = 4 THEN PV(PN) = MAX
840  RETURN
850  REM *----------------
860  REM * VALUE SET:
870  REM *----------------
880  VTAB VP(PN): HTAB 17: NORMAL : PRINT PV(
     PN);: HTAB 17: INPUT "";V$
890  IF  LEFT$ (V$,1) < "0" OR  LEFT$ (V$,1) >
     "9" THEN  HOME : GOSUB 380: GOTO 570
900 PV(PN) =  VAL (V$): GOSUB 810: HOME : GOSUB
     380: GOTO 570
910  REM *----------------
920  REM * PLAY NOTE/SEQUENCE:
930  REM *----------------
940  VTAB VP(PN): HTAB 17: PRINT PV(PN): GOSUB
     1370
950  IF OP$ =  CHR$ (13) THEN 980
960  POKE 206,0: POKE 207,3: & 1
970  RETURN
980  POKE 800,CS: POKE 801,CN: CALL B2T
990  POKE 206,0: POKE 207,64: & CS
1000  RETURN
1010  REM *----------------
1020  REM * CENTER LINE ROUTINE:
1030  REM *----------------
1040  HTAB (20 -  LEN (SP$) / 2): PRINT SP$: RETURN

1050  REM *----------------
1060  REM * APPEND SOUND/SEQ:
1070  REM *----------------
1080  IF OP$ = "A" THEN CHNG = SQ:SE = SE + 1
     :CS = SE:NS = 1:CN = 1: GOSUB 1140: GOTO
     1120
1090  IF CS = SE THEN CHNG = SS: GOTO 1110
1100 CHNG = IS
1110 NS = NS + 1:CN = NS
1120  POKE 800,CS: POKE 801,CN
1130  CALL CHNG: CALL T2B: GOSUB 1310: GOSUB
     380: GOTO 570
1140  VTAB VP(PN): HTAB 17: PRINT PV(PN): VTAB
     9: HTAB 7:A$ = "": CALL  - 868:I = 0
1150  GET OP$: GOSUB 330: IF I = 0 THEN 1190
1160  IF OV <  > 8 THEN 1190
1170 I = I - 1: PRINT OP$;: CALL  - 868: IF I
     = 0 THEN A$ = "": GOTO 1150
1180 A$ =  LEFT$ (A$,I)
1190  IF OV = 13 THEN 1230
1200  IF OV < 32 OR OV > 90 THEN 1150
1210  PRINT OP$;:A$ = A$ + OP$:I = I + 1: IF
     I > 14 THEN 1230
1220  GOTO 1150
1230 N$(CS) = A$: RETURN
1240  REM *----------------
1250  REM * RENAME SEQUENCE:
1260  REM *----------------
1270  GOSUB 1140: GOTO 570
1280  REM *----------------
1290  REM * PEEK BUFFER:
1300  REM *----------------
1310  FOR J = 1 TO 4:PV(J) =  PEEK (772 + J):
     NEXT
1320 PV(5) =  PEEK (777) +  PEEK (778) * 256
1330  RETURN
1340  REM *----------------
1350  REM * POKE BUFFER:
1360  REM *----------------
1370  FOR J = 1 TO 4: POKE (772 + J),PV(J): NEXT

1380 HI =  INT (PV(5) / 256):LO = PV(5) - HI *
     256
1390  POKE 777,LO: POKE 778,HI
1400  RETURN
1410  REM *----------------
1420  REM * INSERT SOUND:
1430  REM *----------------
1440  CALL IS
1450  CALL T2B
1460  GOSUB 1310
1470 NS = NS + 1: GOSUB 380: GOTO 570
1480  REM *----------------
1490  REM * DELETE SOUND:
1500  REM *----------------
1510  CALL DE
1520 SE =  PEEK (16384): IF SE = 0 THEN CS =
     1:SE = 1: CALL 32768
1530  IF CS > SE THEN CS = SE
1540  GOSUB 1820: IF CN > NS THEN CN = NS
1550  POKE 800,CS: POKE 801,CN: CALL T2B: GOSUB
     1310: GOSUB 380: GOTO 570
1560  REM *----------------
1570  REM * CASSETTE/REGULAR:
1580  REM *----------------
1590  IF  NOT MF THEN  &  POP :MF = 1: GOTO 1
     610
1600  IF MF THEN  &  NORMAL :MF = 0
1610  GOSUB 380: GOTO 570
1620  REM *----------------
1630  REM * NEXT-LAST SND/SEQ:
1640  REM *----------------
1650  IF OP$ = ">" OR OP$ = "." OR OP$ = "N" THEN
     C = 1
1660  IF OP$ = "<" OR OP$ = "," OR OP$ = "P" THEN
     C =  - 1
1670  IF OP$ > ">" THEN 1720
1680 CN = CN + C
1690  IF CN < 1 THEN CN = NS
1700  IF CN > NS THEN CN = 1
1710  GOTO 1760
1720 CS = CS + C
1730  IF CS < 1 THEN CS = SE
1740  IF CS > SE THEN CS = 1
1750  GOSUB 1820: IF CN > NS THEN CN = NS
1760  POKE 800,CS: POKE 801,CN
1770  CALL T2B: GOSUB 1310
1780  GOSUB 380: GOTO 570
1790  REM *----------------
1800  REM * FIND SOUND NO.:
1810  REM *----------------
1820 SE =  PEEK (16384):A = 16384:AD = 2 * CS
1830 OFF =  PEEK (A + AD) +  PEEK (A + 1 + AD
     ) * 256
1840 NS =  PEEK (16384 + OFF)
1850  RETURN
1860  REM *----------------
1870  REM * COPY SOUND:
1880  REM *----------------
1890  IF CS = SE THEN CHNG = SS: GOTO 1910
1900 CHNG = IS
1910 NS = NS + 1:CN = NS
1920  POKE 800,CS: POKE 801,CN
1930  CALL CHNG: CALL B2T: GOSUB 380: GOTO 57
     0
1940  REM *----------------
1950  REM * DISK MENU:
1960  REM *----------------
1970  ONERR  GOTO 2680
1980  PRINT : PRINT  CHR$ (4);"CLOSE": CALL C
     L:L =  PEEK (6) +  PEEK (7) * 256: IF L =
     11 AND PV(5) = 0 THEN L = 0
1990  HOME :SP$ = " DISK MENU ": INVERSE : VTAB
     1: GOSUB 1040: NORMAL
2000  VTAB 5: HTAB 3: PRINT "C : CATALOG "
2010  VTAB 7: HTAB 3: PRINT "D : CHANGE CURRE
     NT DRIVE";: HTAB 28: PRINT "(";: INVERSE
     : PRINT D;: NORMAL : PRINT ")": NORMAL
2020  VTAB 9: HTAB 3: PRINT "S : SAVE SOUND T
     ABLE": VTAB 11: HTAB 3: PRINT "L : LOAD
     SOUND TABLE"
2030  VTAB 24: HTAB 7: PRINT "<ESCAPE> : BACK
      TO EDIT";
2040  VTAB 13: HTAB 7: PRINT "FILENAME: ";SN$
2050  VTAB 15: HTAB 13: PRINT "(";L;" BYTES L
     ONG)"
```

```
2060  IF  PEEK ( - 16384) < 128 THEN 2060
2070  GET OP$: PRINT : GOSUB 330
2080  IF OP$ =  CHR$ (27) THEN 2140
2090  IF OP$ = "C" THEN 2250
2100  IF OP$ = "D" THEN 2330
2110  IF OP$ = "S" THEN 2370
2120  IF OP$ = "L" THEN 2540
2130  GOTO 2060
2140  POKE 216,0: HOME : GOSUB 380: GOTO 570
2150  REM *----------------
2160  REM * FILE NAME:
2170  REM *----------------
2180 X =  FRE (0): VTAB 24: HTAB 5: PRINT "<R
     ETURN> : ACCEPT FILENAME";: VTAB 13: HTAB
     17: INPUT "";A$
2190  IF A$ = "" THEN  VTAB 13: HTAB 17: PRINT
     SN$;: CALL  - 868: RETURN
2200  IF  LEFT$ (A$,1) < "A" OR  LEFT$ (A$,1)
     > "Z" OR  LEN (A$) > 10 THEN 2180
2210 SN$ = A$: VTAB 24: HTAB 5: CALL  - 868: RETURN

2220  REM *----------------
2230  REM * CATALOG:
2240  REM *----------------
2250  HOME :SP$ = " CATALOG ": INVERSE : GOSUB
     1040: NORMAL : PRINT  CHR$ (4);CA$;",D"D
     : PRINT
2260  ONERR  GOTO 2680
2270 X =  FRE (0): PRINT : INPUT "DISK COMMAN
     D -> ";DC$: IF DC$ = "" THEN 1980
2280  PRINT  CHR$ (4);DC$;",D"D: GOTO 2270
2290  VTAB 19: HTAB 5: FLASH : PRINT "NO";: NORMAL
     : PRINT " FILE IN MEMORY!": HTAB 5: PRINT
     "PRESS <RETURN> -> ";: GET AN$: GOTO 198
     0
2300  REM *----------------
2310  REM * CHANGE DRIVE NO.:
2320  REM *----------------
2330 D = 3 - D: GOTO 1980
2340  REM *----------------
2350  REM * SAVE SOUND TABLE:
2360  REM *----------------
2370  IF L = 0 THEN 2290
2380  VTAB 9: HTAB 3: INVERSE : PRINT "S": NORMAL
     : GOSUB 2180: IF SN$ = "" THEN 1980
2390 A = 16384: VTAB 17: HTAB 5: INVERSE : PRINT
     " SAVING: ";: NORMAL : PRINT " ";SN$ + S
     D$
2400  ONERR  GOTO 2430
2410  VTAB 5: PRINT : PRINT  CHR$ (4);"VERIFY
     ";SN$ + SD$;",D"D
2420  VTAB 19: FLASH :SP$ = "WARNING": GOSUB
     1040: NORMAL :SP$ = "FILE " + SN$ + SD$ +
     " ALREADY EXISTS!": GOSUB 1040: GOTO 246
     0
2430  ONERR  GOTO 2680
2440  IF  PEEK (222) = 6 THEN 2460
2450  GOTO 2680
2460  VTAB 24: HTAB 1: PRINT "<SPACE> SAVES,
     <ESCAPE> CANCELS -> ";: GET A$: IF A$ =
     CHR$ (27) GOTO 1980
2470  IF A$ <  > " " THEN 2460
2480  VTAB 5: PRINT : PRINT  CHR$ (4);"BSAVE
     ";SN$ + SD$;",A";A;",L";L;",D"D
2490 OP$ = SN$ + NS$: PRINT  CHR$ (4);"OPEN "
     ;OP$;",D"D: PRINT  CHR$ (4);"WRITE ";OP$
2500  PRINT SE: FOR I = 1 TO SE: PRINT N$(I):
     NEXT I: GOTO 1980
2510  REM *----------------
2520  REM * LOAD SOUND TABLE:
2530  REM *----------------
2540  VTAB 11: HTAB 3: INVERSE : PRINT "L": NORMAL
     : GOSUB 2180: IF SN$ = "" THEN 1980
2550  VTAB 17: HTAB 5: INVERSE : PRINT " LOAD
     ING: ";: NORMAL : PRINT " ";SN$ + SD$
2560  IF L = 0 THEN 2590
2570  VTAB 19: FLASH :SP$ = " WARNING ": GOSUB
     1040: NORMAL
2580  VTAB 20:SP$ = "FILE IN MEMORY WILL BE E
     RASED!": GOSUB 1040
2590  VTAB 24: HTAB 1: PRINT "<SPACE> LOADS,
     <ESCAPE> CANCELS -> ";: GET A$: ON A$ =
     CHR$ (27) GOTO 1980: IF A$ <  > " " GOTO
     2590
2600  CALL IT:A = 16384: VTAB 5: PRINT : PRINT
     CHR$ (4);"BLOAD ";SN$ + SD$;",A";A;",D"
     D
2610 CS = 1:CN = 1: POKE 800,1: POKE 801,1: CALL
     T28: GOSUB 1310: FOR I = 1 TO SE:N$(I) =
     "": NEXT : GOSUB 1820
2620 OP$ = SN$ + NS$: PRINT  CHR$ (4);"OPEN "
     ;OP$;",D"D: PRINT  CHR$ (4);"READ ";OP$
2630  INPUT SE: FOR I = 1 TO SE: INPUT N$(I):
     NEXT I
2640  GOTO 1980
2650  REM *----------------
2660  REM * ONERR COME HERE:
2670  REM *----------------
2680 Y =  PEEK (222)
2690  HOME : VTAB 12: HTAB 10: PRINT " DISK "
     ;: FLASH : PRINT "ERROR";: NORMAL : PRINT
     " #";Y;" IN LINE "; PEEK (218) + 256 *  PEEK
     (219)
2700  VTAB 15: PRINT "  PRESS <RETURN> -> ";:
     GET AN$: GOTO 1980
2710  REM *----------------
2720  REM * QUIT:
2730  REM *----------------
2740  HOME : VTAB 2:SP$ = " QUIT ": INVERSE :
     GOSUB 1040: NORMAL
2750  VTAB 12: HTAB 2: PRINT "DO YOU WISH TO
     QUIT (Y/N)? ";: GET OP$: GOSUB 330
2760  IF OP$ <  > "Y" AND OP$ <  > "N" THEN 2
     750
2770  PRINT OP$: IF OP$ = "N" THEN 2810
2780  HOME : VTAB 12:SP$ = "END OF ARCADE SOU
     ND EDITOR": GOSUB 1040
2790  VTAB 14:SP$ = "TYPE 'GOTO 260' TO RE-EN
     TER": GOSUB 1040:SP$ = "WITH SOUND DATA
     INTACT": GOSUB 1040
2800  POKE 1010,191: POKE 1011,157: CALL  - 1
     169: POKE 207,64: VTAB 17: END
2810  VTAB 14: HTAB 2: PRINT "CLEAR THE TABLE
     (Y/N)? ";: GET OP$: GOSUB 330: IF OP$ <
     > "Y" AND OP$ <  > "N" THEN 2810
2820  PRINT OP$: IF OP$ = "N" THEN 2850
2830  CALL IT:NS = 1:CS = 1:CN = 1:SN$ = "": FOR
     I = 1 TO 5:PV(I) = 0: NEXT
2840  FOR I = 1 TO SE:N$(I) = "": NEXT I:SE =
     1
2850  HOME : GOSUB 380: GOTO 570
2860 E =  PEEK (222):EL =  PEEK (218) + 256 *
     PEEK (219): HOME : VTAB 12
2870  PRINT "TROUBLE LOADING " MID$ ("DUOSEU"
     ,EF * 3 - 2,3)
2880  VTAB 22: HTAB 1: PRINT "<ESC> TO QUIT,
     <RETURN> TO TRY AGAIN";: GET Z$: PRINT :
     IF Z$ =  CHR$ (27) THEN  END
2890  ON EF GOTO 200,210
```

END OF LISTING 1

```
KEY PERFECT 5.0
RUN ON
ASE
```

CODE-5.0	LINE# - LINE#	CODE-4.0
7C62FD67	10 - 100	5F73
D6431199	110 - 200	ABF9
D1AB4656	210 - 300	B8C7
CB680F48	310 - 400	9263
40B1EA17	410 - 500	F098
DEAB2A84	510 - 600	79F0
65C65346	610 - 700	8397
3C5F5D69	710 - 800	B7D3
917DC5FD	810 - 900	88C0
242624AA	910 - 1000	511D
52B62589	1010 - 1100	78F6
BAB3F5E8	1110 - 1200	6683
3984BF99	1210 - 1300	57E3
B3D1EF28	1310 - 1400	5AFF
506415E2	1410 - 1500	46A9
46C087D4	1510 - 1600	6A1D
70293318	1610 - 1700	6083
3827B95E	1710 - 1800	4576
0868D0D7	1810 - 1900	57AD
7C55E6AF	1910 - 2000	71EE
711C2FE0	2010 - 2100	8AA9
D8479722	2110 - 2200	7643
5C28D816	2210 - 2300	8FE3
968D7722	2310 - 2400	65A4
822DD4BC	2410 - 2500	B03E
3DA0C6D8	2510 - 2600	AF70
EAF8C4F5	2610 - 2700	9A60
C8D3FBFF	2710 - 2800	9C25
75B34430	2810 - 2890	98CF
80306143	= PROGRAM TOTAL =	1FA5

LISTING 2: DUO

```
1   ********************************************
2   *                                          *
3   *                  DUO                     *
4   *                                          *
5   *            By S. Scott Zimmerman         *
6   *             Copyright (c) 1987           *
7   *             by MicroSPARC, Inc           *
8   *             Concord, MA  01742           *
9   *                                          *
10  *            Assembler:  MERLIN PRO        *
11  *                                          *
12  ********************************************
13
14          ORG   $9400       ;Completely relocatable
15
16  ==========================================
17  *  Zero page equates:                    *
18  ==========================================
19
20  PITCH1   EQU   $00        ;Pitch of sound 1
```

LISTING 2: DUO *(continued)*

```
                        21  PITCH2   EQU   $01          ;Pitch of sound 2
                        22  DELTA1   EQU   $03          ;Delta of sound 1
                        23  DELTA2   EQU   $04          ;Delta of sound 2
                        24  A3L      EQU   $40          ;Auxiliary location
                        25  SNDTBL   EQU   $CE          ;Sound tbl adrs. POKE 206
                        26  DURATION EQU   $19          ;24-bit sound duration
                        27  SNDPTR   EQU   $FA          ;Sound table pointer
                        28  AUXPTR   EQU   $FC          ;Auxiliary pointer
                        29  SPKPTR   EQU   $FE          ;Speaker pointer
                        30
                        31  *-------------------------------------------*
                        32  *  Monitor and Applesoft ROM routines:      *
                        33  *-------------------------------------------*
                        34
                        35  CHRGOT   EQU   $B7          ;Get text character
                        36  STACK    EQU   $100         ;6502 stack
                        37  AMPER    EQU   $3F5         ;& routine address
                        38  ERROR    EQU   $D412        ;Aplsoft error routine
                        39  TXTEND   EQU   $D995        ;Move TXTPTR end statmnt
                        40  GETBYTC  EQU   $E6F5        ;Get a character, then...
                        41  GETBYT   EQU   $E6F8        ;Eval text expression
                        42  SPEAKER  EQU   $C030        ;Speaker switch
                        43  CASSETTE EQU   $C020        ;Cassette switch
                        44  RTNBYTE  EQU   $FF58        ;ROM RTS ($60)
                        45
                        46  *-------------------------------------------*
                        47  *  CONSTANTS:                               *
                        48  *-------------------------------------------*
                        49
                        50  NORMTOK  EQU   $9D          ;Applesoft NORMAL token
                        51  POPTOK   EQU   $A1          ;Applesoft POP token
                        52  STOPTOK  EQU   $B3          ;Applesoft STOP token
                        53
                        54  *-------------------------------------------*
                        55  *  Data storage                             *
                        56  *-------------------------------------------*
                        57
                        58  P1       EQU   $3C8         ;Auxiliary pitch 1
                        59  P2       EQU   P1+1         ;Auxiliary pitch 2
                        60  SNDCNTR   EQU   P1+2         ;Sound counter for loop
                        61  YSAVE    EQU   P1+3         ;Save Y register
                        62  RFLAG1   EQU   P1+4         ;Rest flags
                        63  RFLAG2   EQU   P1+5
                        64  STOPFLG  EQU   P1+6         ;Stop flag
                        65  ASAVE    EQU   P1+7         ;Temp save of A reg
                        66
                        67  *-------------------------------------------*
                        68  *  MACROs:                                  *
                        69  *-------------------------------------------*
                        70
                        71  INCR     MAC
                        72           INC   ]1
                        73           BNE   NC
                        74           INC   ]1+1
                        75  NC       <<<
                        76
                        77  ADD      MAC
                        78           CLC
                        79           LDA   ]1
                        80           ADC   ]2
                        81           STA   ]3
                        82           LDA   ]1+1
                        83           ADC   ]2+1
                        84           STA   ]3+1
                        85           <<<
                        86
                        87  SETPTR   MAC
                        88           LDA   #<]1
                        89           STA   ]2
                        90           LDA   #>]1
                        91           STA   ]2+1
                        92           <<<
                        93
                        94  *-------------------------------------------*
                        95  *  Connect &-hooks:                         *
                        96  *-------------------------------------------*
                        97
9400: 20 58 FF          98  START    JSR   RTNBYTE      ;Put address on stack
9403: BA                99           TSX               ;Get stack pointer
9404: CA               100           DEX               ;Point to LOB
9405: 18               101           CLC               ;Prepare to add
9406: BD 00 01         102           LDA   STACK,X     ;Get the address
9409: 69 23            103           ADC   #<DUO-START-2 ;Point to start
940B: 8D F6 03         104           STA   AMPER+1     ;Put at &-vector
940E: E8               105           INX               ;Point to HOB
940F: BD 00 01         106           LDA   STACK,X     ;Get address
9412: 69 00            107           ADC   #>DUO-START-2
9414: 8D F7 03         108           STA   AMPER+2     ;Save at &-vector
9417: A9 4C            109           LDA   #$4C        ;Get JMP opcode
9419: 8D F5 03         110           STA   AMPER       ;Put at &-vector
                       111           >>>   SETPTR,SPEAKER;SPKPTR
941C: A9 30            111           LDA   #<SPEAKER
941E: 85 FE            111           STA   SPKPTR
9420: A9 C0            111           LDA   #>SPEAKER
9422: 85 FF            111           STA   SPKPTR+1
                       111           <<<
9424: 60               112           RTS               ;End initialization
                       113
                       114  *-------------------------------------------*
                       115  *  DUO (main program):                      *
                       116  *-------------------------------------------*
                       117
                       118  DUO
9425: C9 B3            119           CMP   #STOPTOK    ;Is it the STOP token?
9427: D0 08            120           BNE   CHKNORM     ;No, go check NORMAL
9429: A9 01            121           LDA   #1          ;Set the stop flag
942B: 8D CE 03         122           STA   STOPFLG     ; so no sound is made
942E: 4C 95 D9         123           JMP   TXTEND      ;Exit to end of command
9431: C9 9D            124  CHKNORM  CMP   #NORMTOK    ;Is it the RESUME token?
9433: D0 0C            125           BNE   CHKPOP      ;No, go check POP
9435: A9 00            126           LDA   #0          ;Zero the stop flag
9437: 8D CE 03         127           STA   STOPFLG     ; so sound is made
943A: A9 30            128           LDA   #<SPEAKER   ;Make sure normal
943C: 85 FE            129           STA   SPKPTR      ; speaker, not cassette
943E: 4C 95 D9         130           JMP   TXTEND      ;Exit to end of command
9441: C9 A1            131  CHKPOP   CMP   #POPTOK     ;Is it the POP token?
9443: D0 0C            132           BNE   NOISE       ;No, go do noise
9445: A9 00            133           LDA   #0          ;Zero the stop flag
9447: 8D CE 03         134           STA   STOPFLG     ; so sound thru cassette
944A: A9 20            135           LDA   #<CASSETTE  ;Exchange speaker for
944C: 85 FE            136           STA   SPKPTR      ; cassette out
944E: 4C 95 D9         137           JMP   TXTEND      ;Exit to end of command
                       138
9451: 20 F8 E6         139  NOISE    JSR   GETBYT      ;Evaluate formula after &
9454: 8A               140           TXA               ;Put value in accumulator
9455: F0 08            141           BEQ   ILLEGAL     ;Zero is illegal
9457: A2 00            142           LDX   #0          ;Zero dummy index
9459: C1 CE            143           CMP   (SNDTBL,X)  ;> number of sequences?
945B: 90 07            144           BLT   SOUNDOFF    ;No, so go make sound
945D: F0 05            145           BEQ   SOUNDOFF
945F: A2 35            146  ILLEGAL  LDX   #53         ;Set for ILLEGAL QUANTITY
9461: 4C 12 D4         147           JMP   ERROR       ;Go print error
                       148
                       149  SOUNDOFF
9464: 8D CF 03         150           STA   ASAVE       ;Temp save of A
9467: AD CE 03         151           LDA   STOPFLG     ;Stop the sound?
946A: F0 03            152           BEQ   DOSOUND     ;No, make sound
946C: 4C 95 D9         153           JMP   TXTEND      ;Yes, just quit
                       154
                       155  * Set A3L to point to sound sequence index:
                       156
946F: AD CF 03         157  DOSOUND  LDA   ASAVE       ;Restore A
9472: 85 40            158           STA   A3L         ;Save sound sequ number
9474: A9 00            159           LDA   #0          ;Zero the HOB
9476: 85 41            160           STA   A3L+1
9478: 06 40            161           ASL   A3L         ;Mult by 2
947A: 26 41            162           ROL   A3L+1
947C: 18               163           CLC               ;Prep for addition
947D: A5 CE            164           LDA   SNDTBL      ;Get LOB of sound pntr
947F: 65 40            165           ADC   A3L         ;Add sound number x2
9481: 85 40            166           STA   A3L         ;Stuff back
9483: A5 CF            167           LDA   SNDTBL+1    ;Get HOB of sound pntr
9485: 65 41            168           ADC   A3L+1
9487: 85 41            169           STA   A3L+1
                       170
                       171  * Set SNDPTR to point to current sequence:
                       172
9489: A0 00            173           LDY   #0          ;Zero the index
948B: 18               174           CLC
948C: B1 40            175           LDA   (A3L),Y
948E: 65 CE            176           ADC   SNDTBL
9490: 85 FA            177           STA   SNDPTR
9492: C8               178           INY
9493: B1 40            179           LDA   (A3L),Y
9495: 65 CF            180           ADC   SNDTBL+1
9497: 85 FB            181           STA   SNDPTR+1
                       182
                       183  * Check if a second parameter for sound number:
                       184
9499: 20 B7 00         185           JSR   CHRGOT      ;At end of command?
949C: F0 54            186           BEQ   SNDALL      ;Yes, so sound all
949E: 20 F5 E6         187           JSR   GETBYTC     ;Evaluate next char
94A1: 8A               188           TXA               ;Put value in A
94A2: F0 08            189           BEQ   NOWAY       ;If zero, make error
94A4: A2 00            190           LDX   #0          ;Zero dummy variable
94A6: C1 FA            191           CMP   (SNDPTR,X)  ;Wrong input number?
94A8: F0 07            192           BEQ   SNDOK       ;No, number is okay
94AA: 90 05            193           BLT   SNDOK
94AC: A2 35            194  NOWAY    LDX   #53         ;Illegal quantity error
94AE: 4C 12 D4         195           JMP   ERROR
                       196
                       197  * Make a single sound:
                       198
                       199  SNDOK
94B1: AA               200           TAX               ;Put sound back into X
94B2: CA               201           DEX               ;Set range 0...n
                       202           >>>   INCR,SNDPTR ;Point to first sound
94B3: E6 FA            202           INC   SNDPTR
94B5: D0 02            202           BNE   NC
94B7: E6 FB            202           INC   SNDPTR+1
                       202  NC       <<<
94B9: A9 00            203           LDA   #0          ;Zero HOB
94BB: 85 FD            204           STA   AUXPTR+1
94BD: 86 FC            205           STX   AUXPTR      ;Set the LOB
94BF: 06 FC            206           ASL   AUXPTR      ;Multiply by 6
94C1: 26 FD            207           ROL   AUXPTR+1
94C3: A5 FC            208           LDA   AUXPTR
94C5: 85 40            209           STA   A3L
94C7: A5 FD            210           LDA   AUXPTR+1
94C9: 85 41            211           STA   A3L+1
94CB: 06 FC            212           ASL   AUXPTR
94CD: 26 FD            213           ROL   AUXPTR+1
                       214           >>>   ADD,AUXPTR;A3L;AUXPTR
94CF: 18               214           CLC
94D0: A5 FC            214           LDA   AUXPTR
94D2: 65 40            214           ADC   A3L
94D4: 85 FC            214           STA   AUXPTR
94D6: A5 FD            214           LDA   AUXPTR+1
94D8: 65 41            214           ADC   A3L+1
94DA: 85 FD            214           STA   AUXPTR+1
                       214           <<<
                       215           >>>   ADD,AUXPTR;SNDPTR;SNDPTR
94DC: 18               215           CLC
94DD: A5 FC            215           LDA   AUXPTR
94DF: 65 FA            215           ADC   SNDPTR
94E1: 85 FA            215           STA   SNDPTR
94E3: A5 FD            215           LDA   AUXPTR+1
94E5: 65 FB            215           ADC   SNDPTR+1
94E7: 85 FB            215           STA   SNDPTR+1
                       215           <<<
94E9: B8               216           CLV               ;To force branch
94EA: 20 58 FF         217           JSR   RTNBYTE     ;Put address on stack
94ED: 50 20            218           BVC   PLAYSND     ;Go play the sound
94EF: 4C 95 D9         219           JMP   TXTEND      ;Exit thru text end
                       220
                       221  * Play the entire sound sequence:
                       222
94F2: A0 00            223  SNDALL   LDY   #0          ;Set index to no. sounds
```

LISTING 2: DUO *(continued)*

```
94F4: B1 FA     224           LDA   (SNDPTR),Y ;Get number of sounds
94F6: AA        225           TAX                ;Make an index
94F7: 8E CA 03  226 SNDLOOP   STX   SNDCNTR     ;Save note counter
                227           >>>   INCR.SNDPTR ;Get PITCH1 offset
94FA: E6 FA     227           INC   SNDPTR
94FC: D0 02     227           BNE   NC
94FE: E6 FB     227           INC   SNDPTR+1
                227 NC        <<<
9500: B8        228           CLV                ;To force branch
9501: 20 58 FF  229           JSR   RTNBYTE      ;Put address on stack
9504: 50 09     230           BVC   PLAYSND      ;Go play the sound
9506: AE CA 03  231           LDX   SNDCNTR      ;Get current count
9509: CA        232           DEX                ;End of sounds?
950A: D0 EB     233           BNE   SNDLOOP      ;No, go to next
950C: 4C 95 D9  234           JMP   TXTEND       ;Exit thru end of line
                235
                236 *-------------------------------------------*
                237 *  Get sound parameters and play the sound:  *
                238 *-------------------------------------------*
                239
                240 PLAYSND
950F: BA        241           TSX                ;Decrement the stack
9510: CA        242           DEX                ; pointer so the RTS
9511: CA        243           DEX                ; returns to after
9512: 9A        244           TXS                ; the BVC
9513: 38        245           SEC                ;Make higher values of
9514: A9 00     246           LDA   #0           ; input higher pitches
9516: F1 FA     247           SBC   (SNDPTR),Y   ;Get PITCH1
9518: 8D C8 03  248           STA   P1
                249           >>>   INCR.SNDPTR ;Point to then get P2
951B: E6 FA     249           INC   SNDPTR
951D: D0 02     249           BNE   NC
951F: E6 FB     249           INC   SNDPTR+1
                249 NC        <<<
9521: 38        250           SEC
9522: A9 00     251           LDA   #0
9524: F1 FA     252           SBC   (SNDPTR),Y
9526: 8D C9 03  253           STA   P2
                254           >>>   INCR.SNDPTR ;Point to then get D1
9529: E6 FA     254           INC   SNDPTR
952B: D0 02     254           BNE   NC
952D: E6 FB     254           INC   SNDPTR+1
                254 NC        <<<
952F: B1 FA     255           LDA   (SNDPTR),Y
9531: 85 03     256           STA   DELTA1
                257           >>>   INCR.SNDPTR ;Point to then get D2
9533: E6 FA     257           INC   SNDPTR
9535: D0 02     257           BNE   NC
9537: E6 FB     257           INC   SNDPTR+1
                257 NC        <<<
9539: B1 FA     258           LDA   (SNDPTR),Y
953B: 85 04     259           STA   DELTA2
                260           >>>   INCR.SNDPTR
953D: E6 FA     260           INC   SNDPTR
953F: D0 02     260           BNE   NC
9541: E6 FB     260           INC   SNDPTR+1
                260 NC        <<<
9543: B1 FA     261           LDA   (SNDPTR),Y ;Two-byte duration
9545: 85 19     262           STA   DURATION
                263           >>>   INCR.SNDPTR
9547: E6 FA     263           INC   SNDPTR
9549: D0 02     263           BNE   NC
954B: E6 FB     263           INC   SNDPTR+1
                263 NC        <<<
954D: B1 FA     264           LDA   (SNDPTR),Y
954F: 85 1A     265           STA   DURATION+1
9551: A2 00     266           LDX   #0           ;Zero MSB of 3-byte dur
9553: 86 1B     267           STX   DURATION+2
                268
9555: A2 07     269           LDX   #7           ;Mult by 128
9557: 06 19     270 MULTLOOP  ASL   DURATION
9559: 26 1A     271           ROL   DURATION+1
955B: 26 1B     272           ROL   DURATION+2
955D: CA        273           DEX                ;End of multiply?
955E: D0 F7     274           BNE   MULTLOOP     ;No, proceed
9560: 8C CB 03  275           STY   YSAVE        ;Save pointer location
                276
                277 *-------------------------------------------*
                278 *  Two-tone sound routine:                  *
                279 *-------------------------------------------*
                280
9563: A0 00     281           LDY   #0           ;Zero the dummy variable
9565: AD C8 03  282           LDA   P1           ;Get first pitch
9568: 85 00     283           STA   PITCH1
956A: 8D CC 03  284           STA   RFLAG1       ;Set rest flag (0=rest)
956D: AD C9 03  285           LDA   P2           ;Get second pitch
9570: 85 01     286           STA   PITCH2
9572: 8D CD 03  287           STA   RFLAG2       ;Set rest flag (0=rest)
9575: AD CC 03  288 DURLOOP   LDA   RFLAG1       ;Get rest flag
9578: F0 11     289           BEQ   D1           ;Skip speaker if rest
957A: C6 00     290           DEC   PITCH1       ;Ready for first pitch?
957C: D0 0D     291           BNE   D1           ;No, skip it
957E: B1 FE     292           LDA   (SPKPTR),Y   ;Click the speaker
9580: 38        293           SEC                ;Prepare to subtract
9581: AD C8 03  294           LDA   P1           ;Reset pitch counter
9584: E5 03     295           SBC   DELTA1       ;Change pitch
9586: 8D C8 03  296           STA   P1           ;Save new pitch
9589: 85 00     297           STA   PITCH1
958B: AD CD 03  298 D1        LDA   RFLAG2       ;Get rest flag
958E: F0 11     299           BEQ   D2           ;Skip speaker if rest
9590: C6 01     300           DEC   PITCH2       ;Ready for second pitch?
9592: D0 0D     301           BNE   D2           ;No, skip speaker
9594: B1 FE     302           LDA   (SPKPTR),Y   ;Click the speaker
9596: 38        303           SEC                ;Prepare for subtract
9597: AD C9 03  304           LDA   P2           ;Reset pitch counter
959A: E5 04     305           SBC   DELTA2       ;Change pitch
959C: 8D C9 03  306           STA   P2           ;Save new pitch
959F: 85 01     307           STA   PITCH2
                308
95A1: A5 19     309 D2        LDA   DURATION     ;Do 24-bit decrement
95A3: D0 0C     310           BNE   D6
95A5: A5 1A     311           LDA   DURATION+1
95A7: D0 06     312           BNE   D5
95A9: A5 1B     313           LDA   DURATION+2
95AB: F0 09     314           BEQ   QUIT
95AD: C6 1B     315           DEC   DURATION+2
95AF: C6 1A     316 D5        DEC   DURATION+1
95B1: C6 19     317 D6        DEC   DURATION
95B3: B8        318           CLV                ;To force branch always
95B4: 50 BF     319           BVC   DURLOOP      ;Continue duration loop
                320
95B6: AC CB 03  321 QUIT      LDY   YSAVE        ;Restore the Y register
95B9: 2C 58 FF  322           BIT   RTNBYTE      ;Force set of V bit
95BC: 60        323           RTS                ;End of sound routine

--End assembly, 445 bytes, Errors: 0

    320
95B6: AC CB 03  321 QUIT      LDY   YSAVE        ;Restore the Y register
95B9: 2C 58 FF  322           BIT   RTNBYTE      ;Force set of V bit
95BC: 60        323           RTS                ;End of sound routine
END OF LISTING 2
```

```
KEY PERFECT 5.0
RUN ON
DUO

=========================================
CODE-5.0   ADDR# - ADDR#    CODE-4.0
--------   -------------    --------
E8FB6CEC   9400 - 944F        2439
582FF287   9450 - 949F        296E
A1350809   94A0 - 94EF        2918
BF15E4D0   94F0 - 953F        283F
73DD8EF7   9540 - 958F        28DB
92D65EB7   9590 - 95BC        172E
8001C57B = PROGRAM TOTAL =    01BD
```

LISTING 3: SEU

```
1   .-------------------------------------------.
2   .                                           .
3   .                    SEU                    .
4   .            SOUND EDITOR UTILITY           .
5   .          By S. Scott Zimmerman            .
6   .            Copyright (c) 1987             .
7   .             by MicroSPARC, Inc            .
8   .             Concord, MA  01742            .
9   .                                           .
10  .          Assembler:  MERLIN PRO           .
11  .                                           .
12  .-------------------------------------------.
13
14          ORG   $8000
15
16  .-------------------------------------------.
17  .  Zero page equates:                       .
18  .-------------------------------------------.
19
20  SNDPTR  EQU   $0          ;Sound table pointer
21  GENPTR  EQU   $2          ;General pointer
22  AUXPTR  EQU   $4          ;Auxiliary pointer
23  INSPTR  EQU   $19         ;Insert pointer
24  A1L     EQU   $3C         ;General purpose regs
25  A2L     EQU   $3E         ; used by Monitor MOVE
26  A3L     EQU   $40
27  A4L     EQU   $42
28
29  .-------------------------------------------.
30  .  Game Sound Editor addresses, etc.        .
31  .-------------------------------------------.
32
33  FILELEN EQU   $6          ;Sound table file length
34  SNDBUFF EQU   $305        ;6-byte sound buffer
35  SEQNUM  EQU   $320        ;Current sound seq (800)
36  SNDNUM  EQU   $321        ;Current sound in seq
37  SOUNDTBL EQU  $4000       ;Location of sound table
38  TABLEND EQU   $7FFF       ;End of sound table
39  MOVE    EQU   $FE2C       ;Monitor memory move
40
41  .-------------------------------------------.
42  .  Macro definitions:                       .
43  .-------------------------------------------.
44
45  SETPTR  MAC               ;>> Set pointer
46          LDA   #<]1
47          STA   ]2
48          LDA   #>]1
49          STA   ]2+1
50          <<<
51
52  TRANS   MAC
53          LDA   ]1
54          STA   ]2
55          LDA   ]1+1
56          STA   ]2+1
57          <<<
58
59  INCR    MAC               ;>> Increment 16-bits
60          INC   ]1
61          BNE   NC
62          INC   ]1+1
63  NC      <<<
64
65  DECR    MAC               ;>> Decrement 16-bits
66          LDA   ]1
67          BNE   ND
68          DEC   ]1+1
69  ND      DEC   ]1
70          <<<
71
72  ADD     MAC               ;>> 16-bit addition
73          CLC
```

LISTING 3: SEU *(continued)*

```
                        74              LDA   ]1
                        75              ADC   ]2
                        76              STA   ]3
                        77              LDA   ]1+1
                        78              ADC   ]2+1
                        79              STA   ]3+1
                        80              <<<
                        81
                        82  ADDC        MAC                   ;>> 16-bit const add
                        83              CLC
                        84              LDA   ]1
                        85              ADC   #<]2
                        86              STA   ]3
                        87              LDA   ]1+1
                        88              ADC   #>]2
                        89              STA   ]3+1
                        90              <<<
                        91
                        92  COMPARE     MAC                   ;>> 16-bit compare
                        93              LDA   ]1
                        94              CMP   #<]2
                        95              LDA   ]1+1
                        96              SBC   #>]2
                        97              <<<
                        98
                        99  COMP        MAC                   ;>> 16-bit compare
                       100              LDA   ]1
                       101              CMP   ]2
                       102              LDA   ]1+1
                       103              SBC   ]2+1
                       104              <<<
                       105
                       106 *-------------------------------------------*
                       107 *  Subroutine Jumps:                        *
                       108 *-------------------------------------------*
                       109
8000: 4C 18 80         110              JMP   INITTBL         ;Initialize sound table
8003: 4C 4A 80         111              JMP   TBL2BUF         ;Table sound to buffer
8006: 4C 6E 80         112              JMP   BUF2TBL         ;Buffer sound to table
8009: 4C 92 80         113              JMP   INSERT          ;Insert a sound
800C: 4C 35 81         114              JMP   DELETE          ;Delete a sound
800F: 4C 44 82         115              JMP   STRTSND         ;Start snd on last seq
8012: 4C 57 82         116              JMP   STRTSEQ         ;Start a new sequence
8015: 4C C0 82         117              JMP   CALCLEN         ;Calculate file length
                       118
                       119 *-------------------------------------------*
                       120 *  INITTBL (clear sound table, set up buffer):  *
                       121 *-------------------------------------------*
                       122
                       123 INITTBL
                       124              >>>   SETPTR.SOUNDTBL;SNDPTR
8018: A9 00            124              LDA   #<SOUNDTBL
801A: 85 00            124              STA   SNDPTR
801C: A9 40            124              LDA   #>SOUNDTBL
801E: 85 01            124              STA   SNDPTR+1
                       124              <<<
8020: A2 00            125              LDX   #0              ;Zero the dummy index
8022: A9 00            126 INITLOOP     LDA   #0              ;Zero all the bytes
8024: 81 00            127              STA   (SNDPTR,X)      ;Save a zero there
                       128              >>>   INCR.SNDPTR     ;Go to next memory byte
8026: E6 00            128              INC   SNDPTR
8028: D0 02            128              BNE   NC
802A: E6 01            128              INC   SNDPTR+1
                       128 NC           <<<
                       129              >>>   COMPARE.SNDPTR;TABLEND+1 ;Past end?
802C: A5 00            129              LDA   SNDPTR
802E: C9 00            129              CMP   #<TABLEND+1
8030: A5 01            129              LDA   SNDPTR+1
8032: E9 80            129              SBC   #>TABLEND+1
                       129              <<<
8034: 90 EC            130              BLT   INITLOOP        ;No, so continue
                       131
8036: A0 04            132              LDY   #4              ;Get four bytes at first
8038: B9 45 80         133 SETLOOP      LDA   ONESND,Y        ;Put table for one sound
803B: 99 00 40         134              STA   SOUNDTBL,Y      ;Put in sound table
803E: 99 00 03         135              STA   SNDBUFF-5,Y     ;Put in buffer table
8041: 88               136              DEY                   ;Point to next byte
8042: 10 F4            137              BPL   SETLOOP         ;Done?
8044: 60               138              RTS                   ;End of INITTBL
                       139
8045: 01 00 04 00      140 ONESND       DFB   1,0,4,0,1
8049: 00 01
                       141
                       142 *-------------------------------------------*
                       143 *  TBL2BUF (move table sound to buffer):    *
                       144 *-------------------------------------------*
                       145
                       146 TBL2BUF
                       147              >>>   SETPTR.SNDBUFF;A4L
804A: A9 05            147              LDA   #<SNDBUFF
804C: 85 42            147              STA   A4L
804E: A9 03            147              LDA   #>SNDBUFF
8050: 85 43            147              STA   A4L+1
                       147              <<<
8052: AD 20 03         148              LDA   SEQNUM          ;Get current sequence
8055: AC 21 03         149              LDY   SNDNUM          ;Get current sound
8058: 20 20 83         150              JSR   POINTSND
                       151              >>>   TRANS.GENPTR;A1L
805B: A5 02            151              LDA   GENPTR
805D: 85 3C            151              STA   A1L
805F: A5 03            151              LDA   GENPTR+1
8061: 85 3D            151              STA   A1L+1
                       151              <<<
                       152              >>>   TRANS.AUXPTR;A2L
8063: A5 04            152              LDA   AUXPTR
8065: 85 3E            152              STA   A2L
8067: A5 05            152              LDA   AUXPTR+1
8069: 85 3F            152              STA   A2L+1
                       152              <<<
806B: 4C D7 82         153              JMP   MEMMOVE         ;Go move memory
                       154
                       155 *-------------------------------------------*
                       156 *  BUF2TBL (move buffer sound to table):    *
                       157 *-------------------------------------------*
                       158
                       159 BUF2TBL
                       160              >>>   SETPTR.SNDBUFF;A1L
806E: A9 05            160              LDA   #<SNDBUFF
8070: 85 3C            160              STA   A1L
8072: A9 03            160              LDA   #>SNDBUFF
8074: 85 3D            160              STA   A1L+1
                       160              <<<
                       161              >>>   SETPTR.SNDBUFF+5;A2L
8076: A9 0A            161              LDA   #<SNDBUFF+5
8078: 85 3E            161              STA   A2L
807A: A9 03            161              LDA   #>SNDBUFF+5
807C: 85 3F            161              STA   A2L+1
                       161              <<<
807E: AD 20 03         162              LDA   SEQNUM          ;Get current sequence
8081: AC 21 03         163              LDY   SNDNUM          ;Get current sound
8084: 20 20 83         164              JSR   POINTSND
                       165              >>>   TRANS.GENPTR;A4L
8087: A5 02            165              LDA   GENPTR
8089: 85 42            165              STA   A4L
808B: A5 03            165              LDA   GENPTR+1
808D: 85 43            165              STA   A4L+1
                       165              <<<
808F: 4C D7 82         166              JMP   MEMMOVE         ;Go move memory
                       167
                       168 *-------------------------------------------*
                       169 *  INSERT (insert a sound in the table):    *
                       170 *-------------------------------------------*
                       171
                       172 INSERT
                       173
                       174 * Move everything right six bytes
                       175 *  from current location:
                       176
8092: AD 20 03         177              LDA   SEQNUM
8095: AC 21 03         178              LDY   SNDNUM
8098: 20 20 83         179              JSR   POINTSND
                       180              >>>   TRANS.GENPTR;A1L ;Set START
809B: A5 02            180              LDA   GENPTR
809D: 85 3C            180              STA   A1L
809F: A5 03            180              LDA   GENPTR+1
80A1: 85 3D            180              STA   A1L+1
                       180              <<<
                       181              >>>   TRANS.GENPTR;INSPTR
80A3: A5 02            181              LDA   GENPTR
80A5: 85 19            181              STA   INSPTR
80A7: A5 03            181              LDA   GENPTR+1
80A9: 85 1A            181              STA   INSPTR+1
                       181              <<<
80AB: 20 73 83         182              JSR   POINTEND        ;Set SNDPTR to end
                       183              >>>   TRANS.AUXPTR;A2L
80AE: A5 04            183              LDA   AUXPTR
80B0: 85 3E            183              STA   A2L
80B2: A5 05            183              LDA   AUXPTR+1
80B4: 85 3F            183              STA   A2L+1
                       183              <<<
                       184              >>>   ADDC.A1L;6;A4L
80B6: 18               184              CLC
80B7: A5 3C            184              LDA   A1L
80B9: 69 06            184              ADC   #<6
80BB: 85 42            184              STA   A4L
80BD: A5 3D            184              LDA   A1L+1
80BF: 69 00            184              ADC   #>6
80C1: 85 43            184              STA   A4L+1
                       184              <<<
80C3: 20 D7 82         185              JSR   MEMMOVE
                       186
                       187 * Increase number of sounds by one:
                       188
80C6: AD 20 03         189              LDA   SEQNUM
80C9: 20 91 83         190              JSR   SETOFF          ;Point to offset
80CC: 20 A9 83         191              JSR   SETSEQ          ;Point to sequence
80CF: A2 00            192              LDX   #0
80D1: 18               193              CLC
80D2: A1 00            194              LDA   (SNDPTR,X)
80D4: 69 01            195              ADC   #1
80D6: 81 00            196              STA   (SNDPTR,X)
                       197
                       198 * Add six to all of the pointers following this:
                       199
80D8: AD 20 03         200              LDA   SEQNUM          ;See if it's last seq
80DB: CD 00 40         201              CMP   SOUNDTBL        ; of the sound table
80DE: F0 0D            202              BEQ   ZEROSND         ;Yes, don't adjust index
                       203              >>>   SETPTR.6;ADJNUM
80E0: A9 06            203              LDA   #<6
80E2: 8D CB 83         203              STA   ADJNUM
80E5: A9 00            203              LDA   #>6
80E7: 8D CC 83         203              STA   ADJNUM+1
                       203              <<<
80EA: 20 F7 80         204              JSR   ADJINDEX        ;Go adjust the index
                       205
                       206 * Zero the new inserted sound:
                       207
80ED: A0 05            208 ZEROSND      LDY   #5              ;Zero 6 bytes
80EF: A9 00            209              LDA   #0
80F1: 91 19            210 INSLOOP      STA   (INSPTR),Y
80F3: 88               211              DEY
80F4: 10 F8            212              BPL   INSLOOP
80F6: 60               213              RTS                   ;End of INSERT
                       214
                       215 *  Auxiliary routine to adjust indexes:
                       216
                       217 ADJINDEX                          ;Adjust the index
80F7: AE 20 03         218              LDX   SEQNUM
80FA: E8               219              INX                   ;Point to next index
80FB: 8A               220              TXA                   ;Put back in A
                       221 ADJENTRY
80FC: CD 00 40         222              CMP   SOUNDTBL        ;Past num of sequences?
80FF: F0 02            223              BEQ   ADDIT           ;No, so go add
8101: B0 31            224              BGE   ADJEND          ;Yes, so don't add
8103: 20 91 83         225 ADDIT        JSR   SETOFF          ;Set pntr to index
8106: AD 20 03         226              LDA   SEQNUM          ;Set loop counter
8109: 8D C9 83         227              STA   COUNT
810C: A2 00            228              LDX   #0              ;Zero dummy index
```

continued on next page

LISTING 3: SEU *(continued)*

```
810E: 18          229 ADDLOOP CLC                 ;Add a certain amount
810F: A1 00       230         LDA   (SNDPTR,X)    ;Get the index
8111: 6D CB 83    231         ADC   ADJNUM        ;(Can be negative)
8114: 81 00       232         STA   (SNDPTR,X)
                  233         >>>   INCR.SNDPTR
8116: E6 00       233         INC   SNDPTR
8118: D0 02       233         BNE   NC
811A: E6 01       233         INC   SNDPTR+1
                  233 NC      <<<
811C: A1 00       234         LDA   (SNDPTR,X)
811E: 6D CC 83    235         ADC   ADJNUM+1
8121: 81 00       236         STA   (SNDPTR,X)
                  237         >>>   INCR.SNDPTR
8123: E6 00       237         INC   SNDPTR
8125: D0 02       237         BNE   NC
8127: E6 01       237         INC   SNDPTR+1
                  237 NC      <<<
8129: EE C9 83    238         INC   COUNT
812C: AD C9 83    239         LDA   COUNT
812F: CD 00 40    240         CMP   SOUNDTBL      ;End yet?
8132: 90 DA       241         BLT   ADDLOOP       ;No, so go to next
8134: 60          242 ADJEND  RTS
                  243
                  244 *-------------------------------------------*
                  245 * DELETE (delete a sound from memory):      *
                  246 *-------------------------------------------*
                  247
                  248 DELETE
8135: AD 20 03    249         LDA   SEQNUM        ;Get current sequence
8138: 20 91 83    250         JSR   SETOFF        ;Point to offset
813B: 20 A9 83    251         JSR   SETSEQ        ;Point to sequence
813E: A0 00       252         LDY   #0            ;Zero dummy index
8140: B1 00       253         LDA   (SNDPTR),Y
8142: C9 01       254         CMP   #1            ;Is it equal to one?
8144: F0 03       255         BEQ   DELSEQ        ;Yes, delete sequence
8146: 4C EC 81    256         JMP   DELSND        ;Just delete one sound
                  257
                  258 DELSEQ
                  259
                  260 * Delete 7 bytes of this sound sequence:
                  261
                  262         >>>   TRANS.SNDPTR;A4L ;Make dest
8149: A5 00       262         LDA   SNDPTR
814B: 85 42       262         STA   A4L
814D: A5 01       262         LDA   SNDPTR+1
814F: 85 43       262         STA   A4L+1
                  262         <<<
                  263         >>>   ADDC.A4L;7;A1L ;Set start
8151: 18          263         CLC
8152: A5 42       263         LDA   A4L
8154: 69 07       263         ADC   #<7
8156: 85 3C       263         STA   A1L
8158: A5 43       263         LDA   A4L+1
815A: 69 00       263         ADC   #>7
815C: 85 3D       263         STA   A1L+1
                  263         <<<
815E: 20 73 83    264         JSR   POINTEND
                  265         >>>   TRANS.AUXPTR;A2L ;Set end byte
8161: A5 04       265         LDA   AUXPTR
8163: 85 3E       265         STA   A2L
8165: A5 05       265         LDA   AUXPTR+1
8167: 85 3F       265         STA   A2L+1
                  265         <<<
8169: 20 D7 82    266         JSR   MEMMOVE
                  267
                  268 * Delete the index:
                  269
816C: AD 20 03    270         LDA   SEQNUM        ;Get current sequence
816F: 20 91 83    271         JSR   SETOFF        ;Point to its offset
                  272         >>>   TRANS.SNDPTR;A4L ;Make dest
8172: A5 00       272         LDA   SNDPTR
8174: 85 42       272         STA   A4L
8176: A5 01       272         LDA   SNDPTR+1
8178: 85 43       272         STA   A4L+1
                  272         <<<
                  273         >>>   ADDC.SNDPTR;2;A1L ;Set start
817A: 18          273         CLC
817B: A5 00       273         LDA   SNDPTR
817D: 69 02       273         ADC   #<2
817F: 85 3C       273         STA   A1L
8181: A5 01       273         LDA   SNDPTR+1
8183: 69 00       273         ADC   #>2
8185: 85 3D       273         STA   A1L+1
                  273         <<<
8187: 20 73 83    274         JSR   POINTEND
                  275         >>>   TRANS.AUXPTR;A2L ;Set end
818A: A5 04       275         LDA   AUXPTR
818C: 85 3E       275         STA   A2L
818E: A5 05       275         LDA   AUXPTR+1
8190: 85 3F       275         STA   A2L+1
                  275         <<<
8192: 20 D7 82    276         JSR   MEMMOVE     ;Go move it all
                  277
                  278 * Subtract 2 from all preceding indexes:
                  279
8195: AE 20 03    280         LDX   SEQNUM
8198: CA          281         DEX               ;Point back one index
8199: 8A          282         TXA               ;Put back in A
819A: F0 31       283         BEQ   DELADJ      ;Past num of sequences?
819C: 20 91 83    284         JSR   SETOFF      ;Set pntr to index
819F: AE 20 03    285         LDX   SEQNUM      ;Set loop counter
81A2: CA          286         DEX               ;To one less than curr
81A3: 8E C9 83    287         STX   COUNT
81A6: A2 00       288         LDX   #0          ;Zero dummy index
81A8: 38          289 ADJLOOP SEC               ;Subtr 2 from indexes
81A9: A1 00       290         LDA   (SNDPTR,X)  ;Get the index
81AB: E9 02       291         SBC   #2
81AD: 81 00       292         STA   (SNDPTR,X)
                  293         >>>   INCR.SNDPTR
81AF: E6 00       293         INC   SNDPTR
81B1: D0 02       293         BNE   NC
81B3: E6 01       293         INC   SNDPTR+1
                  293 NC      <<<
81B5: A1 00       294         LDA   (SNDPTR,X)
81B7: E9 00       295         SBC   #0
81B9: 81 00       296         STA   (SNDPTR,X)
                  297         >>>   ADDC.SNDPTR;-3;SNDPTR
81BB: 18          297         CLC
81BC: A5 00       297         LDA   SNDPTR
81BE: 69 FD       297         ADC   #<-3
81C0: 85 00       297         STA   SNDPTR
81C2: A5 01       297         LDA   SNDPTR+1
81C4: 69 FF       297         ADC   #>-3
81C6: 85 01       297         STA   SNDPTR+1
                  297         <<<
81C8: CE C9 83    298         DEC   COUNT
81CB: D0 DB       299         BNE   ADJLOOP     ;Go again if not done
                  300
                  301 * Adjust the succeeding indexes by subtracting 9:
                  302
                  303 DELADJ
81CD: AD 20 03    304         LDA   SEQNUM      ;Is it the last sequence
81D0: CD 00 40    305         CMP   SOUNDTBL    ; in the sound table?
81D3: F0 10       306         BEQ   DECSEQ      ;Yes, don't change index
                  307         >>>   SETPTR.-9:ADJNUM
81D5: A9 F7       307         LDA   #<-9
81D7: 8D CB 83    307         STA   ADJNUM
81DA: A9 FF       307         LDA   #>-9
81DC: 8D CC 83    307         STA   ADJNUM+1
                  307         <<<
81DF: AD 20 03    308         LDA   SEQNUM      ;Get current seq
81E2: 20 FC 80    309         JSR   ADJENTRY    ;Go adjust indexes
                  310
                  311 * Decrement the number of sequences, zero end
                  312
                  313 DECSEQ
81E5: CE 00 40    314         DEC   SOUNDTBL
81E8: 20 37 82    315         JSR   ZEROEND     ;Go zero trailing bytes
81EB: 60          316         RTS
                  317
                  318 DELSND
                  319
                  320 * Move everything down six bytes in memory:
                  321
81EC: AD 20 03    322         LDA   SEQNUM      ;Get sound sequence
81EF: AC 21 03    323         LDY   SNDNUM      ;Get number sounds
81F2: 20 26 83    324         JSR   SETSND      ;Go set this sound
                  325         >>>   TRANS.GENPTR;A4L ;Becomes dest
81F5: A5 02       325         LDA   GENPTR
81F7: 85 42       325         STA   A4L
81F9: A5 03       325         LDA   GENPTR+1
81FB: 85 43       325         STA   A4L+1
                  325         <<<
                  326         >>>   ADDC.A4L;6;A1L ;Set start
81FD: 18          326         CLC
81FE: A5 42       326         LDA   A4L
8200: 69 06       326         ADC   #<6
8202: 85 3C       326         STA   A1L
8204: A5 43       326         LDA   A4L+1
8206: 69 00       326         ADC   #>6
8208: 85 3D       326         STA   A1L+1
                  326         <<<
820A: 20 73 83    327         JSR   POINTEND    ;Get end byte
                  328         >>>   TRANS.AUXPTR;A2L ;Set end byte
820D: A5 04       328         LDA   AUXPTR
820F: 85 3E       328         STA   A2L
8211: A5 05       328         LDA   AUXPTR+1
8213: 85 3F       328         STA   A2L+1
                  328         <<<
8215: 20 D7 82    329         JSR   MEMMOVE
                  330
                  331 * Decrease number of sounds by 1:
                  332
8218: AD 20 03    333         LDA   SEQNUM      ;Get sequence number
821B: 20 91 83    334         JSR   SETOFF
821E: 20 A9 83    335         JSR   SETSEQ
8221: 38          336         SEC               ;Decrease Y by one
8222: A0 00       337         LDY   #0
8224: B1 00       338         LDA   (SNDPTR),Y
8226: E9 01       339         SBC   #1
8228: 91 00       340         STA   (SNDPTR),Y
                  341
                  342 * Subtract 6 from all of the succeeding offsets:
                  343
                  344         >>>   SETPTR.-6:ADJNUM
822A: A9 FA       344         LDA   #<-6
822C: 8D CB 83    344         STA   ADJNUM
822F: A9 FF       344         LDA   #>-6
8231: 8D CC 83    344         STA   ADJNUM+1
                  344         <<<
8234: 20 F7 80    345         JSR   ADJINDEX    ;Go adjust the index
                  346
                  347 * Zero the trailing 7 bytes:
                  348
8237: 20 73 83    349 ZEROEND JSR   POINTEND    ;Go point to the end
823A: A0 07       350         LDY   #7
823C: A9 00       351         LDA   #0
823E: 91 04       352 ZLOOP   STA   (AUXPTR),Y
8240: 88          353         DEY
8241: D0 FB       354         BNE   ZLOOP
8243: 60          355         RTS
                  356
                  357 *-------------------------------------------*
                  358 * STRTSND (start a new sound at end of table): *
                  359 *-------------------------------------------*
                  360
                  361 STRTSND
8244: AD 20 03    362         LDA   SEQNUM
8247: 20 91 83    363         JSR   SETOFF
824A: 20 A9 83    364         JSR   SETSEQ      ;Point to sequence
824D: 18          365         CLC               ;Add one to number snds
824E: A0 00       366         LDY   #0
8250: B1 00       367         LDA   (SNDPTR),Y
8252: 69 01       368         ADC   #1
8254: 91 00       369         STA   (SNDPTR),Y
8256: 60          370         RTS
                  371
                  372 STRTSEQ
```

LISTING 3: SEU *(continued)*

```
8257: 20 73 83  373           JSR   POINTEND    :Point to end
                374           >>>   INCR.AUXPTR :Point to one past end
825A: E6 04     374           INC   AUXPTR
825C: D0 02     374           BNE   NC
825E: E6 05     374           INC   AUXPTR+1
                374 NC        <<<
                375           >>>   TRANS.AUXPTR:A2L :Set end move
8260: A5 04     375           LDA   AUXPTR
8262: 85 3E     375           STA   A2L
8264: A5 05     375           LDA   AUXPTR+1
8266: 85 3F     375           STA   A2L+1
                375           <<<
8268: A0 00     376           LDY   #0
826A: A9 01     377           LDA   #1          :Set num of sounds to 1
826C: 91 04     378           STA   (AUXPTR),Y
                379
                380 *  Insert new pointer into sound table index:
                381
826E: A9 01     382           LDA   #1          :Point to curr 1st seq
8270: 20 91 83  383           JSR   SETOFF
8273: 20 A9 83  384           JSR   SETSEQ
                385           >>>   TRANS.SNDPTR:A1L :Set start of move
8276: A5 00     385           LDA   SNDPTR
8278: 85 3C     385           STA   A1L
827A: A5 01     385           LDA   SNDPTR+1
827C: 85 3D     385           STA   A1L+1
                385           <<<
                386           >>>   ADDC.A1L;2;A4L :Set destination
827E: 18        386           CLC
827F: A5 3C     386           LDA   A1L
8281: 69 02     386           ADC   #<2
8283: 85 42     386           STA   A4L
8285: A5 3D     386           LDA   A1L+1
8287: 69 00     386           ADC   #>2
8289: 85 43     386           STA   A4L+1
                386           <<<
828B: 20 D7 82  387           JSR   MEMMOVE     :Go move everything
                388
                389 *  Calculate offset for new sequence:
                390
828E: A0 00     391           LDY   #0
8290: 38        392           SEC
8291: A5 04     393           LDA   AUXPTR      :Get end byte address
8293: E9 00     394           SBC   #<SOUNDTBL
8295: 91 00     395           STA   (SNDPTR),Y
8297: C8        396           INY
8298: A5 05     397           LDA   AUXPTR+1
829A: E9 40     398           SBC   #>SOUNDTBL
829C: 91 00     399           STA   (SNDPTR),Y
                400
                401 *  Increase the number of sequences by one:
                402
829E: EE 00 40  403           INC   SOUNDTBL    :Set to new (correct) val
                404
                405 *  Add two to all of the indexes:
                406
                407           >>>   SETPTR,2:ADJNUM
82A1: A9 02     407           LDA   #<2
82A3: 8D CB 83  407           STA   ADJNUM
82A6: A9 00     407           LDA   #>2
82A8: 8D CC 83  407           STA   ADJNUM+1
                407           <<<
82AB: A9 01     408           LDA   #1          :Start with sequence #1
82AD: 8D 20 03  409           STA   SEQNUM
82B0: EE 00 40  410           INC   SOUNDTBL    :Temp for routine
82B3: 20 FC 80  411           JSR   ADJENTRY    :Go adjust index
82B6: CE 00 40  412           DEC   SOUNDTBL    :Back to correct value
82B9: AD 00 40  413           LDA   SOUNDTBL    :Set current seq num
82BC: 8D 20 03  414           STA   SEQNUM      : to last of sound table
82BF: 60        415           RTS               :End of STRTSEQ
                416
                417 *---------------------------------------------*
                418 *  CALCLEN (calculate length of file):        *
                419 *---------------------------------------------*
                420
                421 CALCLEN
82C0: 20 73 83  422           JSR   POINTEND    :Point to end of file
                423           >>>   INCR.AUXPTR :Point one past end
82C3: E6 04     423           INC   AUXPTR
82C5: D0 02     423           BNE   NC
82C7: E6 05     423           INC   AUXPTR+1
                423 NC        <<<
82C9: 38        424           SEC               :Take difference
82CA: A5 04     425           LDA   AUXPTR
82CC: E9 00     426           SBC   #<SOUNDTBL
82CE: 85 06     427           STA   FILELEN
82D0: A5 05     428           LDA   AUXPTR+1
82D2: E9 40     429           SBC   #>SOUNDTBL
82D4: 85 07     430           STA   FILELEN+1
82D6: 60        431           RTS
                432
                433 *---------------------------------------------*
                434 *  MEMMOVE (move a range of memory):          *
                435 *---------------------------------------------*
                436
                437 MEMMOVE
                438           >>>   COMP.A1L:A4L
82D7: A5 3C     438           LDA   A1L
82D9: C5 42     438           CMP   A4L
82DB: A5 3D     438           LDA   A1L+1
82DD: E5 43     438           SBC   A4L+1
                438           <<<
82DF: 90 05     439           BLT   MM:1
82E1: A0 00     440           LDY   #0          :Okay, so set Y for MOVE
82E3: 4C 2C FE  441           JMP   MOVE        :Exit thru Mon MOVE rtn
                442
                443 MM:1                        :Dest is above org
82E6: 38        444           SEC               :Calculate top of dest
82E7: A5 3E     445           LDA   A2L
82E9: E5 3C     446           SBC   A1L
82EB: 48        447           PHA
82EC: A5 3F     448           LDA   A2L+1
82EE: E5 3D     449           SBC   A1L+1
82F0: AA        450           TAX
82F1: 18        451           CLC
82F2: 68        452           PLA
82F3: 65 42     453           ADC   A4L
82F5: 85 42     454           STA   A4L
82F7: 8A        455           TXA
82F8: 65 43     456           ADC   A4L+1
82FA: 85 43     457           STA   A4L+1
                458
82FC: A2 00     459           LDX   #0          :Set dummy index
82FE: A1 3E     460 MVLOOP    LDA   (A2L,X)     :Get top byte
8300: 81 42     461           STA   (A4L,X)     :Move to top of dest
                462           >>>   COMP.A1L:A2L
8302: A5 3C     462           LDA   A1L
8304: C5 3E     462           CMP   A2L
8306: A5 3D     462           LDA   A1L+1
8308: E5 3F     462           SBC   A2L+1
                462           <<<
830A: B0 13     463           BGE   MV:END
                464           >>>   DECR.A2L
830C: A5 3E     464           LDA   A2L
830E: D0 02     464           BNE   ND
8310: C6 3F     464           DEC   A2L+1
8312: C6 3E     464 ND        DEC   A2L
                464           <<<
                465           >>>   DECR.A4L
8314: A5 42     465           LDA   A4L
8316: D0 02     465           BNE   ND
8318: C6 43     465           DEC   A4L+1
831A: C6 42     465 ND        DEC   A4L
                465           <<<
831C: 4C FE 82  466           JMP   MVLOOP
831F: 60        467 MV:END    RTS               :End of MEMMOVE
                468
                469 *---------------------------------------------*
                470 *  POINTSND (set pointers to sound in table): *
                471 *---------------------------------------------*
                472
                473 *  Call by putting current sequence # in A-reg.,
                474 *  and current sound in Y-reg.
                475 *  GENPTR points to start of sound, AUXPTR to end
                476
                477 POINTSND
8320: 20 91 83  478           JSR   SETOFF      :Point to offset
8323: 20 A9 83  479           JSR   SETSEQ      :Point to sequence
                480 SETSND                      :Set sound ENTRY point
                481           >>>   TRANS.SNDPTR:GENPTR
8326: A5 00     481           LDA   SNDPTR
8328: 85 02     481           STA   GENPTR
832A: A5 01     481           LDA   SNDPTR+1
832C: 85 03     481           STA   GENPTR+1
                481           <<<
                482           >>>   INCR.GENPTR :Point to first sound
832E: E6 02     482           INC   GENPTR
8330: D0 02     482           BNE   NC
8332: E6 03     482           INC   GENPTR+1
                482 NC        <<<
                483
8334: A9 00     484           LDA   #0          :Zero HOB
8336: 85 05     485           STA   AUXPTR+1
8338: 88        486           DEY               :Set current snd range
8339: 84 04     487           STY   AUXPTR      :Save sound number
833B: 06 04     488           ASL   AUXPTR      :Mult by 6
833D: 26 05     489           ROL   AUXPTR+1
                490           >>>   TRANS.AUXPTR:A3L
833F: A5 04     490           LDA   AUXPTR
8341: 85 40     490           STA   A3L
8343: A5 05     490           LDA   AUXPTR+1
8345: 85 41     490           STA   A3L+1
                490           <<<
8347: 06 04     491           ASL   AUXPTR
8349: 26 05     492           ROL   AUXPTR+1    :Make x4
                493           >>>   ADD.AUXPTR:A3L:AUXPTR
834B: 18        493           CLC
834C: A5 04     493           LDA   AUXPTR
834E: 65 40     493           ADC   A3L
8350: 85 04     493           STA   AUXPTR
8352: A5 05     493           LDA   AUXPTR+1
8354: 65 41     493           ADC   A3L+1
8356: 85 05     493           STA   AUXPTR+1
                493           <<<
                494           >>>   ADD.AUXPTR:GENPTR:GENPTR
8358: 18        494           CLC
8359: A5 04     494           LDA   AUXPTR
835B: 65 02     494           ADC   GENPTR
835D: 85 02     494           STA   GENPTR
835F: A5 05     494           LDA   AUXPTR+1
8361: 65 03     494           ADC   GENPTR+1
8363: 85 03     494           STA   GENPTR+1
                494           <<<
                495           >>>   ADDC.GENPTR;5;AUXPTR
8365: 18        495           CLC
8366: A5 02     495           LDA   GENPTR
8368: 69 05     495           ADC   #<5
836A: 85 04     495           STA   AUXPTR
836C: A5 03     495           LDA   GENPTR+1
836E: 69 00     495           ADC   #>5
8370: 85 05     495           STA   AUXPTR+1
                495           <<<
8372: 60        496           RTS               :End of POINTSND
                497
                498 *---------------------------------------------*
                499 *  POINTEND (set GENPTR to last adrs in table): *
                500 *---------------------------------------------*
```

LISTING 3: SEU *(continued)*

```
                        501
                        502  POINTEND
                        503          >>>   SETPTR.SOUNDTBL;SNDPTR
8373: A9 00             503          LDA   #<SOUNDTBL
8375: 85 00             503          STA   SNDPTR
8377: A9 40             503          LDA   #>SOUNDTBL
8379: 85 01             503          STA   SNDPTR+1
                        503          <<<
837B: A0 00             504          LDY   #0
837D: B1 00             505          LDA   (SNDPTR),Y ;Get number of sequences
837F: 8D C8 83          506          STA   NUMSEQ     ;Save number of sequ's
8382: 20 91 83          507          JSR   SETOFF     ;Get index to last seq
8385: 20 A9 83          508          JSR   SETSEQ     ;Point to last seq
8388: B1 00             509          LDA   (SNDPTR),Y ;Get number of sounds
838A: A8                510          TAY              ;Make sound number
838B: AD C8 83          511          LDA   NUMSEQ     ;Get last sound
838E: 4C 20 83          512          JMP   POINTSND   ;Exit thru point sound
                        513
                        514  *-----------------------------------------*
                        515  *  SETOFF, SETSEQ (set sound pointer routines):  *
                        516  *-----------------------------------------*
                        517
                        518  *  Input:  Current sequence number in A-Reg.
                        519  *  Output: SNDPTR points to current sound sequence
                        520
                        521  SETOFF
8391: A2 00             522          LDX   #0         ;Zero the MSB
8393: 86 01             523          STX   SNDPTR+1
8395: 85 00             524          STA   SNDPTR
8397: 06 00             525          ASL   SNDPTR     ;2 bytes per offset
8399: 26 01             526          ROL   SNDPTR+1   ; so multiply by 2
839B: 18                527          CLC
839C: A9 00             528          LDA   #<SOUNDTBL
839E: 65 00             529          ADC   SNDPTR
83A0: 85 00             530          STA   SNDPTR     ;Save as sound pointer
83A2: A9 40             531          LDA   #>SOUNDTBL ; to point to index
83A4: 65 01             532          ADC   SNDPTR+1
83A6: 85 01             533          STA   SNDPTR+1
83A8: 60                534          RTS              ;End of SETOFF
                        535
                        536  SETSEQ
83A9: 8C CA 83          537          STY   YSAVE      ;Save Y-Reg.
83AC: A0 00             538          LDY   #0         ;Zero the index
83AE: B1 00             539          LDA   (SNDPTR),Y ;Get LOB of offset
83B0: 85 40             540          STA   A3L
83B2: C8                541          INY
83B3: B1 00             542          LDA   (SNDPTR),Y
83B5: 85 41             543          STA   A3L+1      ;Save temporarily
                        544          >>>   ADDC.A3L;SOUNDTBL;SNDPTR
83B7: 18                544          CLC
83B8: A5 40             544          LDA   A3L
83BA: 69 00             544          ADC   #<SOUNDTBL
83BC: 85 00             544          STA   SNDPTR
83BE: A5 41             544          LDA   A3L+1
83C0: 69 40             544          ADC   #>SOUNDTBL
83C2: 85 01             544          STA   SNDPTR+1
                        544          <<<
83C4: AC CA 83          545          LDY   YSAVE      ;Restore register
83C7: 60                546          RTS              ;End of SETSEQ
                        547
83C8: 00                548  NUMSEQ   DS    1          ;Number of sequences
83C9: 00                549  COUNT    DS    1          ;Loop counter
83CA: 00                550  YSAVE    DS    1          ;Temp save of Y-Reg.
83CB: 00 00             551  ADJNUM   DS    2          ;Number to adjust index
```

END OF LISTING 3

```
KEY PERFECT 5.0
RUN ON
SEU

=====================================
CODE-5.0    ADDR# - ADDR#    CODE-4.0
--------    -------------    --------
78D399D0     8000 - 804F        26DE
33AD871D     8050 - 809F        297A
CDCD1C3D     80A0 - 80EF        2B1D
C1B1DDAC     80F0 - 813F        27BA
89C39B6A     8140 - 818F        2842
BC102266     8190 - 81DF        29AB
A0000245     81E0 - 822F        2757
A9C55067     8230 - 827F        2676
7FD2248E     8280 - 82CF        2A5D
BDF63774     82D0 - 831F        23CF
D2F2E15F     8320 - 836F        298D
72422CC0     8370 - 83BF        26F9
5F5DF6A4     83C0 - 83C7        043C
A7D43F3D = PROGRAM TOTAL =      03C8
```

LISTING 4: DUO.DEMO

```
10  REM ************************
20  REM *                      *
30  REM *       DUO.DEMO       *
40  REM * BY SCOTT ZIMMERMAN   *
50  REM * COPYRIGHT (C) 1987   *
60  REM * BY MICROSPARC, INC.  *
70  REM * CONCORD, MA. 01742   *
80  REM *                      *
90  REM ************************
100  REM *----------------
110  REM * INTRODUCTION:
120  REM *----------------
130 HI = 37376: HIMEM: HI: TEXT : HOME
140 A$ = " DUO.DEMO ": INVERSE : GOSUB 480: NORMAL
    : VTAB 3:A$ = "BY S. SCOTT ZIMMERMAN": GOSUB
    480
150 A$ = "COPYRIGHT (C) 1987": GOSUB 480:A$ =
    "BY MICROSPARC, INC": GOSUB 480:NF = 1: ONERR
     GOTO 640
160  PRINT  CHR$ (4)"VERIFY DEMO.SNDS": PRINT
     CHR$ (4)"VERIFY DEMO.NMS"
170 L = 80:AS = HI - L - 1: IF NF THEN  PRINT
    CHR$ (4);"BLOAD DEMO.SNDS,A";AS
180  ONERR  GOTO 650
190 AD = AS - 445: PRINT  CHR$ (4);"BRUN DUO,
    A";AD
200 AX = 256 *  INT (AD / 256) - 512: HIMEM:
    AX
210  ON  NOT NF GOTO 240: PRINT  CHR$ (4);"OP
    EN DEMO.NMS": PRINT  CHR$ (4);"READ DEMO
    .NMS"
220  INPUT N: DIM S$(N),VT(N): FOR I = 1 TO N
    : INPUT S$(I):VT(I) = 6 + I: NEXT I
230  PRINT  CHR$ (4);"CLOSE"
240  IF  NOT NF THEN  GOSUB 550
250  DEF  FN HB(A) =  INT (A / 256): DEF  FN
    LB(A) = A -  FN HB(A) * 256
260  POKE 206, FN LB(AS): POKE 207, FN HB(AS)
270  &  NORMAL
280  REM *----------------
290  REM * SETUP:
300  REM *----------------
310  FOR I = 1 TO N: VTAB VT(I): HTAB 12: PRINT
    I;". ";S$(I): NEXT I
320  VTAB 22:A$ = "PRESS A NUMBER TO MAKE A S
    OUND": GOSUB 480
330 A$ = "Q)UIT, S)TOP, P)OP, N)ORMAL": GOSUB
    480
340  REM *----------------
350  REM * MAIN LOOP:
360  REM *----------------
370  IF  PEEK ( - 16384) < 128 THEN 370
380  GET A$: GOSUB 520: IF A$ < "1" OR A$ > "
    Z" THEN 370
390  IF A$ = "Q" THEN  TEXT : HOME : END
400  IF A$ = "S" THEN  &  STOP
410  IF A$ = "P" THEN  &  POP
420  IF A$ = "N" THEN  &  NORMAL
430 A =  VAL (A$): IF A < 1 OR A > N THEN 370
440  GOSUB 490: GOTO 370
450  REM *----------------
460  REM * SUBROUTINES:
470  REM *----------------
480  HTAB (41 -  LEN (A$)) / 2: PRINT A$: RETURN

490  VTAB VT(A): HTAB 11: INVERSE : PRINT " "
    ;A;". ";S$(A);" ": NORMAL
500  & A
510  VTAB VT(A): HTAB 11: PRINT " ";A;". ";S$
    (A);" ": RETURN
520 A =  ASC (A$): IF A > 95 THEN A = A - 32:
    A$ =  CHR$ (A)
530  RETURN
540  REM  READ IN DEMO SOUND TABLE
550  RESTORE : FOR I = 0 TO 79: READ ML: POKE
    AS + I,ML: NEXT I
560  DATA  4,0,10,0,23,0,36,0,55,0,2,180,0,1,
    6,25,0,210,209,1,0,10,0,2,252,250
570  DATA  1,2,60,0,250,250,255,254,130,0,3,2
    00,199,1,6,30,0,250,250,255,254,20,0,214
    ,213,255
580  DATA  255,200,0,4,170,0,0,0,130,0,190,0,
    0,0,130,0,170,0,0,0,130,0,190,0,0,0
590  DATA  130,0,205
600  REM  READ IN SOUND NAMES
610 N = 4: DIM VT(N),S$(N): FOR I = 1 TO N: READ
    S$(I):VT(I) = 6 + I: NEXT : RETURN
620  DATA  ZAP!,BOOM!,KAPOWEE!,SIREN
630  REM  ERROR TRAP
640 NF = 0: GOTO 170
650  CALL  - 3288: HOME : VTAB 12: PRINT "TRO
    UBLE LOADING DUO": VTAB 22: PRINT "<ESC>
     TO QUIT, <RETURN> TO TRY AGAIN";: GET Z
    $: PRINT : ON Z$ <  >  CHR$ (27) GOTO 17
    0: END
```

END OF LISTING 4