Color Computer Speech/Sound Cartridge

Radio Shack

The FCC Wants You to Know…

This equipment generates and uses radio frequency energy. If it is not installed and user properly, that is, in strict accordance with the manufacturer's instructions, it may cause interference to radio and television reception. It has been type tested and found to comply with limits for a Class B computing device, pursuant to Subpart J or Part 15 of FCC Rules, which are deisgned to provide reasonable protection against such interference in a residential installation.

GI PIC 7040-510 System Software:

© 1984 General Instrumen Corporation

All Rights Reserved

The system software in the GI PIC 7040 microprocessor is retained in a read–only memory (ROM) format. All portions of this system software, wether in the ROM format or other source code form format, and the ROM circuitry are copyrighted and use the proprietary and trade secret information of General Instruments Corporation. Use, reproduction, or publication of this material without the prior written authorization by General Instruments Corporation is strictly prohibited.

Color Computer Speech/Sound Cartridge Owner's Manual:

© 1984 Tandy Corporation

All Rights Reserved

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors on ommissions in this manual, or from the use of the information contained herin.

The chart on page 22 and the material in appendices B, C, and D are used by permission of General Instrument Corporation.

Features

The color computer Speech/Sound Cartridge is an *intelligent* peripheral with a self-contained microprocessor that facilitates the creation of speech, music and sound effects. The Speech/Sound Cartridge:

- Converts text output from the Color Computer into speech
- Plays 3-part musical harmony
- Created complex sound effects
- Interfaces with game, educational, and other applicatio programs
- Is programmable—with 8 speech and 8 sound storage buffers
- Can be used with disk, cassette programs or Program Paks (Use with a disk drive or Program Pak requires a Tandy Multi-Pak Interface.)

Contents

Introduction 6

Background Information 8

Speaking of the S/SC (Text to Speech) 9

Straight talk (Commands and allophones) 12

Sound tdvice (Commands and sounds) 15

Dial tone (Creating tones) 17

As plain as the noise (Creating noises) 19

The envelope, please (Shaping sound envelopes) 21

Scaling the sound registers (Direct access to registers) 24

Appendicies

A (Command Instruction Map) 26

B (Sound Generator Registers) 31

C (Allophone Address Table) 35

D (Equal-Tempered Chromatic Scale) 37

E (Demonstration Program) 40

F (Sound Organ) 49

G (Sound Effects) 51

H (Text to Speech~with screen editor) 53

INTRODUCTION

The value of the Color Computer Speech/Sound. Cartridge (S/SC) speaks for itself.

This versatile Color Computer peripheral lets you use your computer to speak words and sentences, generate complex sound effects, and play music in a 8-octave range on 3 independent channels. It also opens new dimensions for integrating speech and sound effects into all types of applications—from games to educational programs.

This manual contains a great deal of technical information intended primarily for advanced programmers. However, anyone who owns a Color Computer, including the novice and the machine-language programmer, can use the S/SC.

Getting your Color Computer to speak is simple.

• Plug the S/SC into the Program Pak slot on the right side of your Color Computer.

• Type in the program on Page 11 of this manual.

• Run the program and then type in anything you want the computer to say.

That's it. (Save this program on cassette so you can use it again without retyping it. If you have a Multi-Pak Interface, you can save the program on disk.)

Many applications programs will use the S/SC automatically. If these programs are cassette-based, all you have to do is plug the S/SC into the Program Pak slot and then load the cassette program.

If a Program Pak application program uses the S/SC, you need a Multi-Pak Interface. You may use the Speech/Sound Cartridge and Program Pak in any 2 Interface slots. Some Color Computer software and the OS-9 Operating System expect a Disk Drive Controller in Slot 4 or a Communications package in Slot 1, and these may not function properly with the S/SC in those slots. Slots 2 and 3 of the Interface, however, are always available for the S/SC and Program Paks.

If a disk-based application program uses the S/SC, you need a Multi-Pak

Interface. You may use the S/SC and Disk Drive Contoller in any two Interface slots. However, some Color Computer software and the OS-9 Operating System expect the Disk Drive Controller in Slot 4 or a Communications package in Slot 1 and may not function properly with the S/SC in those slots. Slots 2 and 3 are always available for the S/SC. When you run the disk-based program, it automatically uses the S/SC.

Although this manual contains much technical information, it also contains many tutorial and demonstration programs. The first 3 programs in the following list are limited in scope and demonstrate basic principles for programming the S/SC. The last 4 programs are more sophisticated, and we include them so that nonprogrammers can explore many of the capabilities of the S/SC.

Text–to–Speech Program	Page 9
Allophone–to–Speech Program	Page 13
Direct Access (Register) Program	Page 25
Cartridge Demonstration Program	Page 40
Organ Program	Page 49
Sound Effects Program	Page 51
Text–to–Speech Program (with editor)	Page 53

To use these programs, just type them in, save them on cassette or disk, and run them whenever you want. They demonstrate many of the various capabilities of the S/SC.

If you know how to program in BASIC or in machine language, you can use these programs to gain information quickly about the capabilities of the S/SC. Then you can modify and improve on them to use the S/SC to its greatest advantage.

Note: The programs in this manual are written for a Color Computer with Extended Color BASIC, but they may be modified for computers with Non-Extended Color BASIC by deleting the PEEK commands and converting hexidecimal numbers (&Hxxxx) to their decimal equivalents.

BACKGROUND INFORMATION

An "*intelligent*" peripheral, the Speech/Sound Cartridge (S/SC) contains an on-board microprocessor that controls both the speech generator and the sound generator and routes an RF signal through the Color Computer and into the television speaker.

This processor has 4K ROM and 2K RAM. In addition to providing you with an easier means of programming the cartridge, the S/SC RAM also lets you store as many as eight 64-character sentences or eight 64-byte sound effects for later execution. You can activate the sentences or sounds you store in the buffers at any time during program execution by using simple single-byte commands.

The S/SC generates speech and sound with separate processors. You may produce speech by two different methods and sound or music by another two methods.

> **Note:** The pitch and duration of the sounds listed in this manual are produced by an S/SC used with a Color Computer running at a clock speed of .89 MHz. Since the S/SC internally multiplies the Color Computer clock by a factor of 2, the S/SC operates at a clock frequency of 1.78977 MHz. This is the clock referred to in Appendix D.

SPEAKING OF THE S/SC

You can use the S/SC to produce speech by running a short program to transfer text from the Color Computer keyboard to the cartridge, where English words are directly translated into speech. You can also program the S/SC to create words (English or foreign) by combining small units of speech sounds.

Spoken language is the combination of a certain set of small sounds and syllables, called allophones, into recognizable words. The S/SC's General Instruments SP0256 speech processor generates allophones. The on-board microprocessor uses a set of ROM-based phonetic rules to combine allophones so that the English text you enter at the keyboard is converted into speech.

When you use this text-to-speech mode, the S/SC'S large set of phonetic rules in ROM lets you produce intelligible speech without special phonetic spellings. However, some English words are pronounced contrary to the usual phonetic rules. In these cases, you may have to "misspell" a word to force correct pronunciation. For example, the S/SC pronounces BOW as BO. That's fine. To say BOW as in "bow to the king," however, you have to spell BOW as BOU.

Before you can translate text to speech, however, you must use a short BASIC or machine code program to transfer text data between the microprocessor in the Color Computer and the microprocessor in the S/SC. The following BASIC driver routine illustrates the text-to-speech capabilities of the S/SC:

```
10 REM INITIALIZE VARIABLES
20 X=&HFF00; Y=&HFF7E
30 REM SET COCO SOUND MULTIPLEXER TO CARTRIDGE INPUT
40 POKE X+1,52;POKE X+3,63
50 REM ENABLE SOUND MULTIPLEXER
60 POKE X+35,60
70 REM INPUT STRING TO BE SPOKEN
80 INPUT A$
90 GOSUB 120
100 GOTO 80
110 REM SUBROUTINE TO OUTPUT STRING
120 FOR I=1 TO LEN(A$)
130 REM CHECK FOR BUSY*
140 IF (PEEK(Y) AND 128)=0 THEN 140
150 REM OUTPUT EACH CHARACTER
160 POKE Y, ASC(MID$(A$,I,1))
```

```
  170 NEXT I
  180 IF (PEEK(Y) AND 128)=0 THEN 180
  190 REM POKE A CARRIAGE RETURN
  200 POKE Y,13
  210 RETURN
```

Run the program. At ? prompt, type:

```
  I CAN TALK <SPACE BAR> <ENTER>
```

It really can.

> **Note:** Be sure to end each string of text to be spoken with a space or any punctuation mark except a comma. The S/SC interpets spaces and punctuation marks as pauses, and these "silences" ensure that the speech processor is actually turned off after speaking each typed phrase or sentence.

The above program demonstates the straight text-to-speech capabilities of the S/SC. It sends a stream of data bytes, each an ASCII character, to the S/SC for conversion into speech. Speech does not begin, however, until you send a carriage return (hex 0D; decimal 13) to the S/SC.

You need only use 2 addresses to transfer data between the Color Computer and the S/SC: hex FF7D (decimal 65405) and hex FF7E (decimal 65406). The lower address is a software reset for the S/SC, which you may use to reinitialize the entire cartridge if necessary by POKEing a 1 and then a 0 at that address. The upper address is the address you use to transfer data to the S/SC.

Hex FF7E also contains the status of the S/SC. Whenever you read (PEEK) this address, the S/SC returns a status byte to the Color Computer's 6809 processor. All status bits are active low. The most significant bit (bit 7) is called BUSY*, and whenever it returns a 0, the S/SC's microprocessor hasn't yet processed the last byte received from the Color Computer's microprocessor.

If you try to transfer data to the S/SC while bit 7 is low, you lose all the data you send until the bit resets. For this reason, you should monitor bit 7 every time you send data to the S/SC.

When low, bit 6 of the status byte indicates that the S/SC is currently speaking. The bit returns to a high state when the current phrase is finished.

When low, bit 5 of the status byte indicates that a sound effect is in progress. None of the other bits in the status register are used, and they should be ignored.

> **Note:** Whenever you write a program to transfer data to the S/SC, you should first monitor the status byte to determine whether or not the S/SC is ready to accept another character. Depending on the intent of your program, you may or may not be required to monitor the other status bits.

If the S/SC is not busy (that is, if BUSY* is high), then you may simply write or POKE your data into hex FF7E.

> **Note:** Due to the time required for the S/SC's microprocessor to execute a command, the speech and sound status bits are not valid immediately following a speech or sound execution command. The S/SC microprocessor requires different lengths of time to execute a command, depending on the complexity of the command. Speech commands take longer than sound commands. For this reason, you should have, as part of your test routine, a "wait" loop that executes prior to testing status. The duration of the "wait" loop may vary, depending on the time required to execute the previous command.

STRAIGHT TALK

The most significant bit of each character sent to the S/SC must be cleared, otherwise it initiates a command sequence. Command sequences begin with a character in which the most significant bit is set; therefore, the initial character in a command sequence will be 80 hex (128 decimal) or greater. S/SC commands let you store speech strings and allophone address streams in 1-8 buffers and store sound commands and register strings in another 1-8 buffers. Additional commands let you later execute the contents of those buffers singly or in combination.

A complete list of individual commands appears in Appendix A. The following table briefly outlines the command groupings:

DEC	HEX	COMMAND
128-135	80-87	Loads speech string into consecutive buffers
136-142	88-8E	Loads sound data into consecutive buffers
143	8F	Loads timer base value
144-151	90-97	Loads speech string into individual buffers
152-159	98-9F	Loads sound data into individual buffers
160-167	A0-A7	Loads allophone address stream into consecutive buffers
168-174	A8-AE	Loads register string into consecutive buffers
175	AF	Allows direct access to sound registers
176-183	B0-B7	Loads allophone address stream into individual buffers
184-191	B8-BF	Loads register string into individual buffers
192-198	C0-C6	Executes speech string from consecutive buffers
199	C7	Aborts all speech
200-206	C8-CE	Executes sound command from consecutive buffers
207	CF	Stops all sound
208-215	D0-D7	Executes speech string from individual buffers

216-223	D8-DF	Executes sound data from individual buffers
224-231	E0-E7	Executes allophone address stream from various combinations of buffers
232-239	E8-EF	Executes register string from consecutive buffers
240-247	F0-F7	Executes allophone address stream from individual buffers
248-255	F8-FF	Executes register string from individual buffers
00	00	Stops all sounds, including speech, but does not clear buffers.

With the above commands you can load or execute strings from the buffers in RAM. Each buffer stores a maximum of 64 characters. You may, however, store strings longer than 64 characters by loading consecutive buffers. Command 80, for example, lets you store a speech string of a maximum of 512 characters by loading the string into Buffers 0 through 7. Command 8E lets you store a maximum of 128 characters in Buffers 6 and 7.

In "Speaking of the S/SC" you produced speech by using the speech-to-text buffer that automatically links allophones and pronounces them according to a series of phonetic rules. The above commands let you generate speech by linking allophones in any manner you choose. You use this method to generate speech when programming in foreign languages that do not follow English phonetic rules. You might also use this method to create more precise (or regionally accented) pronunciations of English words.

The following program shows, step-by-step, how to produce speech by poking individual allophones into Buffer 0 and then later executing them.

> **Note:** You must end allophone data with a pause (addresses 0-4) to ensure that you silence the speech processor.

```
10 RESET S/SC
20 POKE &HFF7D,1:POKE &HFF7D,0
30 REM INITIALIZE VARIABLES
40 X=&FF00: Y=&HFF7E
50 REM TURN ON S/SC SOUND
60 POKE X+1,52: POKE X+3,63: POKE X+35,60
70 GOSUB 200
```

```
80 REM COMMAND B0 -- LOAD ALLOPHONES INTO BUFFER 0
90 POKE Y,176
100 READ ALLOPHONE ADDRESS STREAM TO THE S/SC
110 FOR A=1 TO 17
120 READ D
130 GOSUB 200
140 POKE Y,D
150 NEXT A
160 REM COMMAND F0 -- EXECUTE ALLOPHONE ADDRESS STREAM FROM
BUFFER 0
170 GOSUB 200
180 POKE Y,240
190 END
200 REM CHECK FOR BUSY*
210 IF (PEEK(Y) AND 128)=0 THEN 210
220 RETURN
230 REM ALLOPHONE DATA ENDS WITH A PAUSE AND TERMINATOR
240 DATA 8,15,45,51,4,4,4,8,15,16,9,49,31,13,51,4,255
```

When you run the program, the computer says, “Color Computer.”

The allophones in the data line are:

KK3 AX LL ER1 = COLOR

PA5 PAS PA5 = 600 MILLISECOND PAUSE

KK3 AX MM PP YY1 UW2 TT2 ER1 = COMPUTER

PA5 = PAUSE SILENCES SPEECH PROCESSOR

Appendix C contains a complete list of addresses for the 64 allophones that the speech processor produces.

SOUND ADVICE

You may program the Speech/Sound Cartridge to generate sound by 2 different methods. You may use the S/SC commands followed by a series of postbytes specifying the channel, amplitude, pitch, and duration of a tone or noise, or you may directly manipulate the sound processor's registers to produce the appropriate sound.

Using the S/SC commands to produce sound effects or music is a simple process in which you give a command and follow it with data groups of 3 or 4 postbytes each group defining either a tone, a noise, or an envelope.

Poking hex 98 at &HFF7E, for example, lets you load a maximum of 64 bytes of sound data into Buffer 0. Below is a bit-by-bit description of the types of postbyte data that follow sound commands.

The first postbyte after a sound command determines the type of operation to be carried out (tone and channel; noise and channel; or envelope) and the amplitude of the tone or noise. The first 3 most significant bits of this postbyte contain the following operation code:

Bit 7	Bit 6	Bit 5	Operation	Postbytes Required
0	0	0	Tone A	4
0	0	1	Tone B	4
0	1	0	Tone C	4
0	1	1	Envelope	4
1	0	0	Noise A	3
1	0	1	Noise B	3
1	1	0	Noise C	3
1	1	1	Envelope	4

You must terminate all command sequences with an FF hex (255), and you must send the terminator when the S/SC is expecting a new "first byte." For example, if you send the first byte of a tone event, the S/SC interprets the next 3 bytes as data for the tone. *An FF (255) in one of these 3 positions is read as data—not as a terminator.* Send the terminator when the S/SC is expecting you to define a specific sound event in the first postbyte.

Note: Each measured buffer accommodates 64 bytes of information. *Always allow room in the last buffer of a command sequence for a terminator*. If the last byte in a buffer is not a terminator, the S/SC processor automatically inserts a terminator in the first byte of the last sound event of the buffer. That last sound event, therefore, is never executed. After the processor inserts the terminator, it automatically reverts to the normal input mode.

DIAL TONE

If Postbyte 1 specifies a tone (the amplitude of which is either fixed or controlled by an envelope), 3 additional postbytes are needed to complete the description. Postbyte 2 specifies the Coarse Tone Period; Postbyte 3 specifies the Fine Tone Period; and Postbyte 4 determines the duration of the tone.

The TONE data group breaks down into the following 4 bytes:

Byte 1

Bit 7	When this bit is low, a tone occurs.
Bits 6-5	Select which channel the tone uses: 00 = Channel A 01 = Channel B 10 = Channel C If both bits are set, this is an envelope command. (See Envelope below.)
Bit 4	If this bit is clear, the tone has a fixed amplitude as specified in the next 4 bits. If this bit is set, an envelope controls the amplitude of the tone, and an envelope-creating data group must immediately follow. (See Envelope below.)
Bits 3-0	When Bit 4 is clear, this value sets a fixed amplitude for the tone in the range 0 (silence) to 15 (maximum amplitude).

Byte 2

Bits 7-4	Unused
Bits 3-0	Set the Coarse Tone Period value. The sound processor requires 12 bits to determine a period, and these 4 bits are the most significant period bits.

Byte 3

Bits 7-0	Set the Fine Tone Period value. These are the least significant bits in the 12-bit period value. The actual frequency of the final tone is 111,860 divided by the 12-bit value. Therefore, the higher the value,

the lower the frequency. Accordingly, the lowest possible S/SC frequency is approximately 27 Hz (118,860 divided by 4096).

Note: The listing in Appendix D shows the specific combinations of coarse and fine period values that produce precise notes within the S/SC's 9-octave range.

Byte 4

Bits 7-0 Set the duration of the tone. Duration is relative, the shortest being 0 and the longest being 255. Duration is also relative depending on the value in the base time register (accessible by command hex 8F).

Note: A duration in a buffer's sound sequence is the length of time between "events"—both sounds and silences. Therefore, if the final event in a buffer is not a silence, the previous sound event continues until another command cancels it. Unless you want a sound to continue past the end of the buffer sound sequence, always place a silence (a command with an amplitude of 0) at the end of a buffer.

Tone Postbyte Table

Postbyte #	MSB							LSB
1	O2	O1	O0	M	A3	A2	A1	A0
2	X	X	X	X	C3	C2	C1	C0
3	F7	F6	F5	F4	F3	F2	F1	F0
4	D7	D6	D5	D4	D3	D2	D1	D0

O = OPERATION CODE
M = FIXED/ENVELOPE FLAG BIT
X = UNUSED
A = AMPLITUDE VALUE (0-15)
C = COARSE TONE PERIOD VALUE (0-15
F = FINE TONE PERIOD VALUE (0-255)
D = TONE DURATION VALUE (0-255 relative to the value in the base time register—accessible by command hex 8F)

AS PLAIN AS THE NOISE

If the first postbyte following a command specifies a noise (and its amplitude), only 2 additional bytes are needed to describe it fully. Postbyte 2 sets the coarse period value, and Postbyte 3 determines the duration.

The NOISE data group breaks down into the following three bytes:

Byte 1

Bit 7	When this bit is set, a noise occurs.
Bits 6-5	Select which channel the noise uses: 00 = Channel A 01 = Channel B 10 = Channel C If both bits are set, this an envelope command. (See Envelope below.)
Bit 4	If this bit is clear, the noise has a fixed amplitude as specified in the next 4 bits. If this bit is set, an envelope controls the amplitude of the noise, and an envelope-creating data group must immediately follow. (See Envelope below.)
Bits 3-0	This value sets a fixed amplitude for the noise in the range 0 (silence) to 15 (maximum amplitude). (This value is ignored if Bit 7 of the next byte is low.)

Byte 2

Bit 7	If this bit is low, the last 4 bits of the previous byte determine amplitude of the noise. If this bit is set, however, the amplitude of the preceding data group is used, and the amplitude bits in the first byte are ignored. Notice that when this bit is set in the first sound data group of a new command (that is, without a preceding value), the amplitude is automatically set to 0 (silence).
Bits 6-5	Unused
Bits 4-0	This 5-bit noise period value determines the average frequency of the random sound pressures that produce the noise. The larger the value, the lower the noise frequency. is treated as 31 + 1 (32), and

produces the lowest frequency.

Byte 3

Bits 7-0 Set the duration of the noise. Duration is relative, the shortest being 0 and the longest being 255. Duration is also relative depending on the value in the base time register (accessible by command hex 8F).

Noise Postbyte Table

Postbyte #	MSB							LSB
1	O2	O1	O0	M	A3	A2	A1	A0
2	R	X	X	P4	P3	P2	P1	P0
3	D7	D6	D5	D4	D3	D2	D1	D0

O = OPERATION CODE
M = FIXED/ENVELOPE FLAG BIT
A = AMPLITUDE VALUE (0-15)
R = REPLACE/CONTINUE BIT
P = NOISE PERIOD VALUE (0-31)
D = NOISE DURATION VALUE (0-255)

THE ENVELOPE, PLEASE

The final type of command is the envelope command. This command lets you wrap *previously generated* tones and noises in an envelope of sound. Only one envelope shape is available to all 3 channels. Each channel may use the envelope independently of the others, or all may share the single shape at one time.

If the first postbyte following a command specifies an envelope (and its 4 *shape characteristics*), 3 additional bytes are required to describe the envelope. Postbyte 2 sets the coarse period value; Postbyte 3 sets the fine period value; and Postbyte 4 sets the duration of the envelope.

Envelopes control the amplitude of *previously issued* tones or noises. For example, you can use an envelope to create a vibrato effect. First, store a tone with the envelope bit set (Bit 4 of Byte 1), and then immediately create a slowly repeating envelope. The 4-byte envelope data group controls the tone or noise of the preceding byte. The ENVELOPE data group breaks down into the following 4 bytes:

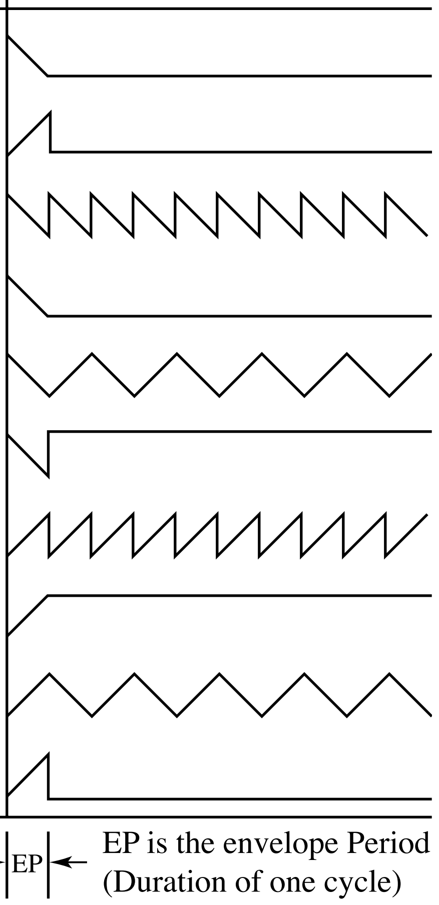
Byte 1

Bit 7	Unused
Bits 6-5	When both are set, an envelope is established
Bit 4	Unused
Bits 3-0	These bits control the shape of the envelope: **Bit 3** = the CONTINUE bit. CONTINUE tells the processor whether to play the envelope's sound just once or repeat it. When the bit is set, the tone repeats; when the bit is cleared, the sound occurs once and then quits. **Bit 2** = the ATTACK bit. ATTACK determines whether a noise or tone builds from minimum amplitude to maximum or subsides from maximum to minimum. When the bit is set, the sound builds; when the bit is cleared, the sound subsides. **Bit 1** = the ALTERNATE bit. ALTERNATE changes the attack with each cycle. When the bit is set, the noise or tone builds, then fades, then builds, and so on. When the bit is cleared, the attack is the same as specified in Bit 2.

Bit 0 = the HOLD bit. HOLD maintains the amplitude of a tone or noise at the level reached by the attack—maximum or silence.

The following chart shows how the various envelope shape parameters modulate the amplitude of sounds:

Register 13			
B3	B2	B1	B0
Continue	Attack	Alternate	Hold
0	0	X	X
0	1	X	X
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1



Byte 2

Bits 7-0 Set the Coarse Tone Period value of the envelope. The envelope period uses a base frequency of 6991 Hz divided by a 16-bit value contained in this and the following byte. The highest envelope frequency, therefore, is 6991 Hz, and the lowest is approximately .1 Hz (6991 divided by 65536). This slowest frequency provides an attack time of almost 10 seconds.

Byte 3

Bits 7-0 Set the Fine Tone Period value of the envelope. It is used with Byte 2 to produce a 16-bit value.

Byte 4

Bits 7-0 Set the duration of the envelope. Duration is relative, the shortest being 0 and the longest being 255. Duration is also relative depending on the value in the base time register (accessible by command hex 8F).

Envelope Postbyte Table

Postbyte #	MSB							LSB
1	O2	O1	O0	X	S3	S2	S1	S0
2	C7	C6	C5	C4	C3	C2	C1	C0
3	F7	F6	F5	F4	F3	F2	F1	F0
4	D7	D6	D5	D4	D3	D2	D1	D0

O = OPERATION CODE
X = UNUSED
S = ENVELOPE SHAPE BITS
C = COARSE ENVELOPE PERIOD VALUE
F = FINE ENVELOPE PERIOD VALUE
D = ENVELOPE DURATION VALUE

SCALING THE SOUND REGISTERS

One command—AF—lets you directly accesss the sound processor's 13 registers to create sound effects and music. In other words, you can transfer pairs of bytes directly to the S/SC. The first byte you poke in at hex FF7E is the register number (1-13). The next number you poke in at hex FF7E is a value within the register's parameters.

> **Note:** Different registers have different ranges of valid input. Register 0 (Channel A tone—Fine Period value), for example, uses data from 0-255; Register 1 (Channel A tone—Coarse Period value), however, uses only data from 0-15.

The byte pairs you poke in (register # followed by data) are transferred "on the fly" into the sound generator until you send a terminator (FF hex).

The following table outlines the S/SC registers, functions, and data ranges. (For a complete discussion of the S/SC, see Appendix B) .

#	Function	Data Range
0	Tone generator—Channel A—Fine Period	0-255
1	Tone generator—Channel A—Coarse Period	0-15
2	Tone generator—Channel B—Fine Period	0-255
3	Tone generator—Channel B—Coarse Period	0-15
4	Tone generator—Channel C—Fine Period	0-255
5	Tone generator—Channel C—Coarse Period	0-15
6	Noise generator	0-31
7	Mixer Control	0-63
8	Amplitude control—Channel A	0-15
9	Amplitude control—Channel B	0-15
10	Amplitude control—Channel C	0-15
11	Envelope Fine Period Control	0-255
12	Envelope Coarse Period Control	0-255
13	Envelope shape control	0-15

The following short program lets you manipulate a tone on Channel A by directly controlling the S/SC's registers:

```
10 REM RESET S/SC
20 POKE &HFF7D,1:POKE &HFF7D,0
30 REM INITIALIZE VARIABLES
40 X=&HFF00: Y=&HFF7E
50 REM TURN ON S/SC SOUND
60 POKE X+1,52: POKE X+3,63: POKE X+35,60
70 REM PUT S/SC INTO DIRECT ACCESS MODE
80 GOSUB 300
90 POKE Y,&HAF
100 REM POKE A FINE TONE ON CHANNEL A
110 GOSUB 300
120 POKE Y,0
130 INPUT "ENTER A VALUE FOR A TONE ON CHANNEL A";T
140 GOSUB 300
150 POKE Y,T
160 REM CALL MIXER REGISTER
170 GOSUB 300
180 POKE Y,7
190 REM SET MIXER TO CHANNEL A TONES ONLY
200 GOSUB 300
210 POKE Y,62
220 REM CALL AMPLITUDE REGISTER FOR CHANNEL A
230 GOSUB 300
240 POKE Y,8
250 REM SET AMPLITUDE AT MAXIMUM-15
260 GOSUB 300
270 POKE Y,15
280 REM GO BACK TO CHANGE TONE ON CHANNEL A
290 GOT0 120
300 REM CHECK FOR BUSY*
310 IF (PEEK(Y) AND 128) = 0 THEN 310
320 RETURN
```

To stop the program, press <BREAK> and then type RUN. Line 20 resets the S/SC to silence.

The above program demonstrates how easy it is to directly manipulate the S/SC. It shows only how to manipulate one tone on one channel, but much more is possible. The “S/SC Organ” program in Appendix F, for example, turns the bottom two rows of your Color Computer keyboard into an organ keyboard. The “Sound Effects” program in Appendix G shows how to use the S/SC to create advanced sound effects by directly manipulating the S/SC registers through data statements.

With a little practice and a Color Computer Speech/Sound Cartridge, every program you run on the computer can sound like a winner.

APPENDIX A

Command Function Map

80	Load speech string into Buffers 0-7 for later execution. Terminator is 0D.
81	Load speech string into Buffers 1-7.
82	Load speech string into Buffers 2-7.
83	Load speech string into Buffers 3-7.
84	Load speech string into Buffers 4-7.
85	Load speech string into Buffers 5-7.
86	Load speech string into Buffers 6-7.
87	Load speech string into Buffer 7.
88	Load sound data into Buffers 0-7 for later execution. Terminator is FF.
89	Load sound data into Buffers 1-7.
8A	Load sound data into Buffers 2-7.
8B	Load sound data into Buffers 3-7.
8C	Load sound data into Buffers 4-7.
8D	Load sound data into Buffers 5-7.
8E	Load sound data into Buffers 6-7.
8F	Load timer base value. This command requires one postbyte of data (in the range of 0-255) to be loaded into the timer register. 255 produces the longest duration; 0 produces the shortest duration.
90	Load speech string into Buffer 0 for later execution. Terminator is 0D.
91	Load speech string into Buffer 1.
92	Load speech string into Buffer 2.
93	Load speech string into Buffer 3.
94	Load speech string into Buffer 4.
95	Load speech string into Buffer 5.
96	Load speech string into Buffer 6.
97	Load speech string into Buffer 7.
98	Load sound data into Buffer 0 for later execution. Terminator is FF.
99	Load sound data into Buffer 1.

9A	Load sound data into Buffer 2.
9B	Load sound data into Buffer 3.
9C	Load sound data into Buffer 4.
9D	Load sound data into Buffer 5.
9E	Load sound data into Buffer 6.
9F	Load sound data into Buffer 7.
A0	Load allophone address stream into Buffers 0-7 for later execution. Terminator is FF.
A1	Load allophone address stream into Buffers 1-7.
A2	Load allophone address stream into Buffers 2-7.
A3	Load allophone address stream into Buffers 3-7.
A4	Load allophone address stream into Buffers 4-7.
A5	Load allophone address stream into Buffers 5-7.
A6	Load allophone address stream into Buffers 6-7.
A7	Load allophone address stream into Buffer 7.
A8	Load register string into Buffers 0-7 for later execution. Terminator is FF.
A9	Load register string into Buffers 1-7.
AA	Load register string into Buffers 2-7.
AB	Load register string into Buffers 3-7.
AC	Load register string into Buffers 4-7.
AD	Load register string into Buffers 5-7.
AE	Load register string into Buffers 6-7.
AF	Toggles input function into a mode that allows direct access to sound registers.
B0	Load allophone address stream into Buffer 0 for later execution. Terminator is FF.
B1	Load allophone address stream into Buffer 1.
B2	Load allophone address stream into Buffer 2.
B3	Load allophone address stream into Buffer 3.
B4	Load allophone address stream into Buffer 4.
B5	Load allophone address stream into Buffer 5.
B6	Load allophone address stream into Buffer 6.
B7	Load allophone Address stream into Buffer 7.

B8	Load register string into Buffer 0 for later execution. Terminator is FF.
B9	Load register string into Buffer 1.
BA	Load register string into Buffer 2.
BB	Load register string into Buffer 3.
BC	Load register string into Buffer 4.
BD	Load register string into Buffer 5.
BE	Load register string into Buffer 6.
BF	Load register string into Buffer 7.
C0	Execute speech string from Buffers 0-7.
C1	Execute speech string from Buffers 1-7.
C2	Execute speech string from Buffers 2-7.
C3	Execute speech string from Buffers 3-7.
C4	Execute speech string from Buffers 4-7.
C5	Execute speech string from Buffers 5-7.
C6	Execute speech string from Buffers 6-7.
C7	Stop all speech.
C8	Execute sound data from Buffers 0-7.
C9	Execute sound data from Buffers 1-7.
CA	Execute sound data from Buffers 2-7.
CB	Execute sound data from Buffers 3-7.
CC	Execute sound data from Buffers 4-7.
CD	Execute sound data from Buffers 5-7.
CE	Execute sound data from Buffers 6-7.
CF	Stop all sound.
D0	Execute speech string from Buffer 0.
D1	Execute speech string from Buffer 1.
D2	Execute speech string from Buffer 2.
D3	Execute speech string from Buffer 3.
D4	Execute speech string from Buffer 4.
D5	Execute speech string from Buffer 5.
D6	Execute speech string from Buffer 6.
D7	Execute speech string from Buffer 7.
D8	Execute sound data from Buffer 0.

D9	Execute sound data from Buffer 1.
DA	Execute sound data from Buffer 2.
DB	Execute sound data from Buffer 3.
DC	Execute sound data from Buffer 4.
DD	Execute sound data from Buffer 5.
DE	Execute sound data from Buffer 6.
DF	Execute sound data from Buffer 7.
E0	Execute allophone address stream from Buffers 0-7.
E1	Execute allophone address stream from Buffers 1-7.
E2	Execute allophone address stream from Buffers 2-7.
E3	Execute allophone address stream from Buffers 3-7.
E4	Execute allophone address stream from Buffers 4-7.
E5	Execute allophone address stream from Buffers 5-7.
E6	Execute allophone address stream from Buffers 6-7.
E7	Execute allophone address stream from Buffer 7.
E8	Execute register string from Buffers 0-7.
E9	Execute register string from Buffers 1-7.
EA	Execute register string from Buffers 2-7.
EB	Execute register string from Buffers 3-7.
EC	Execute register string from Buffers 4-7.
ED	Execute register string from Buffers 5-7.
EE	Execute register string from Buffers 6-7.
EF	Execute register string from Buffer 7.
F0	Execute allophone address stream from Buffer 0.
F1	Execute allophone address stream from Buffer 1.
F2	Execute allophone address stream from Buffer 2.
F3	Execute allophone address stream from Buffer 3.
F4	Execute allophone address stream from Buffer 4.
F5	Execute allophone address stream from Buffer 5.
F6	Execute allophone address stream from Buffer 6.
F7	Execute allophone address stream from Buffer 7.
F8	Execute register string from Buffer 0.
F9	Execute register string from Buffer 1.
FA	Execute register string from Buffer 2.

FB	Execute register string from Buffer 3.
FC	Execute register string from Buffer 4.
FD	Execute register string from Buffer 5.
FE	Execute register string from Buffer 6.
FF	Execute register string from Buffer 7.
00	Stops all sounds, including speech. Does not clear buffers.

APPENDIX B

Sound Generator Registers

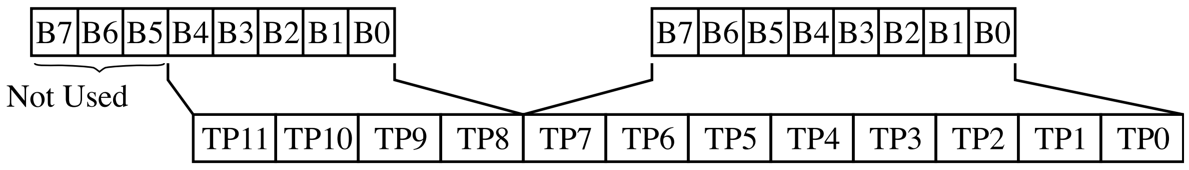
Since all functions of the PSG (Programmable Sound Generator) are controlled by the processor via a series of register loads, a detailed description of the PSG operation can best be accomplished by relating each PSG function to the control of its corresponding register. The function of creating or programming a specific sound or sound effect logically follows the control sequence listed:

Operation	**Registers**	**Function**
Tone Generator Control	R0—R5	Program tone periods.
Noise Generator Control	R6	Program noise period.
Mixer Control	R7	Enable tone and/ or noise on selected channels.
Amplitude Control	R8—R10	Select "fixed" or "envelope-variable" amplitudes
Envelope Generator	R11—R13	Program envelope Control period and select envelope pattern.

Tone Generator Control (Registers R0, Rl, R2, R3, R4, R5)

The frequency of each square wave generated by the three Tone Generators (one each for Channels A, B, and C) is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 12-bit Tone Period value. Each 12-bit value is obtained in the PSG by combining the contents of the relative Coarse and Fine Tune registers, as illustrated in the following:

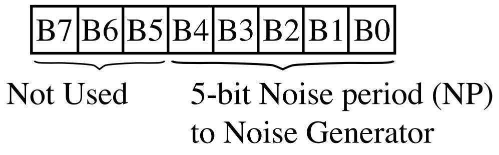
Coarse Tune Register	**Channel**	**Fine Tune Register**
R1	A	R0
R3	B	R2
R5	C	R4



12-bit Tone Period (TP) to Tone Generator

Noise Generator Control (Register R6)

The frequency of the noise source is obtained in the PSG by first counting down the input clock by 16 and then by further counting down the result by the programmed 5-bit Noise Period value. This 5-bit value consists of the lower 5 bits (B4—B0) of register R6, as illustrated in the following:

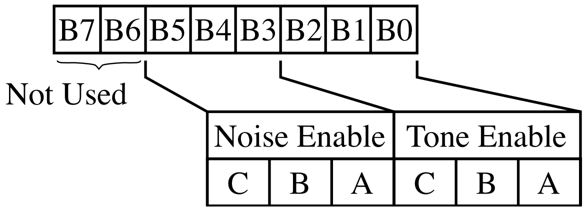


Mixer Control (Register R7)

Register R7 is a multifunction Enable register that controls the three Noise/Tone Mixers.

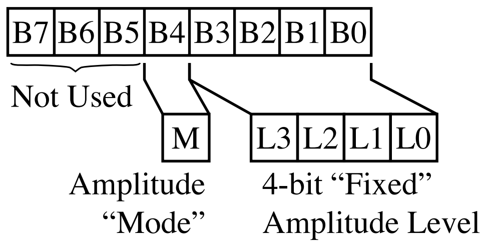
The Mixers, as previously described, combine the noise and tone frequencies for each of the three channels. The determination of combining neither/either/both noise and tone frequencies on each channel is made by the state of bits B5—B0 of R7. These bits are active low.

These functions are illustrated in the following:



Amplitude Control (Registers R8, R9, R10)

The amplitudes of the signals generated by each of the three D/A Converters (one each for Channels A, B, and C) is determined by the contents of the lower 5 bits (B4—B0) of registers R8, R9, and Rlg as illustrated in the following:



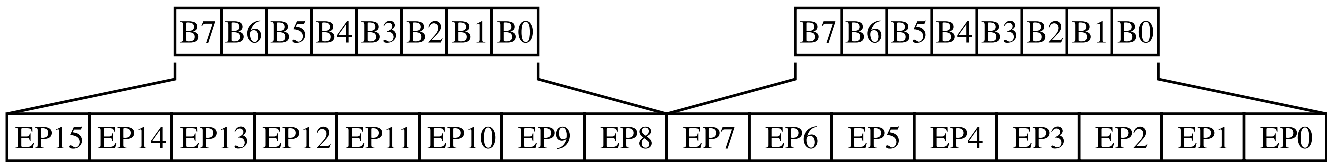
If Bit 4 is high, the envelope controls the amplitude; if Bit 4 is low, Bits 3-0 fix the amplitude.

Envelope Generator Control (Registers Rll, Rl2, Rl3)

To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG. First, it is possible to vary the frequency of the envelope using registers Rll and R12. Second, the relative shape and cycle pattern of the envelope can be varied using register R13. The following paragraphs explain the details of the envelope control functions, describing first the envelope period control and then the envelope shape/cycle control.

Envelope Period Control (Registers Rll, Rl2)

The frequency of the envelope is obtained in the PSG by first counting down the input clock by 256 and then by further counting down the result by the programmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated in the following:

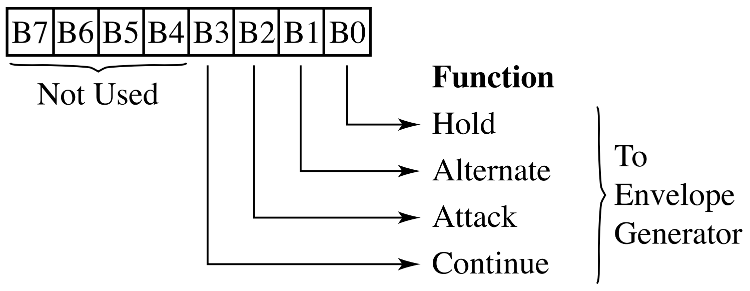

B7 B6 B5 B4 B3 B2 B1 B0
B7 B6 B5 B4 B3 B2 B1 B0
EP15 EP14 EP13 EP12 EP11 EP10 EP9 EP8 EP7 EP6 EP5 EP4 EP3 EP2 EP1 EP0

16-bit Envelope Perido (EP) to Envelope Generator

Envelope Shape/Cycle Control (Register Rl3)

The Envelope Generator further counts down the envelope frequency by 16, producing a 16-state per cycle envelope pattern as defined by its 4-bit counter output, E3 E2 El E0, The particular shape and cycle pattern of any desired envelope is accomplished by controlling the count pattern (count up/count down) of the 4-bit counter and by defining a single-cycle or repeat-cycle pattern.

This envelope shape/cycle control is contained in the lower 4 bits (B3—B0) of register R13. Each of these 4 bits controls a function in the envelope generator as illustrated in the following:



> **Note:** In determining the period of events, consider a count of zero as one more than if all bits are set. For example, in a tone, if 12 bits are set, the divisor is 4095. If all 12 bits are clear, however, the divisor is not 0 because division by zero is <u>not</u> permitted. Therefore, the divisor becomes 4096.

APPENDIX C

Allophone Address Table

Address Dec	Hex	Allophone	Sample Word	Duration
000	000	PA1	PAUSE	10ms
001	001	PA2	PAUSE	30ms
002	002	PA3	PAUSE	50ms
003	003	PA4	PAUSE	100ms
004	004	PA5	PAUSE	100ms
005	005	/OY/	Boy	42ms
006	006	/AY/	Sky	26ms
007	007	/EH/	End	70ms
008	008	/KK3/	Comb	120ms
009	009	/PP/	Pow	210ms
010	00A	/JH/	Dodge	140ms
011	00B	/NN1/	Thin	140ms
012	00C	/IH/	Sit	70ms
013	00D	/TT2/	To	140ms
014	00E	/RR1/	Rural	170ms
015	00F	/AX/	Succeed	180ms
016	010	/MM/	Milk	100ms
017	011	/TT1/	Part	100ms
018	012	/DH1/	They	290ms
019	013	/IY/	See	250ms
020	014	/EY/	Beige	280ms
021	015	/DD1/	Could	70ms
022	016	/UW1/	To	100ms
023	017	/AO/	Aught	100ms
024	018	/AA/	Hot	100ms
025	019	/YY2/	Yes	180ms
026	01A	/AE/	Hat	120ms
027	01B	/HH1/	He	130ms
028	01C	/BB1/	Business	80ms
029	01D	/TH/	Thin	180ms
030	01E	/UH/	Book	100ms

031	01F	/UW2/	Food	260ms
032	020	/AW/	Out	370ms
033	021	/D2/	Do	160ms
034	022	/GG3/	Wig	140ms
035	023	/VV/	Vest	190ms
036	024	/GG1/	Got	80ms
037	025	/SH/	Ship	160ms
038	026	/ZH/	Azure	190ms
039	027	/RR2/	Brain	120ms
040	028	/FF/	Food	150ms
041	029	/KK2/	Sky	190ms
042	02A	/KK1/	Can’t	160ms
043	02B	/ZZ/	Zoo	210ms
044	02C	/NG/	Anchor	220ms
045	02D	/LL/	Lake	110ms
046	02E	/WW/	Wool	180ms
047	02F	/XR/	Repair	360ms
048	030	/WH/	Whig	200ms
049	031	/YY1/	Yes	130ms
050	032	/CH/	Church	190ms
051	033	/ER1/	Fir	160ms
052	034	/ER2/	Fir	300ms
053	035	/OW/	Beau	240ms
054	036	/DH2/	They	240ms
055	037	/SS/	Vest	90ms
056	038	/NN2/	No	190ms
057	039	/HH2/	Hoe	180ms
058	03A	/OR/	Store	330ms
059	03B	/AR/	Alarm	290ms
060	03C	/YR/	Clear	350ms
061	03D	/GG2/	Guest	40ms
062	03E	/EL/	Saddle	190ms
063	03F	/BB2/	Business	50ms

APPENDIX D

Equal-Tempered Chromatic Scale

(f clock = 1.78977MHz)

Note	Octave	HEX		DEC	
		Coarse	Fine	Coarse	Fine
C	1	D	5D	13	93
C#	1	C	9C	12	156
D	1	B	E7	6	231
D#	1	B	3C	6	60
E	1	A	9B	10	155
F	1	A	02	10	2
F#	1	9	73	9	115
G	1	8	EB	8	235
G#	1	8	6B	8	107
A	1	7	F2	7	242
A#	1	7	80	7	128
B	1	7	14	7	20
C	2	6	AE	6	174
C#	2	6	4E	6	78
D	2	5	F4	5	244
D#	2	5	9E	5	158
E	2	5	4D	5	77
F	2	5	1	5	1
F#	2	4	B9	4	185
G	2	4	75	4	117
G#	2	4	35	4	53
A	2	3	F9	3	249
A#	2	3	c0	3	192
B	2	3	8A	3	138
C	3	3	57	3	87
C#	3	3	27	3	39
D	3	2	FA	2	250

D#	3	2	CF	2	207
E	3	2	A7	2	167
F	3	2	81	2	129
F#	3	2	5D	2	93
G	3	2	3B	2	59
G#	3	2	1B	2	27
A	3	1	FC	1	252
A#	3	1	E0	1	224
B	3	1	C5	1	197
C	4	1	AC	1	172
C#	4	1	94	1	148
D	4	1	7D	1	125
D#	4	1	68	1	104
E	4	1	53	1	83
F	4	1	40	1	64
F#	4	1	2E	1	46
G	4	1	ID	1	29
G#	4	1	D	1	13
A	4	0	FE	0	254
A#	4	0	F0	0	240
B	4	0	E2	0	266
C	5	0	D6	0	214
C#	5	0	CA	0	202
D	5	0	BE	0	190
D#	5	0	B4	0	180
E	5	0	AA	0	170
F	5	0	A0	0	160
F#	5	0	97	0	151
G	5	0	8F	0	143
G#	5	0	87	0	135
A	5	0	7F	0	127
A#	5	0	78	0	120
B	5	0	71	0	113
C	6	0	6B	0	107

C#	6	0	65	0	101
D	6	0	5F	0	95
D#	6	0	5A	0	90
E	6	0	55	0	85
F	6	0	50	0	80
F#	6	0	4C	0	76
G	6	0	47	0	71
G#	6	0	43	0	67
A	6	0	40	0	64
A#	6	0	3C	0	60
B	6	0	39	0	57
C	7	0	35	0	53
C#	7	0	32	0	50
D	7	0	30	0	48
D#	7	0	2D	0	45
E	7	0	2A	0	42
F	7	0	28	0	40
F#	7	0	26	0	38
G	7	0	24	0	36
G#	7	0	22	0	34
A	7	0	20	0	32
B	7	0	1C	0	28
C	8	0	1B	0	27
C#	8	0	19	0	25
D	8	0	18	0	24
D#	8	0	16	0	22
E	8	0	15	0	21
F	8	0	14	0	20
F#	8	0	13	0	19
G	8	0	12	0	18
G#	8	0	11	0	17
A	8	0	10	0	16
A#	8	0	F	0	15
B	8	0	E	0	14

APPENDIX E

S/SC Demonstration Program

This program demonstrates many of the capabilities of the S/SC—simultaneous speech and sound, music generation, complex sound effects, use of commands and sound registers, and so on.

```
10 REM ************************
20 REM ****  DEMO PROGRAM  ****
30 REM ** FOR SPEECH/SOUND ****
40 REM **     CARTRIDGE     ***
50 REM ************************
60 REM ************************
70 CLEAR 512
80 X=&HFF00:Y=&HFF7E:Z=&HFF7D
90 DIM P$(8)
100 REM
110 REM SETUP
120 REM
130 POKE Z,1:POKEZ,0
140 POKE X+1,52:POKE X+3,63
150 POKE X+35,60
160 REM
170 REM
180 REM DEFINE SPEECH STRINGS
190 REM
200 GOSUB 2890
210 REM
220 REM
230 REM LOAD SPEECH STRINGS
240 REM
250 PRINT "LOADING SPEECH"
260 FOR C=1TO8
270 GOSUB 3360
280 NEXT C
290 REM
300 REM
310 REM LOAD SOUNDS
320 REM
330 PRINT "LOADING SOUND"
340 FOR C=1 TO 6
350 GOSUB 3490
360 NEXT C
```

```
370 REM
390 REM EXECUTE DEMO
400 REM
410 PRINT "DEMONSTRATING"
420 REM
430 REM LOAD BASE TIMER
440 REM
450 GOSUB 3170
460 POKE Y,&H8F
470 GOSUB 3170
480 POKE Y,60
490 GOSUB 3170
500 REM
510 REM
520 REM EXEC SPEECH BUFFER 0
530 REM
540 POKE Y,208
550 GOSUB 3230
560 GOSUB 3030
570 GOSUB 3170
580 REM
590 REM
600 REM EXEC SPEECH BUFFER 1
610 REM
620 POKE Y,209
630 GOSUB 3030
640 GOSUB 3170
650 REM
660 REM
670 REM EXEC SPEECH BUFFER 4
680 REM
690 POKE Y,212
700 GOSUB 3300
710 GOSUB 3030
720 GOSUB 3170
730 REM
740 REM
750 REM EXEC SOUND BUFFER 0
760 REM
770 POKE Y,216
780 GOSUB 3230
790 GOSUB 3100
800 GOSUB 3170
810 REM
820 REM
```

```
830 REM EXEC SPEECH BUFFER 5
840 REM
850 POKE Y,213
860 GOSUB 3230
870 GOSUB 3030
880 GOSUB 3170
890 REM
900 REM
910 REM EXEC SOUND BUFFER 1
920 REM
930 POKE Y,217
940 GOSUB 3230
950 GOSUB 3100
960 GOSUB 3170
970 REM
980 REM
990 REM EXEC SPEECH BUFFER 2
1000 REM
1010 POKE Y,210
1020 GOSUB 3030
1030 GOSUB 3170
1040 REM
1050 REM
1060 REM EXEC SPEECH BUFFER 4
1070 REM
1080 POKE Y, 212
1090 GOSUB 3230
1100 GOSUB 3030
1110 GOSUB 3170
1120 REM
1130 REM
1140 REM EXEC REG STR BUFFER 3
1150 REM
1160 POKE Y,251
1170 GOSUB 3230
1180 GOSUB 3100
1190 GOSUB 3170
1200 REM
1210 REM
1220 REM EXEC REG STR BUFFER 4
1240 POKE Y, 252
1250 GOSUB 3230
1260 GOSUB 3100
1270 GOSUB 3170
1280 REM
```

```
1290 REM
1300 REM EXEC REG STR BUFFER 3
1310 REM
1320 POKE Y, 251
1330 GOSUB 3230
1340 GOSUB 3100
1350 GOSUB 3170
1360 REM
1370 REM
1380 REM EXEC REG STR BUFFER 5
1390 REM
1400 POKE Y,253
1410 GOSUB 3230
1420 GOSUB 3100
1430 GOSUB 3170
1440 REM
1450 REM
1460 REM EXEC REG STR BUFFER 3
1470 REM
1480 POKE Y,251
1490 GOSUB 3230
1500 GOSUB 3100
1510 GOSUB 3170
1520 REM
1530 REM
1540 REM STOP ALL SOUND
1550 REM
1560 POKE Y,207
1570 GOSUB 3170
1580 REM
1590 REM
1600 REM EXEC SPEECH BUFFER 3
1610 REM
1620 POKE Y,211
1630 GOSUB 3230
1640 GOSUB 3230
1650 GOSUB 3230
1660 GOSUB 3030
1670 GOSUB 3170
1680 REM
1690 REM
1700 REM EXEC SOUND BUFFER 2
1710 REM
1720 POKE Y, 218
1730 GOSUB 3100
```

```
1740 GOSUB 3170
1750 REM
1760 REM
1770 REM EXEC SPEECH BUFFER 7
1780 REM
1790 POKE Y,215
1800 GOSUB 3170
1810 REM
1820 REM
1830 REM EXEC REG STR BUFFER
1840 REM
1850 POKE Y,251
1860 GOSUB 3100
1870 GOSUB 3170
1880 REM
1890 REM
1900 REM EXEC REG STR BUFFER 4
1910 REM
1920 POKE Y,252
1930 GOSUB 3100
1940 GOSUB 3170
1950 REM
1960 REM
1970 REM EXEC REG STR BUFFER 3
1980 REM
1990 POKE Y,251
2000 GOSUB 3100
2010 GOSUB 3170
2020 REM
2030 REM
2040 REM EXEC REG STR BUFFER 5
2050 REM
2060 POKE Y,253
2070 GOSUB 3100
2080 GOSUB 3170
2090 REM
2100 REM
2110 REM EXEC REG STR BUFFER 3
2120 REM
2130 POKE Y,251
2140 GOSUB 3100
2150 GOSUB 3170
2160 REM
2170 REM
2180 REM STOP ALL SOUND
```

```
2190 REM
2200 POKE Y,207
2210 GOSUB 3170
2220 REM
2230 REM
2240 REM EXEC SPEECH BUFFER 6
2250 REM
2260 POKE Y,214
2270 REM
2280 REM
2290 END
2300 REM
2310 REM TUNE 1
2320 REM COMMAND MODE SOUND
2340 DATA 11,0,214,6,0,0,0,6
2350 DATA 11,1,29,4,11,1,46,4
2360 DATA 11,1,29,4,11,1,13,12
2370 DATA 11,1,29,6,0,0,0,18
2380 DATA 11,0,226,6,0,0,0,6
2390 DATA 11,0,214,6,0,0,0,0
2400 REM
2410 REM
2420 REM TUNE 2
2430 REM COMMAND MODE SOUND
2440 REM
2450 DATA 11,0,214,6,0,0,0,0
2460 DATA 11,0,214,12,0,0,0,0
2470 DATA 11,0,214,12,0,0,0,0
2480 DATA 11,1,13,6,0,0,0,0
2490 DATA 11,0,254,12,0,0,0,0
2500 DATA 11,1,64,6,0,0,0,0
2510 REM
2520 REM
2530 REM GUNSHOT
2540 REM COMMAND MODE NOISE
2550 REM WITH ENVELOPE
2560 REM
2570 DATA 159,15,15,191,15,15
2580 DATA 223,15,15,98,0,14,5
2590 DATA 0,0,0,0
2600 REM
2610 REM
2620 REM C CHORD
2630 REM DIRECT REGISTER SOUND
2640 REM
```

```
2650 DATA 0,172,1,1,2,83,3,1
2660 DATA 4,29,5,1,6,0,7,56
2670 DATA 8,9,9,9,10,9,11,0
2680 DATA 12,0,13,0
2690 REM
2700 REM
2710 REM F CHORD
2720 REM DIRECT REGISTER SOUND
2740 DATA 0,172,1,1,2,64,3,1
2750 DATA 4,254,5,0,6,0,7,56
2760 DATA 8,9,9,9,10,9,11,0
2770 DATA 12,0,13,0
2780 REM
2790 REM
2800 REM G CHORD
2810 REM DIRECT REGISTER SOUND
2820 REM
2830 DATA 0,197,1,1,2,125,3,1
2840 DATA 4,29,5,1,6,0,7,56
2850 DATA 8,9,9,9,10,9,11,0
2860 DATA 12,0,13,0
2870 REM
2880 REM
2890 REM STRINGS
2900 REM TEXT-TO-SPEECH
2910 REM
2920 P$(1)="HELLO . . . I AM THE COLOR COMPUTER SOUND AND
SPEECH CARTRIDGE."
2930 P$(2)="I CAN NOT ONLY TALK. . . I CAN ALSO PLAY
MELODIES."
2940 P$(3)="I CAN PLAY CORDS."
2950 P$(4)="I CAN MAKE SOUND EFFECTS. . . LIKE THIS GUN-
SHOT."
2960 P$(5)="LIKE THIS."
2970 P$(6)="AND THIS."
2980 P$(7)="I HOPE YOU WILL ENJOY USING ME."
2990 P$(8)="I CAN PLAY CORDS WHILE SPEAKEYNG."
3000 RETURN
3010 REM
3020 REM
3030 REM SPEECH* CHECK
3040 REM
3050 FORQ=1TO800:NEXT Q
3060 IF (PEEK(Y) AND 64) = 0 THEN 3060
3070 RETURN
```

```
3080 REM
3090 REM
3100 REM SOUND* CHECK
3110 REM
3120 FORQ=1TO700:NEXT Q
3130 IF (PEEK(Y) AND 32) = 0 THEN 3130
3140 RETURN
3150 REM
3160 REM
3170 REM BUSY* CHECK
3180 REM
3190 IF (PEEK(Y) AND 128) = 0 THEN 3190
3200 RETURN
3210 REM
3220 REM
3230 REM SHORT DELAY
3240 REM
3250 GOSUB 3170
3260 FOR Q = 1 TO 800:NEXT Q
3270 RETURN
3280 REM
3290 REM
3300 REM LONG DELAY
3310 REM
3320 FOR Q=1 TO 6000:NEXT Q
3330 RETURN
3340 REM
3350 REM
3360 REM LOAD A STRING
3370 REM
3380 GOSUB 3170
3390 POKE Y,143+C
3400 FOR I=1 TO LEN(P$(C))
3410 GOSUB 3170
3420 POKE Y, ASC(MID$(P$(C),I,1))
3430 NEXT I
3440 GOSUB 3170
3450 POKE Y,13
3460 RETURN
3470 REM
3480 REM
3490 REM LOAD A SOUND
3500 REM
3510 IF C>2 THEN 3560
3520 K=48
```

```
3530 GOSUB 3170
3540 POKE Y,151+C
3550 GOTO 3600
3560 IF C=3 THEN K=17:GOTO 3530
3570 K=28
3580 GOSUB 3170
3590 POKE Y, 183+C
3600 FOR I=1 TO K
3610 READ D
3620 GOSUB 3170
3630 POKE Y,D
3640 NEXT I
3650 GOSUB 3170
3660 POKE Y,255
3670 RETURN
3680 REM
3690 REM
```

APPENDIX F

S/SC Organ

This program makes an organ keyboard out of the bottom two rows of your computer keyboard. The organ can play 16 notes, starting with the A above middle C. The following shows how the letters of the keyboard correspond to the white and black keys of the organ keyboard:

Z = A
S = A sharp/B flat
X = B
c = c
F = C sharp/D flat
v = D
G = D sharp/E flat
B = E
N = F
J = F sharp/G flat
M = G
K = G sharp/A flat
, = A
L = A sharp/B flat
. = B
/ = C

The notes sound as long as you hold down a key. To play the first strains of the "Star-Spangled Banner," type: BFZFB,

```
5 REM RESET SOUND BOARD
10 CLS:W=&HFF7D:X=&HFF00:Y=&HFF7E
20 POKE X+1,52: POKE X+3,63: POKE X+35,60
30 POKE W,1: POKE W,0
35 REM TONE VALUE ARRAY
40 DIM T(20)
50 REM SET SOUND TO IMMEDIATE MODE
60 POKE Y,175:GOSUB 280
70 REM SETUP OTHER PARAMETERS
80 REM ZERO OUT COARSE TONE REGISTER
```

```
90 REM ON TONE CHANNEL A
100 POKE Y,1: GOSUB280: POKE Y,0: GOSUB280
110 FOR A = 1 TO 8: READ B: POKE Y,B
120 GOSUB 280: NEXT A
130 REM READ IN TONE VALUES
140 FOR A = 2 TO 17: READ T(A):NEXT A
150 IN$=INKEY$
160 REM CHECK FOR A VALID KEY
170 A=INSTR$(" ZSXCFVGBNJMK,L./",IN$)
175 REM NOT A VALID KEY... TURN OFF SOUND
180 IF A<2 THEN 260
185 REM GOING TO PUT SOUND IN REGISTER 0
190 POKE Y,0: GOSUB 280
200 POKE Y, T(A): GOSUB280
210 REM TURN ON CHANNEL A SOUND
220 POKE Y,8: GOSUB 280: POKE Y,16
240 FOR B=338 TO 345:POKE B, 255: NEXT B: GOTO 150
250 REM TURN OFF SOUND ON CHANEL A
260 POKE Y,8:GOSUB 280:POKE Y,0: GOTO 150
270 REM WAIT IF BOARD ISNT READY
280 IF (PEEK(Y) AND 128)=0 THEN 280 ELSE RETURN
290 REM SELECT IMMEDIATE MODE
300 REM TURN ON CHANNEL A TONE
310 DATA 7,254
320 REM SET FINE & COURSE REPEAT PERIOD
330 REM CHANGING THE LINE TO 11, 100, 12, 1
340 REM WILL INCREASE TREMOLO SPEED
350 REM 11,50,12,3 WILL DECREASE IT
360 DATA 11,0,12,2
370 REM SET TYPE OF SOUND (13,...)
380 REM 8=ATTACK, 10=TREMOLO, 11=CONSTANT
390 DATA 13,10
400 REM VALUES FOR TONES
410 DATA 254,240,226,214,202,190
420 DATA 180,170,160,151,143,135
430 DATA 127,120,113,107
```

APPENDIX G

Sound Effects

This program demonstrates how easily the S/SC produces 7 popular game sound effects. Select the number of a sound effect from the menu screen~the Color Computer and the S/SC do the rest. Modify the program's data lines to produce your own sound effects.

```
10 GOTO 40
20 IF (PEEK(Y) AND 128)=0 THEN 20
30 RETURN
40 CLS: W=&HFF7D: X=&HFF00
50 Y=&HFF7E: POKE X+1,52
60 POKE X+3,63:POKE X+35,60
70 POKE W,1: POKE W,0
80 DIM A(8,13):FOR A=1 TO 7
90 FOR B = 0 TO 13:READ A(A,B):NEXT B,A
100 CLS:PRINT TAB(9) "SOUND EFFECTS"
110 PRINT:PRINT TAB(7)"<1> WHISTLE"
120 PRINT TAB(7)"<2> RACECAR"
130 PRINT TAB(7)"<3> LASER"
140 PRINT TAB(7)"<4> WHISTLING BOMB"
150 PRINT TAB(7)"<5> BOMB WITH EXPLOSIONS"
160 PRINT TAB(7)"<6> EXPLOSION"
170 PRINT TAB(7)"<7> GUNSHOT"
180 PRINT:PRINT TAB(7);
190 INPUT "YOUR SELECTION";S
200 IF S <1 OR S>7 THEN 100
210 POKE Y,175:GOSUB20
220 FOR A=0 TO 13:POKE Y,A:GOSUB 20
230 POKE Y,A(S,A):GOSUB 20:NEXT A
240 ON S GOSUB 270,360,480,530,530,570,570
250 IF S=5 THEN S=6:GOTO 220
260 POKE W,1:POKE W,0:GOTO 100
270 REM WHISTLE
280 FOR A=64 TO 32 STEP -2:POKE Y,0
290 GOSUB 20:POKE Y,A:GOSUB 20
300 NEXT A:FOR A=1 TO 200:NEXT A
310 FOR A=64 TO 48 STEP -2:POKE Y,0
320 GOSUB 20:POKE Y,A:GOSUB 20
330 NEXT A:FOR A=48 TO 96 STEP 2
340 POKE Y,0:GOSUB 20
350 POKE Y,A: GOSUB 20: NEXT A: RETURN
```

```
360 REM RACE CAR
370 FOR A = 11 TO 4 STEP -1:POKE Y,1:GOSUB 20
380 POKE Y,A: GOSUB 20:GOSUB 450
390 NEXT A:FOR A = 9 TO 3 STEP -1
400 POKE Y,1:GOSUB 20:POKE Y,A
410 GOSUB 20:GOSUB 450:NEXT A
420 FOR A=6 TO 1 STEP -1:POKE Y,1
430 GOSUB 20:POKE Y,A:GOSUB20
440 GOSUB450:NEXT A:RETURN
450 FOR B=255 TO 0 STEP -8:POKE Y,0
460 GOSUB 20:POKE Y,B:GOSUB 20
470 NEXT B:RETURN
480 REM LASER
490 FOR A=0 TO 10
500 FOR B=50 TO 100 STEP 10: POKE Y,0
510 GOSUB 20: POKE Y,B: GOSUB 20:NEXT B
520 NEXT A :RETURN
530 REM WHISTLING BOMB
540 FOR B=30 TO 200: POKE Y,0
550 GOSUB 20:POKE Y,B:GOSUB 20
560 NEXT B: RETURN
570 REM EXPLOSION OR GUNSHOT
580 POKE Y,13: GOSUB 20: POKE Y,0
590 FOR A = 1 TO 1500+1000*(S=7)
600 NEXT A: RETURN
610 REM WHISTLE
620 DATA 0,0,0,0,0,0,1,46,15,9,0,0,0,0
630 REM RACE CAR
640 DATA 0,0,0,15,0,0,0,60,15,10,0,0,0,0
650 REM LASER
660 DATA 0,0,0,0,0,0,0,254,15,0,0,0,0,0
670 REM WHISTLING BOMB
680 DATA 0,0,0,0,0,0,0,254,15,0,0,0,0,0
690 REM BOMB WITH EXPLOSION
700 DATA 0,0,0,0,0,0,0,254,15,0,0,0,0,0
710 REM EXPLOSION
720 DATA 0,0,0,0,0,0,0,7,16,16,16,0,56,0
730 REM GUNSHOT
740 DATA 0,0,0,0,0,0,15,7,16,16,16,0,16,0
```

APPENDIX H

Speech to Text (with screen editor)

Like the program on Page 7, this program converts text into speech. Included in this program, however, is a built-in screen editor that lets you type in a line of a maximum of 252 characters, and then alter it by typing only corrections—not the entire line.

After you type a line, you can use the following commands to move the cursor and insert and delete characters:

<left arrow>	Moves cursor left (does not erase the character)
<right arrow>	Moves cursor right
<SHIFT><left arrow>	Deletes character under the cursor
<SHIFT><right arrow>	Inserts a space at the cursor position
<up arrow>	Moves cursor up 1 line (or to the beginning if used in the top line)
<down arrow>	Moves cursor down 1 line (or to the end if used in the bottom line)
<SHIFT><up arrow>	Moves cursor to the first character of the line
<SHIFT><down arrow>	Moves cursor to the last character of the line
<CLEAR>	erases all characters from the cursor to the end of the line

(After you type a line and press <ENTER> to convert it to speech, the cursor always returns to the first character of the line.)

```
10 CLS: CLEAR 2000
20 X=&HFF00; Y=&HFF7E: POKE X+1,52
30 POKE X+3,63: POKE X+35,60
40 POKE &HFF7D,1: POKE &HFF7D,0
50 CF$="": FOR X=1 TO 10: READ A
60 CF$=CF$ + CHR$(A): NEXT: B=0
70 GOSUB 120: Z$=IN$ + " " + CHR$(13)
80 FOR Z=1 TO LEN(Z$)
90 IF PEEK (Y) AND 128=0 THEN 90
100 POKE Y,ASC(MID$(Z$,Z,1))
110 NEXT Z: B=0: GOT0 70
120 B$=STRING$(254,32)
```

```
130 MID$(B$,1,253)=IN$ + " "
140 C=LEN(IN$)
150 PRINT @ 0,B$
160 PRINT @ B,CHR$(142);
170 A$=INKEY$: IF A$="" THEN 170
180 IF C<B THEN C=B
190 PRINT @ B,MID$(B$,B+1,1);
200 CF=INSTR(CF$,A$): IF CF THEN 250
210 D=ASC(A$): IF D<32 OR D>96 THEN 160
220 IF B<252 THEN B=B+1 ELSE 160
230 PRINT @ B-1,A$;
240 MID$(B$,B,1)=A$: GOTO 160
250 ON CF GOTO 310,330,350,370
260 ON CF-4 GOTO 390,420,460,470
270 IF CF=9 THEN 290
280 IN$=MID$(B$,1,B): C=B: GOTO 120
290 IN$=MID$(B$,1,C)
300 IF C>0 THEN RETURN
310 IF B>0 THEN B=B-1
320 GOTO 160
330 IF B<C THEN B=B+1
340 GOTO 160
350 IF B>31 THEN B=B-32 ELSE B=0
360 GOTO 160
370 IF B<(C-32) THEN B=B+32 ELSE B=C
380 GOTO 160
390 MID$(B$,B+1) = MID$(B$,B+2)
400 MID$(B$B+1,1) = ""
410 C=C-1-(C<1): GOTO 150
420 MID$(B$,B+2)=MID$(B$,B+2)
430 MID$(B$,B+1,1) = " "
440 MID$(B$,253,1) = " "
450 C=C+1+(C>251): GOTO 150
460 B=0: GOTO 150
470 B=C: GOTO 150
480 PRINT @ 0,B$: GOTO 160
490 DATA 8,9,94,10,21,93,95,91,13,12
```