mockingboard

mockingboard
mockingboard
mockingboard

user's manual

sweet micro systems
150 chestnut street
providence, rhode island
02903

mockingboard
mockingboard
mockingboard™

mockingboard™

user's manual

sound I
sound II
speech I
sound/speech I

sweet micro systems
150 chestnut street, providence, rhode island 02903

150 Chestnut Street
Providence, RI 02903 (401) 273-5333

The information contained in this manual is correct as far as we can determine; however, Sweet Micro Systems reserves the right to make improvements in the product and/or manual at any time without notice.

Package Design by Teresa Level, Providence, RI.

TABLE OF CONTENTS

INTRODUCTION

Mockingboard peripherals are a series of Apple II compatible boards devoted to sound effects generation and speech synthesis. The concept behind the Mockingboard series is to provide you, the user, with a sophisticated new tool that can be utilized to enhance your programs without complex programming techniques. We want you to enjoy the experience and be able to utilize this tool in all your programs. Therefore, much of the complexity of generating the various sound effects and speech have been assumed by the hardware.

Mockingboard series currently features the Sound I, a sound effects generator; Sound II, a dual sound effects generator; Speech I, a speech synthesizer; and Sound/Speech I, a combination sound effects generator and speech synthesizer. Sound I board can produce a wide variety of sounds, ranging from a gunshot to musical notes. The Sound II board was designed to produce two independent sounds simultaneously. The board can produce a continuous sound, such as a train, from one speaker and at the same time, produce a gunshot from the other speaker. Speech I board produces synthesized speech by sequencing phonemes (basic sound units of speech) to form words. Finally, all of the above features were combined onto one board to produce Sound/Speech I. This board can effectively produce a variety of sound effects and continuous speech, while drastically reducing the complexity of the software normally required.

Each section in this manual will be devoted primarily to programming information for each board. Some program listings have been included to illustrate the ease of programming. The demonstration disk, included with your board, is not copy protected so that you may view all of the program listings. This will enable you to see and hear the assortment of sounds produced from variations of the same program. One added feature is that each sample program is portable so that you can utilize the individual modules in your programs.

The boards for the Mockingboard series were very carefully designed to ensure versatility and ease of programming. Therefore, the selection of the major components played a

significant role in their development. A brief description of these components and their contributing features is given below.

SYNERTEK SY6522 VERSATILE INTERFACE ADAPTER*

Mockingboard peripherals are interfaced to the Apple II by the Synertek SY6522 Versatile Interface Adapter, which will be referred to as the "6522." The 6522 was selected as an I/O (input/output) control device because of its flexibility including features which provide the user with greater programming capabilities.

Each function (sound effect and speech) is controlled by its own separate interface chip. This combines the versatility of independent control with the power of merging sound effects or speech and sound. In fact, the capability of producing simultaneous sounds can only be achieved through this added feature.

This chip interface also has features which enhance programming capabilities. The 6522 is a register oriented device. Any one of sixteen internal registers can be referenced directly through the Apple, providing the user with full control over I/O functions. One of the more powerful features is the two programmable on-chip timers which can be utilized in your programs to achieve real-time interrupts without tying up the processor in long loops. With the Sound II and Sound/Speech I boards, you can use one 6522 as a timer and the other to produce continuous sound or speech while your program continues to run. The 6522 also contains two 8-bit bidirectional ports. Each line of each port can be individually programmed as either an input or an output line, thus bidirectional. (See the 6522 block diagram in Appendix A).

*chip manufacturer may be substituted based on availablity

GENERAL INSTRUMENT AY-3-8910 PROGRAMMABLE SOUND GENERATOR

Sound I, Sound II and Sound/Speech I utilize General Instrument's AY-3-8910 Programmable Sound Generator to produce a remarkable variety of sound effects, from harpsichords to gunshots. This chip was selected for features which allow full software control of sound

generation and continuous sound without constant processor control.

The AY-3-8910, which will be referred to as PSG, is also a register oriented device containing 16 directly addressable registers. All sound generation is controlled through these registers. These registers are broken down into functional blocks: tone generators (fine and coarse tune), noise generator, mixers, amplitude control, envelope generator and D/A Converters.

Sound is produced through the manipulation of these blocks. For example, to produce a basic tone without noise, you would reference and set each of the following: the tone period register for one of the three channels, the channel for tone only, the amplitude for the channel and finally, turn off the channel to end the sound effect. More complex sound effects can be achieved by manipulating any number of combinations of the functional blocks. A more detailed explanation of the registers, functions and programming information will be given in the sections to follow.

VOTRAX SC-01 SPEECH SYNTHESIZER

Speech synthesis is accomplished through the use of Votrax's SC-01 Speech Synthesizer. This single chip phoneme synthesizer can produce continuous speech with a low data rate of 70-100 bits per second. The SC-01 contains 64 different phonemes (basic sound units of speech) of the English language. The phonemes are sequenced to form continuous speech. Since the SC-01 stores phonemes rather than encoded words, it can produce unlimited vocabulary, yet requires lower data rates to output words.

The 64 phonemes are broken down into 36 vowel sounds, 25 consonant sounds, and 3 pause phonemes (which emit no sounds). These phonemes can also be classified by the manner in which these sounds are produced in the human vocal tract, such as voiced sounds, fricative sounds (sounds produced by constricting the airflow), plosive sounds (sounds produced by stopping the airflow and then releasing it), and nasal sounds. (See Appendix C for tables of the 64 phonemes by classification.)

Each phoneme is assigned a code which is expressed in hexadecimal, the largest being $3F. Therefore, all phonemes can be accessed by a 6-bit code. The remaining two bits are accessed to set the pitch level of voiced phonemes. However, this feature is not necessary for normal pitch levels because inflection is automatically generated by the SC-01. Programming information and a more detailed explanation of the phonemes and their concatenation will be given in the sections to follow.

The appendices in the back of this manual contain some technical information provided by the manufacturers of the components described above, as well as a brief explanation of the demonstration program on the disk.

EQUIPMENT REQUIREMENTS

Apple II or Apple II Plus
48K RAM
Disk Drive
1 External 8 ohm Speaker for Sound I and Speech I
2 External 8 ohm Speakers for Sound II and Sound/Speech I

VOLUME AND FREQUENCY CONTROL

The control devices on each board have been specially selected and placed on the board for ease of use. Volume or frequency is controlled by a thumb wheel device located at the top of the board as it sits in the slot. This allows you to adjust the volume or frequency while the boards are in the computer. No screw driver is needed so you do not have to pull the board out of the slot to make adjustments.

Each board has a volume control device for each speaker. Sound I and Speech I contain one each and Sound II and Sound/Speech I contain two each. The volume control devices are located at the rear of the board as it sits in the slot.

Speech I and Sound/Speech I also contain a frequency control device to adjust the output pitch. This device is located in front of the volume control or towards the middle of the board as it sits in the slot.

INSTALLATION INSTRUCTIONS

Mockingboard peripherals are equipped with one (or two for Sound II or Sound/Speech I) 1/2 watt amplifier(s). The boards can also drive 8 ohm speakers directly or can be connected to the auxiliary inputs of your stereo amplifier. The board installation instruction is outlined below for both direct speaker interface and external amplification.

Care should be taken in handling the boards. Always turn off the computer before inserting or removing the boards from the slot. Handle the boards by the edges, taking care not to grab the gold plated pins.

Direct Speaker Interface

1. Make sure that the computer is off before proceeding with the installation.
2. The board may be plugged into any slot number from 1 to 7; however, the demonstration program will access the board in slot 4. Insert the gold plated fingers of the board into the slot connectors and gently rock until firmly seated.
3. Plug the cable, included in the package, into the board with the imprinted series number side faced down.
4. Plug the speaker(s) into the cable. The plugs on the end of the cable will plug into any RCA type phone jack.

External Amplification

1. Turn the computer off before proceeding.
2. The board may be plugged into any slot number from 1 to 7; however, the demonstration program will access the board in slot 4. Insert the gold plated fingers of the board into the slot connectors and gently rock until firmly seated.
3. Plug the cable, included in the package, into the board with the imprinted series number side faced down.
4. Connect the cable to the stereo amplifier auxiliary inputs. The plugs on the end of the cable will plug into any RCA type phone jack.

Sweet Micro Systems will support the Mockingboard Sound/Speech I board with a series of sophisticated new adventure games. The board adds new dimensions to adventure games, making each situation more credible and more exciting. Characters will speak to you. Recognizable sounds and voices will identify your location. You may be required to make decisions in real time. Check with your local computer store for the games or contact us for more information.

WARRANTY REGISTRATION

Please fill out the Warranty Registration Card enclosed at the back of this manual and return it to us. When we receive this card, we will register you as the original owner of this product and provide you with these services:

1. You will receive information about updates and new products. Since we will continue to support our hardware products with new and exciting software, we would like to let you know first. You may also be eligible to purchase these products at a discount.
2. We will also have a record of your purchase in case you lose your sales slip. If this product should require service during the warranty period, you must send proof of purchase. Send in the card now so you won't have to worry about it later.
3. Finally, your comments and answers to the questions on the form will help us to design future products with our customers in mind.

Take a moment to complete and return the registration card now. Thank you.

SOUND I MANUAL

The Mockingboard Sound I is a single sound effects generator peripheral which can be incorporated into any program you create. Sound I produces a remarkable repertoire of quality sounds, suitable for any program enhancement. Programming is very simple and efficient. Best of all, your program will not suffer from this enhancement because the Programmable Sound Generator (PSG) chip requires very little processor time to output the sound and no processor time to sustain it. Although the board can be accessed by a program in any language, our discussion will be limited to the demonstration program and specific listings which are in 6502 assembly language. Some information on decimal equivalent address locations is also given.

We have included in this manual a program listing of the primary routines utilized on the demonstration program disk. These routines consist of the INITialization of the input/output ports; the LATCH address, WRITE to PSG and RESET functions of the bus control lines. These functions are used in the transfer of data and addresses to the PSG for output of the desired sound effect. Each routine will be explained following a brief explanation of board and register selection.

BOARD AND REGISTER SELECTION

Each time you wish to access the Sound I board, you must directly address the board location and a 6522 register. The board can be selected in the following manner:

From assembly language, the address format is:

[$CsOr]

"$Cs" the page number (in hex) of the area in the Apple's memory map allocated for input/output where "s" is the slot number of the Sound I board .

"Or" one of the sixteen registers (in hex) of the 6522 you wish to access

For example, if you wish to access register "1" and the board is installed in slot 4, the address would be "$C401."

From Basic, the address format is:

[-16384 + (256 * s) + r]

"-16384 + (256 * s)": the decimal equivalent of "$Cs" explained above
"s": the slot number of the Sound I board
"r": one of 16 registers of the 6522

Again, if you wish to access register "1" and the board is installed in slot 4, the address would be [-16384 + (256 * 4) + 1] or -15359.

INITIALIZATION

The 6522 contains 16 internal registers in which each register may be accessed directly. Registers 0 and 1 are the two 8-bit bidirectional ports through which information is transferred. The ports are labelled "B" (register 0) and "A" (register 1). Each line of each port can be individually programmed as either an input or an output line. For all intents and purposes both ports should be initialized as output ports.

To configure the ports, a specific number must be written to each of the Data Direction Registers, register 2 for port "B" (DDRB) and register 3 for port "A" (DDRA). If a bit in the number written to either DDR is designated as a zero, the corresponding line will be an input line. However, if the bit is a one, the corresponding line will be an output line.

Since the Sound I board utilizes port "A" for tranferring data and addresses to the PSG, all lines in the DDRA must be configured for output. In other words, all 8 bits must be set to "1." This binary number is equivalent to a "FF" in hexadecimal or "255" in decimal and should be stored or poked into the DDRA (or location $Cs03), respectively. See Figure 1-1.

Figure 1-1: Primary Routines in Assembly Language

```
                        1   *PRIMARY ROUTINES
                        2   *FOR SOUND I
                        3   *BOARD IN SLOT 4
                        4   *
                        5   *
                        6           ORG   $9000
                        7           OBJ   $9000
                        8   *                        ; DECIMAL
                        9   *
                        10  ORB     EQU   $C400      ; -15360
                        11  ORA     EQU   $C401      ; -15359
                        12  DDRB    EQU   $C402      ; -15358
                        13  DDRA    EQU   $C403      ; -15357
                        14  *
                        15  *
9000:  A9 FF            16  INIT    LDA   #$FF
9002:  8D 03 C4         17          STA   DDRA
9005:  A9 07            18          LDA   #$07
9007:  8D 02 C4         19          STA   DDRB
900A:  60               20          RTS
900B:  A9 07            21  LATCH   LDA   #$07
900D:  8D 00 C4         22          STA   ORB
9010:  A9 04            23          LDA   #$04
9012:  8D 00 C4         24          STA   ORB
9015:  60               25          RTS
9016:  A9 06            26  WRITE   LDA   #$06
9018:  8D 00 C4         27          STA   ORB
901B:  A9 04            28          LDA   #$04
901D:  8D 00 C4         29          STA   ORB
9020:  60               30          RTS
9021:  A9 00            31  RESET   LDA   #$00
9023:  8D 00 C4         32          STA   ORB
9026:  A9 04            33          LDA   #$04
9028:  8D 00 C4         34          STA   ORB
902B:  60               35          RTS
```

The DDRB should also be configured for output to the PSG. However, Sound I utiliizes only three of the eight lines in port "B." Two lines are connected to the two bus control pins and the other line is connected to the reset pin of the PSG. There are actually three bus control pins (Bus

Direction, Bus Control 2, Bus Control 1); however, Bus Control 2 is tied high (+5V). The first three lines should be configured as "1" with the remaining five lines as "0." This binary number is equivalent to "07" in hexadecimal or "7" in decimal and should be stored or poked into the DDRB (or location $Cs02), respectively. See Figure 1-1.

The 6522 will automatically reset the ports for input when the computer is turned on or reset. Therefore, the initialization of the ports is required only when the computer is powered up or reset. Once the ports have been initialized, all data written to those ports will be output to the PSG chip. In the demonstration program, the INIT routine begins at location $9000 (hex) or -28672 (decimal) and is referenced immediately after the primary routines are loaded by a CALL to the location.

LATCH, WRITE AND RESET

Now that the ports of the 6522 are initialized for output, information can be sent to the PSG through output register 1, or port "A" (ORA). The information written to register 0, or port "B" (ORB) then tells the PSG what to do with the information in port "A" (register 1). The signals sent by the 6522 to the bus control pins of the PSG indicates whether the contents of the bus in ORA is register data or a register address. If the bus contains a register address, the signal will indicate to the PSG that this register should be latched (referred to as Latch Address or Latch). If the bus contains data, the signal will indicate that the data should be written into the currently addressed register (referred to as Write to PSG or Write). In addition to the Latch and Write, the ORB can also be utilized to signal a reset to the PSG. This will clear all of the registers in the PSG. The "signal" refers to the value of the bit configuration of each function. See Figure 1-2 (adapted from a diagram in the General Instrument's Programmable Sound Generator Data Manual).

Please note that in each of these routines the bus control lines are always brought back to an inactive state. Inactive refers to the state of the bus line between the PSG and 6522. The 6522 signals to the PSG that no action is to be taken with the data on the line. The bit configuration for Inactive is equivalent to "$04" in hexadecimal or "4" in

decimal because the Reset pin, in normal operation, is set at "1." See Figure 1-1 for Primary Routines listing.

Figure 1-2: Truth Table for Bus Control Line

PSG FUNCTION	RESET	BDIR	BC1	HEX	DEC
INACTIVE	1	0	0	#$04	4
READ FROM PSG	1	0	1	#$05	5
WRITE TO PSG	1	1	0	#$06	6
LATCH ADDRESS	1	1	1	#$07	7
RESET	0	0	0	#$00	0

The Read from PSG function, shown in Figure 1-2, enables data to be output by the PSG. This will enable you to read data from any PSG register. The Data Direction Register of the 6522 must be re-initialized for input. Although not possible on the Sound I board, this function also allows you to read from other peripheral via the PSG I/O ports. This function is not covered in this manual as we have limited our discussion to those functions involved in the output of sounds.

PSG REGISTERS

Figure 1-3 is a table of PSG registers and their bit configurations (printed with permission of General Instrument). As each register function is described below, it may be helpful to refer to this table from time to time.

(Please note that the registers are labelled in decimal with the hexadecimal equivalence in the margin.)

Figure 1-3 PSG Registers

	REGISTER / BIT		B7	B6	B5	B4	B3	B2	B1	B0
	R0	Channel A Tone Period	8-BIT Fine Tune A							
	R1					4-BIT Coarse Tune A				
	R2	Channel B Tone Period	8-BIT Fine Tune B							
	R3					4-BIT Coarse Tune B				
	R4	Channel C Tone Period	8-BIT Fine Tune C							
	R5					4-BIT Coarse Tune C				
	R6	Noise Period				5-BIT Period Control				
	R7	Enable	IN/OUT		Noise			Tone		
			IOB	IOA	C	B	A	C	B	A
	R8	Channel A Amplitude				M	L3	L2	L1	L0
	R9	Channel B Amplitude				M	L3	L2	L1	L0
RA	R10	Channel C Amplitude				M	L3	L2	L1	L0
RB	R11	Envelope Period	8-BIT Fine Tune E							
RC	R12		8-BIT Coarse Tune E							
RD	R13	Envelope Shape/Cycle					CONT	ATT	ALT	HOLD
RE	R14	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
RF	R15	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

The first six registers (R0-R5) set the tone period for each of the three channels (A, B, C). Each tone period is broken down into an 8-bit fine tune register and a 4-bit coarse tune register. The tone period together with Apple's clock frequency produces the desired tone frequency. The relationship is expressed in the following equation:

Tone Frequency = Clock Frequency/(16 * Tone Period)

where: Tone Period = 256 Coarse Tune + Fine Tune
(all values are expressed in decimal)

Register 6 is the noise generator control. The noise period is set by the lower 5-bits of register 6. The noise frequency output is produced by the 5-bit noise period value and Apple's clock frequency and is expressed in the following relationship:

Noise Frequency = Clock Frequency/(16 * Noise Period)
(Noise Period expressed in decimal)

Register 7 is the mixer control which enables tone and/or noise outputs on selected channels. The lower 3-bits enable the tone generators for each channel. The next 3-bits enable the noise generator for each channel. The last 2-bits control the direction of the two general purpose I/O ports which are not accessible on the Sound I board.

The tone/noise on any channel is disabled by a high bit or "1" and enabled by a low bit or "0." A table of bit configuration for possible combinations is given in Figure 1-4 (adapted from a diagram in General Instrument's Programmable Sound Generator Data Manual).

Figure 1-4: Noise and Tone Enable Truth Table

I/O PORTS NOT USED		NOISE ENABLE BIT VALUE			TONE ENABLE BIT VALUE			NOISE/TONE ENABLED ON
B7	B6	B5	B4	B3	B2	B1	B0	CHANNEL
X	X	0	0	0	0	0	0	C, B, A
X	X	0	0	1	0	0	1	C, B
X	X	0	1	0	0	1	0	C, A
X	X	0	1	1	0	1	1	C
X	X	1	0	0	1	0	0	B, A
X	X	1	0	1	1	0	1	B
X	X	1	1	0	1	1	0	A
X	X	1	1	1	1	1	1	NONE

Where X = Not Significant 0 = Enable 1 = Disable*
*Disabling does not turn off a channel.
See Amplitude Control Register.

Since the I/O ports cannot be utilized on the Sound I board, the bit value has no significance except when determining the hexadecimal or decimal equivalent of the eight bit register value.

If, for example, we wish to output a tone frequency on channel A only, we would look up in the table the necessary bit configuration for tone enabled on channel A. The 8-bit value would be: XX111110. The noise would be disabled in all three channels (111) and the tone would be enabled on channel A only (110). If the "X's" were replaced with "0's," the value in hexadecimal would be "#$3E" (0011, 1110) and in decimal it would be "62."

Registers 8, 9 and A (or 8, 9 and 10) control the amplitude for each channel (A, B and C, respectively). The amplitude is determined by the lower 5-bits of each register. The fifth bit, referred to as the amplitude mode, determines whether the amplitude level is fixed or variable. When this bit is set equal to "0" (fixed), the amplitude level, in a sense, is manually controlled by your program since any variation in amplitude levels must be individually written to the PSG. The lower 4-bits fix the amplitude at one of sixteen levels. When all 4-bits are set to zero, a channel is turned off.

If the fifth bit is set equal to "1" (variable), the amplitude level is determined by the envelope pattern and the lower 4-bits become inactive. This will free the processor to continue with the program at hand. Both sound output and program processing can occur simultaneously for any length of time, as long as the envelope generator is not interrupted. The envelope generator will be discussed next.

Registers B, C and D (or 11, 12 and 13) control the generation of envelope patterns. Registers B and C control the frequency of the envelope while register D controls the relative shape and cycle pattern of the envelope.

The contents of registers B (fine tune) and C (coarse tune) are combined to produce a 16-bit envelope period value. This value in relationship with Apple's clock frequency produces the frequency of the envelope. The equation which expresses this relationship is as follows:

Envelope Frequency = Clock Frequency/(256 * Envelope Period)

where: Envelope Period = 256 * Coarse Tune + Fine Tune
(all values are expressed in decimal)

The envelope shape and cycle pattern is controlled by the bit configuration of the lower 4-bits in register D. Each of the 4-bits controls a function in the envelope generator. The first bit is the Hold function which when set to "1," limits the envelope to one cycle and holds the last count. The second bit is the Alternate function which when set to "1," reverses the count direction (up-down) after each cycle. The third bit is the Attack function which when set to "1," will count up (attack) and will count down when set to "0" (decay). The last bit is the Continue function which

when set to "1," will take its cue from the Hold bit. If this bit is set to "0," the envelope generator will reset to "0000" after one cycle and hold that count. See Figure 1-5 (printed with permission of General Instrument).

The last two registers are the 8-bit I/O ports. These registers are not used in the production of sound and cannot be used on the Sound I board.

Figure 1-5 Envelope Shape/Cycle Control

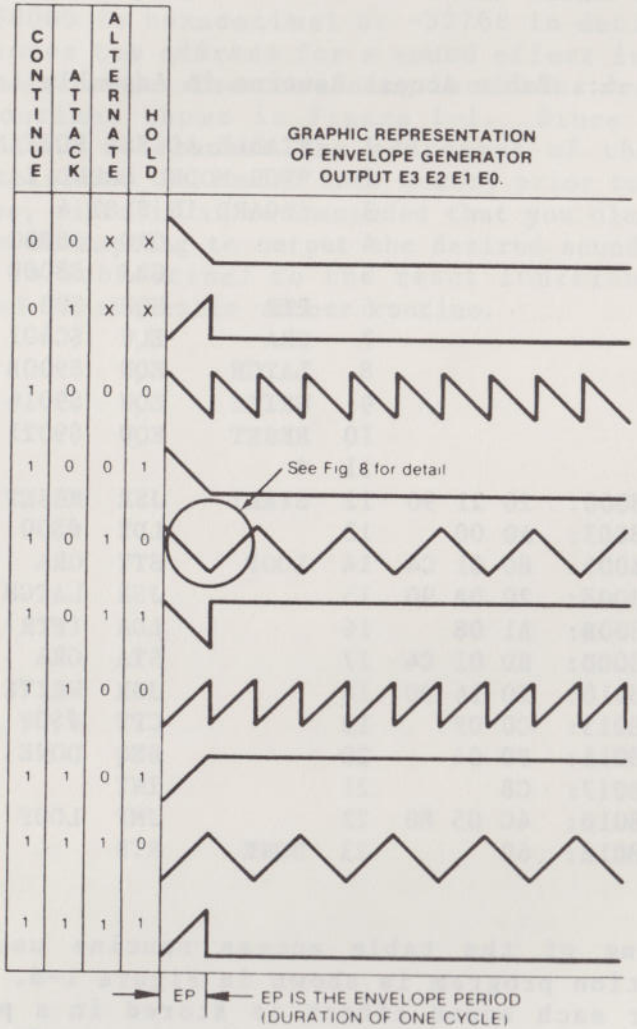


TABLE ACCESS ROUTINE

In theory, a sound is generated by the Sound I through a process of addressing and storing values in the appropriate registers of the PSG. A simple LDA (load the accummulator) and STA (store the accummulator) command in assembly language or a POKE command in Basic would achieve the desired sound effect. This would also prove to be a very tedious programming task and an inefficient use of memory space. A better and more efficient method is to set up a table of values and use a table access routine to take the values and store them in each of the 16 registers in succession.

Figure 1-6: Table Access Routine in Assembly Language

```
                        1   *TABLE ACCESS ROUTINE
                        2   *FOR MODEL SOUND I
                        3   *BOARD IN SLOT 4
                        4           ORG  $8000
                        5           OBJ  $8000
                        6   PTR     EQU  $08
                        7   ORA     EQU  $C401
                        8   LATCH   EQU  $900B
                        9   WRITE   EQU  $9016
                        10  RESET   EQU  $9021
                        11  *
8000:  20 21 90         12  START   JSR  RESET
8003:  A0 00            13          LDY  #$00
8005:  8C 01 C4         14  LOOP    STY  ORA
8008:  20 0B 90         15          JSR  LATCH
800B:  B1 08            16          LDA  (PTR),Y
800D:  8D 01 C4         17          STA  ORA
8010:  20 16 90         18          JSR  WRITE
8013:  C0 0F            19          CPY  #$0F
8015:  F0 04            20          BEQ  DONE
8017:  C8               21          INY
8018:  4C 05 80         22          JMP  LOOP
801B:  60               23  DONE    RTS
```

The listing of the table access routine used in the demonstration program is shown in Figure 1-6. The data values for each sound effect is stored in a particular memory location in register order. In other words, the first data value corresponds to register 0, the second value to register 1, etc. You must poke or load the address of the first data value at the memory location specified in the pointer. The content of the pointer will be indexed by the Y-register (in assembly language this is accomplished through indirect indexed addresssing) so that each data value will be written to its corresponding register. The routine will latch the register address (the counter value) and write the data to the PSG in successive order. Each sound effect will have its own location, which may be spaced 16 bytes (one byte for each register) apart.

The demonstration program loads the table access routine at location $8000 in hexadecimal or -32768 in decimal. The pointer stores the address for a sound effect in location $08 or 8 in decimal. The remaining locations are from the primary routines shown in Figure 1-1. Since the table access routine references the addresses of the primary routines, this must be loaded into memory prior to using the table. Also, since it is recommended that you clear the PSG chip before attempting to output the desired sound effect, a JSR (Jump to SubRoutine) to the reset function has been incorporated in the table access routine.

SOUND II MANUAL

The Mockingboard Sound II is a dual sound effects generator peripheral which can be incorporated into any program you create. The attractive feature of this board design is that two independent sounds can occur simultaneously, through one speaker or two. This is achieved through two separate sound generation systems. Two programmable sound generator chips (PSG) together with their own 6522 interface chip outputs the sounds independent of each other and still allows the Apple's processor to continue with the program. This versatile sound generation system will allow you to create richer and more dynamic sound effects. By taking the same sound effect and slightly delaying the output of one chip, you can produce a very full and powerful sound. Delay it even longer and you can produce a totally different sound effect. The Sound II is actually made up of two Sound I boards but can be accessed through one slot.

Since the Sound II board contains the same programming features as the Sound I, the information that follows is intended only to point out the differences. Please read the Sound I Manual, which precedes this section.

BOARD AND REGISTER SELECTION

Accessing the Sound II board involves the same process; however, in order to produce two independent sounds you must address each 6522 from a different entry point, $Cs0r or $Cs8r. The board can be selected in the following manner:

From assembly language, the address format is:

[$Cs0r] [$Cs8r]

"$Cs" the page number (in hex) of the area in the Apple's memory map allocated for input/output where "s" is the slot number of the Sound II board

"0r" one of the sixteen registers (in hex) of the first 6522 you wish to access

"8r" one of the sixteen registers (in hex) of the second 6522 for outputting another sound

For example, if you wish to access register "1" of the first 6522 and the board is installed in slot 4, the address would be "$C401. The address for the second 6522's register "1" would be $C481.

From Basic, the address format is:

[-16384 + (256 * s) + r] [-16384 + (256 * s) + 128 + r]

"-16384 + (256 * s)": the decimal equivalent of "$Cs" explained above
"s": the slot number of the Sound II board
"r": one of 16 registers of the first 6522

"128 + r": one of 16 registers of the second 6522

Again, if you wish to access register "1" of the first 6522 and the board is installed in slot 4, the address would be [-16384 + (256 * 4) + 1] or -15359. The address for the board and register "1" of the second 6522 would be [-16384 + (256 * 4) + 128 + 1] or -15231.

INITIALIZATION, LATCH, WRITE AND RESET ROUTINES

The definitions and explanations of each of these routines in the Sound I Manual also apply to the Sound II. Again, since there are two Sound I systems on this board, these routines apply to both separately. To be consistent in our explanation of the Sound II as two independent sound generating systems, the following listing of the Primary Routines contain two sets of routines. Two sets are not necessary and other more efficient methods can be used.

Figure 2-1: Primary Routines in Assembly Language

```
                        1   *PRIMARY ROUTINES
                        2   *FOR SOUND II
                        3   *BOARD IN SLOT 4
                        4   *
                        5   *
                        6           ORG   $9000
                        7           OBJ   $9000
                        8   *                        ; DECIMAL
                        9   *
                        10  ORB     EQU   $C400      ; -15360
                        11  ORA     EQU   $C401      ; -15359
                        12  DDRB    EQU   $C402      ; -15358
                        13  DDRA    EQU   $C403      ; -15357
                        14  ORB2    EQU   $C480      ; -15232
                        15  ORA2    EQU   $C481      ; -15231
                        16  DDRB2   EQU   $C482      ; -15230
                        17  DDRA2   EQU   $C483      ; -15229
                        18  *
                        19  *
9000: A9 FF             20  INIT    LDA   #$FF
9002: 8D 03 C4          21          STA   DDRA
9005: A9 07             22          LDA   #$07
9007: 8D 02 C4          23          STA   DDRB
900A: 60                24          RTS
900B: A9 07             25  LATCH   LDA   #$07
900D: 8D 00 C4          26          STA   ORB
9010: A9 04             27          LDA   #$04
9012: 8D 00 C4          28          STA   ORB
9015: 60                29          RTS
9016: A9 06             30  WRITE   LDA   #$06
9018: 8D 00 C4          31          STA   ORB
901B: A9 04             32          LDA   #$04
901D: 8D 00 C4          33          STA   ORB
9020: 60                34          RTS
9021: A9 00             35  RESET   LDA   #$00
9023: 8D 00 C4          36          STA   ORB
9026: A9 04             37          LDA   #$04
9028: 8D 00 C4          38          STA   ORB
902B: 60                39          RTS
902C: 00                40          BRK
902D: A9 FF             41  INIT2   LDA   #$FF
902F: 8D 83 C4          42          STA   DDRA2
9032: A9 07             43          LDA   #$07
```

```
9034:  8D 82 C4   44           STA   DDRB2
9037:  60         45           RTS
9038:  A9 07      46  LATCH2   LDA   #$07
903A:  8D 80 C4   47           STA   ORB2
903D:  A9 04      48           LDA   #$04
903F:  8D 80 C4   49           STA   ORB2
9042:  60         50           RTS
9043:  A9 06      51  WRITE2   LDA   #$06
9045:  8D 80 C4   52           STA   ORB2
9048:  A9 04      53           LDA   #$04
904A:  8D 80 C4   54           STA   ORB2
904D:  60         55           RTS
904E:  A9 00      56  RESET2   LDA   #$00
9050:  8D 80 C4   57           STA   ORB2
9053:  A9 04      58           LDA   #$04
9055:  8D 80 C4   59           STA   ORB2
9058:  60         60           RTS
```

In the demonstration program, the INIT routine begins at location $9000 (hex) or -28672 (decimal) and INIT2 begins at location $902D (hex) or -28627 (decimal). Both should be referenced immediately after the primary routines are BLOADed by a CALL to the locations.

PSG REGISTERS

The last two registers are the 8-bit I/O ports. These registers are not used in the production of sound but can be used on the Sound II board for add-on peripherals. Pin holes to both chips have been provided on the board. (The pin holes are not available on the Sound I board.)

TABLE ACCESS ROUTINE

The demonstration program stores the table access routine at location $8000 in hexadecimal or -32768 in decimal for the first sound generating system and at $801D in hexadecimal or -32739 for the second. The first pointer, PTR, stores the address for a sound effect in location $08 or 8 and the PTR2 stores it in $0A or 10. The remaining locations are from the primary routines shown in Figure 2-1. Since the table access routine references the addresses of

the primary routines, this must be loaded into memory prior to using the table. Also, since it is recommended that you clear both PSG chips before attempting to output the desired sound effect, a JSR (Jump to SubRoutine) to RESET has been incorporated in the routines.

Figure 2-2: Table Access Routine in Assembly Language

```
                          1   *TABLE ACCESS ROUTINE
                          2   *FOR MODEL SOUND II
                          3   *BOARD IN SLOT 4
                          4   *
                          5   *
                          6           ORG   $8000
                          7           OBJ   $8000
                          8   *
                          9   *
                          10  PTR     EQU   $08
                          11  ORA     EQU   $C401
                          12  LATCH   EQU   $900B
                          13  WRITE   EQU   $9016
                          14  RESET   EQU   $9021
                          15  PTR2    EQU   $0A
                          16  ORA2    EQU   $C481
                          17  LATCH2  EQU   $9038
                          18  WRITE2  EQU   $9043
                          19  RESET2  EQU   $904E
                          20  *
                          21  *
8000:  20 21 90           22  START   JSR   RESET
8003:  A0 00              23          LDY   #$00
8005:  8C 01 C4           24  LOOP    STY   ORA
8008:  20 0B 90           25          JSR   LATCH
800B:  B1 08              26          LDA   (PTR),Y
800D:  8D 01 C4           27          STA   ORA
8010:  20 16 90           28          JSR   WRITE
8013:  C0 0F              29          CPY   #$0F
8015:  F0 04              30          BEQ   DONE
8017:  C8                 31          INY
8018:  4C 05 80           32          JMP   LOOP
801B:  60                 33  DONE    RTS
801C:  00                 34          BRK
801D:  20 4E 90           35  START2  JSR   RESET2
8020:  A0 00              36          LDY   #$00
8022:  8C 81 C4           37  LOOP2   STY   ORA2
```

```
8025:  20 38 90   38          JSR  LATCH2
8028:  B1 0A      39          LDA  (PTR2),Y
802A:  8D 81 C4   40          STA  ORA2
802D:  20 43 90   41          JSR  WRITE2
8030:  C0 0F      42          CPY  #$0F
8032:  F0 04      43          BEQ  DONE2
8034:  C8         44          INY
8035:  4C 22 80   45          JMP  LOOP2
8038:  60         46  DONE2   RTS
```


SPEECH I MANUAL

The Mockingboard Speech I is a speech synthesizer peripheral which utilizes phonemes as a basis for producing continuous speech. Phonemes are basic units of sound in a language which when blended together forms speech. The 64 phonemes present on the SC-01 are identified as the phonemes of the English language. Therefore, the vocabulary in English will be unlimited and you will probably be able to create some foreign words as well. (Please refer to Appendix C for table of phonemes.)

According to the linguist, phonemes are identified through contrast. For example, in contrasting the pronunciation of the words "bat" and "cat," you find they only differ by one phoneme (the phonemes /b/ and /c/). This example shows the contrast between monosyllabic words and identifying the phoneme sequence to produce these words should be an easy task. However, identifying the phonemes for multisyllabic words may be a bit more difficult because some phonemes vary minimally and other phonemes differ in duration. Some also serve as transitional phonemes to produce a smoother pronunciation.

Some guidelines for sequencing phonemes are available (see appendix C for sources), however, the most effective method is pronouncing each sound of the word out loud. You will find yourself becoming an active listener and analyzing subtle differences. BEWARE: a strange metamorphosis may occur as you begin speaking like the machine in order to make your machine speak like you.

BOARD AND REGISTER SELECTION

Each time you wish to access the Speech I board, you must directly address the board location and a 6522 register. The board can be selected in the following manner:
From assembly language, the address format is:

[$CsOr]

"$Cs" the page number (in hex) of the area in the Apple's memory map allocated for input/output where "s" is the slot number of the Speech I board

"Or" one of the sixteen registers (in hex) of the 6522 you wish to access (to be explained shortly)

For example, if you wish to access register "1" and the board is installed in slot 4, the address would be "$C401.

From Basic, the address format is:

[-16384 + (256 * s) + r]

"-16384 + (256 * s)": the decimal equivalent of "$Cs" explained above
"s": the slot number of the Speech I board
"r": one of 16 registers of the 6522

Again, if you wish to access register "1" and the board is installed in slot 4, the address would be [-16384 + (256 * 4) + 1] or -15359.

INITIALIZATION

The 6522 contains 16 internal registers in which each register may be accessed directly. Registers 0 and 1 are the two 8-bit bidirectional ports through which information is transferred. The ports are labelled "B" (register 0) and "A" (register 1). Each line of each port can be individually programmed as either an input or an output line. Since each phoneme requires only a 6-bit code to output speech through the SC-01, only port "B" will be initialized for output.

To configure the port, a specific number must be written to the Data Direction Registers (register 2 for port "B" (DDRB)). If a bit in the number written to DDRB is designated as a zero, the corresponding line will be an input line. However, if the bit is a one, the corresponding line will be an output line.

Since the Speech I board utilizes port "B" for tranferring the phoneme code to the SC-01, all lines in the DDRB must be configured for output. In other words, all 8 bits must be set to "1." This binary number is equivalent to a "FF" in hexadecimal or "255" in decimal and should be stored or poked into the DDRB (or location $Cs02), respectively. See Figure 3-1.

Also initialized are registers 13 and 14 (or C and D), the Peripheral Control Register (PCR) and the Interrupt Flag Register (IFR), respectively. The 6522 allows handshake control of the data transfer between the Apple's processor and the SC-01 through the PCR. Handshaking refers to the interaction of the "Data Ready" signal generated by the 6522 and the "Data Taken" signal from the SC-01 which regulates the flow of data. This handshaking operation is used on the Speech I board to check for the completion of the output of a phoneme before the next phoneme is sent. This check routine will be discussed next.

Port B lines handshake on a write operation only, or when data is transferred from the Apple's processor to the SC-01 (which is what we want). The configuration of the high order bits of the PCR select the operating mode of the port B lines. Bits 7-5 act as a "Data Ready" output and bit 4, which is the interrupt control, accepts the "Data Taken" signal from the SC-01. The binary equivalent for initializing the PCR in hexadecimal is #$B0 and in decimal it is 176. (The configuration of the lower order bits are insignificant since port A lines are not utilized by the Speech I.) The interrupt flag is set when the "Data Taken" signal is accepted and must be cleared before proceeding. This is accomplished by the IFR. The binary equivalent for initializing the IFR in hexadecimal is #$10 and in decimal it is 16.

CHECK ROUTINE

The check routine, as mentioned before, looks for the completion of an output. More specifically, it is looking for the interrupt flag to be set before clearing the flag and outputting the next phoneme. The routine will continue in a loop until it is detected.

Figure 3-1: Primary Routines in Assembly language

```
                        1   *PRIMARY ROUTINES
                        2   *FOR SPEECH I
                        3   *BOARD IN SLOT 4
                        4   *
                        5   *
                        6           ORG  $9000
                        7           OBJ  $9000
                        8*                      ;DECIMAL
                        9   ORB     EQU  $C400  ; -15360
                        10  DDRB    EQU  $C402  ; -15358
                        11  PCR     EQU  $C40C  ; -15348
                        12  IFR     EQU  $C40D  ; -15347
                        13  *
9000:  A9 FF            14  INIT    LDA  #$FF
9002:  8D 02 C4         15          STA  DDRB
9005:  AD 0C C4         16          LDA  PCR
9008:  09 B0            17          ORA  #$B0
900A:  8D 0C C4         18          STA  PCR
900D:  A9 10            19          LDA  #$10
900F:  8D 0D C4         20          STA  IFR
9012:  60               21          RTS
9013:  48               22  CHK     PHA
9014:  A9 10            23  LOOP    LDA  #$10
9016:  2C 0D C4         24          BIT  IFR
9019:  F0 F9            25          BEQ  LOOP
901B:  8D 0D C4         26          STA  IFR
901E:  68               27          PLA
901F:  8D 00 C4         28          STA  ORB
9022:  60               29          RTS
```

TABLE ACCESS ROUTINE

A table access routine can be utilized to output continuous speech by setting up a table of phoneme codes for a particular word or phrase you wish to output. This routine will output each phoneme code and check for its completion before accessing the next code.

The listing of the table access routine used in the demonstration program is shown in Figure 3-2. The phoneme codes for each word or phrase are stored in a table which can be accessed by referencing the location of the first

phoneme code. Each word or phrase will have its own location in memory. The table location for the demonstration program is located at $8100 (in hex) or -32512 (in decimal). You must poke or load the address of the first phoneme code at the memory location specified in the pointer. This phoneme will then be spoken by the SC-01. During this process, the table access routine will jump to the "CHK" subroutine in the primary routines and return for the next phoneme when the SC-01 has finished speaking. The content of the pointer will be indexed by the Y-register (in assembly language this is accomplished through indirect indexed addresssing) so that the next phoneme can be accessed. The counter will continue to increment until it reaches a marker which will indicate the end of the word or phrase. The marker in the table access routine of this demonstration program is #$3F or 63. This value is also the phoneme code for "stop" (stop speaking). The marker should be the last code in each word or phrase in the table.

The demonstration program stores the table access routine at location $8000 in hexadecimal or -32768 in decimal. The pointer stores the address for the word or phrase in location $06 or 6. The remaining location is from the primary routines shown in Figure 3-1. Since the table access routine references the addresses of the primary routines, this must be loaded into memory prior to using the table.

Figure 3-2: Table Access Routine in Assembly Language

```
                        1   *TABLE ACCESS ROUTINE
                        2   *FOR MODEL SPEECH I
                        3   *BOARD IN SLOT 4
                        4   *
                        5   *
                        6           ORG  $8000
                        7           OBJ  $8000
                        8   *
                        9   *
                        10  PTR     EQU  $06
                        11  ORB     EQU  $C400
                        12  CHK     EQU  $9013
                        13  *
                        14  *
8000:  A0 00            15  START   LDY  #$00
```

```
8002:  B1 06       16  LOOP    LDA  (PTR),Y
8004:  8D 00 C4    17          STA  ORB
8007:  20 13 90    18          JSR  CHK
800A:  C9 3F       19          CMP  #$3F     ;MARKER
800C:  F0 04       20          BEQ  DONE
800E:  C8          21          INY
800F:  4C 02 80    22          JMP  LOOP
8012:  60          23  DONE    RTS
```


SOUND/SPEECH I MANUAL

The Mockingboard Sound/Speech I is a combination sound effects generator and speech synthesizer peripheral. The Sound/Speech I is a powerful board because of its versatility. Not only can this board produce a single or continuous sound or speech, it can also produce simultaneous sound and speech. A continuous sound effect can be a very effective background for speech. Creating interaction between the two sounds is unlimited. You can also have simultaneous sound effects since speech sounds can be intentionally garbled to produce an effect rather than intelligable speech.

The board incorporates the Mockingboard Sound I and Speech I in their entirety and without compromise. They are in actuality a separate sound generating system and a speech generating system physically placed on one board. The advantage to this design is that both systems can be accessed and controlled through the same peripheral slot and still output sounds simultaneously. The features in each system have been previously explained. The following information is intended only to point out the differences. Please read the Sound I and Speech I sections.

BOARD AND REGISTER SELECTION

The board is accessed in the same manner as described in the previous sections by directly addressing the board location and a 6522 register. The difference is that since each system is independent of the other and interfaced by its own 6522, the address locations must also be different. The speech generating system uses the same address format as described in the Speech I manual. The sound effects system uses a slightly different address format as described below. From assembly language, the address format is:

Speech = [$Cs0r] Sound = [$Cs8r]

"$Cs" the page number (in hex) of the area in the Apple's memory map allocated for input/output where "s" is the slot number of the Sound/Speech I board (the same for both systems)

"0r" one of the sixteen registers (in hex) of the speech system's 6522 you wish to access
"8r" one of the sixteen registers (in hex) of the sound system's 6522 you wish to access

For example, if you wish to access register "1" of the speech system's 6522 and the board is installed in slot 4, the address would be "$C401." The address for register "1" of the sound system's 6522 in slot 4 would be "$C481."

From Basic, the address format is:

```
Speech = [-16384 + (256 * s) + r]
Sound  = [-16384 + (256 * s) + 128 + r]
```

"-16384 + (256 * s)": the decimal equivalent of "$Cs" explained above
"s": the slot number of the Sound/Speech I board
"r": one of 16 registers of the speech system's 6522
"128 + r": one of 16 registers of the sound system's 6522

Again, if you wish to access register "1" and the board is installed in slot 4, the address would be [-16384 + (256 * 4) + 1] or -15359 for speech and [-16384 + (256 * 4) + 128 + 1] or -15231 for sound.

PRIMARY ROUTINES

To be consistent in our explanation of the Sound/Speech I as two independent systems, the listing of the Primary Routines given in Figure 4-1 contain both primary routines for the Sound I and Speech I. Only the address for the sound system is different, as explained above. Again, it is important to stress that these are two independent systems and must be programmed as separate entities. By simply referencing the appropriate 6522, you will be able to program that particular system.

Figure 4-1: Primary Routines for Sound/Speech I in Assembly

```
                          1  *PRIMARY ROUTINES
                          2  *FOR SOUND/SPEECH I
                          3  *BOARD IN SLOT 4
                          4  *
                          5  *
                          6        ORG  $9000
                          7        OBJ  $9000
                          8  *                    ;DECIMAL
                          9  *
                         10  ORB    EQU  $C400  ; -15360
                         11  DDRB   EQU  $C402  ; -15358
                         12  PCR    EQU  $C40C  ; -15348
                         13  IFR    EQU  $C40D  ; -15347
                         14  ORB2   EQU  $C480  ; -15232
                         15  ORA2   EQU  $C481  ; -15231
                         16  DDRB2  EQU  $C482  ; -15230
                         17  DDRA2  EQU  $C483  ; -15229
                         18  *
                         19  *
9000: A9 FF              20  INIT   LDA  #$FF
9002: 8D 02 C4           21         STA  DDRB
9005: AD 0C C4           22         LDA  PCR
9008: 09 B0              23         ORA  #$B0
900A: 8D 0C C4           24         STA  PCR
900D: A9 10              25         LDA  #$10
900F: 8D 0D C4           26         STA  IFR
9012: 60                 27         RTS
9013: 48                 28  CHK    PHA
9014: A9 10              29  LOOP   LDA  #$10
9016: 2C 0D C4           30         BIT  IFR
9019: F0 F9              31         BEQ  LOOP
901B: 8D 0D C4           32         STA  IFR
901E: 68                 33         PLA
901F: 8D 00 C4           34         STA  ORB
9022: 60                 35         RTS
9023: A9 FF              36  INIT2  LDA  #$FF
9025: 8D 83 C4           37         STA  DDRA2
9028: A9 07              38         LDA  #$07
902A: 8D 82 C4           39         STA  DDRB2
902D: 60                 40         RTS
902E: A9 07              41  LATCH2 LDA  #$07
9030: 8D 80 C4           42         STA  ORB2
9033: A9 04              43         LDA  #$04
9035: 8D 80 C4           44         STA  ORB2
```

```
9038:  60          45          RTS
9039:  A9 06       46  WRITE2  LDA  #$06
903B:  8D 80 C4    47          STA  ORB2
903E:  A9 04       48          LDA  #$04
9040:  8D 80 C4    49          STA  ORB2
9043:  60          50          RTS
9044:  A9 00       51  RESET2  LDA  #$00
9046:  8D 80 C4    52          STA  ORB2
9049:  A9 04       53          LDA  #$04
904B:  8D 80 C4    54          STA  ORB2
904E:  60          55          RTS
```

The demonstration program stores the INIT routines beginning at location $9000 (hex) or -28672 (decimal) for initializing the registers for speech output. INIT2 begins at location $9023 (hex) or -28637 (decimal) for initializing the registers for sound output. Both should be referenced immediately after the primary routines are BLOADed, by a CALL to the locations.

PSG REGISTERS

The last two registers of the PSG chip on the sound system are the 8-bit I/O ports. These registers are not used in the production of sound but can be used on the Sound/Speech I board for add-on peripherals. Pin holes to this chip have been provided on the board. (This feature is not available on the Sound I board.)

TABLE ACCESS ROUTINE

The demonstration program stores the table access routine at location $8000 in hexadecimal or -32768 in decimal for the speech system and at $8014 or -32748 for the sound system. The first pointer stores the address for the word or phrase in location $06 or 6 and PTR2 stores the address for a sound effect in location $08 or 8. The remaining location is from the primary routines shown in Figure 4-1. Since the table access routine references the addresses of the primary routines, this must be loaded into memory prior to using the table. Also, since it is recommended that you clear the PSG chip before attempting to output the desired sound effect, a JSR (Jump to SubRoutine) to RESET has been incorporated in the routine.

Figure 4-2: Table Access Routine in Assembly Language

```
                                1  *TABLE ACCESS ROUTINE
                                2  *FOR MODEL SOUND/SPEECH I
                                3  *BOARD IN SLOT 4
                                4  *
                                5  *
                                6          ORG  $8000
                                7          OBJ  $8000
                                8  *
                                9  *
                                10 PTR     EQU  $06
                                11 ORB     EQU  $C400
                                12 CHK     EQU  $9013
                                13 PTR2    EQU  $08
                                14 ORA2    EQU  $C481
                                15 LATCH2  EQU  $902E
                                16 WRITE2  EQU  $9039
                                17 RESET2  EQU  $9044
                                18 *
                                19 *
8000:  A0 00       20 START    LDY  #$00
8002:  B1 06       21 LOOP     LDA  (PTR),Y
8004:  8D 00 C4    22          STA  ORB
8007:  20 13 90    23          JSR  CHK
800A:  C9 3F       24          CMP  #$3F     ;MARKER
800C:  F0 04       25          BEQ  DONE
800E:  C8          26          INY
800F:  4C 02 80    27          JMP  LOOP
8012:  60          28 DONE     RTS
8013:  00          29          BRK
8014:  20 44 90    30 START2   JSR  RESET2
8017:  A0 00       31          LDY  #$00
8019:  8C 8A C4    32 LOOP2    STY  ORA2
801C:  20 2E 90    33          JSR  LATCH2
801F:  B1 06       34          LDA  (PTR2),Y
8021:  8D 81 C4    35          STA  ORA2
8024:  20 39 90    36          JSR  WRITE2
8027:  C0 0F       37          CPY  #$0F
8029:  F0 04       38          BEQ  DONE2
802B:  C8          39          INY
802C:  4C 19 80    40          JMP  LOOP2
802F:  60          41 DONE2    RTS
```

APPENDIX A

SYNERTEK SY6522 VERSATILE INTERFACE ADAPTER

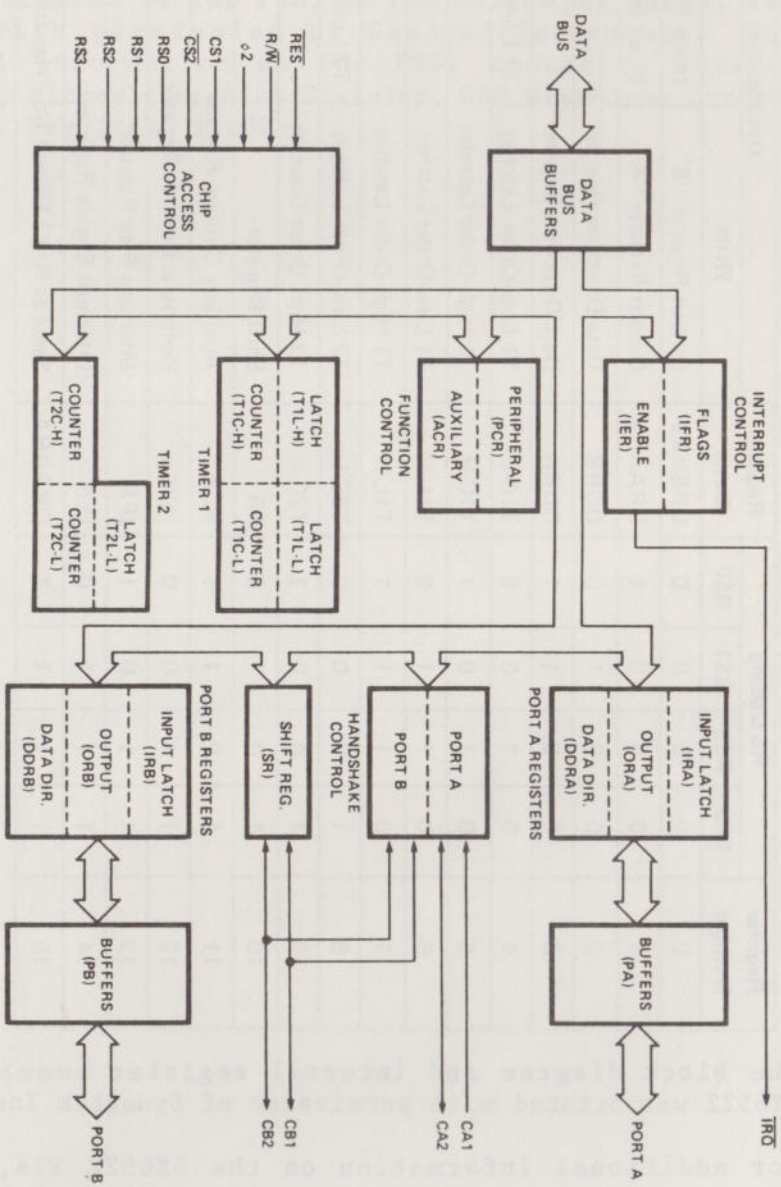


Figure 1. SY6522 Block Diagram

Register Number	RS Coding				Register Desig.	Description	
	RS3	RS2	RS1	RS0		Write	Read
0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
2	0	0	1	0	DDRB	Data Direction Register "B"	
3	0	0	1	1	DDRA	Data Direction Register "A"	
4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	0	1	0	1	T1C-H	T1 High-Order Counter	
6	0	1	1	0	T1L-L	T1 Low-Order Latches	
7	0	1	1	1	T1L-H	T1 High-Order Latches	
8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	1	0	0	1	T2C-H	T2 High-Order Counter	
10	1	0	1	0	SR	Shift Register	
11	1	0	1	1	ACR	Auxiliary Control Register	
12	1	1	0	0	PCR	Peripheral Control Register	
13	1	1	0	1	IFR	Interrupt Flag Register	
14	1	1	1	0	IER	Interrupt Enable Register	
15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

SY6522 Internal Register Summary

The block diagram and internal register summary of the SY6522 was printed with permission of Synertek Inc.

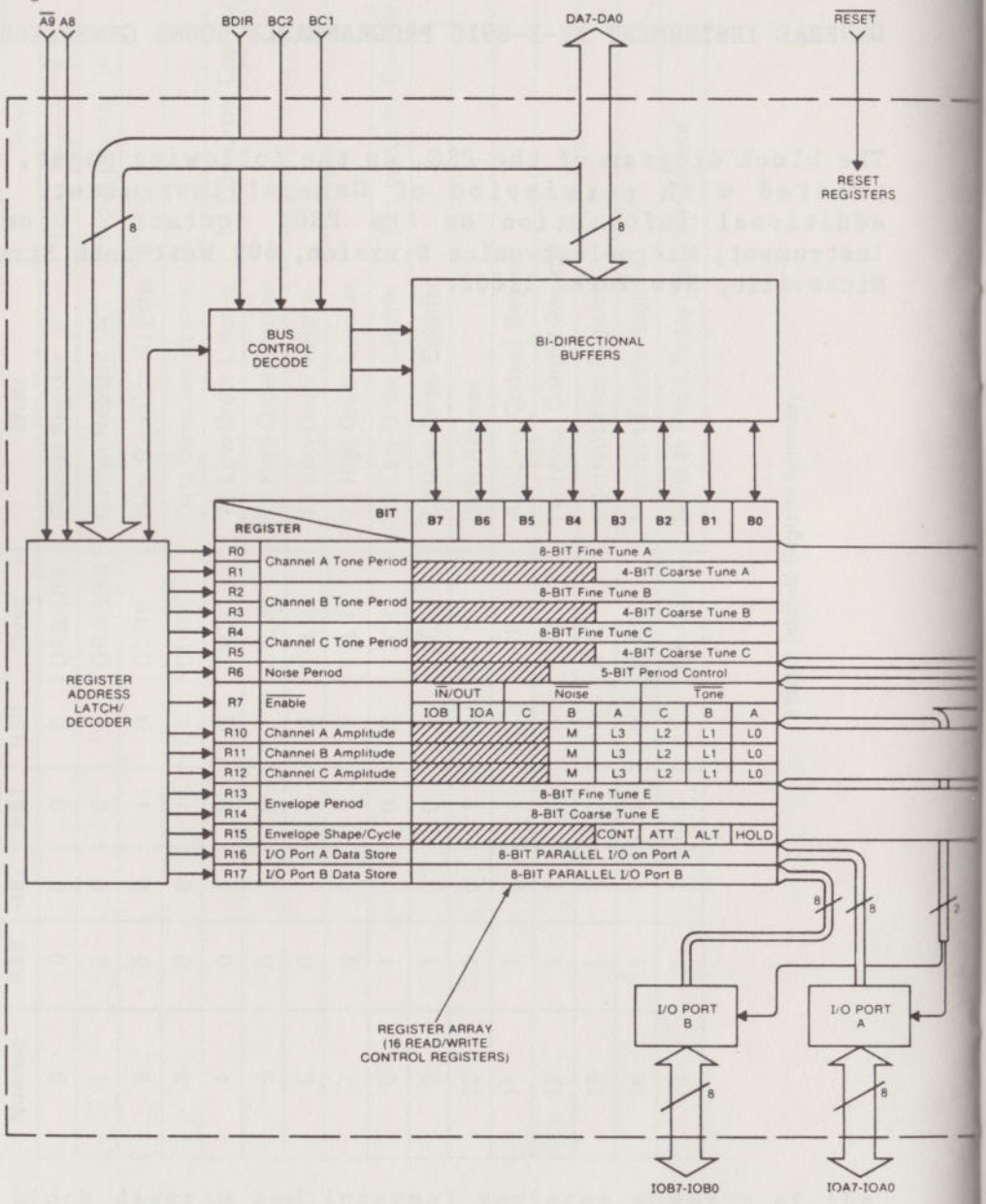
For additional information on the SY6522 VIA, contact: Microprocessor Applications, Synertek, P.O. Box 552 MS60, Santa Clara, CA 95052.

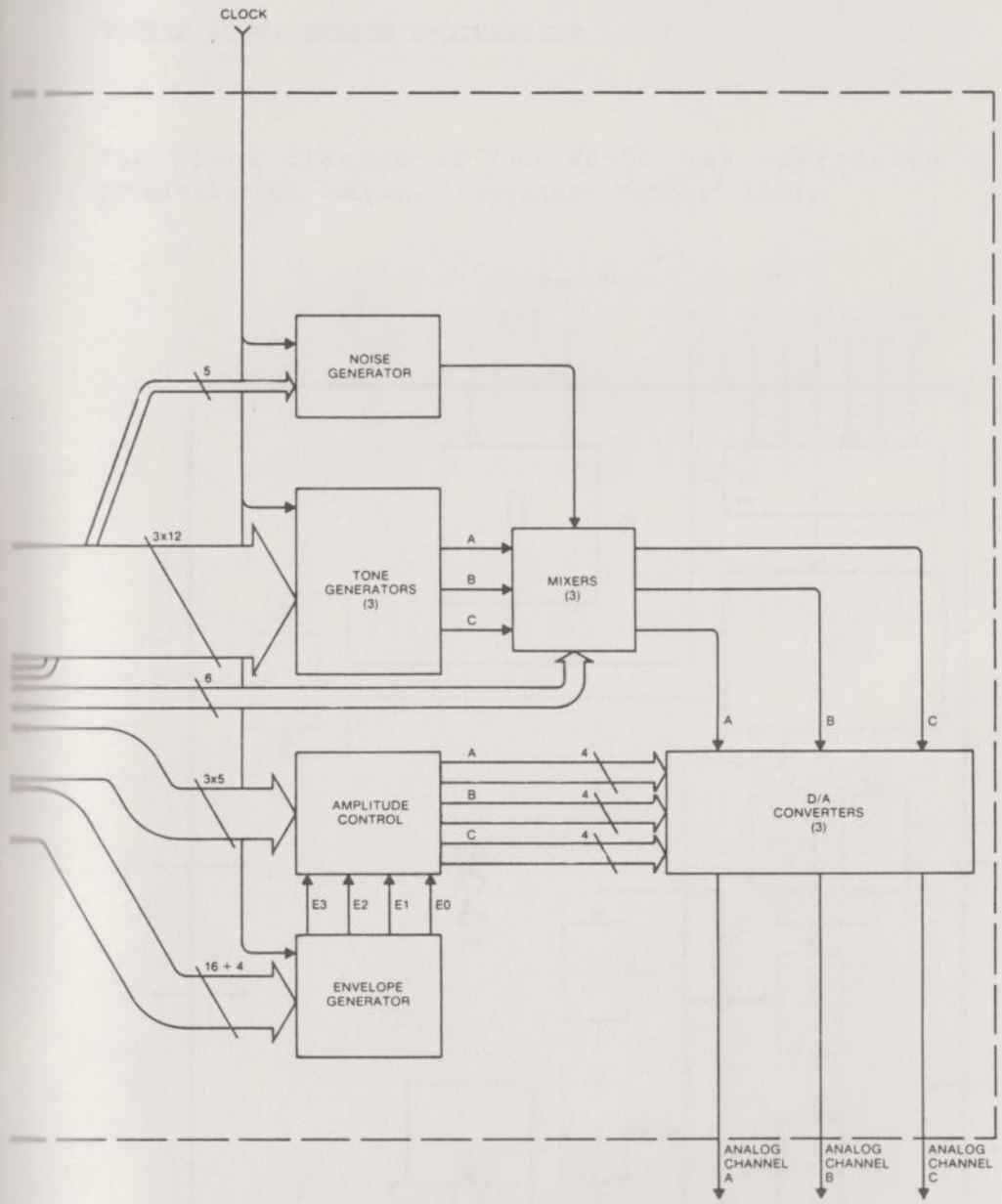
APPENDIX B

GENERAL INSTRUMENT AY-3-8910 PROGRAMMABLE SOUND GENERATOR

The block diagram of the PSG, on the following pages, was printed with permission of General Instrument. For additional information on the PSG, contact: General Instrument, Microelectronics Division, 600 West John Street, Hicksville, New York 11802.

Fig. 2 PSG BLOCK DIAGRAM





APPENDIX C

VOTRAX SC-01 SPEECH SYNTHESIZER

The block diagram of the SC-01 was reproduced with permission of Votrax. Copyright Votrax® 1980.

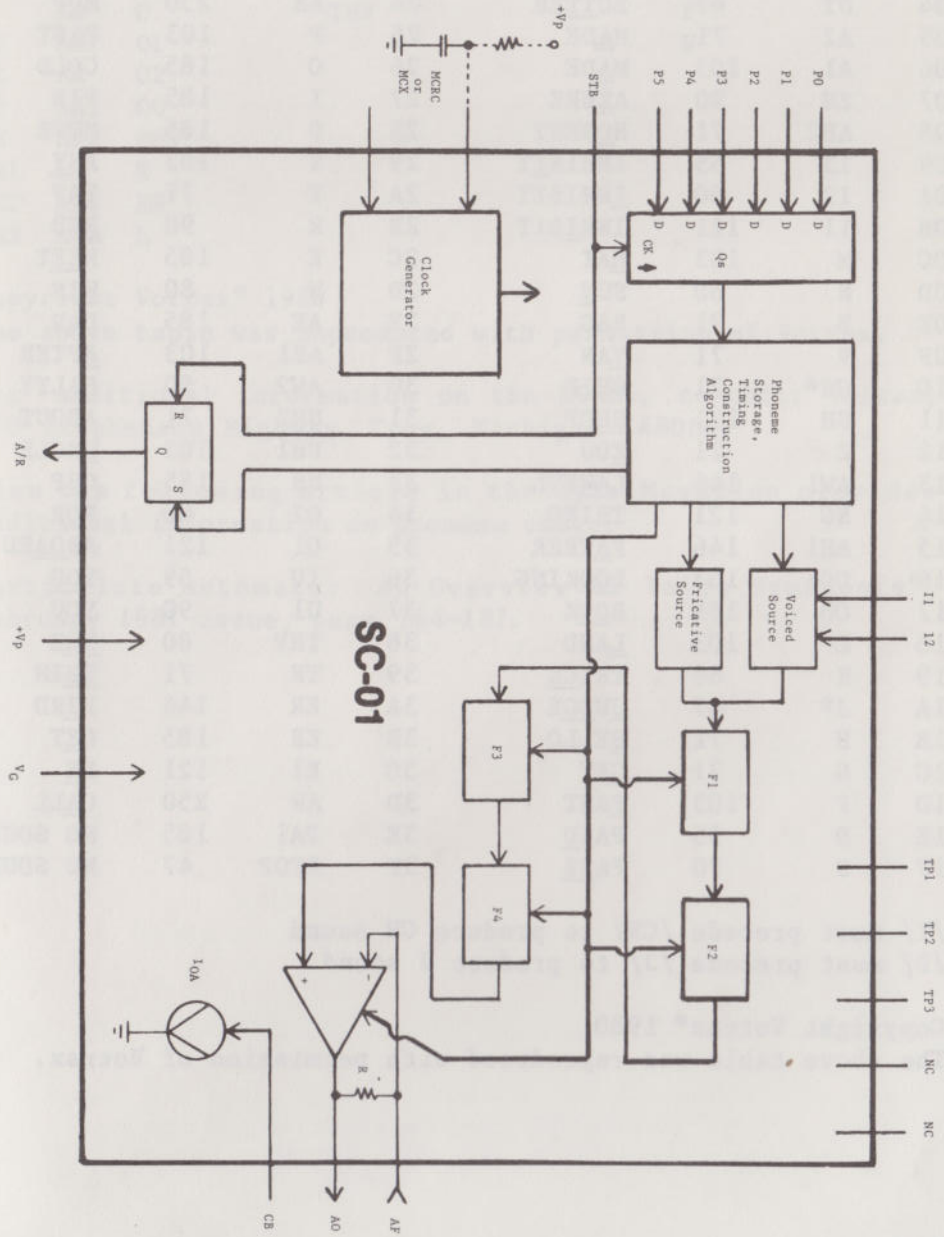


Table 1 Phoneme Chart

Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word	Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word
00	EH3	59	JACKET	20	A	185	DAY
01	EH2	71	ENLIST	21	AY	65	DAY
02	EH1	121	HEAVY	22	Y1	80	YARD
03	PA0	47	NO SOUND	23	UH3	47	MISSION
04	DT	47	BUTTER	24	AH	250	MOP
05	A2	71	MADE	25	P	103	PAST
06	A1	103	MADE	26	O	185	COLD
07	ZH	90	AZURE	27	I	185	PIN
08	AH2	71	HONEST	28	U	185	MOVE
09	I3	55	INHIBIT	29	Y	103	ANY
0A	I2	80	INHIBIT	2A	T	71	TAP
0B	I1	121	INHIBIT	2B	R	90	RED
0C	M	103	MAT	2C	E	185	MEET
0D	N	80	SUN	2D	W	80	WIN
0E	B	71	BAG	2E	AE	185	DAD
0F	V	71	VAN	2F	AE1	103	AFTER
10	CH*	71	CHIP	30	AW2	90	SALTY
11	SH	121	SHOP	31	UH2	71	ABOUT
12	Z	71	ZOO	32	UH1	103	UNCLE
13	AW1	146	LAWFUL	33	UH	185	CUP
14	NG	121	THING	34	O2	80	FOR
15	AH1	146	FATHER	35	O1	121	ABOARD
16	OO1	103	LOOKING	36	IU	59	YOU
17	OO	185	BOOK	37	U1	90	YOU
18	L	103	LAND	38	THV	80	THE
19	K	80	TRICK	39	TH	71	THIN
1A	J*	47	JUDGE	3A	ER	146	BIRD
1B	H	71	HELLO	3B	EH	185	GET
1C	G	71	GET	3C	E1	121	BE
1D	F	103	FAST	3D	AW	250	CALL
1E	D	55	PAID	3E	PA1	185	NO SOUND
1F	S	90	PASS	3F	STOP	47	NO SOUND

/T/ must precede /CH/ to produce CH sound
/D/ must precede /J/ to produce J sound

Table 2 Phoneme Categories According to Production Features

Voiced				Voiced Fricat.	Voiced Stop	Fricat. Stop	Fricat.	Nasal	No Sound
E	A	UH	IU	Z	B	T	S	M	PA0
E1	A1	UH1	U	ZH	D	DT	SH	N	PA1
Y	A2	UH2	U1	J	G	K	CH	NG	STOP
Y1	AY	UH3	W	V		P	TH		
I	AE	O		THV			F		
I1	AE1	O1					H		
I2	AH	O2							
I3	AH1	OO							
EH	AH2	OO1							
EH1	AW	R							
EH2	AW1	ER							
EH3	AW2	L							

For additional information on the SC-01, contact: Votrax, 500 Stephenson Highway, Troy, Michigan 48084.

Also the following article in the Byte Magazine provides additional information on phoneme usage:

"Articulate Automata: An Overview of Voice Synthesis" February 1981 issue, page 164-187.

APPENDIX D

THE DEMONSTRATION PROGRAM

The demonstration program has been designed in modules so that any part of the program can be used in any program you create. These modules, therefore, were not necessarily written for efficiency but rather for portability. Since the entire program can be listed, viewed and copied, we recommend you list and study them for better understanding. In addition, ample space has been provided between data tables on this disk for experimental use.

Each demonstration program includes a boot program, primary routines, table access routines, data tables, menus and demonstrations. The boot program will automatically load all the necessary utility programs when the computer is turned on and run the menu program. The main menu will offer you a selection of demonstration sound effects and/or speech.

Each demo will poke the location of the selected sound into the pointer address and call the appropriate table access routines (as discussed in the individual manuals). The access routine will begin loading the registers with the data from the data table at the pointer location and then output the sound. The utility programs have been written in assembly language for speed and in some instances, a smoother sound.

The data tables are in machine language and are located at the following address for each demonstration program:

SOUND I:	Sound table at 8100.81AF
SOUND II:	Sound table at 8100.81AF
SPEECH I:	Speech table at 8100.8177
SOUND/SPEECH I:	Speech table at 8100.8177
	Sound table at 8900.89AF

To list the data table, enter the monitor of the Apple with a CALL-151, then enter the address as shown above. For example, to list the data table for Sound/Speech I, you should enter after the asterisk "8100.8177" for the speech table or "8900.89AF" for the sound table.

The sound table contains all the sound effects in the demonstration program, in successive order. The values stored in this table represent the values for each of the 16 registers of the PSG chip. Each sound effect requires only two rows of values. If a register is not used to produce a sound, the value "00" is stored for that register. For example, the "gunshot" sound effect appears as follows:

```
8100- 00 00 00 00 00 00 0F 07
8108- 10 10 10 00 0A 00 00 00
```

Registers 0-5 are not set because the gunshot sound is a noise effect not a tone effect. Register 6 is set to #$0F (decimal eqivalent is 15) which sets the 5 bit noise period to mid-value (highest value is 31 in decimal). Register 7 is set to #$07 (7 in decimal) to enable noise in all three channels. The binary equivalent would appear as "00000111" (See Figure 1-4). Registers 8-A (or 8-10) are each set to #$10 (16 in decimal) which sets the amplitude mode to a variable level amplitude. The amplitude of all three channels will be under the direct control of the Envelope Generator. Register B (or 11) is not set. Register C (or 12) is set to #$0A (10 in decimal) to set the envelope period. Register D (or 13) is set to #$00 which selects the envelope decay. Registers E-F (or 14-15) are not set.

Many sound effect variations are created by changing one or two register values. For example, the Explosion sound effect (address 8130.813F) and the Ocean sound effect (address 8140.814F) differ only by the envelope shape cycle. The Explosion (#$00) is a one cycle decay pattern and the Ocean (#$0E) is a continuous pattern (see Figure 1-5).

The speech table contains the phoneme codes for the words and phrases used in the demonstration program. They are listed sequentially. For example, the first word in the table is "TOOT" and is composed of six phonemes (including the stop code to designate the end).

```
8100- 2A 37 37 2A 03 3F 00 00
```

The word uses only three phoneme codes. The phoneme code for #$2A (T) is the sound for the consonant "T" and the phoneme code for #$37 (U1) is the shorter duration sound for the vowel "U." The code #$03 (PA0) is a no-sound phoneme and is used as a pause before the word is completed. Some

words may sound "cut off" if a pause is not used. The pause helps to complete the previous sound or helps to begin the first sound. You will have to listen for these subtle differences. Although this word appears to be produced by the same number of phonemes as letters, there is no connection. Phonemes are not associated with letters but with sounds. For example, the word "OOPS" is composed of eight phonemes.

```
8108- 03 37 37 25 25 1F 1F 3F
```

To achieve the correct pronunciation, the phoneme code #$37 (U1) is repeated twice followed by two #$25 (P) and two #$1F (S). Again, the pause is used in the beginning to achieve the correct sound and the stop code is used to end the word.

To produce a phrase, words are strung together. The no-sound codes can be used to separate the words and maintain rhythm. The number of no-sound codes between words depends upon your preference. Generally a pause should also be used to separate two hard or plosive sounds.

LIMITED WARRANTY

Sweet Micro Systems warrants, to the original purchaser only, that this product shall be free from defects in materials and faulty workmanship under normal use and service for a period of ninety (90) days from the date of purchase. Defects covered by this Warranty shall be corrected either by repair or replacement, at our option. In the event replacement is elected by Sweet Micro Systems, any replacement product shall be warranted under the terms of this warranty for the remainder, if any, of the original ninety (90) day period. Sweet Micro Systems' liability is limited to the cost of repair or replacement of any defective part or product and Sweet Micro Systems shall not under any circumstances be liable for special, incidental or consequential damages of any kind resulting from use or possession of this product. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

If this product should require service, please contact us for a Return Authorization Number. Sweet Micro Systems will assume no liabilities for unauthorized returns. Return it to Sweet Micro Systems, 150 Chestnut Street, Providence, RI 02903, postage prepaid.

THE ABOVE WARRANTIES FOR GOODS ARE IN LIEU OF ALL WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND OF ANY OTHER WARRANTY OBLIGATION ON THE PART OF SWEET MICRO SYSTEMS. Some states do not allow the limitations on how long an implied warranty lasts so the above limitation may not apply to you.

This Warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

RRANTY REGISTRATION CARD: Please fill out this card and drop the mail. No stamp or envelope required.

me _____

dress _____

_____ State _____ Zip _____

duct Name _____ Date Purchased _____

ler _____

dress _____

_____ State _____ Zip _____

se specify your system configuration: _____

se indicate the primary uses for your computer:
e ☐ Business ☐ Education ☐ Other ☐

did you hear about Sweet Micro Systems' products?

puter Store ☐ Magazine: Byte ☐
t Mail ☐ Creative Computing ☐
puter Show ☐ Personal Computing ☐
onal/Friend ☐ Infoworld ☐
er ☐ Other ☐

t is your age?
er 18 ☐ 18-24 ☐ 25-29 ☐ 30-39 ☐ over 40 ☐

much is your individual ☐ or family ☐ income?
er $10,000 ☐ $10,000-$20,000 ☐ $20,000-$30,000 ☐
000-$40,000 ☐ over $40,000 ☐

QUICK REFERENCE CARD

Table 1 Phoneme Chart

Phoneme Code	Symbol	Duration (ms)	Example Word	Phoneme Code	Symbol	Duration (ms)	Example Word
00	EH3	59	JACKET	20	A	185	DAY
01	EH2	71	ENLIST	21	AY	65	DAY
02	EH1	121	HEAVY	22	Y1	80	YARD
03	PA0	47	NO SOUND	23	UH3	47	MISSION
04	DT	47	BUTTER	24	AH	250	MOP
05	A2	71	MADE	25	P	103	PAST
06	A1	103	MADE	26	O	185	COLD
07	ZH	90	AZURE	27	I	185	PIN
08	AH2	71	HONEST	28	U	185	MOVE
09	I3	55	INHIBIT	29	Y	103	ANY
0A	I2	80	INHIBIT	2A	T	71	TAP
0B	I1	121	INHIBIT	2B	R	90	RED
0C	M	103	MAT	2C	E	185	MEET
0D	N	80	SUN	2D	W	80	WIN
0E	B	71	BAG	2E	AE	185	DAD
0F	V	71	VAN	2F	AE1	103	AFTER
10	CH*	71	CHIP	30	AW2	90	SALTY
11	SH	121	SHOP	31	UH2	71	ABOUT
12	Z	71	ZOO	32	UH1	103	UNCLE
13	AW1	146	LAWFUL	33	UH	185	CUP
14	NG	121	THING	34	O2	80	FOR
15	AH1	146	FATHER	35	O1	121	ABOARD
16	OO1	103	LOOKING	36	IU	59	YOU
17	OO	185	BOOK	37	U1	90	YOU
18	L	103	LAND	38	THV	80	THE
19	K	80	TRICK	39	TH	71	THIN
1A	J*	47	JUDGE	3A	ER	146	BIRD
1B	H	71	HELLO	3B	EH	185	GET
1C	G	71	GET	3C	E1	121	BE
1D	F	103	FAST	3D	AW	250	CALL
1E	D	55	PAID	3E	PA1	185	NO SOUND
1F	S	90	PASS	3F	STOP	47	NO SOUND

/T/ must precede /CH/ to produce CH sound
/D/ must precede /J/ to produce J sound

Figure 1-3 PSG Registers

REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period				5-BIT Period Control				
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R8	Channel A Amplitude				M	L3	L2	L1	L0
R9	Channel B Amplitude				M	L3	L2	L1	L0
R10	Channel C Amplitude				M	L3	L2	L1	L0
R11	Envelope Period	8-BIT Fine Tune E							
R12		8-BIT Coarse Tune E							
R13	Envelope Shape/Cycle					CONT	ATT	ALT	HOLD
R14	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R15	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							