MSX Article

Digitalização de sons para o MSX

Resumo

Esse artigo tem como objetivo apresentar técnicas para a geração de sons digitalizados para sistema de som nativo do MSX. Além disso, propõe a criação de um modelo de som digitalizado com 4 bits de resolução.

1. Introdução

Na década de 80, alguns jogos de videogame incluíam vozes digitalizadas. No caso do MSX, temos como exemplo os jogos Snake-it que reproduzia as frases “welcome to Snake-it”, “next screen” e “game over”; Bosconian com “Blast Off”; Macattack com “welcome to macattack” e “prepare your hamburgers”; e o jogo Oh Shit com “This is Joey, Paul, Willie and Frankie” e “oh, shiiiit!”.

Alguns programas foram lançados, onde era possível reproduzir até 24 segundos de uma música, como é caso do demo brasileiro Video Hits [1]. Na França, foi lançado o programa Digivoix [2], que era capaz de não só reproduzir sons digitalizados, como digitalizar sons a partir de um microfone acoplado ao MSX.

Apesar do Livro Vermelho do MSX [3] e do MSX2 Technical Handbook [4] darem um a boa pista de como criar tais arquivos, a literatura do MSX é pobre em relação ao assunto. Dessa forma, o presente artigo propõe o estudo aprofundado sobre o formato dos arquivos digitais utilizados nesses programas, bem como os programas fazem a reprodução dos sons digitalizados.

2. Ondas Sonoras e Som Digital

O som é uma onda mecânica [5], produzida pela vibração das moléculas do ar. Esse tipo de onda é produzida quando falamos, fazemos vibrar a corda de um violão, através da batida de um bumbo ou prato de bateria, a colisão entre objetos, uma explosão etc.

Uma onda pode ser representada graficamente, onde o eixo *x* contém o tempo de propagação da onda, enquanto que o eixo *y* contém a intensidade da onda em um determinado instante de tempo. A figura 2.1 apresenta um exemplo de representação gráfica de uma onda.

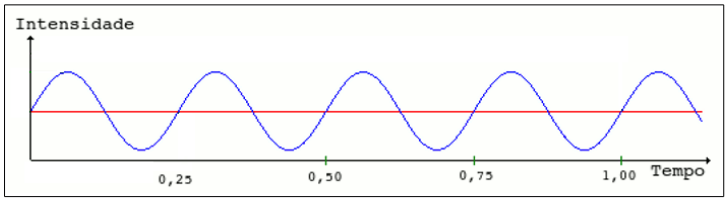

Figura 2.1. Exemplo de representação gráfica de uma onda.

A onda é um fenômeno repetitivo, composta de ciclos. O tempo de duração de cada ciclo varia de acordo com o tipo de fenômeno que produz a onda. A esse tempo chamamos de período. A freqüência é o número de ciclos produzidos por uma onda em uma unidade

de tempo, geralmente em segundos. A figura 2.2 ilustra os conceitos de período (T) e frequência (F). Nesse exemplo, o período é de 0,25 segundos e a frequência de 4 ciclos por segundo ou Herz, ou seja, durante 1 segundo passam exatamente 4 ondas.

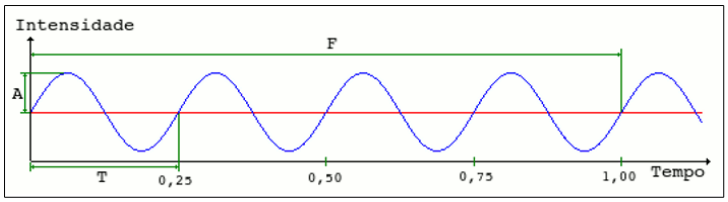

Figura 2.2. Período e freqüência.

Para eventos sonoros do mundo real, a onda produzida poderá ter diversas frequências e formatos. A figura 2.3 ilustra um trecho do formato de onda produzido por uma voz humana.

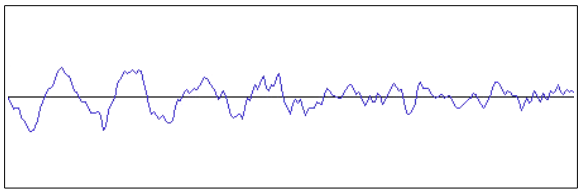

Figura 2.3. Onda sonora produzida por voz humana.

Mais informações sobre ondas pode ser visto no curso de Basic em [6].

2.1. O processo de digitalização de um sinal analógico

A onda sonora é um sinal analógico, ou seja, é composto por intervalos contínuos (infinitos) de tempo. Já o som digital é composto por intervalos discretos (finitos) de tempo, obtidos a partir de um sinal analógico. Dessa forma, a digitalização de um sinal analógico consiste em obter as intensidades do sinal em intervalos regulares de tempo, processo esse chamado de amostragem.

A figura 2.4 mostra um exemplo de digitalização de um sinal analógico. Nela, podemos ver as intensidades obtidas (quadrados vermelhos) em intervalos regulares.

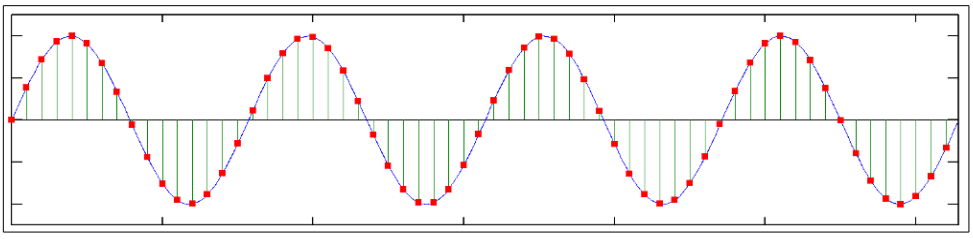

Figura 2.4. Digitalização de um sinal analógico.

A frequência de amostragem deverá ser maior do que a frequência do sinal. Segundo Nyquist, necessitamos amostrar com pelo menos uma frequência com 2 vezes o valor da frequência do sinal [9] para podermos reconstruí-lo de modo satisfatório. No caso de sinais

mais complexos, partimos do princípio que o ouvido humano é capaz de perceber frequências de 20 até 20 KHz. Dessa forma, utiliza-se normalmente uma taxa de amostragem de 44 KHz para gravar sons de alta qualidade [9]. Valores de amostragem abaixo desse começam a deteriorar a qualidade do sinal sonoro para a percepção humana.

2.2. O formato PCM

Pulse-code modulation (PCM) é um método utilizado para representar sinais analógicos amostrados. Em uma sequência PCM, a amplitude de um sinal analógico é amostrada em intervalos regulares de tempo, onde cada amostra é quantizada para o valor de amplitude mais próximo. Já o Linear pulse-code modulation (LPCM) é um tipo de PCM onde os níveis de quantização são uniformes [8].

Normalmente, possuímos níveis de intensidades LPCM com 8, 16, 24 e 32 bits. Por exemplo, um arquivo digital com 8 bits de resolução poderá discriminar até 256 níveis de intensidades do sinal sonoro.

Para valores discretos de intensidade, há um pequeno deslocamento em relação à sua posição original, pois o valor da intensidade deverá ser o do valor correspondente mais próximo (quantização). A figura 2.5 apresenta um exemplo disso. Observe que cada retângulo vermelho está sobre a linha cinza horizontal mais próxima da interseção com a linha verde vertical com o sinal.

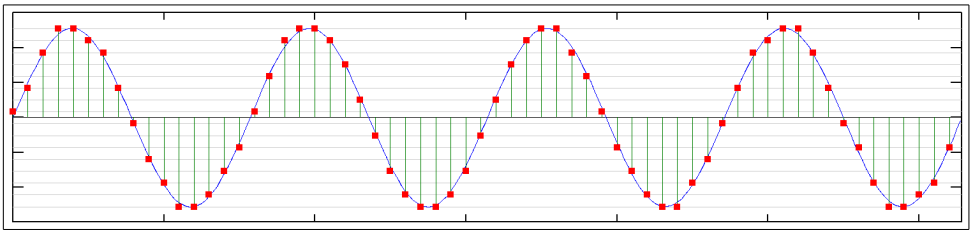

Figura 2.5. Deslocamento dos valores de intensidade devido à quantização dos valores reais.

O MSX apresenta um caso a parte, pois possui um formato de onda com resolução de apenas 1 bit, que será visto mais adiante.

Os formatos de arquivo do tipo Wave (WAV) contém os dados da digitalização com os respectivos níveis de sinal enfileirados. Ele possui um cabeçalho com a descrição dos dados, bem como a frequência de amostragem. Os dados podem ser comprimidos ou não. O formato RAW contém os mesmos tipos de dados, porém não possui compressão e nem cabeçalho.

3. Como o MSX pode gerar ondas

A porta C da PPI do MSX possui um bit que é capaz de gerar sons de clique de teclado. Essa porta é acessada através da porta número &HAA do MSX. A tabela a seguir apresenta a porta C da PPI.

Bit	7	6	5	4	3	2	1	0
Descrição	Clique do Teclado	LED do CAPs	Cas out	Cas Motor	Seleção de linha do teclado			

Segundo O Livro Vermelho do MSX [3], “a saída do clique de teclado é atenuada e misturada com a saída do PSG. Para realmente gerar um som, esse bit deve ser alternado em ligado e desligado”.

Na realidade, um clique somente é produzido quando se altera o estado do bit 7. Caso se altere de 0 para 1, um clique forte é gerado e seu valor é atenuado até o silêncio, caso não haja a alteração de seu valor. Quando a alteração é feita de 1 para 0, um clique fraco é gerado e atenuado até o silêncio, caso o estado do bit 7 não seja alterado. A freqüência de ambos os eventos é de cerca de 100 Hz.

A figura 3.1 ilustra isso. Observe os valores do bit 7 ao longo do tempo. Quando há transições entre 0 e 1, há a geração de um pulso (um para cima, outro para baixo).

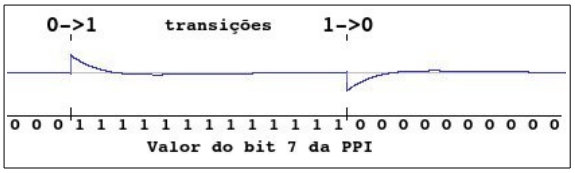

Figura 3.1. Pulsos gerados pelo clique da porta C da PPI.

Dessa forma, para reproduzirmos um determinado som no MSX, temos que reproduzir o formato de onda alterando o bit 7 da porta C da PPI para ligado e desligado. Como somente temos dois níveis de sinal, o formato PCM da onda deverá ser de 1 bit.

Esse tipo de onda possui a forma quadrada, e é facilmente reconhecida quando analisada em um gráfico. Veja o exemplo da figura 3.2.

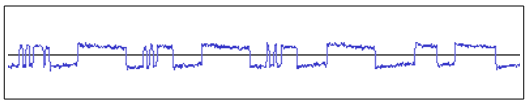

Figura 3.2. Onda quadrada resultante da resolução de 1 bit.

Como a unidade de informação desse formato é um bit, podemos colocar 8 sequências de amostras do sinal dentro de um byte. A esse conjunto chamamos de octeto. A figura 3.3 apresenta um exemplo de octeto, onde o valor binário &B10101100 pode ser representado pelo byte 172 (&HAC). A leitura dos bits é sempre feita da esquerda para a direita.

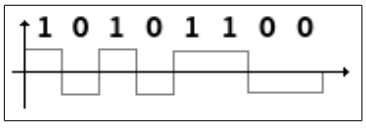

Figura 3.3. Um octeto para PCM de 1 bit.

3.1. O ajuste de frequência do sinal PCM no MSX

O programa que reproduz o arquivo de som PCM de 1 bit no MSX deverá simplesmente identificar se cada bit lido deverá ligar o desligar o clique do teclado. Porém, aí surge um problema: a frequência de leitura do programa versus a frequência de amostragem do som.

Analisando o funcionamento dos programas Video Hits e Digivoix, percebemos a existência de um termo chamado *delay* (atraso). O *delay* nada mais é do que um *loop* que gera atrasos na leitura e execução dos bits do arquivo PCM. Com esse mecanismo, é possível ajustar a frequência de leitura para a frequência próxima à de amostragem do sinal sonoro. O *delay* é um valor inteiro que corresponde a quantidade de vezes que o *loop* é executado.

Então, foram analisados diversos valores de *delay* e suas respectivas frequências geradas para o mesmo arquivo de som, de forma a encontrar uma relação entre as essas duas grandezas. Constatamos que esta relação é do tipo:

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

Ao realizar uma regressão, confirmamos esse fato. Assim, foi possível realizar a relação entre *delay* e frequência. A figura 3.4 mostra a regressão realizada a partir de 15 pares de valores de *delay* e frequência (quadrados vermelhos) obtidos em experimento. A linha azul representa o resultado da regressão, onde é possível calcular o valor de uma frequência de acordo com um dado valor de *delay*.

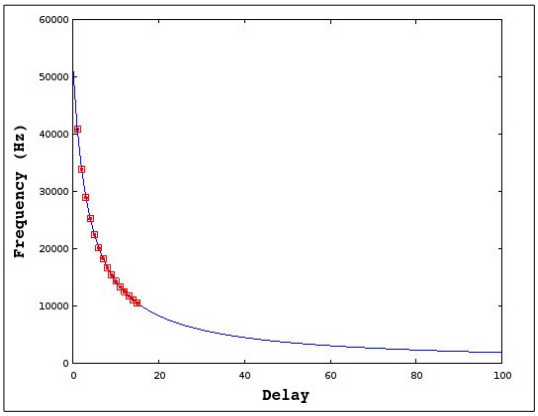

Figura 3.4. Relação entre *delay* e frequência para o Video Hits.

Obs: o valor da frequência é obtido através do cálculo do número de bits que compõe o arquivo dividido pela duração do evento sonoro.

3.2. Algoritmo básico de leitura e execução de sequencia de bits

O algoritmo básico para a leitura e execução de um arquivo PCM de 1 bit é apresentado no pseudo-código a seguir.

```
p ← end_inicial;

repita

  octet ← ler_memoria(p);

  para i ← 1 até 8 faça
    bit ← bit_mais_significativo(octeto);
    octet ← rotaciona_esquerda(octeto);
    data ← leia_porta(&HAA);
    data ← set_bit(data, 7);
    escreva_porta(&AA, data);

    para t ← 1 ate delay faça;

  fim_para

  p ← p+1;

até p >= endereço_final;
```

4. Conversão de PCM para o MSX

Nesse capítulo iremos ver o passo a passo para gerar um sinal sonoro de 1 bit para o MSX, a partir de formatos de arquivo de áudio do PC com 44 KHz. Para isso, iremos utilizar o programa Audacity [10] para PC, de modo a nos auxiliar a gerar um arquivo RAW de 8 bits pronto para ser convertido para o MSX.

4.1. Primeiros passos no Audacity

O Audacity é um poderoso programa de edição de som multiplataforma e open source, ou seja, pode ser obtido gratuitamente. Com ele, podemos abir e editar vários formatos de arquivos de sons, inclusive arquivos de vídeo MP4.

Abra um arquivo de som qualquer no Audacity. O arquivo é então exibido, onde à esquerda temos um painel com diversas opções e à direita o gráfico de onda do arquivo. A figura 4.1 apresenta um arquivo de som estéreo, onde duas faixas de onda são apresentadas.

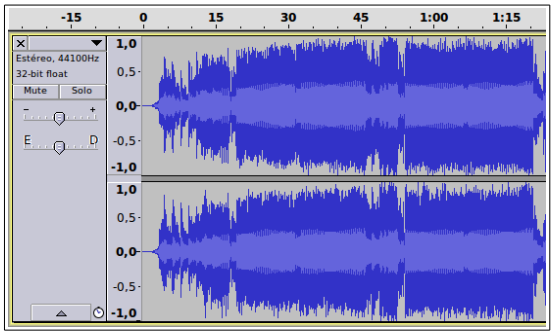

Figura 4.1. Audacity.

Para gerar um formato de som compatível com o MSX, devemos ter apenas um canal (mono). Se o áudio for estéreo, devemos convertê-lo para mono. Para isso, escolha a opção do menu do Audacity “Faixas” e depois “Faixa estéreo para mono”. A figura 4.2 apresenta o resultado.

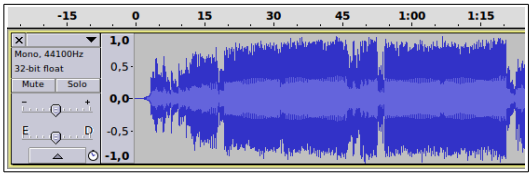

Figura 4.2. De estéreo para mono.

A frequência padrão do Audacity é de 44,1 KHz. Entretanto, a frequência de amostragem deverá corresponder ao *delay* utilizado no MSX. No Video Hits, por exemplo, o valor default é 14. Para esse *delay*, frequência correspondente é de 11.130 Hz.

Para alterar a taxa de amostragem do som, deve-se alterar o valor da “Taxa de Projeto”, localizado na parte inferior da janela do Audacity, de 44100 para 11130. Uma vez feito isso, deve-se selecionar a opção do menu “Faixas” e depois “Mudar taxa de amostragem”. O valor do projeto deverá ser repetido (11130). Clique em Ok.

Um arquivo de som no MSX ocupa bastante espaço, mesmo sendo de 1 bit. O ideal é utilizarmos arquivos com até 16 KB de tamanho. Para esse tamanho, a duração máxima é de 11,78 segundos para o *delay* 14. Portanto, para a frequência de 11 KHz, devemos selecionar um trecho de música ou som com duração de até 11,78 segundos.

Na figura 4.2, podemos observar uma régua situada na parte superior da forma de onda. Ela é a linha de tempo da música e nos auxiliará a selecionar um trecho do som.

Escolha um trecho qualquer do som. Clique na forma de onda sobre o instante inicial do trecho. Apague o trecho anterior, arrastando o mouse até o inicio ou apertando as teclas SHIFT + HOME e em seguida apague com a tecla DELETE. Agora ficou fácil medir o tempo, pois o inicio do trecho foi para o instante 0,0 segundos. Marque o instante 11,78 com o auxílio a régua, clicando sobre a onda no local exatamente abaixo a essa marca. Ao pressionar SHIFT + END, selecione tudo até o final e depois apagamos com DELETE.

Agora o arquivo está pronto para ser exportado no formato RAW. Clique no menu superior em “Ficheiro”, “Exportar...”. Ao abrir o diálogo de salvar arquivo, escolha a opção no combo box localizada na parte inferior à direita “Outros ficheiros sem compressão”. Agora, clique em “Opções” logo abaixo e escolha o cabeçalho “Raw (header-less)” e a opção “unsigned 8 bit PCM”. Dê um nome e utilize a extensão “.raw” e salve.

O arquivo salvo é um arquivo binário, sem cabeçalho, onde cada byte corresponde a uma amostra, variando de 0 a 255. o valor 0 corresponde a parte mais inferior do gráfico, o valor 255 a parte mais superior do gráfico e o valor 128 corresponde à linha central.

4.2. O processo de conversão de 8 bits para 1 bit

Iremos apresentar dois métodos de conversão de um arquivo RAW de 8 bits não sinalizado para 1 bit do MSX. O primeiro corresponde à limiarização, onde os valores menores que 128 (linha média) são convertidos para 0, enquanto que os demais são convertidos para 1. O segundo é método do gradiente, onde assume-se o valor 1 quando o

valor atual é maior ou igual ao anterior (a curva sobe) e 0 quando o valor anterior é menor (a curva desce).

O primeiro método apresentou deficiências quando a onda oscila abaixo ou acima da linha média do gráfico, pois a forma de onda não é reproduzida corretamente. Já o segundo método apresentou melhores resultados, pois é capaz de representar melhor a forma de onda. A figura 4.3 mostra a comparação gráfica entre os métodos linear e gradiente.

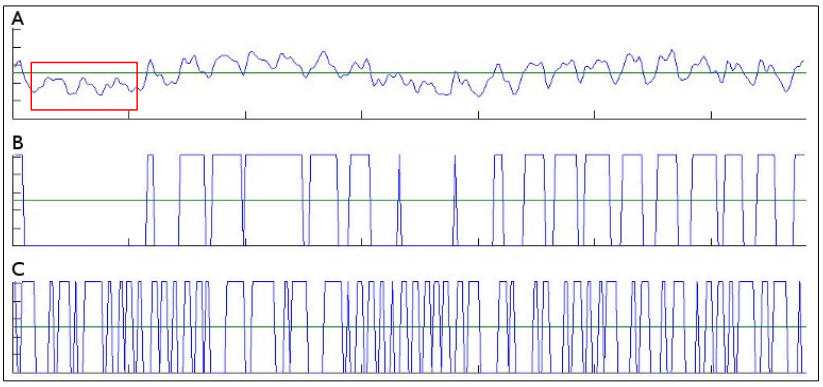

Figura 4.3. Comparação dos métodos linear (B) e gradiente (C) para uma dada onda de 8 bits (A).

Observe no retângulo vermelho da figura 4.3, que as oscilações abaixo da linha média em (A) não foram corretamente representadas em (B), onde há somente um valor. Já em (C), podemos observar que essas oscilações foram bem representadas.

Uma vez convertidos os valores para 0 ou 1, devemos ainda converter esses valores de byte para bit. Veja o exemplo a seguir.

```
Valores 8-bit unsigned: 200, 145, 180, 128, 130, 80, 45, 80
Valores obtidos pela limiarização: 1, 1, 1, 1, 0, 0, 0, 0
Conversão para bits: &B11110000
Valor do byte: 240 (&HF0), que contém 8 amostras de som.
```

5. O Formato 4-bit

Podemos observar na figura 4.3 do capítulo anterior, que a onda de 1 bit resultante da conversão é quadrada, prejudicando a qualidade do som.

Além do bit do clique do teclado, podemos também modelar uma onda alterando a intensidade de volume de um canal do PSG [11]. Como o volume do PSG possui 16 intensidades, podemos quantizar a amostra da onda em 4 bits em vez de apenas 1. Esse formato permite uma melhor representação da forma de onda. Entretanto, ele possui um preço alto: o arquivo possui 4 vezes o tamanho do arquivo de 1 bit. Assim, em vez de executar, por exemplo, 12 segundos para 11 KHz, irá executar somente 3 segundos.

A figura 5.1 apresenta o resultado da conversão da mesma onda utilizada na figura 4.3 para o formato 4 bits. Compare os resultados.

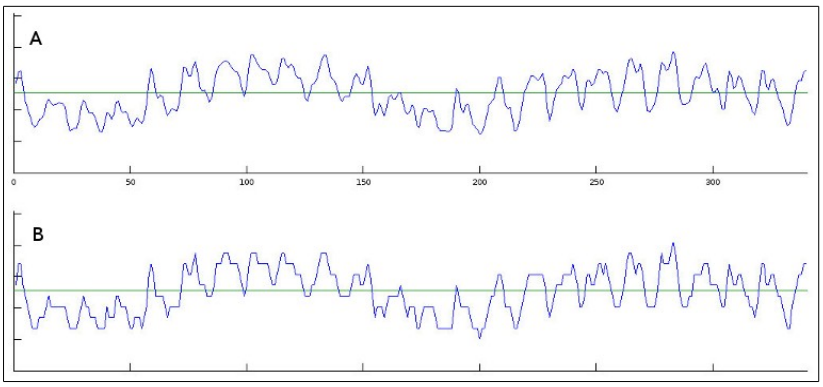

Figura 5.1. Resultado para conversão no formato 4 bits (mesma onda da fig. 4.3).

A conversão para esse formato consiste em quantizar o espaço de intensidades em 4 bits. Dessa forma, cada valor de 8 bits é atribuído à faixa de 4 bits em que ele se situar.

Deve-se notar que o formato de onda de 1 bit produz som sempre com o volume máximo. Entretanto, esse formato varia o volume e, consequentemente, o som fica mais baixo que o formato de 1 bit.

O procedimento de execução é semelhante ao do formato 1 bit e é apresentado a seguir.

```
p ← end_inicial;

ajuste_reg_PSG(7, 254); // Seleciona canal A
ajuste_reg_PSG(0, 0); // Freq. fina canal A
ajuste_reg_PSG(1, 0); // Freq. grossa canal A

repita

  dado ← ler_memoria(p);

  para i ← 1 até 2 faça
    volume ← nibble(dado, i);
    ajuste_reg_PSG(8, volume);
    para t ← 1 ate delay faça;
  fim_para

  p ← p+1;

até p >= endereço_final;
```

O *delay* deverá sempre estar entre duas amostras de som.

Por fim, devido ao fato de uma frequência ser utilizada em vez de um pulso para a geração do som, o sinal da frequência é misturado com o sinal do som, produzindo uma interferência. Foi observado que para as frequências mais altas só sinal, a interferência é menor.

6. Créditos e Referências

Este artigo foi escrito por Marcelo Teixeira Silveira.

Data: Junho de 2020.
Site: http://marmsx.msxall.com
E-mail: flamar98@hotmail.com

Referências:

[1] - Video Hits, Produzido pelo grupo CPM.
[2] - Digivoix, Produzido por Hartard Frederic e publicado na revista francesa Hebdogiciel, números 164 a 168, 1986.
[3] - O Livro Vermelho do MSX, Avalon Software, editora McGraw Hill.
[4] - MSX2 Technical Handbook, ASCII Corporation, 1987.
[5] - Fundamentos de Física 2, Halliday e Resnick. Editora Livros Técnicos e Científicos, 1994.
[6] - Sons, Curso de Basic, Site MarMSX. Em http://marmsx.msxall.com
[7] - Digital audio. Em http://en.wikipedia.org/wiki/Digital_audio
[8] - Pulse-code Modulation. Em: http://en.wikipedia.org/wiki/Puse-code_modulation
[9] - 44,100 Hz. Em http://en.wikipedia.org/wiki/44,100_Hz
[10] - Audacity. Em http://www.adacityteam.org
[11] - MSX Música, José Maurício Bussab. Editora McGraw-Hill, 1987.