MSX Article

MARMSX

Digitized Sounds for MSX

Summary

This articles aims at describing the techniques used to generate digitized sounds for MSX native sound chips. Thus, it proposes the creation of a 4-bit PCM format.

1. Introduction

Some games included digitized voices in the 80's and 90's, where we could hear voices speaking short sentences. The MSX had also some games with digitized sounds like Snake-it, pronouncing the phrases “welcome to Snake-it”, “next screen” and “game over”; Bosconian with “Blast Off”; Macattack with “welcome to macattack” and “prepare your hamburgers”; and the game Oh Shit with “This is Joey, Paul, Willie and Frankie” and “oh, shiiiit!”.

Some programs to digitize and reproduce sounds were also created. In Brazil, the CPM group created Video Hits [1], a 1-bit PCM player that could play up to 24 seconds of digitized musics. In France, the Digivoix [2] was created and published on a French magazine. This tools is able to digitize sounds from a microphone attached on a MSX tape recorder, play digitized sounds and save/load the digitized sounds.

Besides the MSX Red Book [3] and MSX2 Technical Handbook [4] present some stuff about digitized sounds, the MSX literature is quite poor on this subject. According to that, this article proposes a deep study on digitized sounds for MSX, the sound formats and the players used to reproduce the sounds.

2. Sound Waves and Digital Sound

The sound is a mechanical wave [5] produced by the vibration of the air's molecules. This type of wave is created when we speak, make a guitar string vibrate, hit a drum or plate, when objects collide, when a explosion occurs etc.

A wave can be graphically represented, where the x-axis denotes the propagation time of the wave, while y-axis contains the wave intensity in a certain time. Figure 2.1 shows an example of graphical wave representation.

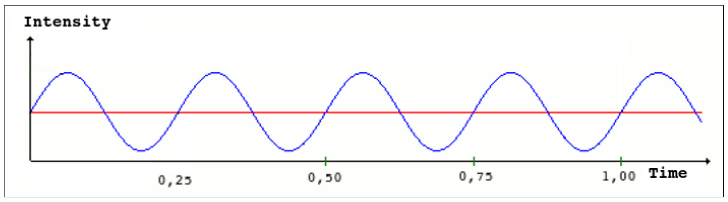

Figure 2.1. Example of graphical wave representation.

The wave is a repetitive phenomena composed by cycles. The duration of each cycle varies according to the phenomena that created the wave. The time spent by one cycle is called period. The frequency is the number of cycles generated by the wave in a determined period of time, generally one second. Figure 2.2 illustrates the period (T) and frequency (F).

In this example, the period is 0,25 seconds and the frequency is 4 cycles per second or Herz. This means that for each second, 4 cycles are produced.

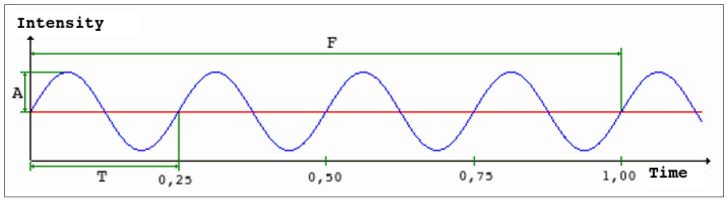

Figure 2.2. Period and Frequency.

For the real world sound events, we can have many frequencies and formats on the wave resulted by the event. Figure 2.3 shows a wave segment produced by a human voice.

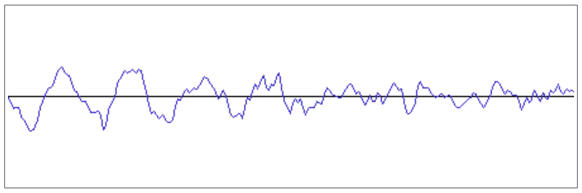

Figure 2.3. Sound wave resulted by a human voice.

More information about waves can be seen on Basic course [6].

2.1. Analog signal digitizing process

The sound wave is a analog signal composed by infinite intervals of time. The digital sound is composed by discrete (finite) time intervals, created from an analog signal. According to that, the digitizing process consists in sampling the analog signal in regular intervals of time. This process is so called sampling.

Figure 2.4 presents an example of analog signal sampling. We can see the intensities obtained (red squares) in regular intervals.

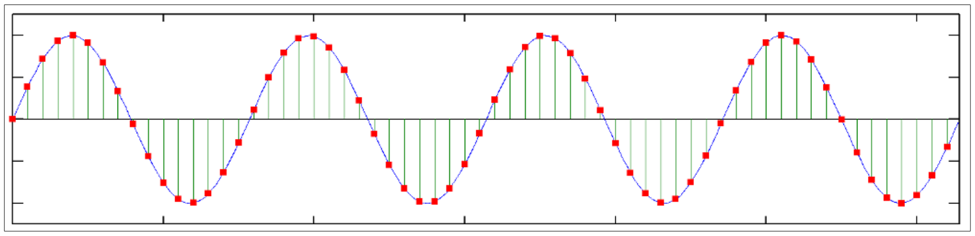

Figure 2.4. Digitizing an analog signal.

The sampling frequency must be greater than the signal frequency. According to Nyquist, it is necessary to sample using at least twice the signal frequency [9] in order to be able to reconstruct the original wave in a satisfactory way. For more complex signals, we base on the principle that human beings are able to perceive frequencies between 20 up to 20 KHz. So, generally people use 44 KHz sampling frequency to record high quality sounds [9]. Values below 44 KHz start to deteriorate sound quality for humans.

2.2. The PCM format

Pulse-code modulation (PCM) is a method used to represent sampled analog signals. In a PCM sequence, the analog signal amplitude is sampled in regular intervals of time. The Linear pulse-code modulation (LPCM) is a type of PCM where the quantize levels are uniform [8].

Generally we have PCM levels of 8, 16, 24 and 32 bits. For example, an 8-bit digital sound file is able to discriminate up to 256 levels of signal intensity.

For discrete values, the resulting values are shifted from their original position, once this value must be one who is closer than the real value (quantization). Figure 2.5 shows an example. Notice that each red square is shifted to the closest interception between the horizontal line and its respective vertical line.

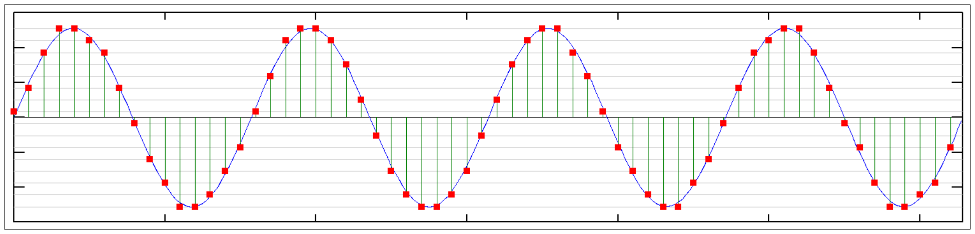

Figure 2.5. Intensities values shift sue to the quantization process.

MSX format is quite singular, once the wave is represented by only one bit.

The Wave (WAV) file format includes the sample values data in sequence. Also, it has a header describing the data and the frequency. The data can be compressed or not. The RAW format contains the same type of data, but without header and compression.

3. How MSX can generate sound waves

The MSX PPI port C has a bit that is able to generate key clicks. This port is accessed through MSX port number &HAA. The next table describes the PPI port C.

Bit	7	6	5	4	3	2	1	0
Description	Key Click	CAPs LED	Cas out	Cas Motor	Keyboard row selection			

According to the MSX Red Book [3], “the key click output is attenuated and mixed with the audio output from the Programmable Sound Generator. To actually generate a sound this bit should be flipped on and off.”

In fact, a click is only generated when the bit 7 state is changed. If we change the state from 0 to 1, a strong click is generated and its value is attenuated until silence, if the bit 7 state is not changed. When the state is changed from 1 to 0, a weak click is generated and attenuated until silence, if the bit 7 state is not changed. The frequency for both events is about 100 Hz.

Figure 3.1 shows the click events. Take a look on the bit 7 values throughout the time. When the state changes, there is a pulse generation (one up, one down).

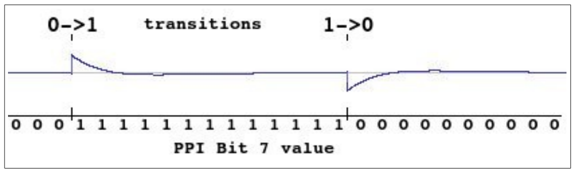

Figure 3.1. Pulses generated by key click on PPI port C.

According to that, to hear correctly a sound on MSX, we must reproduce the wave form by flipping on/off PPC port C bit 7.

This type of wave is squared and it is quite simple to identify when seen on a wave graphic. See the example on figure 3.2.

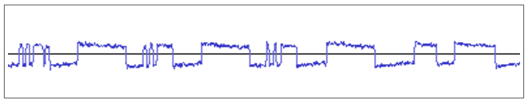

Figure 3.2. 1-bit squared wave resulted.

Once the shortest information for this type of wave is 1 bit, we can put 8 samples in each byte. This set is called octet. The figure 3.3 shows an example of a octet, where the binary value &B10101100 is represented by the byte 172 (&HAC). The bit reading is always performed from the left to the right.

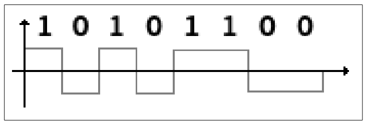

Figure 3.3. One octet for 1-bit PCM.

3.1. PCM signal frequency adjustment on MSX

The 1-bit PCM player created for MSX must read each bit from the octets and, based on this information, change the PPI port C bit 7 state. Nevertheless, there is a problem here: the program reading frequency versus the sound sampling frequency.

By analyzing how Video Hits and Digivoix works, we noticed that they introduced an adjustment called delay. The delay is simple a loop that generates a read and execution delay for each bit read. Using this, we can adjust the program execution frequency next to the sound sampling frequency. The delay value is the number of times that the loop runs.

After that, we analyzed some delay values and their respective frequencies, based on the same sound file, in order to find the relationship between the delay and frequency. After analyzing the resulting graphic, we noticed that the relationship was probably of type:

$$Y = \frac{1}{X}$$

After performing data regression, we confirmed that. So, we could find the relationship between delay and frequency. Figure 3.4 shows the result of a regression applied on 15 pairs of delay and frequency (red squares) measured on a experiment. The blue line represents the regression results. Using this line, we can calculate a frequency for a given delay value.

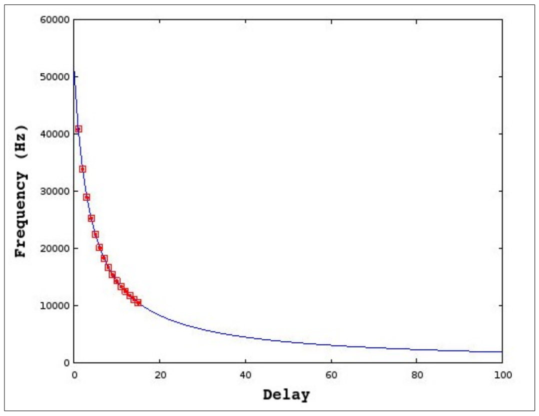

Figure 3.4. Relationship between delay and frequency on Video Hits.

Obs: to calculate the frequency, divide the number of sound file samples by the sound duration.

3.2. Basic algorithm for bits sequence reading and execution

The basic algorithm for reading and play a 1-bit PCM file is described by the following pseudo-code.

```
p ← initial_address;

repeat

  octet ← read_memory(p);

  for i ← 1 until 8 do
    bit ← most_significant_bit(octet);
    octet ← rotate_left(octet);
    data ← read_port(&HAA);
    data ← set_bit(data, 7);
    write_port(&AA, data);

    for t ← 1 to delay do;

  end_for

  p ← p+1;

until p >= final_address;
```

4. Conversion from PCM to MSX

In this chapter we will see how to convert from PC sound formats with 44 KHz to MSX 1 bit format. For that, we will use the PC sound editor Audacity [10] in order to create an intermediary unsigned 8-bit PCM RAW file.

4.1. First steps in Audacity

Audacity is a powerful sound editor program, open source, free and multi-platform. It is compatible with many sound files types, including MP4 video format.

Open a sound file in Audacity. The file is then shown, where on the left we have a panel with some options and on the right the file wave graphic. Figure 4.1 shows a stereo sound file, where two tracks are shown (stereo).

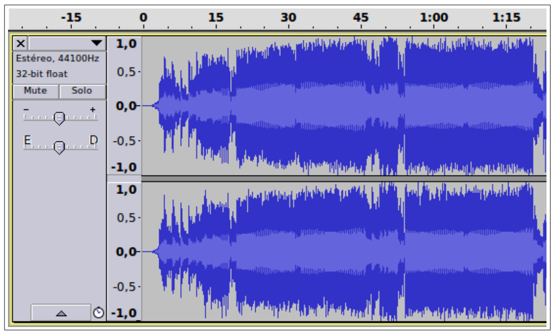

Figure 4.1. Audacity.

MSX formats are all mono. So, if the sound file is stereo, we must convert it to mono. For that, choose menu option “Tracks” and then “Stereo to mono”. Figure 4.2 shows the resulted mono track.

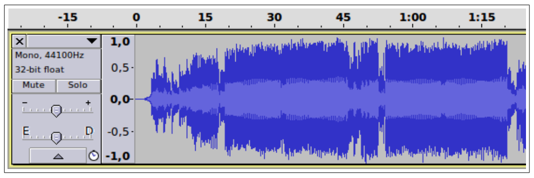

Figure 4.2. From stereo to mono.

The default project frequency in Audacity is 44,1 KHz. But, the frequency of the file used in the MSX must be the one according to the player delay used. For instance, Video Hits default delay value is 14. For that delay, the corresponding frequency is 11,130 Hz.

For changing sound file sample value, we must change first the “Project Rate (Hz)” located at the left bottom of the Audacity's window. Change the value from 44100 to 11130. After that, we must go to menu “Track” option and then “Resample”. Keep the value on the dialog box (that must be 11130). Click on Ok.

A sound file is quite big to MSX standards, even for the 1-bit format. So, the players analyzed uses up to 16 KB blocks. For this size, the maximum duration is 11.78 seconds for the delay 14. So, by using this configuration, we can only select up to 11.78 seconds from our song.

On figure 4.2 we can notice a rule right above the waves. This is the time line and it help us on selecting the sound duration.

Choose any part of the sound file. This choice is made by eliminating the not used parts. For that, click on the initial part of our sound selection part. You may drag the mouse to the beginning, but prefer to press SHIFT + HOME. After selecting the part before to be eliminated, press DELETE key to erase it. Now it is easy to measure the time, once the initial part of our selection is moved to time 0.0 seconds. Click on the time 11.78, aided by the rule. Press SHIFT + END to select the part after our region of interest. Press DELETE to eliminate this part.

By this time, the file is ready to be exported on RAW format. Click on menu “File” and then “Export...”. On the save dialog, change the file type option to “Other uncompressed files”. Now, click on “Options” and choose header “Raw (header-less)” and option “unsigned 8 bit PCM”. Give the file a name using the “.raw” extension. Save it.

The file is then saved in binary format, headless, where each byte corresponds to a sample ranging from 0 to 255. The value 0 is the lowest part on the graphic, the value 255 is the upper part of the graphic and 128 is the central line.

4.2. The 8-bit to 1-bit conversion process

We will present two methods for converting unsigned 8-bit PCM RAW files to MSX 1-bit. The first one is the value thresholding, where values lower than 128 are converted to 0, while the other values are converted to 1. The second method is the gradient, where the value 1 is set always when the current sample value is greater or equal the previous and 0 otherwise.

Some problems were found on the first method when the wave oscillates above or below the central line. When this happens, the wave form is not represented correctly. The second method presented better results, once it reproduced better the original wave form. Figure 4.3 shows a comparison between both methods.

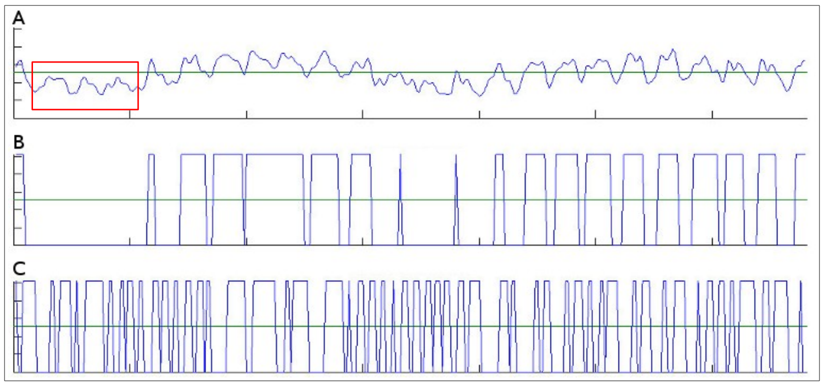

Figure 4.3. Comparing the methods threshold (B) and gradient (C) for a given 8-bit wave (A).

Take a look on the red rectangle on figure 4.3. The wave oscillations below the central line (A) could not be correctly represented in (B), but were well represented in (C).

After converting the 8-bit format to 1-bit, we still have to fit theses values into octets. See the next example.

```
8-bit unsigned values: 200, 145, 180, 128, 130, 80, 45, 80
Thresholding values: 1, 1, 1, 1, 0, 0, 0, 0
Octet fitting: &B11110000
Byte value: 240 (&HF0), which has 8 sound samples.
```

5. The 4-bit format

We can also model a wave form using the PSG volume [11]. Once each PSG channel volume has 16 values, we can quantize the wave sample to 4 bits instead of 1 bit. This format allow us to represent much better the wave form. Nevertheless, it has a high cost: the sound file is 4 times greater than the 1 bit format. For that 11 KHz sound example, we can only play 3 seconds instead of 12.

Figure 5.1 shows the result for 4-bit conversion applied on the same wave form used on figure 4.3. Compare the results.

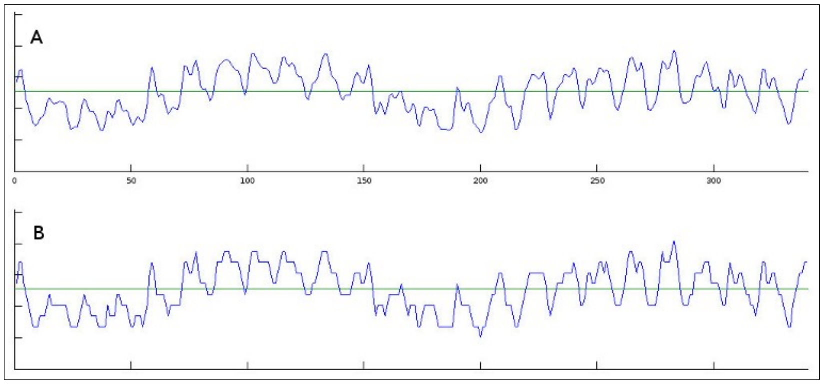

Figure 5.1. Result for 4-bit conversion (the same wave used on fig. 4.3).

For converting this, we quantize the intensity space into 16 values. For each original sample value, we take the closest value.

Notice that 1-bit format plays the sound at maximum volume. Thus, the 4-bit format ranges the volume and, consequently, the sound is lower than the 1 bit.

The basic playing process is presented below.

```
p ← initial_address;

adjust_reg_PSG(7, 254); // Selects channel A
adjust_reg_PSG(0, 0); // Low part of period in channel A
adjust_reg_PSG(1, 0); // High part of period in channel A

repeat

  data ← read_memory(p);

  for i ← 1 to 2 do
```

```
    volume ← nibble(data, i);
    adjust_reg_PSG(8, volume);
    for t ← 1 to delay do;
  end_for;

  p ← p+1;

until p >= final_address;
```

The delay must be always introduced between two sound samples.

Due to the fact of a frequency is used to generate sound, this signal is mixed with sound signal creating an interference. The highest frequencies produces the lowest interferences.

6. Credits and References

This article was written by Marcelo Teixeira Silveira.

Date: July 2020.
Site: http://marmsx.msxall.com
E-mail: flamar98@hotmail.com

References:

[1] - Video Hits, Produced by grupo CPM.
[2] - Digivoix, Produced by Hartard Frederic and published on French magazine Hebdogiciel, numbers 164 to 168, 1986.
[3] – TheMSX Red Book, Avalon Software, published by McGraw Hill.
[4] - MSX2 Technical Handbook, ASCII Corporation, 1987.
[5] - Fundamentos de Física 2, Halliday and Resnick. Publisher Livros Técnicos e Científicos, 1994.
[6] - Sons, Curso de Basic, Site MarMSX. At http://marmsx.msxall.com
[7] - Digital audio. At http://en.wikipedia.org/wiki/Digital_audio
[8] - Pulse-code Modulation. At: http://en.wikipedia.org/wiki/Puse-code_modulation
[9] - 44,100 Hz. At http://en.wikipedia.org/wiki/44,100_Hz
[10] - Audacity. At http://www.adacityteam.org
[11] - MSX Música, José Maurício Bussab. Publisher McGraw-Hill, 1987.