

Programming the AY-3-8910 sound chip

- [Specifications](#)
- [Memory allocation](#)
- [Memory layout](#)
- [I/O Port mapping](#)
- [Cartridge/Extension port](#)
- [ROM Recognition](#)
- [System block diagram](#) (Blue Sky Rangers)
- [PCBoard](#)
- Build an [16K+ Cartridge](#)
- An [external keyboard](#)
- [AY-3-8910 programming](#)
- [The handcontrollers](#)
- [Cassette Cable](#)
- [PC Connected](#)



Programming the AY-3-8910 Programmable Sound Generator (PSG) is a relative simple task. But since the Aquarius BASIC doesn't support any command's for INPUT / OUTPUT the task has to be solved by machinecode. The info on this page is therefor for the advanced programmer with some assembler knowledge.

Before we start programming the PSG, we have to know something about it's internal structure. It has 16 registers, which are used as followed:

Register \ bit	B7	B6	B5	B4	B3	B2	B1	B0
R0	8-Bit Fine Tune A							
R1	Channel A Tone Period				4-Bit Coarse Tune A			
R2	8-Bit Fine Tune B							
R3	Channel B Tone Period				4-Bit Coarse Tune B			
R4	8-Bit Fine Tune C							
R5	Channel C Tone Period				4-Bit Coarse Tune C			
R6	Noise period				5-Bit Period control			
R7	IN/OUT		Noise			Tone		
	IOB	IOA	C	B	A	C	B	A
R8	Channel A Envelope on/off, Volume		Env	volume				
R9	Channel B Envelope on/off, Volume		Env	volume				
R10	Channel C Envelope on/off, Volume		Env	volume				

R11	Envelope Period	8-Bit Fine Tune Envelope			
R12		4-Bit Coarse Tune Envelope			
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 on Port B			

The Mattel Aquarius has two I/O ports in order to program the PSG. Port 0xF7 is used to select the register and port 0xF6 is used to read or write the selected register. In order to generate a tone on Channel A we do the following:

```

data      label  opcode  operand          comment
 62,   8  start:  LD      A,0x08    ; Select register #8
211, 247                OUT     (0xF7),A
 62,  15                LD      A,0x0F    ; Volume channel A full
211, 246                OUT     (0xF6),A
 62,   0                LD      A,0x00    ; Select register #0
211, 247                OUT     (0xF7),A
 62,  93                LD      A,0x5D    ; Write #93 into register #0
211, 246                OUT     (0xF6),A
 62,   1                LD      A,0x01    ; Select register #1
211, 247                OUT     (0xF7),A
 62,  13                LD      A,0x0D    ; Write #13 into register #1
211, 246                OUT     (0xF6),A
 62,   7                LD      A,0x07    ; Select register #7
211, 247                OUT     (0xF7),A
 62,  62                LD      A,0x3E    ; Enable output Channel A (0011 1110)
211, 246                OUT     (0xF6),A
201                                RET                    ; Return to BASIC

```

Translated to BASIC that would be:

```

10 DATA 62,8,211,247,62,15,211,246
20 DATA 62,0,211,247,62,93,211,246
30 DATA 62,1,211,247,62,13,211,246
40 DATA 62,7,211,247,62,62,211,246
50 DATA 201
60 FORX=32000TO32032:READA:POKEX,A:NEXT
70 POKE14340,0:POKE1341,125:X=USR(0)

```

Which results in a nice everlasting tone. The tone you hear is the note C. In order to generate certain tones you can use these periods:

<i>Note</i>	<i>Period</i>	<i>Note</i>	<i>Period</i>
C	3421	F#	2419
C#	3228	G	2283
D	3047	G#	2155
D#	2876	A	2034
E	2715	A#	1920
F	2562	B	1892

The lower the period, the higher the note. In order to get a higher octave, just divide the period by two.

The registers 0 & 1 are used for the period of Channel A. To calculate the correct values for these registers you can use the following formula:

$$\text{Register 1} = \text{INT}(\text{period} / 256)$$

$$\text{Register 0} = \text{period} - (\text{Register}_1 * 256)$$

(Ofcourse this also applies to the registers 2 & 3 for channel B and the registers 4 & 5 for channel C)
To work out the frequency of the note, use this equation:

$$\text{Frequency} = 111861 / \text{Period}$$

Now let's take it a little step further and try working with an effect. The following will generate a short decaying burst of noise, just like a gunshot:

```

data   label  opcode  operand  comment
62, 11 start: LD      A,0x0B  ; Register 11
211,247      OUT     (0xF7),A
62,160      LD      A,0xA0
211,246      OUT     (0xF6),A
62, 12      LD      A,0x0C  ; Register 12
211,247      OUT     (0xF7),A
62, 15      LD      A,0x0F
211,246      OUT     (0xF6),A
62,  6      LD      A,0x06  ; Register 6
211,247      OUT     (0xF7),A
62, 15      LD      A,0x0F
211,246      OUT     (0xF6),A
62,  7      LD      A,0x07  ; Register 7
211,247      OUT     (0xF7),A
62, 55      LD      A,0x37  ; 00110111
211,246      OUT     (0xF6),A

```

```

62, 13      LD      A,0x0D      ; Register 13
211,247    OUT     (0xF7),A
62, 0       LD      A,0x00
211,246    OUT     (0xF6),A
62, 8       LD      A,0x08      ; Register 8
211,247    LD      (0xF7),A
62, 16      LD      A,0x10      ; Enable Enveloppe Channel A
211,246    OUT     (0x246),A
201        RET                    ; Return to BASIC

```




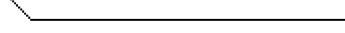





Again, translated to BASIC:

```

10 DATA 62,11,211,247,62,160,211,246
20 DATA 62,12,211,247,62,15,211,246
30 DATA 62,6,211,247,62,15,211,246
40 DATA 62,7,211,247,62,55,211,246
50 DATA 62,13,211,247,62,0,211,246
60 DATA 62,8,211,247,62,16,211,246
70 DATA 201
80 FORX=32000TO32048:READA:POKEX,A:NEXT
90 POKE14340,0:POKE1341,125:X=USR(0)

```

The effect is generated by the Enveloppe register #13. It can make several effects which are graphical represented in the next table:

R13 Bits					Decimal representation	GRAPHICAL REPRESENTATION OF ENVELOPE GENERATOR OUTPUT (E3 E2 E1 E0)
B3 Continue	B2 Attack	B1 Alternate	B0 Hold			
0	0	x	x	0,1,2,3		
0	1	x	x	4,5,6,7		
1	0	0	0	8		
1	0	0	1	9		
1	0	1	0	10		
1	0	1	1	11		
1	1	0	0	12		
1	1	0	1	13		
1	1	1	0	14		

1	1	1	1	15	

After running the last BASIC program one can experiment with the envelope effects by poking the wanted effect on address 32037 and calling the USR(0) function. For example:

```
POKE32037,4:X=USR(0)
POKE32037,8:X=USR(0)
```

This concludes the article on generating sound using the AY-3-8910 PSG. Be sure to read the next article on reading I/O with the AY-3-8910 PSG using the [handcontrollers](#).