RAY TESKE

AY-3-8910/8912 PROGRAMMABLE SOUND GENERATO DATA MANUAL

ARCHITECTURE

OPERATION

INTERFACING

MUSIC GENERATION

SOUND EFFECTS GENERATION

ELECTRICAL SPECIFICATIONS

TABLE OF CONTENTS—PAGE 2

FEBRUARY 1979

Table of Contents

List of Illustrations

List of Illustrations (cont.)

1 INTRODUCTION

It is apparent that any microprocessor is capable of producing acceptable sounds with only a transducer if the processor has no other tasks to perform while the sound is sustained. In real world microprocessor use, however, video games need refreshing, keyboards need scanning, etc. For example, in order to produce a single channel of ninth octave C (8372 Hz) the signal needs attention every sixty microseconds. Software required to produce this simple effect and still perform other activities would in the least be very complex if not impossible. In the extreme, random noise requires periodic attention even more frequently.

This need for software-produced sounds without the constant attention of the processor is now satisfied with the availability of the General Instrument AY-3-8910 and AY-3-8912 Programmable Sound Generators.

1.1 Description

The AY-3-8910/8912 Programmable Sound Generator (PSG) is a Large Scale Integrated Circuit which can produce a wide variety of complex sounds under software control. The AY-3-8910/8912 is manufactured in GI's N-Channel Ion Implant Process. Operation requires a single 5V power supply, a TTL compatible clock, and a microprocessor controller such as the GI 16-bit CP1600/1610 or one of GI's PIC 1650 series of 8-bit microcomputers.

The PSG is easily interfaced to any bus oriented system. Its flexibility makes it useful in applications such as music synthesis, sound effects generation, audible alarms, tone signalling and FSK modems. The analog sound outputs can each provide 4 bits of logarithmic digital to analog conversion, greatly enhancing the dynamic range of the sounds produced.

In order to perform sound effects while allowing the processor to continue its other tasks, the PSG can continue to produce sound after the initial commands have been given by the control processor. The fact that realistic sound production often involves more than one effect is satisfied by the three independently controllable channels available in the PSG.

All of the circuit control signals are digital in nature and intended to be provided directly by a microprocessor/microcomputer. This means that one PSG can produce the full range of required sounds with no change in external circuitry. Since the frequency response of the PSG ranges from sub-audible at its lowest frequency to post-audible at its highest frequency, there are few sounds which are beyond reproduction with only the simplest electrical connections.

Since most applications of a microprocessor/PSG system would also require interfacing between the outside world and the microprocessor, this facility has been designed into the PSG. The AY-3-8910 has two general purpose 8-bit I/O ports and is supplied in a 40 lead package; the AY-3-8912 has one port and 28 leads.

1.2 Features

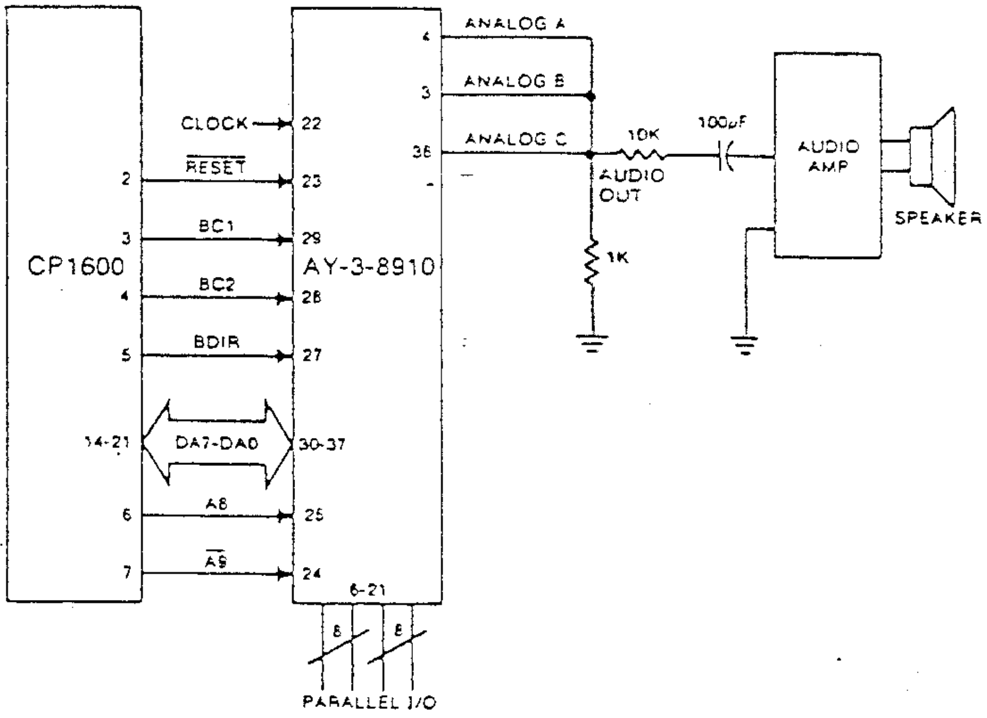
- ☐ Full software control of sound generation.
- ☐ Interfaces to most 8-bit and 16-bit microprocessors.
- ☐ Three independently programmed analog outputs.
- ☐ Two 8-bit general purpose I/O ports (AY-3-8910).
- ☐ One 8-bit general purpose I/O port (AY-3-8912).
- ☐ Single +5 Volt Supply.

1.3 Scope

This Data Manual is intended to introduce the techniques needed to cause the AY-3-8910/8912 Programmable Sound Generator to perform in its intended fashion. All of the programs, programming, and hardware designs have been tested to ensure that the methods are practical rather than purely theoretical.

Although the techniques described will produce powerful results, the range of sounds to be synthesized is so vast and the PSG capabilities so varied that this guide should be viewed merely as an introduction to the applications possibilities of the PSG.

Fig. 1 TYPICAL SYSTEM DIAGRAM



2 ARCHITECTURE

The AY-3-8910/8912 is a register oriented Programmable Sound Generator (PSG). Communication between the processor and the PSG is based on the concept of memory-mapped I/O. Control commands are issued to the PSG by writing to 16 memory-mapped registers. Each of the 16 registers within the PSG is also readable so that the microprocessor can determine, as necessary, present states or stored data values.

All functions of the PSG are controlled through its 16 registers which once programmed, generate and sustain the sounds, thus freeing the system processor for other tasks.

2.1 Basic Functional Blocks

An internal block diagram of the PSG showing the various functional blocks and data flow is shown in Fig. 2.

2.1.1 REGISTER ARRAY

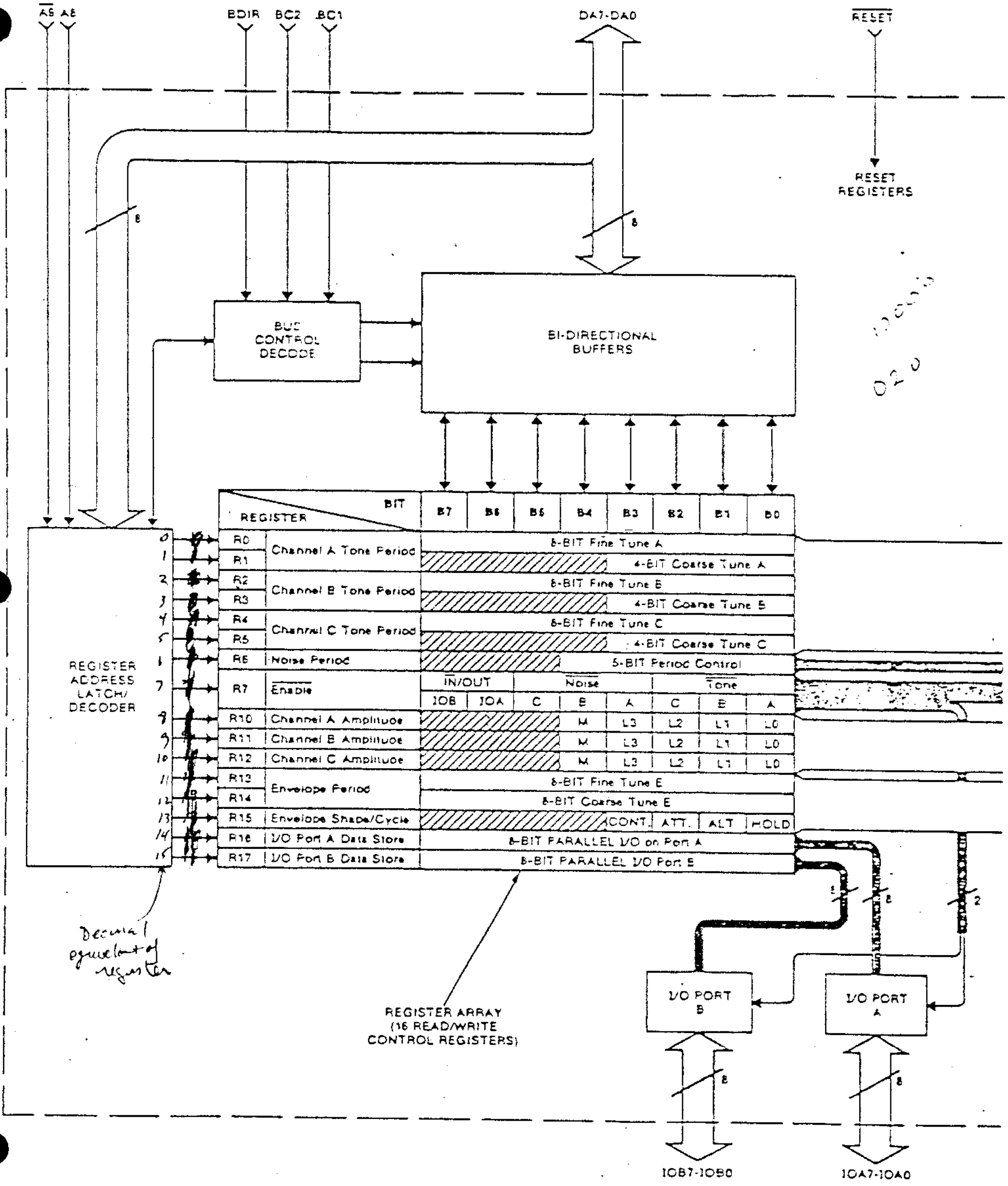
The principal element of the PSG is the array of 16 read/write control registers. These 16 registers look to the CPU as a block of memory and as such occupy a 16 word block out of 1,024 possible addresses. The 10 address bits (8 bits on the common data/address bus, and 2 separate address bits A8 and $\overline{A9}$) are decoded as follows:

* $\overline{A9}$	A8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
0	1	0	0	0	0	0	0	0	0
				THRU					
0	1	0	0	0	0	1	1	1	1
HIGH ORDER (Chip Select)						LOW ORDER (Register #)			

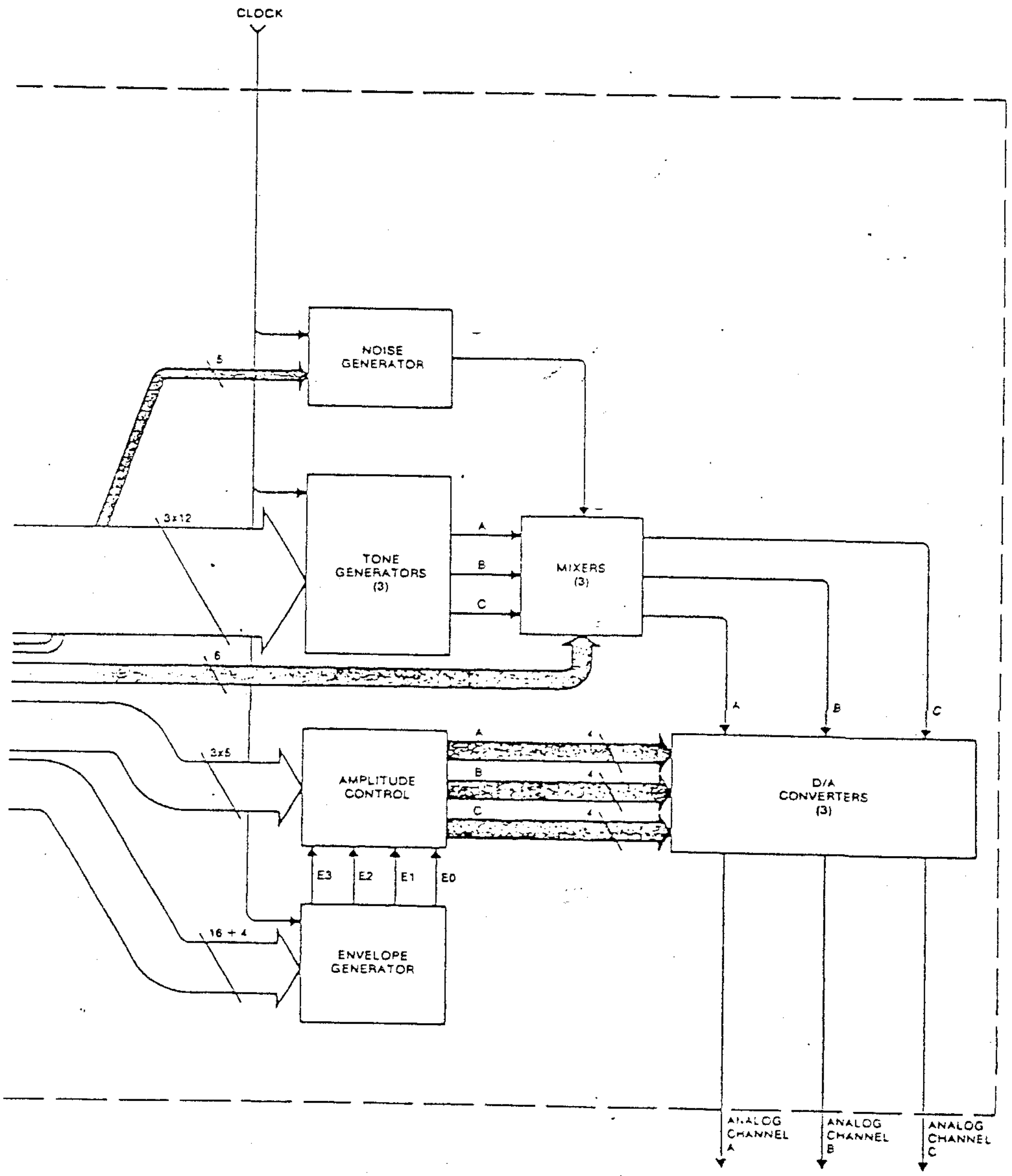
* $\overline{A9}$ is not provided on the AY-3-8912.

The four low order address bits select one of the 16 registers (R0--$R17_8$). The six high order address bits function as "chip selects" to control the tri-state bidirectional buffers (when the high order address bits are "incorrect", the bidirectional buffers are forced to a high impedance state). High order address bits $\overline{A9}$ A8 are fixed in the PSG design to recognize a 01 code; high order address bits DA7--DA4 may be mask-programmed to any 4-bit code by a special order factory mask modification. Unless otherwise specified, address bits DA7--DA4 are programmed to recognize only a 0000 code. A valid high order address latches the register address (the low order 4 bits) in the Register Address Latch/Decoder block. A latched address will remain valid until the receipt of a new address, enabling multiple reads and writes of the same register contents without the need for redundant re-addressing.

Fig. 2 PSG BLOCK DIAGRAM



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	$\overline{\text{Enable}}$	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R11	Channel B Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R12	Channel C Amplitude	▨	▨	▨	M	L3	L2	L1	L0
R13	Envelope Period	8-BIT Fine Tune E							
R14		8-BIT Coarse Tune E							
R15	Envelope Shape/Cycle	▨	▨	▨	▨	CONT.	ATT.	ALT	HOLD
R16	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R17	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							


CLOCK
NOISE GENERATOR
5
TONE GENERATORS (3)
3x12
A
B
C
MIXERS (3)
6
AMPLITUDE CONTROL
3x5
A
B
C
4
4
4
D/A CONVERTERS (3)
A
B
C
E3
E2
E1
E0
ENVELOPE GENERATOR
16 + 4
ANALOG CHANNEL A
ANALOG CHANNEL B
ANALOG CHANNEL C

2.1 Basic Functional Blocks (cont.)

Conditioning of the Register Address Latch/Decoder and the Bidirectional Buffers to recognize the bus function required (inactive, latch address, write data, or read data) is accomplished by the Bus Control Decode block.

The function of each of the 16 PSG registers and the data flow of each register's contents are shown in context in Fig. 2 and explained in detail in Section 3, "Operation". For reference purposes, the Register Array details are reproduced in Fig. 3.

2.1.2 SOUND GENERATING BLOCKS

The basic blocks in the PSG which produce the programmed sounds include:

Tone Generators	produce the basic square wave tone frequencies for each channel (A,B,C)
Noise Generator	produces a frequency modulated pseudo random pulse width square wave output.
Mixers	combine the outputs of the Tone Generators and the Noise Generator: One for each channel (A,B,C).
Amplitude Control	provides the D/A Converters with either a fixed or variable amplitude pattern. The fixed amplitude is under direct CPU control; the variable amplitude is accomplished by using the output of the Envelope Generator.
Envelope Generator	produces an envelope pattern which can be used to amplitude modulate the output of each Mixer.
D/A Converters	the three D/A Converters each produce up to a *16 level output signal as determined by the* Amplitude Control.

2.1.3 I/O PORTS

Two additional blocks are shown in the PSG Block Diagram which have nothing directly to do with the production of sound—these are the two I/O Ports (A and B). Since virtually all uses of microprocessor-based sound would require interfacing between the outside world and the processor, this facility has been included in the PSG. Data to/from the CPU bus may be read/written to either of two 8-bit I/O Ports without affecting any other function of the PSG. The I/O Ports are TTL-compatible and are provided with internal pull-ups on each pin. Both Ports are available on the AY-3-8910; only I/O Port A is available on the AY-3-8912.

2.2 Pin Assignments

The AY-3-8910 is supplied in a 40 lead dual in-line package with the pin assignments as shown in Fig. 4. The AY-3-8912 is supplied in a 28 lead dual in-line package with the pin assignments as shown in Fig. 5.

Fig. 4 AY-3-8910 PIN ASSIGNMENTS

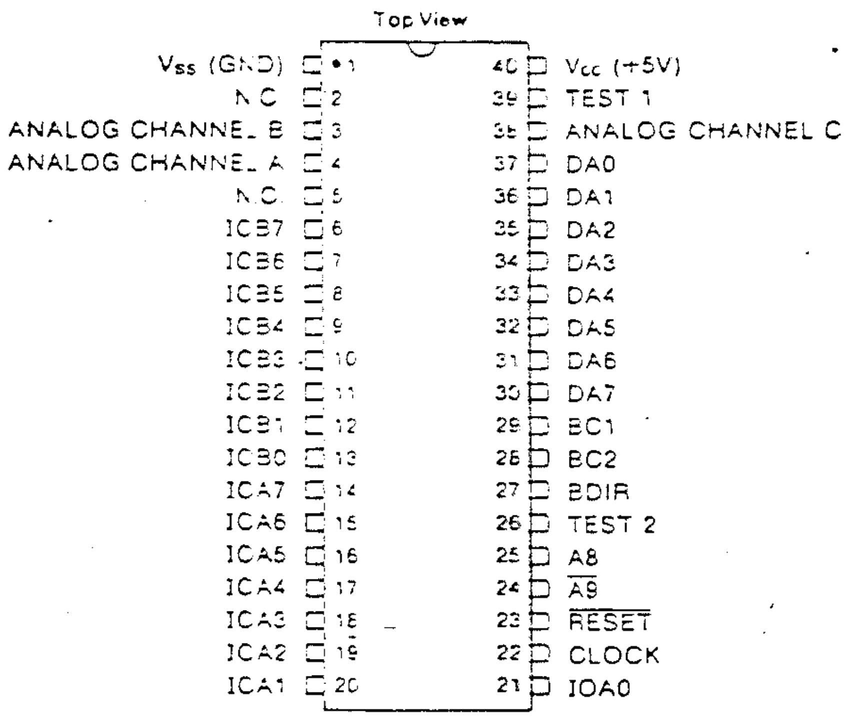
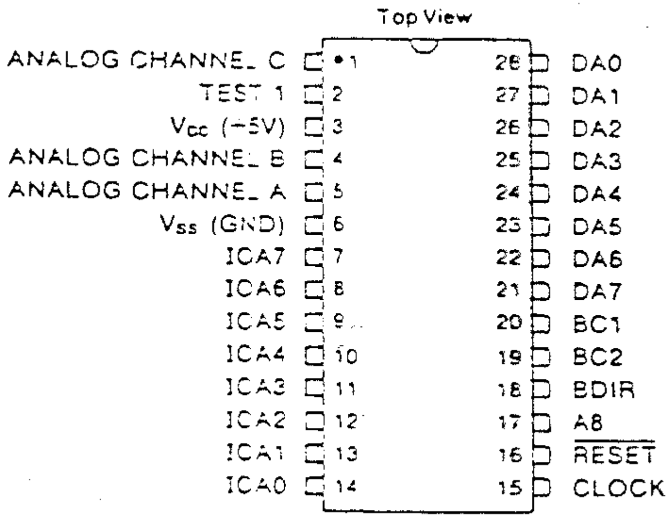


Fig. 5 AY-3-8912 PIN ASSIGNMENTS



2.3 Pin Functions

DA7--DA0 (input/output/high impedance): pins 30--37 (AY-3-8910)
Data/Address 7--0: pins 21--28 (AY-3-8912)

These 8 lines comprise the 8-bit bidirectional bus used by the microprocessor to send both data and addresses to the PSG and to receive data from the PSG. In the data mode, DA7--DA0 correspond to Register Array bits B7—B0. In the address mode, DA3--DA0 select the register # (0--17_8) and DA7—DA4 in conjunction with address inputs $\overline{A9}$ and A8 form the high order address (chip select).

A8 (input): pin 25 (AY-3-8910)
pin 17 (AY-3-8912)
$\overline{A9}$ (input): pin 24 (AY-3-8910)
(not provided on AY-3-8912)

Address 9, Address 8

These "extra" address bits are made available to enable the positioning of the PSG (assigning a 16 word memory space) in a total 1,024 word memory area rather than in a 256 word memory area as defined by address bits DA7--DA0 alone. If the memory size does not require the use of these extra address lines they may be left unconnected as each is provided with either an on-chip pull down ($\overline{A9}$) or pull-up (A8) resistor. In "noisy" environments, however, it is recommended that $\overline{A9}$ and A8 be tied to an external ground and +5V, respectively, if they are not to be used.

$\overline{\text{RESET}}$ (input): pin 23 (AY-3-8910)
pin 16 (AY-3-8912)

For initialization/power-on purposes, applying a logic "0" (ground) to the $\overline{\text{Reset}}$ pin will reset all registers to "0". The $\overline{\text{Reset}}$ pin is provided with an on-chip pull-up resistor.

CLOCK (input): pin 22 (AY-3-8910)
pin 15 (AY-3-8912)

This TTL-compatible input supplies the timing reference for the Tone, Noise and Envelope Generators.

BDIR, BC2, BC1 (inputs): pins 27,28,29 (AY-3-8910)
pins 18,19,20 (AY-3-8912)

Bus DIRection, Bus Control 2,1

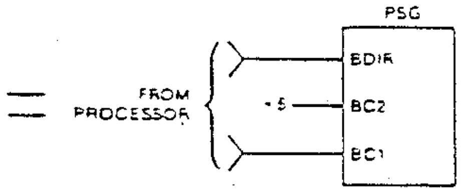
These bus control signals are generated directly by GI's CP1600 series of microprocessors to control all external and internal bus operations in the PSG. When using a processor other than the CP1600, these signals can be provided either by comparable bus signals or by simulating the signals on I/O lines of the processor. The PSG decodes these signals as illustrated in the following:

2.3 Pin Functions (cont.)

BDIR	BC2	BC1	CP1600 FUNCTION	PSG FUNCTION
0	0	0	NACT	INACTIVE. See 010 (IAB) below
0	0	1	ADAR	LATCH ADDRESS. See 111 (INTAK) below.
0	1	0	IAB	INACTIVE. The PSG/CPU bus is inactive. DA7--DA0 are in a high impedance state.
0	1	1	DTB	READ FROM PSG. This signal causes the contents of the register which is currently addressed to appear on the PSG/CPU bus. DA7—DA0 are in the output mode.
1	0	0	BAR	LATCH ADDRESS. See 111 (INTAK) below.
1	0	1	DW	INACTIVE. See 010 (IAB) above.
1	1	0	DWS	WRITE TO PSG. This signal indicates that the bus contains register data which should be latched into the currently addressed register. DA7—DA0 are in the input mode.
1	1	1	INTAK	LATCH ADDRESS. This signal indicates that the bus contains a register address which should be latched in the PSG. DA7—DA0 are in the input mode.

While interfacing to a processor other than the CP1600 would simply require simulating the above decoding, the redundancies in the PSG functions vs. bus control signals can be used to advantage in that only four of the eight possible decoded bus functions are required by the PSG. This could simplify the programming of the bus control signals to the following, which would only require that the processor generate two bus control signals (BDIR and BC1, with BC2 tied to +5V):

BDIR	BC2	BC1	PSG FUNCTION
0	1	0	INACTIVE.
0	1	1	READ FROM PSG.
1	1	0	WRITE TO PSG.
1	1	1	LATCH ADDRESS.



ANALOG CHANNEL A, B, C (outputs): pins 4, 3, 38 (AY-3-8910)
pins 5, 4, 1 (AY-3-8912)

Each of these signals is the output of its corresponding D/A Converter, and provides an up to 1V peak-peak signal representing the complex sound waveshape generated by the PSG.

IOA7--IOA0 (input/output): pins 14--21 (AY-3-8910)
pins 7--14 (AY-3-8912)

IOB7--IOB0 (input/output): pins 6--13 (AY-3-8910)
(not provided on AY-3-8912)

Input/Output A7--A0, B7--B0

Each of these two parallel input/output ports provides 8 bits of parallel data to/from the PSG/CPU bus from/to any external devices connected to the IOA or IOB pins. Each pin is provided with an on-chip pull-up resistor, so that when in the "input" mode, all pins will read normally high. Therefore, the recommended method for scanning external switches, for example, would be to ground the input bit.

TEST 1: pin 39 (AY-3-8910)
pin 2 (AY-3-8912)
TEST 2: pin 26 (AY-3-8910)
(not connected on AY-3-8912)

These pins are for GI test purposes only and should be left open—do not use as tie-points.

V_{CC}: pin 40 (AY-3-8910)
pin 3 (AY-3-8912)

Nominal +5Volt power supply to the PSG.

V_{SS}: pin 1 (AY-3-8910)
pin 6 (AY-3-8912)

Ground reference for the PSG.

2.4 Bus Timing

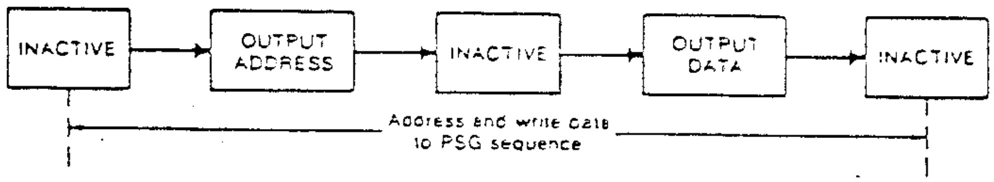
Since the PSG functions are controlled by commands from the system processor, the common data/address bus (DA7--DA0) requires definition as to its function at any particular time. This is accomplished by the processor issuing bus control signals, previously described, defining the state of the bus; the PSG then decodes these signals to perform the requested task.

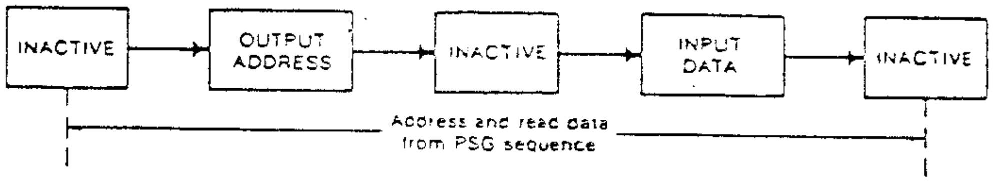
The conditioning of these bus control signals by the processor is the same as if the processor were interacting with RAM: (1) the processor outputs a memory address; and (2) the processor either outputs or inputs data to/from the memory. The "memory" in this case is the PSG's array of 16 read/write control registers.

The timing relationships in issuing the bus control signals relative to the data or address signals on the bus are reviewed in general in the following section, and in detail in Section 7, Electrical Specifications.

2.5 State Timing

While the state flow for many microprocessors can be somewhat involved for certain operations, the sequence of events necessary to control the PSG is simple and straightforward. Each of the three major state sequences (Latch Address, Write to PSG, and Read from PSG) consists of several operations (indicated below by rectangular blocks), defined by the pattern of bus control signals (BDIR, BC2, BC1).

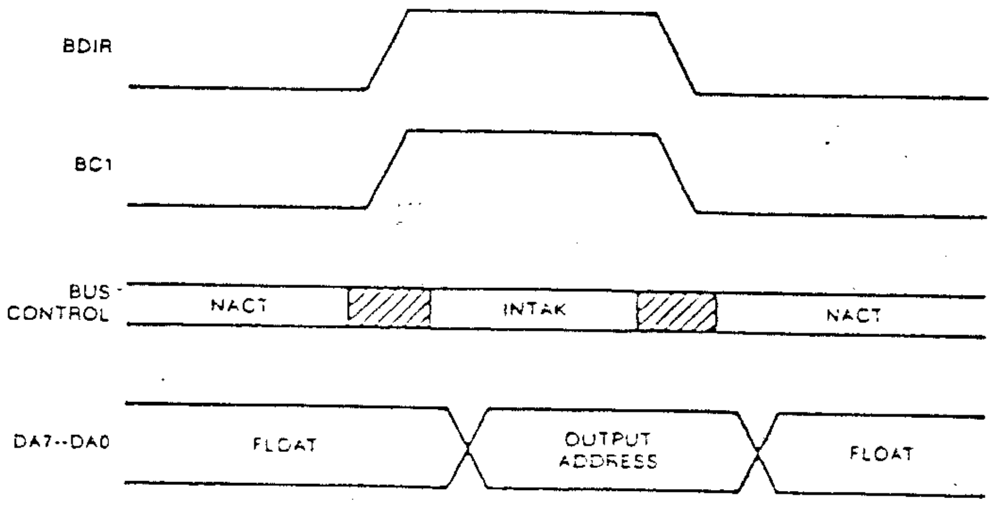




The functional operation and relative timing of the PSG control sequences are described in the following paragraphs (in all examples, BC2 has been assumed to be tied to logic "1", +5V).

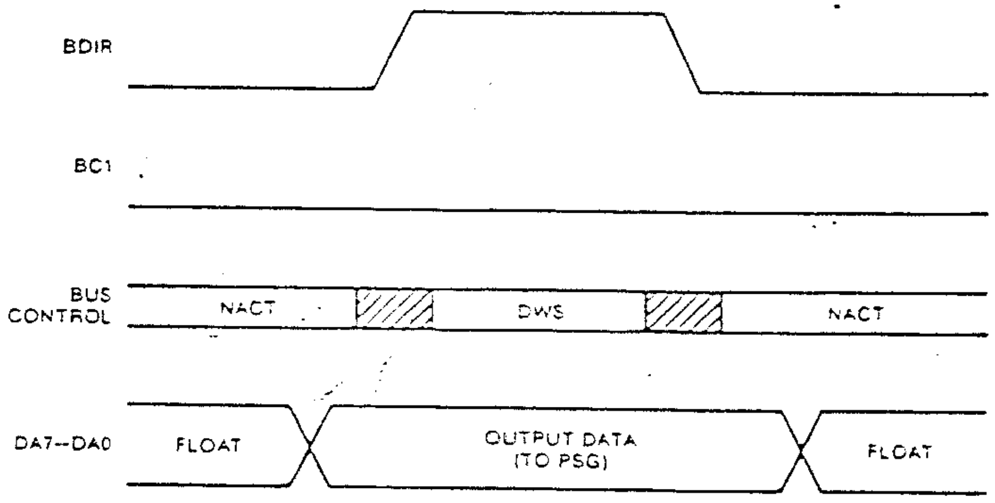
2.5.1 ADDRESS PSG REGISTER SEQUENCE

The "Latch Address" sequence is normally an integral part of the write or read sequences, but for simplicity is illustrated here as an individual sequence. Depending on the processor used the program sequence will normally require four principal microstates: (1) send NACT (inactive); (2) send INTAK (latch address); (3) put address on bus; (4) send NACT (inactive). [Note: within the timing constraints detailed in Section 7, steps (2) and (3) may be interchanged.]



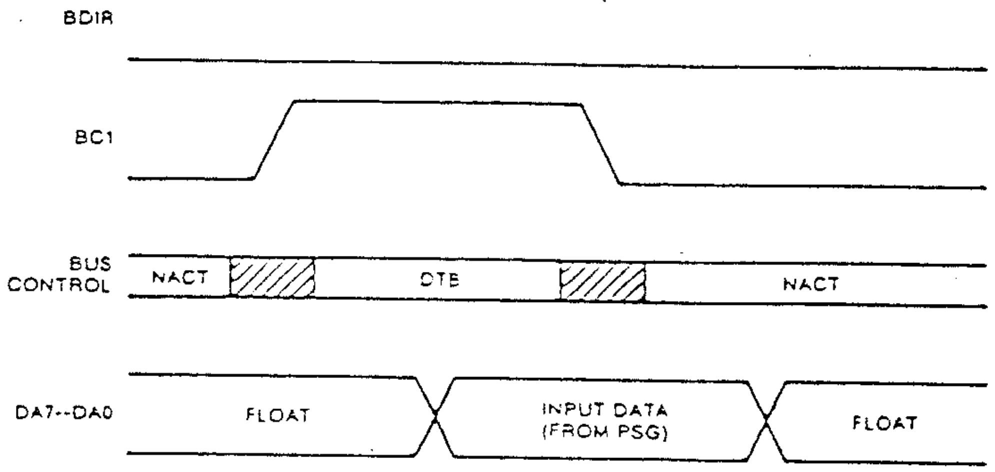
2.5.2 WRITE DATA TO PSG SEQUENCE

The "Write to PSG" sequence, which would normally follow immediately after an address sequence, requires four principal microstates: (1) send NACT (inactive); (2) put data on bus; (3) send DWS (write to PSG); (4) send NACT (inactive).



2.5.3 READ DATA FROM PSG SEQUENCE

As with the "Write to PSG" sequence, the "Read from PSG" sequence would also normally follow immediately after an address sequence. The four principal microstates of the read sequence are: (1) send NACT (inactive); (2) send DTB (read from PSG); (3) read data on bus; (4) send NACT (inactive).



2.5.4 WRITE TO/READ FROM I/O PORT SEQUENCE

Since the two I/O Ports (A and B) each have an 8-bit register assigned as a data store, writing to or reading from either port is identical to writing or reading to any other register. Hence, the state sequences are exactly the same as described in the preceding paragraphs.

3 OPERATION

Since all functions of the PSG are controlled by the host 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:

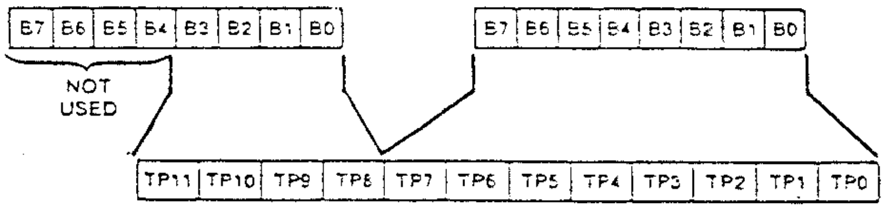
Section	Operation	Registers	Function
3.1	Tone Generator Control	R0--R5	Program tone periods.
3.2	Noise Generator Control	R6	Program noise period.
3.3	Mixer Control	R7	Enable tone and/or noise on selected channels.
3.4	Amplitude Control	R10--R12	Select "fixed" or "envelope-variable" amplitudes.
3.5	Envelope Generator Control	R13--R15	Program envelope period and select envelope pattern.

3.1 Tone Generator Control

(Registers R0, R1, 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

Note that the 12-bit value programmed in the combined Coarse and Fine Tune registers is a period value—the higher the value in the registers, the lower the resultant tone frequency.

Note also that due to the design technique used in the Tone Period count-down, the lowest period value is 000000000001 (divide by 1) and the highest period value is 111111111111 (divide by 4,095$_{10}$).

The equations describing the relationship between the desired output tone frequency and the input clock frequency and Tone Period value are:

(a) $f_T = \frac{f_{CLOCK}}{16TP_{10}}$ (b) $TP_{10} = 256CT_{10} + FT_{10}$

Where: f_T = desired tone frequency
f_{CLOCK} = input clock frequency
TP_{10} = decimal equivalent of the Tone Period bits TP11--TP0.
CT_{10} = decimal equivalent of the Coarse Tune register bits B3--B0 (TP11--TP8)
FT_{10} = decimal equivalent of the Fine Tune register bits B7--B0 (TP7--TP0)

From the above equations it can be seen that the tone frequency can range from a low of $\frac{f_{CLOCK}}{65,520}$ (wherein: TP_{10}=4,095$_{10}$) to a high of $\frac{f_{CLOCK}}{16}$ (wherein: TP_{10}=1). Using a 2 MHz input clock, for example, would produce a range of tone frequencies from 30.5 Hz to 125 kHz.

To calculate the values for the contents of the Tone Period Coarse and Fine Tune registers, given the input clock and the desired output tone frequencies, we simply rearrange the above equations, yielding:

(a) $TP_{10} = \frac{f_{CLOCK}}{16f_T}$ (b) $CT_{10} + \frac{FT_{10}}{256} = \frac{TP_{10}}{256}$

Example 1: f_T = 1kHz
f_{CLOCK} = 2MHz

$$TP_{10} = \frac{2\times10^6}{16(1\times10^3)} = 125$$

Substituting this result into equation (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{125}{256}$$

$\therefore CT_{10} = 0 \quad = 0000$ (B3--B0)
$FT_{10} = 125_{10} = 01111101$ (B7--B0)

Example 2: f_T = 100Hz
f_{CLOCK} = 2MHz

$$TP_{10} = \frac{2\times10^6}{16(1\times10^2)} = 1250$$

Substituting this result into equation (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{1250}{256} = 4 + \frac{226}{256}$$

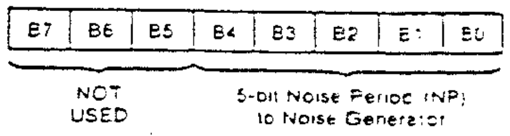
$\therefore CT_{10} = 4_{10} = 0100$ (B3--B0)
$FT_{10} = 226_{10} = 11100010$ (B7--B0)

3.2 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, 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:

Noise Period
Register R6



Note that the 5-bit value in R11 is a <u>period</u> value—the higher the value in the register, the lower the resultant noise frequency. Note also that, as with the Tone Period, the <u>lowest</u> period value is 00001 (divide by 1); the <u>highest</u> period value is 11111 (divide by 31_{10}).

The noise frequency equation is:

$$f_N = \frac{f_{CLOCK}}{16\ NP_{10}}$$

Where: f_N = desired noise frequency
f_{CLOCK} = input clock frequency
NP_{10} = decimal equivalent of the Noise Period register bits B4—B0.

From the above equation it can be seen that the noise frequency can range from a low of $\frac{f_{CLOCK}}{496}$ (wherein: $NP_{10} = 31_{10}$) to a high of $\frac{f_{CLOCK}}{16}$ (wherein: $NP_{10} = 1$). Using a 2 MHz input clock, for example, would produce a range of noise frequencies from 4 kHz to 125 kHz.

To calculate the value for the contents of the Noise Period register, given the input clock and the desired output noise frequencies, we simply rearrange the above equation, yielding:

$$NP_{10} = \frac{f_{CLOCK}}{16\ f_N}$$

3.3 Mixer Control-I/O Enable

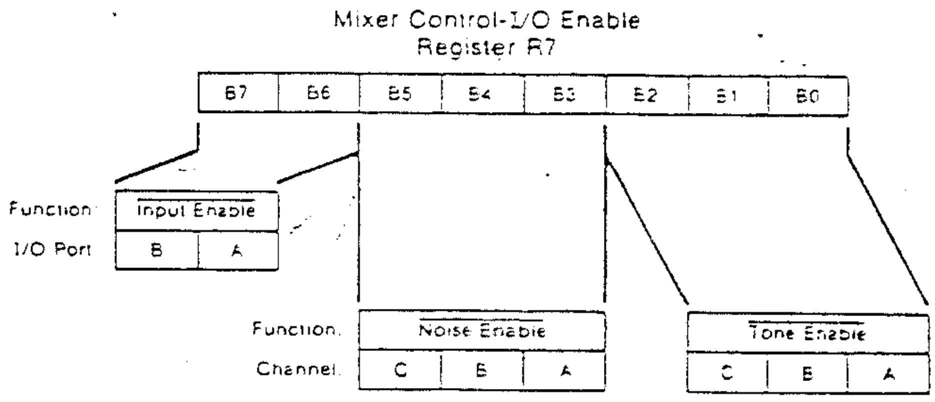
(Register R7)

Register 7 is a multi-function $\overline{\text{Enable}}$ register which controls the three Noise/Tone Mixers and the two general purpose I/O Ports.

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.

The direction (input or output) of the two general purpose I/O Ports (IOA and IOB) is determined by the state of bits B7 and B6 of R7.

These functions are illustrated in the following:



Noise Enable Truth Table:

R7 Bits			Noise Enabled on Channel		
B5	B4	B3			
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

Tone Enable Truth Table:

R7 Bits			Tone Enabled on Channel		
B2	B1	B0			
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

I/O Port Truth Table:

R7 Bits		I/O Port Status	
B7	B6	IOB	IOA
0	0	Input	Input
0	1	Input	Output
1	0	Output	Input
1	1	Output	Output

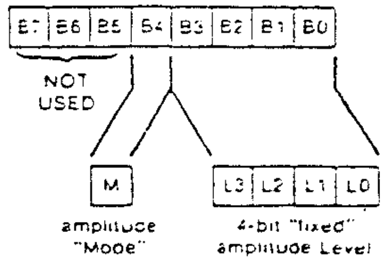
NOTE: Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register, R10, R11, or R12 (see Section 3.4).

3.4 Amplitude Control

(Registers R10, R11, R12)

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 R10, R11, and R12 as illustrated in the following:

Amplitude Control Register #	Channel
R10	A
R11	B
R12	C



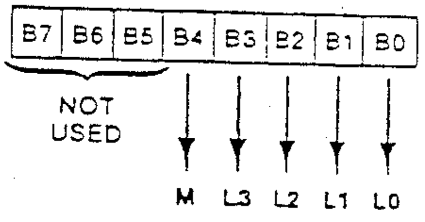
The amplitude "mode" (bit M) selects either fixed level amplitude (M=0) or variable level amplitude (M=1). It follows then that bits L3—L0, defining the value of a "fixed" level amplitude, are only active when M=0. When fixed level amplitude is selected, it is "fixed" only in the sense that the amplitude level is under the direct control of the system processor (via bits D3--D0). Varying the amplitude when in this "fixed" amplitude mode requires in each instance the direct intervention of the system processor via an address latch/write data sequence to modify the D3—D0 data.

When M=1 (select "variable" level amplitudes), the amplitude of each channel is determined by the envelope pattern as defined by the Envelope Generator's 4-bit output E3 E2 E1 E0.

The amplitude "mode" (bit M) can also be thought of as an "envelope enable" bit; i.e., when M=0 the envelope is not used, and when M=1 the envelope is enabled. (A full description of the Envelope Generator function follows in Section 3.5).

The full chart describing all combinations of the 5-bit Amplitude Control is as follows:

Amplitude Control Register #	Channel
R10	A
R11	B
R12	C



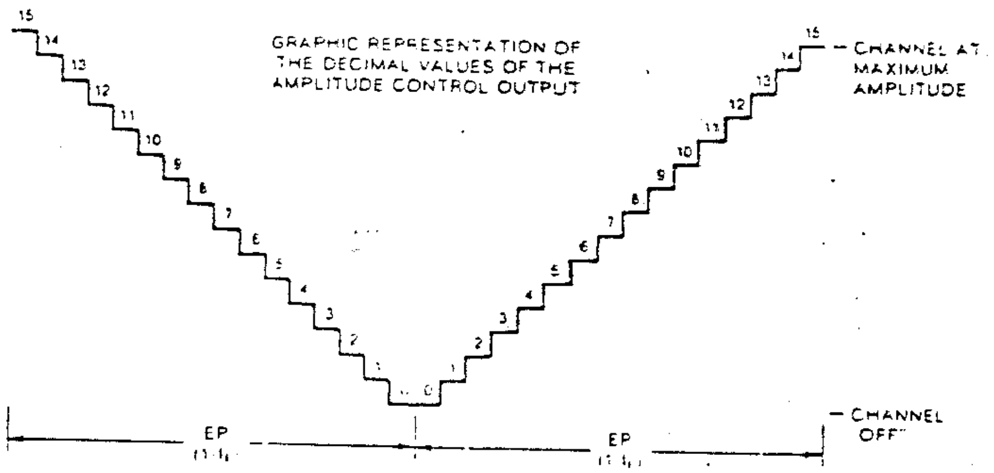
M	L3	L2	L1	L0	Amplitude Control Output				
0	0	0	0	0	*0	0	0	0	The amplitude is fixed at 1 of 16 levels as determined by L3 L2 L1 L0.
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
0	1	1	1	1	1	1	1	1	
1	X	X	X	X	E3	E2	E1	E0	The amplitude is variable at 16 levels as determined by the output of the Envelope Generator.

(X=Don't Care)

*The all zeroes code is used to turn a channel "off".

Fig. 6 graphically illustrates a selection of variable level (envelope-controlled) amplitude where the 16 levels directly reflect the output of the Envelope Generator. A fixed level amplitude would correspond to only one of the levels shown, with the level directly determined by the decimal equivalent of bits L3 L2 L1 L0.

Fig. 6 VARIABLE AMPLITUDE CONTROL (M=1)


GRAPHIC REPRESENTATION OF THE DECIMAL VALUES OF THE AMPLITUDE CONTROL OUTPUT
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
CHANNEL AT MAXIMUM AMPLITUDE
CHANNEL OFF
EP
EP

To calculate the values for the contents of the Envelope Period Coarse and Fine Tune registers, given the input clock and the desired envelope frequencies, we rearrange the above equations, yielding:

(a) $EP_{10} = \frac{f_{CLOCK}}{256 f_E}$ (b) $CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$

Example: $f_E = 0.5$ Hz
$f_{CLOCK} = 2$ MHz

$$EP_{10} = \frac{2 \times 10^6}{256(0.5)} = 15{,}625$$

Substituting this result into equation (b):

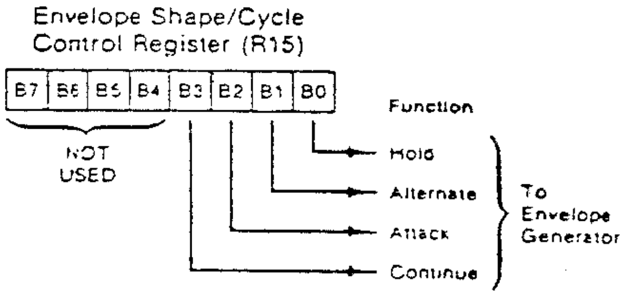
$$CT_{10} + \frac{FT_{10}}{256} = \frac{15.625}{256} = 61 + \frac{9}{256}$$

$$CT_{10} = 61_{10} = 00111101 \text{ (B7--B0)}$$
$$FT_{10} = 9_{10} = 00001001 \text{ (B7--B0)}$$

3.5.2 ENVELOPE SHAPE/CYCLE CONTROL (Register R15)

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 E1 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 R15. Each of these 4 bits controls a function in the envelope generator, as illustrated in the following:

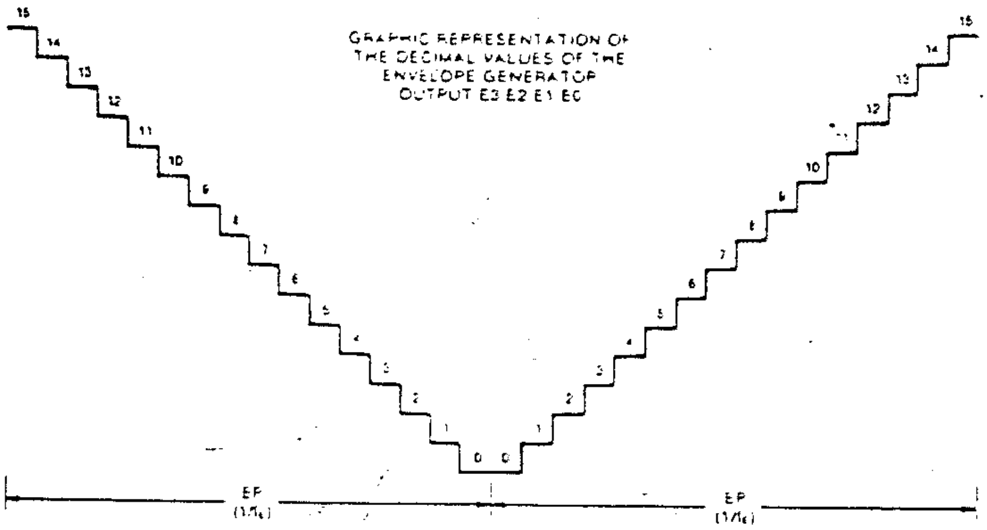


The definition of each function is as follows:

Hold	when set to logic "1", limits the envelope to one cycle, holding the last count of the envelope counter (E3--E0=0000 or 1111, depending on whether the envelope counter was in a count-down or count-up mode, respectively).
Alternate	when set to logic "1", the envelope counter reverses count direction (up-down) after each cycle.

NOTE: When both the Hold bit and the Alternate bit are ones, the envelope counter is reset to its initial count before holding.

Fig. 8 DETAIL OF TWO CYCLES OF Fig. 7
(ref. waveform "1010" in Fig. 7)



GRAPHIC REPRESENTATION OF THE DECIMAL VALUES OF THE ENVELOPE GENERATOR OUTPUT E3 E2 E1 E0
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
EP
EP

3.6 I/O Port Data Store

(Registers R16, R17)

Registers R16 and R17 function as intermediate data storage registers between the PSG/CPU data bus (DA0--DA7) and the two I/O ports (IOA7--IOA0 and IOB7--IOB0). Both ports are available in the AY-3-8910; only I/O Port A is available in the AY-3-8912. Using registers R16 and R17 for the transfer of I/O data has no effect at all on sound generation.

To output data from the CPU bus to a peripheral device connected to I/O Port A would require only the following steps:

1. Latch address R7 (select $\overline{\text{Enable}}$ register)
2. Write data to PSG (setting B6 of R7 to "1")
3. Latch address R16 (select IOA register)
4. Write data to PSG (data to be output on I/O Port A)

To input data from I/O Port A to the CPU bus would require the following:

1. Latch address R7 (select $\overline{\text{Enable}}$ register)
2. Write data to PSG (setting B6 to R7 to "0")
3. Latch address R16 (select IOA register)
4. Read data from PSG (data from I/O Port A)

Note that once loaded with data in the output mode, the data will remain on the I/O port(s) until changed either by loading different data, by applying a reset (grounding the Reset pin), or by switching to the input mode.

Note also that when in the input mode, the contents of registers R16 and/or R17 will follow the signals applied to the I/O port(s). However, transfer of this data to the CPU bus requires a "read" operation as described above.

3.7 D/A Converter Operation

Since the primary use of the PSG is to produce sound for the highly imperfect amplitude detection mechanism of the human ear, the D/A conversion is performed in logarithmic steps with a normalized voltage range of from 0 to 1 Volt. The specific amplitude control of each of the three D/A Converters is accomplished by the three sets of 4-bit outputs of the Amplitude Control block, while the Mixer outputs provide the base signal frequency (Noise and/or Tone).

Fig. 9 illustrates the D/A Converter output which would result if noise and tones were disabled and an envelope-controlled variable amplitude were selected.

Figs. 10 through 13 illustrate other typical output waveforms.

Fig. 9 D/A CONVERTER OUTPUT (ref. Fig. 6)

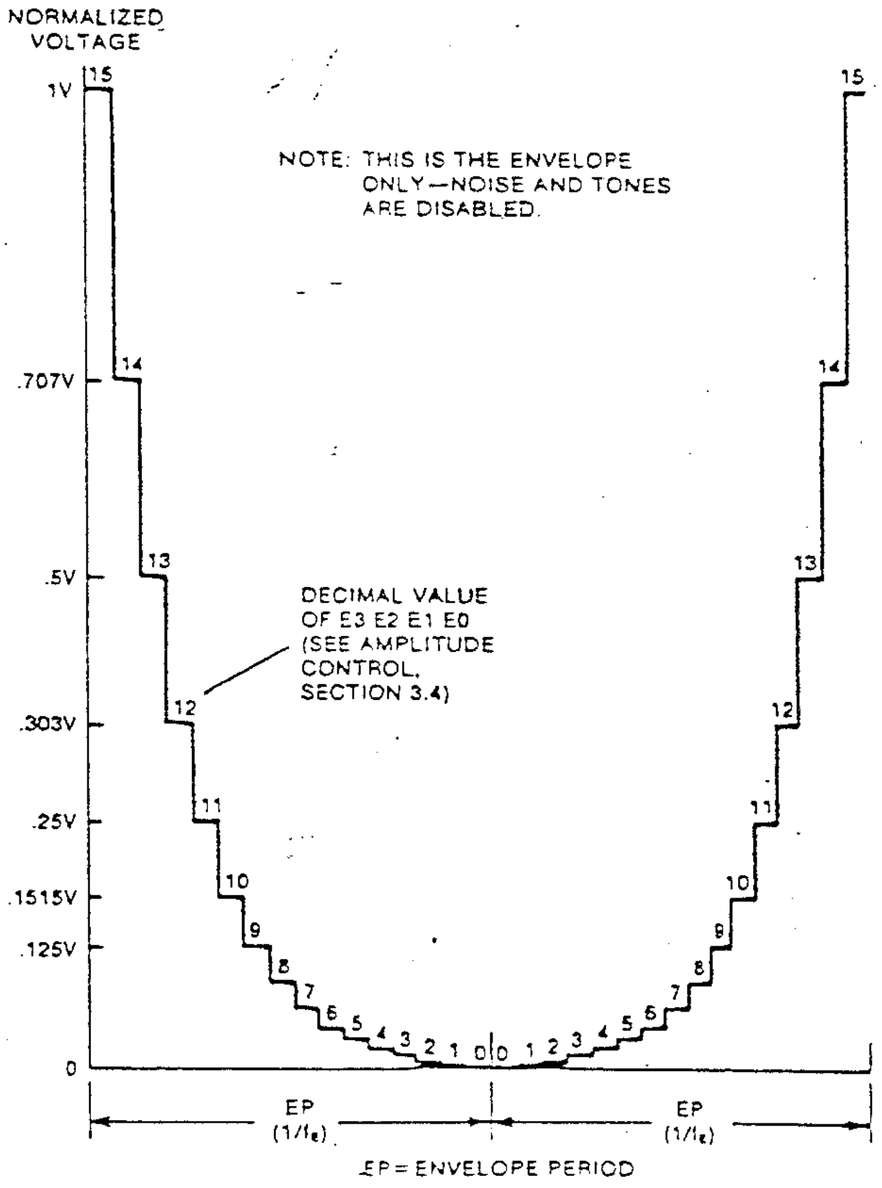
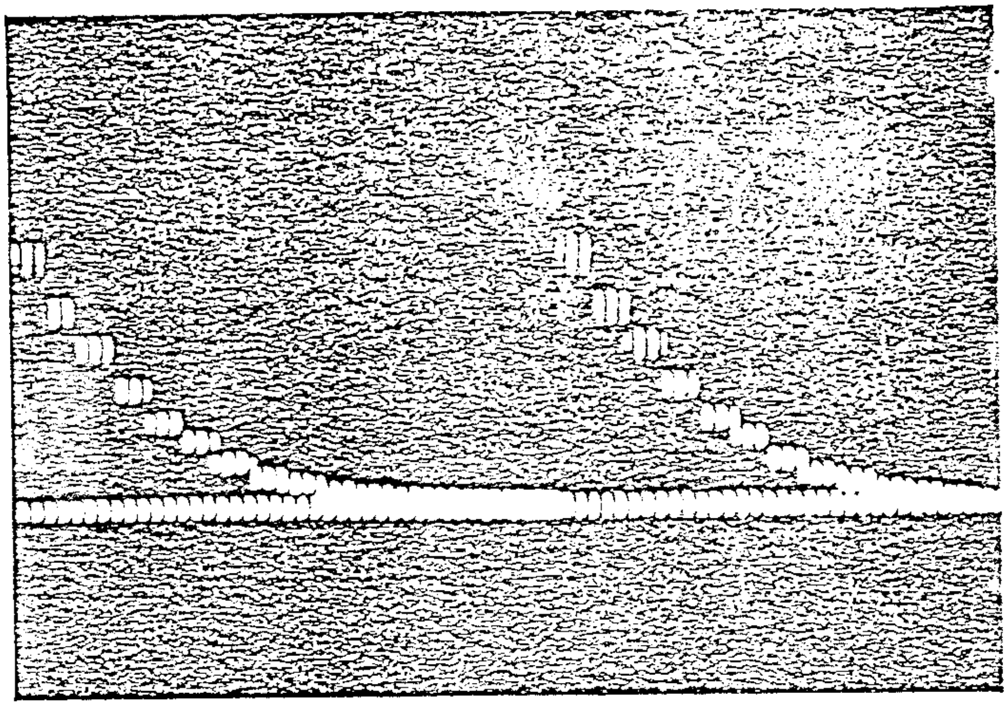


Fig. 10 SINGLE TONE WITH ENVELOPE SHAPE [illegible] 1100
(R0=14, R1=37, R7=76, R12=20, R15=10 [illegible] 0)



[illegible] TONE WITH ENVELOPE SHA[illegible] 1100
(R15=14 all other registers same as Fig [illegible]

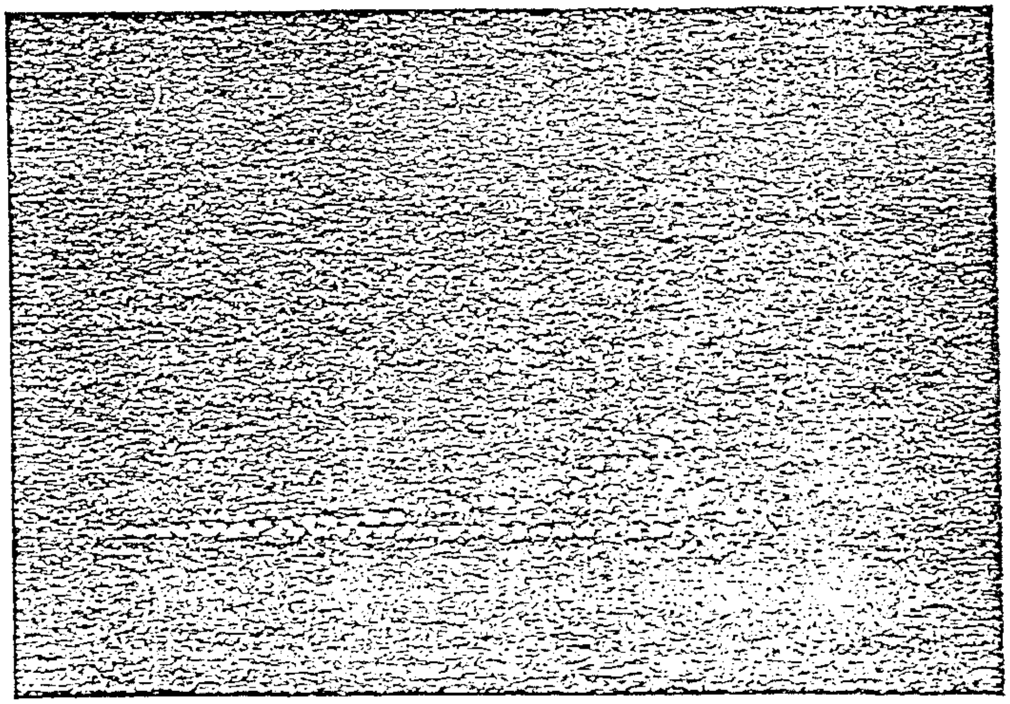


Fig. 12 SINGLE TONE WITH ENVELOPE SHAPE [illegible] 1010 (R15=12[illegible] all other registers same as Fig. [illegible]

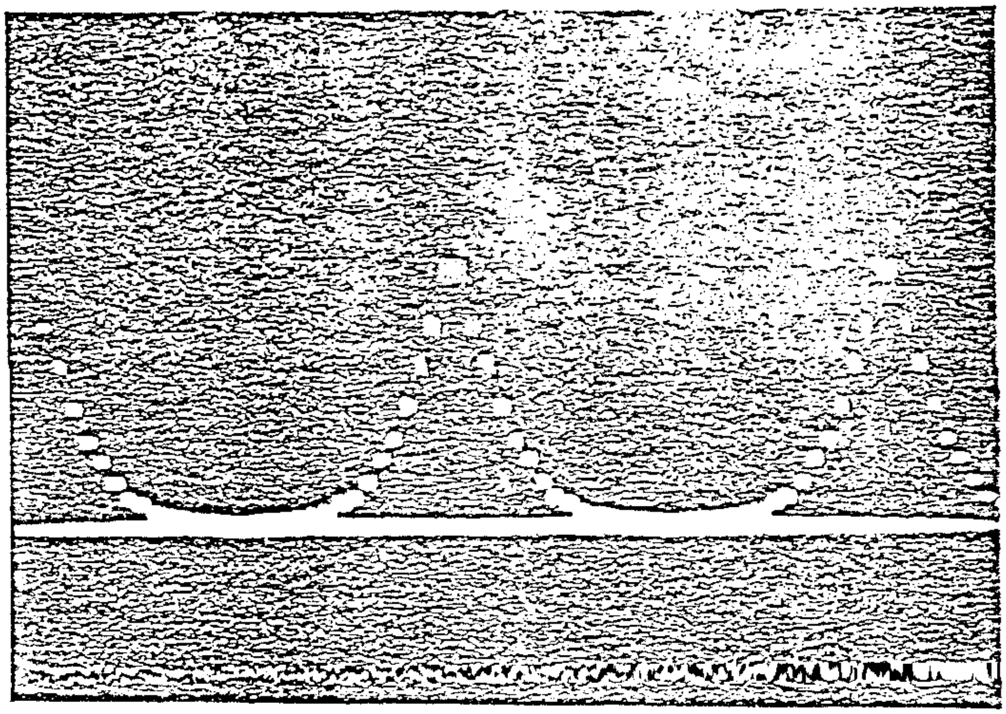
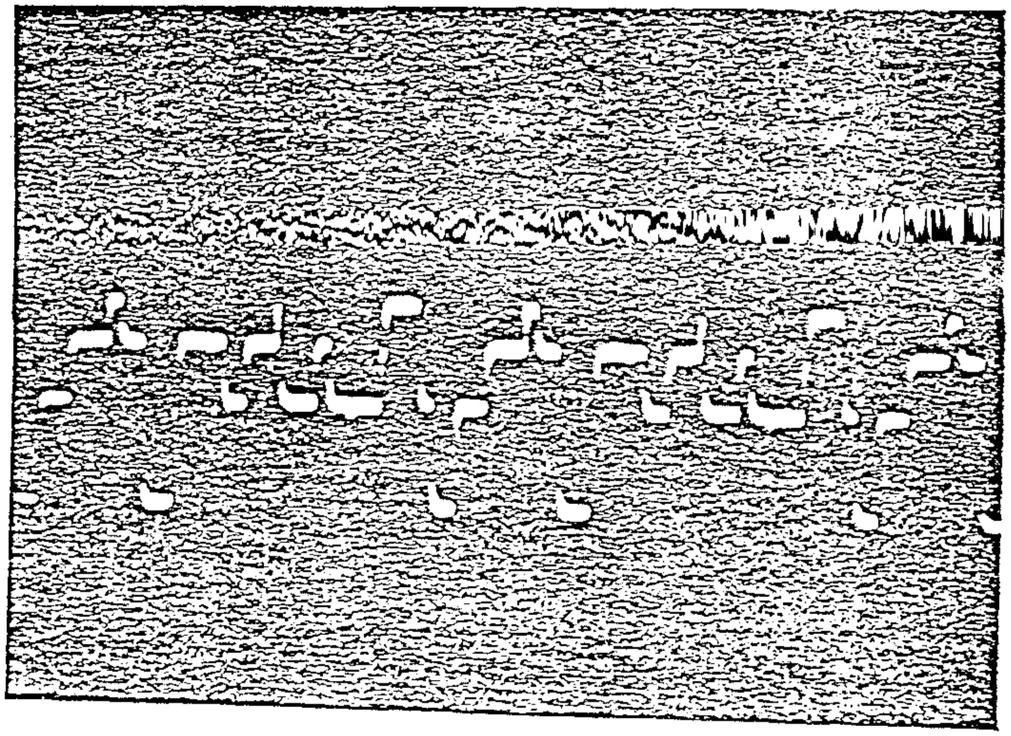


Fig. 13 MIXTURE OF THREE TONES WITH FIXED AMPLITUDES



4.9 Interfacing to the 8080 S100 Bus

The sample S100 bus design provides for reading and writing the PSG using only an 8080 "IN" or "OUT" instruction to the prop address. Another feature of the design is the provision for multiple PSG devices to be connected to a single bus. The system described is presently running two PSG's, one to each of two stereo channels.

As can be seen from the read and write routines in the illustrative program, the program overhead necessary to communicate with the PSG is minimal.

4.9.1 LATCH ADDRESS ROUTINE

```
PORTADDR EQU 80H ;ADDRESS TRANSFER PORT ADDRESS
PORTDATA EQU 81H ;DATA TRANSFER PORT ADDRESS
;
;THIS ROUTINE WILL TRANSFER THE CONTENTS OF
;8080 REGISTER C TO THE PSG ADDRESS REGISTER
PSGBAR     MOV    A,C ;GET C IN A FOR OUT
           OUT    PORTBAR ;SEND TO ADDRESS PORT
           RET
```

4.9.2 WRITE DATA ROUTINE

```
;
;
;ROUTINE TO WRITE THE CONTENTS OF 8080 REGISTER B
;TO THE PSG REGISTER SPECIFIED BY 8080 REGISTER C
;
PSGWRITE   CALL   PSGBAR ;GET ADDRESS LATCHED
           MOV    A,B, ;GET VALUE IN A FOR TRANSFER
           OUT    PORTDATA ;PUT TO PSG REGISTER
           RET
```

4.9.3 READ DATA ROUTINE

```
;
;
;ROUTINE TO READ THE PSG REGISTER SPECIFIED
;BY THE 8080 REGISTER C AND RETURN THE DATA
;IN 8080 REGISTER B
;
PSGREAD    CALL   PSGBAR
           IN     PORTDATA ;GET REGISTER DATA
           MOV    B,A GET IN TRANSFER REGISTER
           RET
```

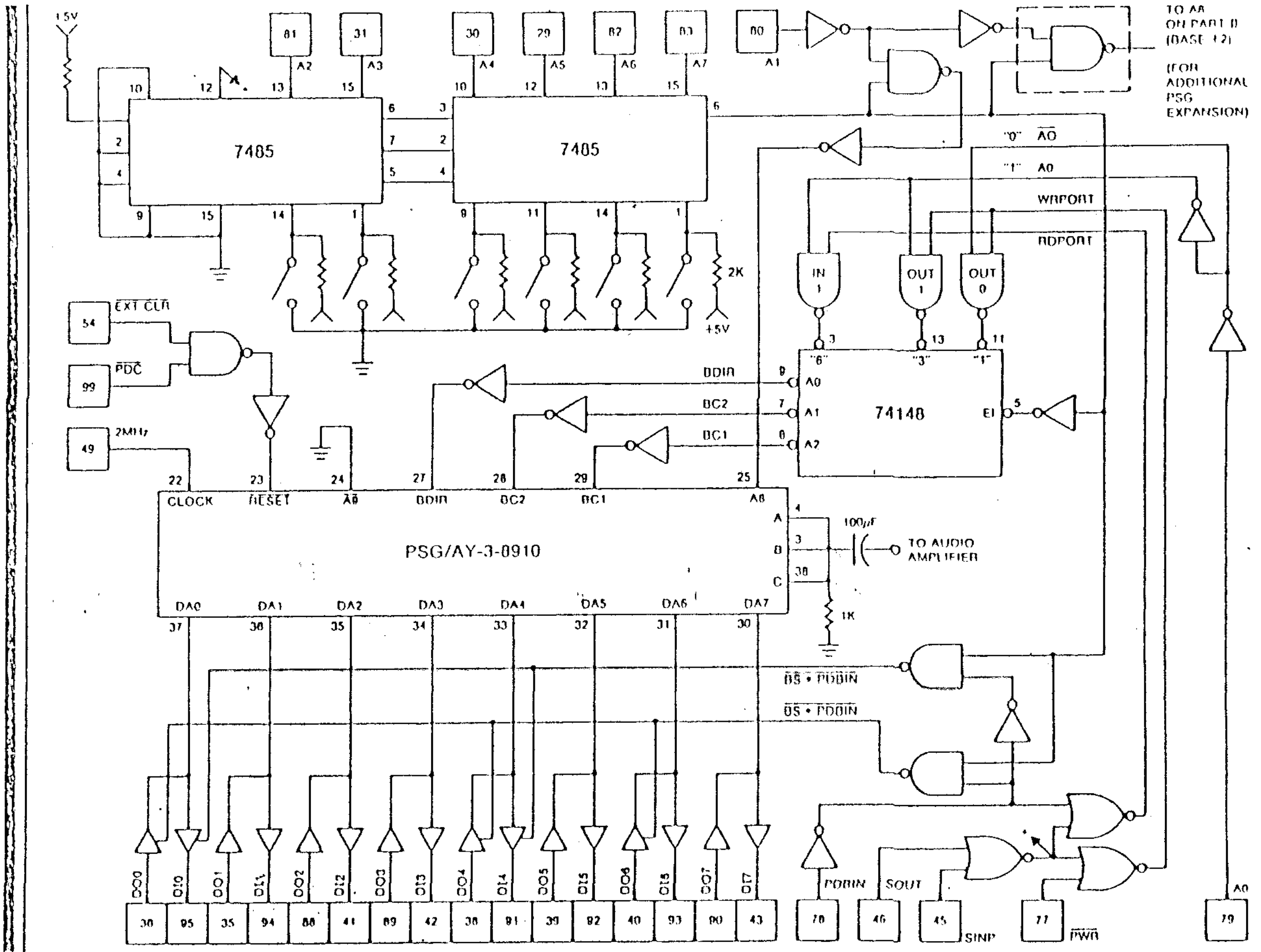


Fig. 22 8080 S100 BUS/AY-3-8910 INTERFACE

5 MUSIC GENERATION

The production of music involves the creation of series of frequencies which are pleasing to the human ear (setting critical evaluation aside). This involves essentially mathematical relationships, making the application ideal for digital devices. For example, the shifting up or down in octaves is a multiplication or division by a power of 2, which is a simple shift operation for most microprocessors.

Another factor in music generation is "communication". The composer must be able to convey his tune ideas so that a musician or group of musicians can reproduce the composer's ideas—often on widely differing instruments. This concept involves "tuning" the instruments to a standard set of frequencies and following a set rhythm pattern. The tuning frequency most widely used is based on the third octave note "A" of 440Hz, the "Equal Tempered Chromatic Scale".

Although it is easy to construct recognizable tunes using only one note at a time, the simultaneous sounding of more than one note to produce chords and counterpoint vastly increases the quality of the sound. This feature is easily achieved in the PSG since three channels are provided, each independently programmable.

5.1 Note Generation

Since notes are formed by sustaining a particular frequency for a preset period of time at a varying amplitude, the PSG performs this function with a series of simple register loads. The method used in many cases is to obtain register load values for first octave notes and to shift to the correct octave at playtime.

The chart in Fig. 23 lists a full 8 octaves of notes from a low of C1 (32.703Hz) to a high of B8 (7902.080Hz). Assuming an input clock frequency of 1.78977MHz (one half the standard "color" crystal frequency of 3.579545MHz), and applying the formulas of Section 3.1 for calculating Tone Period register load values, results in the register values shown. The nature of the PSG divider scheme produces a high degree of accuracy for low frequencies, less for high frequencies. This can be seen in the chart in the comparison of "ideal frequencies" and "actual frequencies", with the ideal frequencies being those of the Equal Tempered Chromatic Scale, and the actual frequencies being the "best fit" values from the formula calculation.

5.2 Tune Entry/ Playback

One of the methods of entering a composition into a computer memory would be to utilize a keyboard to pass number and alphabetic information concerning the composer's wishes. An alternate method would be to scan a positional series of switches (like a piano keyboard) to determine note, volume and duration data.

Since flexibility in tune entry is desired, it is important to allow the composer to specify certain constants of entry such as octave, pitch or tempo, and have these entries normalized to a known value.

5.3 Tune Variations

One of the significant features of a microcomputer based music player is the ability to modify the tune once it has been recorded. Among the simpler variations are:

5.3.1 OCTAVE SHIFT

If an octave constant is added to the octave of the recorded note prior to storing the value in the PSG register, dynamic pitch changes can be obtained. The programming effect would be to shift one bit left for each lower octave and one bit right for each higher octave. For example, the effect will be that a tune written to play on a piano will sound like bells if a multiple octave up modification is performed.

5.3.2 KEY

One measure of the virtuosity of a musician is his ability to modify the "key" or suboctave shift of a composition. The logical description of key transposition is to shift each note up or down by a predetermined number of notes from the original. For example, a piece written in C and played in C# would have all C notes shifted to C#, C# shifted to D, etc. (Note that the case must be considered where B of one octave is shifted to C of the next higher octave.) All of these operations require that the one of twelve note identification must be retained in the recorded representation.

5.3.3 TEMPO

The duration of each recorded note is best expressed in terms of "ticks" of an overall "tempo clock". At playtime, the total duration can be obtained by programatically multiplying the individual note to "slow down" or "speed up" the tune without changing the crucial time relationship between the notes. This can be accomplished by imbedding the note timing loops within the tempo timing loops for simple operation.

5.3.4 CHORDS

There are certain combinations of notes which when played simultaneously produce pleasant combinations. These "chords" can be easily formed from a base note by performing octave and key changes on two notes, which are played with the main note. These relationships are illustrated in Fig. 24, which lists the various note constants which will produce musical chords. A chord with a particular quality may be formed by playing its root, a 3rd Minor or Major, and other notes from the chord chart. For example, a C Major chord is formed from C(÷2), E(÷2), and G(÷2).

Fig. 24 CHORD SELECTION CHART

Chord Selection	Root	3rd Minor	3rd Major	4th	5th	6th	7th
C	C (÷2)	D# (÷2)	E (÷2)	F (÷2)	G (÷2)	A (÷2)	A# (÷2)
C#	C# (÷2)	E (÷2)	F (÷2)	F# (÷2)	G# (÷2)	A# (÷2)	B (÷2)
D	D (÷2)	F (÷2)	F# (÷2)	G (÷2)	A (÷2)	B (÷2)	C (÷1)
D#	D# (÷2)	F# (÷2)	G (÷2)	G# (÷2)	A# (÷2)	C (÷1)	C# (÷1)
E	E (÷2)	G (÷2)	G# (÷2)	A (÷2)	B (÷2)	C# (÷1)	D (÷1)
F	F (÷2)	G# (÷2)	A (÷2)	A# (÷2)	C (÷1)	D (÷1)	D# (÷1)
F#	F# (÷4)	A (÷4)	A# (÷4)	B (÷4)	C# (÷2)	D# (÷2)	E (÷2)
G	G (÷4)	A# (÷4)	B (÷4)	C (÷2)	D (÷2)	E (÷2)	F (÷2)
G#	G# (÷4)	B (÷4)	C (÷2)	C# (÷2)	D# (÷2)	F (÷2)	F# (÷2)
A	A (÷4)	C (÷2)	C# (÷2)	D (÷2)	E (÷2)	F# (÷2)	G (÷2)
A#	A# (÷4)	C# (÷2)	D (÷2)	D# (÷2)	F (÷2)	G (÷2)	G# (÷2)
B	B (÷4)	D (÷2)	D# (÷2)	E (÷2)	F# (÷2)	G# (÷2)	A (÷2)

5.4 Sound Variation

5.4.1 RELATIVE CHANNEL VOLUME

The independently programmable amplitude control for each channel allows up to 16 levels if using the processor controlled amplitude mode (bit 4 of registers 10, 11 or 12=0). In the case of a decaying or steady note, when a note is played or "fired", a frequency may be set up in the coarse and fine tune registers and then an amplitude value placed in the respective register 10, 11 or 12. The value which is placed to play the tune can be an independent variable, allowing channels to play their respective melody lines with *varying force.*

5.4.2 DECAY

The main difference between a "piano" sound and an "organ" sound is the speed with which the note loses volume. If all of the notes can be decayed at a uniform rate, the automatic envelope generator can be set to produce a decaying waveform. Each of the three channels can have the same decay constant but differing playing times to simulate the same instrument with differing note-strike times.

5.4.3 OTHER EFFECTS

The addition of variable noise to any or all of the channels can produce modification effects such "breathing" with a wind instrument. Or noise can be used alone to produce a drum rhythm. The fact that the noise dominant frequencies are variable allows "synthesizer" type effects with simple processor interaction.

Other pleasing effects include vibrato and tremolo, the cyclical variation of the frequency and volume. Because an intelligent microprocessor is controlling the effect, they can be all keyed to the tune itself or to other external stimuli.

6 SOUND EFFECTS GENERATION

One of the main uses of the PSG is to produce non-musical sound effects to accompany visual action or as a feature in itself. The following sections outline techniques and provide actual examples of some popular effects. All examples are based on a 1.78977MHz PSG clock.

6.1 Tone Only Effects

Many effects are possible using only the tone generation capability of the PSG without adding noise and without using the PSG's envelope generation capability. Examples of this type of effect would include telephone tone frequencies (two distinct frequencies produced simultaneously) or the European Siren effect listed in Fig. 27 (two distinct frequencies sequentially produced).

Fig. 27 EUROPEAN SIREN SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R0	376	Set Channel A Tone period to 2.27ms
R1	000	(440Hz).
R7	076	Enable Tone only on Channel A only.
R10	017	Select maximum amplitude on Channel A.
(Wait approximately 350ms before continuing)		
R0	126	Set Channel A Tone period to 5.346ms
R1	001	(187Hz).
(Wait approximately 350ms before continuing)		
R10	000	Turn off Channel A to end sound effect.

6.2 Noise Only Effects

Some of the more commonly required sounds require only the use of noise and the envelope generator (or processor control of chann envelope if other channels are using the envelope generator).

Examples of this, which can be seen in Figs. 28 and 29, are gunshot and explosion. In both cases pure noise is used with a decaying envelope. In the examples shown the only changes are in the length of the envelope as modified by the coarse tune register and in the noise period. Note that a significantly lower explosion can be obtained by using all three channels operating with the same parameters.

Fig. 28 GUNSHOT SOUND EFFECT CHART

Register #	Octal Load Value	15 Explanation
Any not specified	000	—
R6	017	Set Noise period to mid-value.
R7	007	Enable Noise only on Channels A,B,C.
R10	020	Select full amplitude range under direct control of Envelope Generator.
R11	020	
R12	020	
R14	020	Set Envelope period to 0.586 seconds.
R15	000	Select Envelope "decay", one cycle only.

Fig. 29 EXPLOSION SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R6	000	Set Noise period to max. value.
R7	007	Enable Noise only, on Channels A,B,C.
R10 8	16 020	Select full amplitude range under direct control of Envelope Generator.
R11 9	16 020	
R12 10	16 020	
R14 12	070	Set Envelope period to 2.05 seconds.
R15 13	000	Select Envelope "decay", one cycle only.

6.3 Frequency Sweep Effects

The Laser, Whistling Bomb, Wolf Whistle, and Race Car sounds in Figs. 30 thru 33 all utilize frequency sweeping effects. In all cases they involve the increasing or decreasing of the values in the tone period registers with variable start, end, and time between frequency changes. For example, the sweep speed of the Laser is much more rapid than the high gear accelerate in the race car, yet both use the same computer routine with differing parameters.

Other easily achievable results include "doppler" and noise sweep effects. The sweeping of the noise clocking register (R6) produces a "doppler" effect which seems well suited for "space war" type games.

Fig. 30 LASER SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R7	076	Enable Tone only on Channel A only.
R10	017	Select maximum amplitude on Channel A.
R0	060 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 3ms wait time between each step from 060 to 160 (0.429ms/2330Hz to 1.0ms/1000Hz).
R0	160 (end)	
R10	000	Turn off Channel A to end sound effect.

Fig. 31 WHISTLING BOMB SOUND EFFECT CHART

octal value

Register #	Octal Load Value	Explanation
Any not specified	000	—
R7	076	Enable Tone only on Channel A only.
R10	017	Select maximum amplitude on Channel A.
R0	060 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 25ms wait time between each step from 060 to 300 (0.429ms/2330Hz to 1.72ms/582Hz).
R0	300 (end)	

After above loop is completed, follow with sequence in Fig. 28.

6.4 Multi-Channel Effects

Because of the independent architecture of the PSG, many rather complex effects are possible without burdening the processor. Fc example, the Wolf Whistle effect in Fig. 32 shows two channels in use to add constant breath hissing noise to the three concentrated frequency sweeps of the whistle. Once the noise is put on the channel, the processor only need be concerned with the frequency sweep operation.

Fig. 32 WOLF WHISTLE SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R6	001	Set Noise period to minimum value.
R7	056	Enable Tone on Channel A. Noise on Channel B.
R10	017	Select maximum amplitude on Channel A.
R11	011	Select lower amplitude on Channel B.
R0 R0	100 (start) 040 (end)	Sweep effect for Channel A Tone period via a processor loop with approximately 12ms wait time between each step from 100 to 040 (0.572ms/1748Hz to 0.286ms/3496Hz).
	(Wait approximately 150ms before continuing)	
R0 R0	100 (start) 060 (end)	A processor loop with approximately 25ms wait time between each step from 100 to 060 (0.572ms/1748Hz to 0.429ms/2331Hz).
R0 R0	060 (start) 150 (end)	A processor loop with approximately 6ms wait time between each step from 060 to 150 (0.429ms/2331Hz to 0.930ms/1075Hz).
R10 R11	000 000	Turn off Channels A and B to end effect.

Fig. 33 RACE CAR SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R3	017	Set Channel B Tone period to 34.33ms (29Hz).
R7	074	Enable Tones only on Channels A and B.
R10 8	017	Select maximum amplitude on Channel A.
R11 9	012	Select lower amplitude on Channel B.
*R1/R0 *R1/R0	013/000 (start) 004/000 (end)	Sweep effect for Channel A Tone period via a processor loop with approximately 3ms wait time between each step from 013/000 to 004/000 (25.17ms/39.7Hz to 9.15ms/109.3Hz).
R1/R0 R1/R0	011/000 (start) 003/000 (end)	A processor loop with approximately 3ms wait time between each step from 011/000 to 003/000 (20.6ms/48.5Hz to 6.87ms/145.6Hz).
R1/R0 R1/R0	006/000 (start) 001/000 (end)	A processor loop with approximately 6ms wait time between each step from 006/000 to 001/000 (13.73ms/72.8Hz to 2.29ms/436.7Hz).
R10 R11	000 000	Turn off Channels A and B to end effect.

* Decrement R1/R0 as a whole number, e.g. start at 013/000, then 012/377, then 012/376, etc