MSX Article

MARMSX

8-bit sounds on MSX PSG

Summary

In this article is based on article published in MSX Assembly Page [1], where we will discuss on how it is possible to play a good quality PC 8-bit samples using the native MSX sound chip, the PSG AY-3-8910-A.

1- Introduction

It was discussed in the article “Digitized sounds for MSX” [2] two ways to digitize sound for MSX. The first one is made through the PPI keyboard click where only two values can be discriminated, resulting in a 1-bit sound. The other way is by controlling the 16 levels of PSG volume, resulting in a 4-bit sound. Thus, it is possible to combine the three PSG channels in a way that the sound resolution increases to 816 levels, making the sound quality even better. This topic will be explained in the next chapters.

2- MSX volume logarithmic shape

MSX PSG volume control operates in a logarithmic scale. For each two levels of volume we decrease, we actually decrease the sound amplitude in half [1][3]. The General Instrument datasheet [3] presents a graphic showing the relationship between the PSG nominal volume value (digital) and the normalized generated voltage (analog).

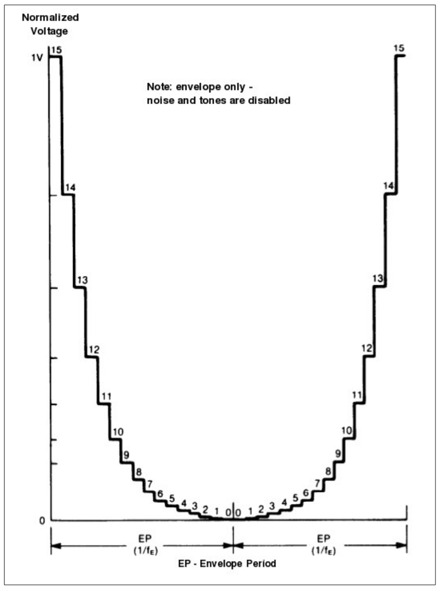


Figure 2.1 – PSG digital to analog conversion [3].

The D/A conversion (DAC) formula [1] is presented next.

$$y = 2^{\frac{-(15-n)}{2}} \qquad \text{formula (1)}$$

Where *y* denotes the DAC response for the digital volume from 0 to 15.

The formula (1) can be understood by the following way: suppose a set of elements that increases *step* by *step* by a constant *ratio*. In order calculate an element *y* of this set by the index value, we have:

$$y = ratio^{\frac{index}{step}}$$

For example, consider a set of base 2 exponential progression:

1, 2, 4, 8, 16, 32, etc.

The ration between members is constant and the next member has always the double value of the current member. So, the *ratio* is 2 and the *step* is 1.

$$y = 2^{\frac{i}{1}} = 2^i$$

The index value starts at 0. By using the formula, we found for the fourth element of the set, which index is 3, the value 8.

Now suppose that our set increased the value by 2 for each 3 units. So:

$$y = 2^{\frac{i}{3}}$$

The set members are: 1, 1.26, 1,59, 2, 2.52, 3.18; 4, etc

For exponential expressions, the value of *y* “explodes” as *i* value increases. In the case of PSG, the *y* values are attenuated. In that case, we must replace *i* by an expression that decreases the value of the numerator while *i* increases. In addition, we know that when *i=15*, the corresponding DAC value is 1.0. As any number raised by 0 is equal 1, the expression that replaces *i* is *15-i*.
At last, a negative sign in an exponential expression inverts the value of the ratio. So, we have the following equivalence:

$$y = 2^{\frac{-(15-n)}{2}} = \frac{1}{2}^{\frac{15-n}{2}}$$

Using formula (1), we can calculate all the corresponding DAC and PCM unsigned 8-bit values for each PSG volume value. The PCM is found multiplying the DAC by 255. DAC is 0 for PSG volume 0.

PSG volume	DAC	8-bit PCM
0	0	0
1	0.0078125	2
2	0.0110485435	3
3	0.015625	4
4	0.0220970869	6
5	0.03125	8
6	0.0441941738	11
7	0.0625	16
8	0.0883883476	23
9	0.125	32
10	0.1767766953	45
11	0.25	64
12	0.3535533906	90
13	0.5	128
14	0.7071067812	180
15	1	255

Table 2.1. PSG to DAC and PCM conversion values.

The backwards formula to calculate the PSG from a DAC value is described next.

As we know:

$$\log_a b = X \quad \text{and} \quad a^X = b$$

So:

$$b = 2^{\frac{-(15-n)}{2}}$$

Where:

- *a = 2*
- *X = -(15-n) / 2*
- *b = DAC or y*

The unknown here is the variable *n*.

According to that:

$$\log_2 y = \frac{-(15-n)}{2}$$

$$2 \times \log_2 y = -15 + n$$

$$2 \times \log_2 y + 15 = n \qquad \text{formula (2)}$$

3- Putting the three PSG channels together

The three PSG channels PSG can be put together, resulting in a sum of each channel DAC value [1]. Figure 3.1 shows the PSG waveforms for one, two and three PSG channels used at the same time, playing the Do note (octave 4) and volume 15 for each channel.

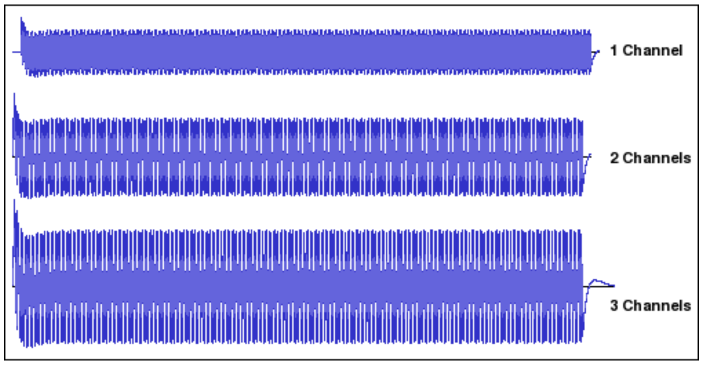


Figure 3.1 – PSG response for 1, 2, or 3 channels playing the Do note.

By combining different channels, we get more discrete DAC values than the 16 values per channel. For example, the DAC value 0.51 can be achieved by the sum of volume 13 (0.50) from channel A and volume 2 (0.01) from channel B [1].

As each channel has 16 levels or 3 bits, putting them together results in 4096 levels or 12 bits. Nevertheless, after eliminating the duplicated levels, remains only 816. This is quite enough for playing the 256 levels of a PC 8-bit file format.

In the sound file formats presented in [2], the data represent the right values to the keyboard click or PSG channel volume. For the PCM unsigned 8-bit file, the data value is PCM and it is necessary to convert it to the right volume combination for each PSG channel. For such task, a conversion table located in MSX memory will be used, where each index represents the PCM value and each table's row holds a volume combination for the three PSG channels.

The challenge now is to find out the best 256 volume combinations subset from the 816 available for the PCM values, where the fit error is the least.

3.1. Building up the conversion table between PCM and PSG

The conversion table between PCM and PSG that will be used in the MSX has 256 entries. Thus, the first step is to create a table with the 816 possible combinations of PSG volumes that will help us to analyze which is the best 256 subset for the PCM values. The number 816 comes from the mathematics combination with repetition formula, presented next.

$$C_{n,p}=\frac{(n+p-1)!}{k!(n-1)!}$$ formula (3)

Then:

$$C_{16,3}=\frac{(16+3-1)!}{3!(16-1)!}=\frac{18!}{3!15!}=\frac{18\times17\times16\times\cancel{15!}}{3!\cancel{15!}}=\frac{18\times17\times16}{3\times2\times1}=\frac{4896}{6}=816$$

The next code is a C adaptation for the program available in [1]. Here, we create a table with all 816 possible combinations of the three channels of PSG volume.

```
#include <math.h>

double psg_dac[16];
double psg_table[816][4];
unsigned int map[256];

void psg_values() {
  int n;
  psg_dac[0] = 0;
  for (n=1; n<16; n++)
    psg_dac[n] = pow(2.0, -(15-n)/2.0);
}

void create_psg_table()
 {
  int p=0, i, j, k;
  for (i=0; i<16; i++) {
    for (j=i; j<16; j++) {
      for (k=j; k<16; k++) {
        psg_table[p][0] = (double) i;
        psg_table[p][1] = (double) j;
        psg_table[p][2] = (double) k;
        psg_table[p][3] = psg_dac[i] + psg_dac[j] + psg_dac[k];
        p++;
      }
    }
  }
}
```

The table *psg_table* has the volume of each PSG channel (columns 0 to 2) plus the corresponding DAC from PSG (column 3), which ranges from 0.0 to 3.0. For each PCM value, we must find out the PSG DAC value closest to the PCM DAC value in the table.

The next code calculates the best subset of PSG.

```
void get_best_combinations(double scale) {
  int i, p, best_idx = 0;
  double total_error = 0, dac, diff, min_diff;
  for (i=0; i<256; i++) {
    dac = scale * (i / 255.0);

    best_idx = 0;
    min_diff = fabs(dac - psg_table[0][3]);
    for (p=1; p<816; p++) {
      diff = fabs(dac - psg_table[p][3]);
      if (diff < min_diff) {
        min_diff = diff;
        best_idx = p;
      }
    }

    map[i] = best_idx;
    total_error += min_diff;
  }

  printf(" Total error: %f\n", total_error);
  printf(" Total error normalized: %f\n", total_error / scale);
}
```

The PSG DAC value is calculated as follows:

$$DAC=\frac{PCM}{255} \quad \text{formula (4)}$$

The total error is the sum of the errors for all the 256 PCM values. This error will be used as reference for a little trick used next to find out the best 256 subset from *psg_table*.

The article [1] suggest us to scale the PCM values from 0.0 to 3.0. By doing that, we could spread these values through the *psg_table* in the search of the subset that could return the least total error. According to that:

$$DAC=\frac{PCM}{255}\times scale \quad \text{formula (5)}$$

It is also necessary to normalize the total error value, according to the scale used.

```
// Normalize error
total_error = total_error / scale;
```

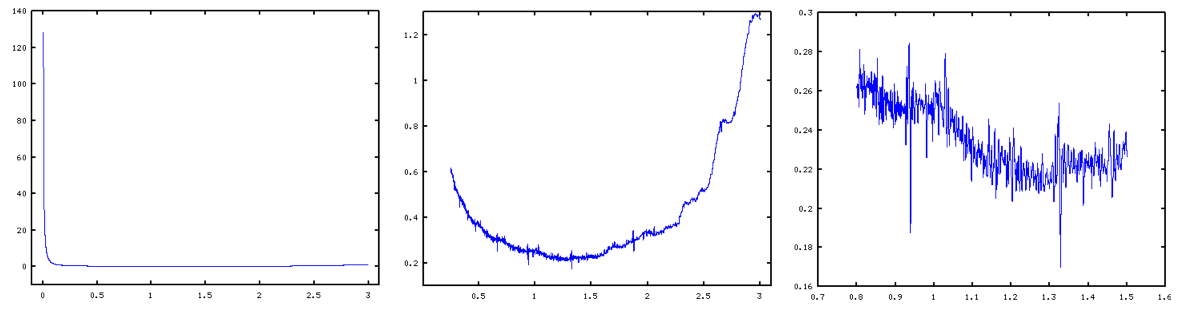
After performing some tests [1], the article suggests that the best scale values could be found between 0.5 and 2.5, and the value 1.328 is the one that returned the least error.

4. The PSG 3 channel player analysis

In this chapter we will analyze the PSG 3 channel player proposed by the authors of the MSX Assembly Page – MAP [1], which will be called here MAP player.

4.1. Graphic analysis for each scale value error calculated

By ranging the scale values from 0.001 to 3.000 and using the step of 0.001, three graphics were generated (see figure 3.2). The graphic in (a) represents all the scales used. From the elbow of the curve, we deducted that scale values below 0,25 could be eliminated to a better visualization of the data. So, we plotted a new graphic in (b). This graphic suggests us that the scale values between 0.8 and 1.5 should be the better errors values. At last, a new graphic were plotted in (c) for a even better visualization. The scale value 1,3281 presented the least error (1,16969).



(a) (b) (c)

Figure 4.1 – Graphic analysis for each scale value error calculated.

The next picture shows the 256 PCM levels distribution over the 816 three channel PSG possible combinations (dark-blue circles) using scale 1.3281. Notice that the PCM data form a straight line, finishing right before to start the high deforming curve.

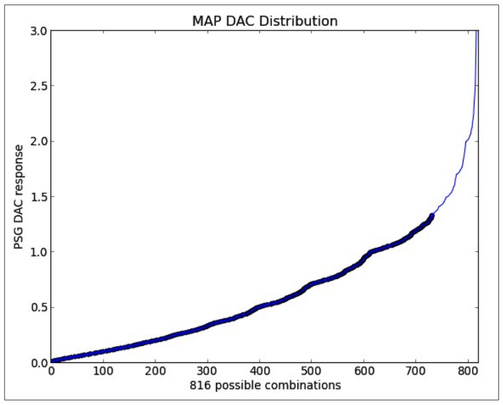


Figure 4.2. - PSG levels distribution.

If the scale is greater than 1.3281, the data distribution advances to the right and climbs the curve (figure 4.2). If it is smaller, the distribution goes to the left. The next figure shows the PSG DAC response for each PCM unsigned 8-bit value in different scales. Notice what happens to the higher scales, when the PCM values are over 170.

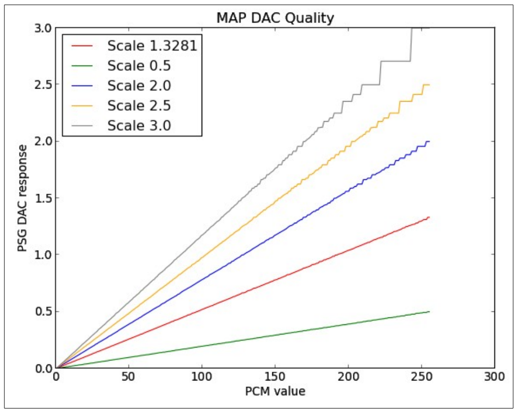


Figure 4.3. - PSG DAC responses for PCM values in different scales.

Whenever we raise the scale, then sound volume raises. Nevertheless, some distortions can be seen in higher PCM values. Thus, in all the cases presented in figure 4.3, the PSG response was always linear. This is a proof that MAP player does well in fixing the logarithmic PSG response curve issue.

Notice that scale 3.0 (gray line) is quite louder than the good scale 1.3281 (red line). Must we reject a louder sound in the name of better quality? We will discuss it in the next section.

4.2. Comparing MAP with different MSX sample players

In this section we will compare MAP player with two other players: MarMSX S4b and PSG Sampler. The S4b uses only one PSG channel whereas PSG Sampler uses all.

Figure 4.4 compares different PSG responses for PCM unsigned 8bit values for MAP player (scale 1.3281) and the other players. Figure out that MAP is the only one who had a linear response.

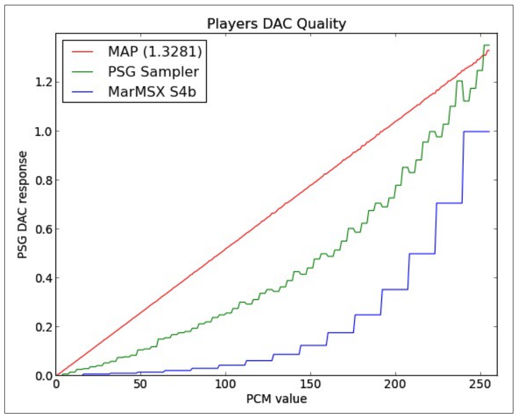


Figure 4.4. - PSG DAC responses for PCM values for different players.

The next table compares four MAP player scales with the other players using different criteria. Take in account that for the PSG Sampler, we calculated the error creating a polygonal fit curve based on PSG Sample DAC curve presented in figure 4.4 (green curve).

Player	Scale	Total Error	Error in Scale	Unique Levels	PSG Response
MAP Player	1.3281	0.2253	0.1696408403	244	Linear
MAP Player	0.5	0.66705	1.3341	221	Linear
MAP Player	2.5	1.27542	0.510168	193	Linear
MAP Player	3	3.78722	1.2624066667	170	Linear
PSG Sampler	1.3536	3.4309	2.5346483452	64	Exponential
S4b Player	1	6.3228	6.3228	16	Exponential

Table 4.1. Comparing different MSX sample players.

Even for the worst scale 3.0, the MAP player presented better results than the other tested sample players, if we consider the error in scale, unique levels and PSG response.

Next, we will compare the generated sound waves by the MSX PSG (openMSX emulator), using the MAP player with scale 3.0 and PSG Sampler for the same PCM sound file.

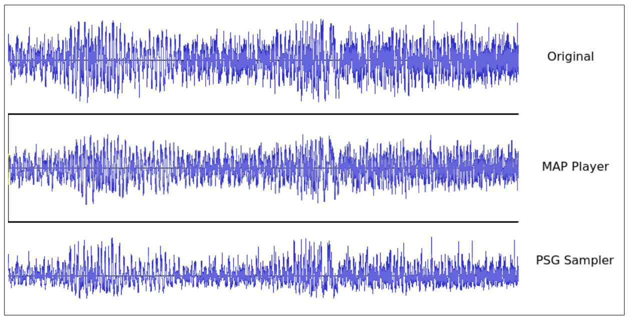


Figure 4.5. - Sound waves generated by MAP player and PSG Sampler.

The resulting MAP player wave is quite more similar to the original wave than the PSG Sampler wave. Notice that for values close to the central line (PCM value equal 128) or below it, the PSG Sampler wave's distortions are greater.

Credits and references

This article was written by Marcelo Silveira in January 2022, based on the article [1] published in the MSX Assembly Page.

E-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com

References:

[1] – Playing Samples on the PSG. MSX Assembly Page, at:
http://map.grauw.nl/articles/psg_sample.php (mais atual)
[2] - Digitized sounds for MSX. MarMSX Development, articles at:
http://marmsx.msxall.com

[3] – General Instrument AY-3-8910, AY-3-8912, AY-3-8913 datasheet. Found at: MSX Assembly Page - http://map.grauw.nl/.