

The Integration of the AY-3-8910 with the 6502 Microprocessor

Strahinja Marinkovic

July 2023

Abstract

One of the most fundamental parts within a computer design is audio. It brings any software to life and gives the user another dimension of interaction previously missing. To explore how computer systems use audio, this paper will explore the utilization of the programmable sound generator (PSG), AY-3-8910 chip, with the eight bit 6502 microprocessor. It will also discuss the implementation of adding the AY-3-8910 into an already existing system and the challenges that come along with it.

1 PSG Integration

1.1 AY-3-8910

The impact of sound in a system is immense. It is a fundamental part of a computer like video is. Thus, implementing a simple solution in an eight bit system can be a challenge. A solution is a programmable sound generator. For instance, the AY-3-8910 which this paper will explore. This device is a "Large Scale Integrated Circuit which can produce a wide variety of complex sounds under software control."¹ It requires a single 5V power supply and TTL compatible clock. Becuase the PSG is a "bus oriented" system, it should be realitivy easy to implement with a 6502 microprocessor. The following shows the registers that are under software control:

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
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							

Figure 1: The memory layout of the AY-3-8910 registers. The register numbers are in octal.

The choice of the AY-3-8910 is very clear, compared to other sound generators, the AY-3-8910 has the ability to continue to perform sound effects while the processor is doing other tasks. After the initial commands have been sent by the MPU, it can execute other programs while letting sound to play and only have to update registers for a sound change. Another nice

¹From Datasheet

feature of this chip is that it has three channels that can each be controlled, amplitude control, envelope shape, and two general purpose IO ports that can be used for anything. The pinout of the chip is as shown:

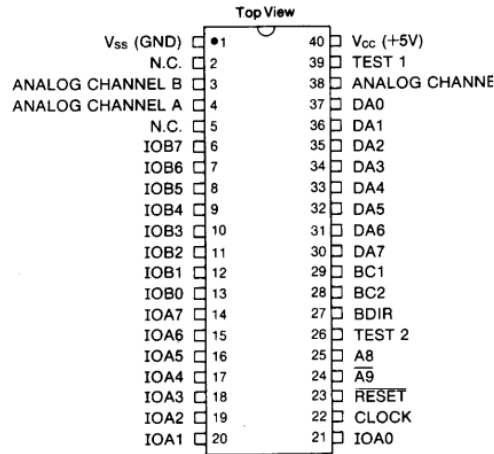


Figure 2: AY-3-8910 Pinout

Interfacing with this chip is quite simple. Data is written and read through the eight bit data/address bus, DA0 - DA7, with three control signals: BDIR, BC1, and BC2. There are no address lines as selecting register addresses occurs through the data bus. There are a total of 16 registers. Addresses are sent through the bus in addition to two extra bits acting as "chip selects" which come from pins A8 and A9. However, according to the data sheet, these are not totally necessary and even have their own pull up and pull down resistors respectively. However, I tied A8 to 5 volts through a 1K resistor and tied A9 to ground through a 1K resistor just in case (the datasheet recommends it, especially in noisy environments).

The control signals allow for four different functionalities: inactive, read, write, and latch. Inactive is when PSG to CPU bus is inactive and goes into a high impedance state. Read is for when the CPU reads from the PSG. Write is for when the CPU wants to write data to the PSG and latch is when the CPU writes an address to latch to a register. The following shows a table of these functions with the control signals.

BDIR	BC1	BC2	Function
0	0	1	INACTIVE
0	1	1	READ (FROM PSG)
1	0	1	WRITE (TO PSG)
1	1	1	LATCH ADDR

This table shows how there are only two signals the MPU needs to control: BDIR and BC1. BC2 can be tied high through a resistor. However, before I can connect this PSG to the MPU, I will first go over how devices can be added to my 6502.

1.2 6502 Device Interface

Because the 6502 uses a bus oriented system, there must be a way to map devices to specific locations in memory. In my computer, I use a GAL chip to handle the memory mapping. The following is a table of the memory layout:

Memory Range	Device
0x0000 - 0x7EFF	RAM (BANKED)
0x7F00 - 0x7FFF	IO
0x8000 - 0xFFFF	ROM

Furthermore, the IO is broken up using a 4 to 16 line decoder. This means that from 0x7F00 to 0x7FFF we can connect 16 devices each having 16 bytes of IO! For example, One of these lines is used to control a ACIA while two others are used to control two VIAS. This control line will be used for our decoding.

Now, one thing I noticed was that the BDIR control signal is in reference to if the MPU is either reading or writing. That means we can use the MPU's RWB pin. However, it is opposite of the PSG. According to the 6502's datasheet, a high on the RWB pin indicates a read and a low indicates a write. This is flipped from the PSG. For the control signal BC1, I decided to use address line A0 from the MPU. This way it uses different address locations for writing data and writing an address. However, the RWB and A0 can not be connected directly to the PSG as there is no way to control

when the PSG bus is active or not.² I needed a way to flip the RWB control signal while also making sure that the bus was active in its memory location.

1.3 Solution

A solution I thought of was using two NOR gates. A NOR gate is simply an OR gate with its output inverted. The following is its logic table:

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

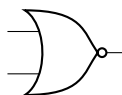


Figure 3: NOR Gate

The logic circuit for decoding uses two of these NOR gates for one AY-3-8910. One NOR gate's inputs are the IO control signal and RWB with its output connected to the BDIR control signal. The other NOR gate's inputs are again the IO control signal from the CPU and A0 with its output connected to BC1. Both NOR gates have the IO control as an input so that there is activity only in the designated memory location. The following is the logic table:

²We cannot connect the IO control line to BC2 because specific logic combinations from the datasheet with the control signal causes latching addresses outside of the PSG designated memory location.

IO	A0	RWB	BDir	BC1	FUNCTION
0	0	0	1	1	LATCH ADDRESS
0	0	1	0	1	READING FROM PSG
0	1	0	1	0	WRITING TO PSG
0	1	1	0	0	INACTIVE
1	0	0	0	0	INACTIVE
1	0	1	0	0	INACTIVE
1	1	0	0	0	INACTIVE
1	1	1	0	0	INACTIVE

It can be seen that when IO is high, meaning it is inactive, the PSG bus is also inactive. Then, when IO is low, the CPU can either read or write or latch an address depending on if its in byte 0 or byte 1. This is extremely useful as it means that the PSG does not have to be controlled by a VIA, which would take up 11 pins! A technique I tried prior to this was using the VIA but then connecting a second PSG to the firsts IO lines, this way they could be nested and only need 11 pins of the VIA. This can still work with the new set up or a different IO line can be connected to the second PSG since there are 16 lines to choose from meaning it would be difficult to run out. Otherwise the nesting technique for controlling them is also suitable.

1.4 Circuit Layout

The rest of the pins on the AY-3-8910 are quite simple to connect. Pin 23, RESET can be directly connected to the 6502 reset. Vss can be connected to ground and Vcc to 5 volts. Pin 22, CLOCK could be connected directly to the CPU's clock, however, the datasheet recommends a clock speed of 2 MHz. In my system, I am using a clock speed of 1.8432 MHz because I have an ACIA that already requires that clock speed.

The IO lines can be left unconnected, connected to a header or socket of some sort, or used to control other devices in your system like another PSG. The N.C and TEST pins are left unconnected. The AY-3-8910 has three analog channels. This means that there must be a circuit to merge these channels into one audio signal. The amplification of the signal can depend on what speaker you use. In my case, the speakers I have do their own amplification thus I don't need to build my own. To get the analog channels together, I developed the circuit shown in figure 4. It should be noted that

this circuit does no amplification of the audio. It simply merges the channels together and does some filtering.

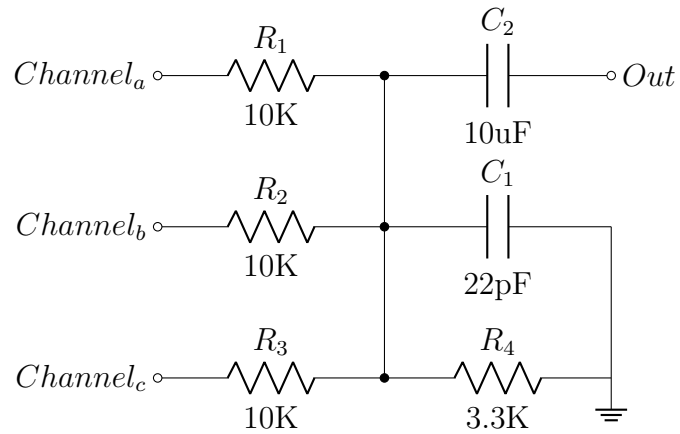


Figure 4: Three to One Channel Circuit

1.5 Software

The software that controls the PSG is quite simple.

References

- [1] General Instrument, "Programmable Sound Generator", AY-3-8910, 1978.
- [2] The Western Design Center Inc, "W65C02S 8-bit Microprocessor", W65C02S, 1983, [Revised Jan. 2022].