MSX Article

MARMSX

Menu de Barras na Screen 0

Resumo

O objetivo desse artigo é mostrar que é possível fazer um menu de barras com inversão das cores dos caracteres na screen 0 do MSX, utilizando a tabela de caracteres do VDP.

1- Introdução

Um programa pode possuir diversos recursos disponíveis ao usuário. Entretanto, o acesso a esses recursos pode ser complicado, dependendo de como o autor os estabelece. É muito comum o uso de teclas de atalho para esse fim, obrigando ao usuário um longo estudo e memorização dessas teclas.

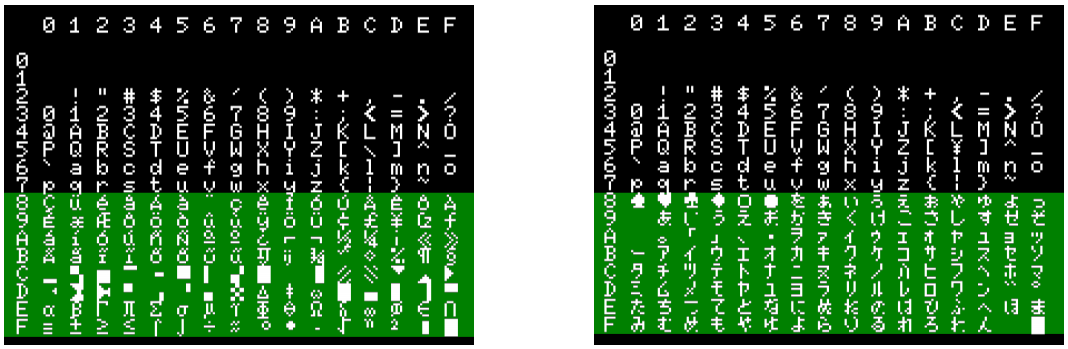
O menu de barras é uma forma que o programador oferece a seus usuários a facilidade de descobrir e acessar os diversos recursos disponíveis no produto, de maneira bastante clara e amigável. Para construí-lo, é necessário entender primeiramente como o MSX gera os caracteres na tela.

2- A geração de caracteres no modo texto

As telas em modos texto do MSX possuem duas áreas de memória VRAM destinadas à geração dos caracteres na tela. A primeira é uma tabela que mapeia a tela, indicando qual caractere será exibido em uma determinada posição, chamada de tabela de nomes. A segunda é uma tabela que contém o desenho de cada caractere correspondente ao código da primeira, chamada de tabela de padrões de caracteres.

O MSX segue um padrão internacional de codificação de caracteres chamado ASCII. O padrão ASCII define 128 caracteres, variando de 0 a 127. Entretanto, como um byte pode codificar até 256 caracteres, é possível utilizar os 128 bytes restantes para representar símbolos quaisquer. Normalmente, essa área é customizada em cada micro.

A tabela 2.1 apresenta os caracteres de dois modelos de MSX, onde a área preta corresponde ao padrão ASCII, enquanto que a área verde corresponde à área livre. Note que na área preta os símbolos são iguais nos dois micros. Entretanto, na área verde, os símbolos são diferentes.



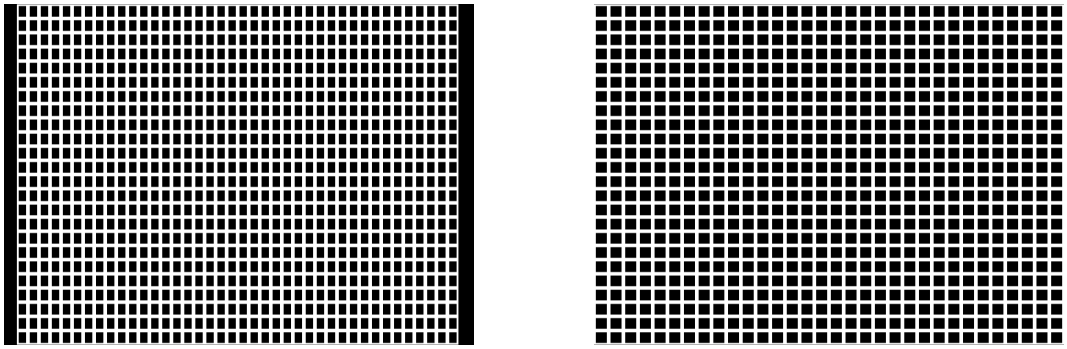
a) MSX Expert Gradiente 1.1 b) MSX Turbo-R GT

Figura 2.1. Exemplos de tabelas de caracteres do MSX.

Observando a figura 2.1, a letra “M” maiúscula está na linha 4 e na coluna D, formando o número hexadecimal 4D. Convertendo esse valor para decimal, concluímos a letra “M” possui código ASCII igual a 77.

2.1. A Divisão da Tela

A tela é dividida em blocos de 8x6 pixels na screen 0, e 8x8 pixels na screen 1, onde cada bloco contém o desenho de um caractere. Assim, é possível ter 40 colunas de texto para a screen 0, 32 colunas para a screen 1 e 24 linhas em ambas as screens (figura 2.2).



a) Screen 0 b) Screen 1

Figura 2.2. Divisão da tela do MSX no modo texto.

2.2. A Tabela de Nomes

A tabela de nomes irá conter o mapeamento de cada bloco ou caractere contido na tela (vide figura 2.2), indicando qual é o caractere que será desenhado no bloco. Os blocos são numerados da seguinte forma:

Screen 0	Screen 1
0 1 2 3 … 39 40 41 42 43 … 79 … 920 921 922 923 … 959	0 1 2 3 … 31 32 33 34 35 … 63 … 736 737 738 739 … 767

Tabela 2.1. Numeração dos blocos.

Assim, o bloco localizado no canto superior esquerdo da screen 0 é a entrada 0 da tabela de nomes, enquanto que o bloco superior direito da screen 1 possui a entrada igual a 31 na tabela. Ao alterarmos o código de caractere dessas duas entradas na tabela de nomes, estaremos modificando o caractere exibido nas respectivas posições da tela, conforme mostra a figura 2.3.

a) Screen 0 b) Screen 1

Figura 2.3. Exemplo de modificação na tabela de nomes.

O endereço da tabela de nomes na screen 0 começa na posição 0 da VRAM, enquanto que na screen 1 começa na posição 6144. Assim, temos:

	Endereço inicial	Endereço final	Tamanho
Screen 0	0	959	960 bytes
Screen 1	6144	6911	768 bytes

Tabela 2.2. Endereço da tabela de nomes.

O comando em Basic VPOKE permite alterar um determinado endereço de memória de vídeo ou VRAM. Para alterar o caractere de um bloco na tela, devemos utilizar:

VPOKE endereço_inicial + número_do_bloco, código_do_caractere

Na figura 2.3, foi utilizado o comando VPOKE 0, 67 para desenhar a letra “A” no canto superior esquerdo da screen 0. Já para a screen 1, foi utilizado o comando VPOKE 6144+31, 67 para desenhar a letra “A” no canto superior direito.

Para facilitar a identificação do número do bloco da tela, utilize as seguintes equações:

$NB = X + (Y \times 40)$ Screen 0

$NB = X + (Y \times 32)$ Screen 1

Onde *X* corresponde à coluna, *Y* corresponde à linha e NB é o número do bloco.

2.3. A Tabela de Caracteres

A tabela de caracteres contém o desenho de cada letra, numero ou símbolo que será exibido na tela. Essa tabela possui 256 entradas, permitindo desenhar 256 caracteres. O valor de cada entrada está diretamente relacionado ao código de caracteres padrão mencionado anteriormente.

Independente do modo de tela, cada caractere é desenhado em um bloco de 8x8 pixels monocromáticos, ocupando 8 bytes. Dessa forma, a tabela de caracteres ocupa 2048 bytes no total. No caso da screen 0, ela irá somente utilizar 6 colunas de cada caractere.

A localização da tabela de caracteres é apresentada a seguir.

	Endereço inicial	Endereço final	Tamanho
Screen 0	2048	4095	2048 bytes
Screen 1	0	2047	2048 bytes

Tabela 2.3. Endereço da tabela de caracteres.

O endereço inicial de cada caractere é calculado utilizando-se as seguintes fórmulas:

$P = 2048 + (C \times 8)$ Screen 0

$P = (C \times 8)$ Screen 1

Onde *P* é a posição na tabela e *C* é o código do caractere em questão.

Por exemplo, a letra “A” maiúscula possui código ASCII igual a 65. Portanto, a posição da VRAM onde começa o desenho dessa letra na screen 0 fica em:

2048 + (65 x 8) = 2568

Cada caractere possui as dimensões de 8x8 pixels. Cada byte representa uma linha do desenho, onde cada bit representa uma coluna, indicando se é a cor de fundo (0) ou de frente (1) a ser exibida. Essa representação é a mesma utilizada pelos sprites do MSX.

A letra “A” maiúscula tem a seguinte representação na tabela de caracteres da screen 0:

```
VRAM   Bits
2568 = 0 0 1 0 0 0 0 0
2569 = 0 1 0 1 0 0 0 0
2570 = 1 0 0 0 1 0 0 0
2571 = 1 0 0 0 1 0 0 0
2572 = 1 1 1 1 1 0 0 0
2573 = 1 0 0 0 1 0 0 0
2574 = 1 0 0 0 1 0 0 0
2575 = 0 0 0 0 0 0 0 0
```

VPOKE 2568, &B00100000

O desenho do caractere possui sempre o tamanho de 8x8 pixels. Entratanto, a screen 0 só irá utilizar 6x8 pixels desse desenho.

O desenho da letra “A” está definido nos endereços da VRAM de 2568 a 2575.

Podemos modificar os desenhos dos caracteres, ou seja, modificar a fonte de texto do MSX, alterando os dados da tabela de caracteres. Além disso, é possível salvar a fonte modificada para posterior carregamento. Exs:

BSAVE “ALFABETO.ALF”, 2048, 4095, S
BSAVE “ALFABETO.ALF”, 0, 2047, S

Obs: Ao alterarmos o desenho da letra “A” na tabela de caracteres, estaremos alterando o desenho de todos blocos da tela que possuam o valor igual a 65 na tabela de nomes.

O programa a seguir escreve na tela do MSX o desenho de um dado caractere, de acordo com o seu código ASCII.

```
10 INPUT "Codigo ASCII";A
20 E = 2048 + A*8
30 FOR I=E TO E+7
40 V$ = BIN$(VPEEK(I))
50 V$ = RIGHT$("0000000"+V$,8)
60 PRINT V$
70 NEXT I
```

3- Invertendo as cores dos caracteres

O menu de barras normalmente inverte as cores do texto da opção corrente. Dessa forma, podemos inverter as cores dos pixels de uma letra, alterando a tabela de caracteres. Para tal, basta aplicar a operação booleana NOT aos bits do desenho de cada caractere. Por exemplo, podemos inverter as cores do caractere “A” através do seguinte programa:

```
10 FOR E=2568 TO 2575
20 VPOKE E,&HFF AND (NOT VPEEK(E))
30 NEXT E
```

Como as operações de cálculo do MSX retornam um número de 2, 4 ou 8 bytes, ao utilizarmos o operador NOT, será retornado um número maior que 1 byte. Assim, aplicamos a operação lógica *&HFF AND <valor>*, que irá retornar um número inteiro entre 0 e 255.

Conforme dito na seção 2.3, quando alteramos o desenho de um caractere, isso irá afetar todos os caracteres que possuírem o mesmo código na tela. Assim, todas as letras “A” da tela serão invertidas, conforme mostra a figura 3.1.

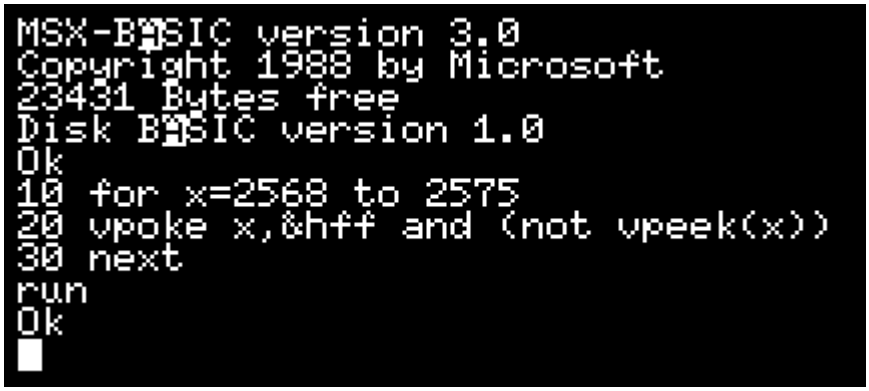


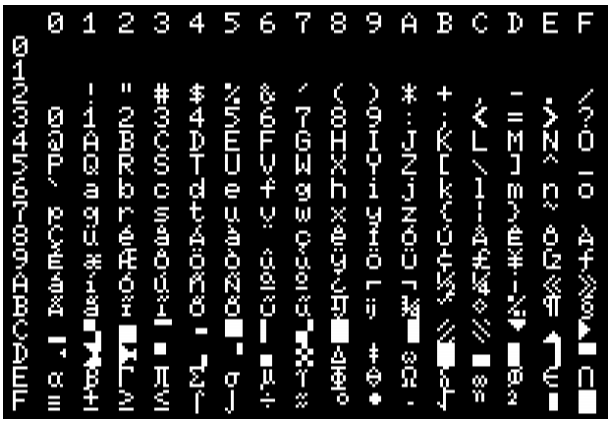
Figura 3.1. Inversão das cores do caractere “A”.

4- Estratégias para a inversão de cores da barra do menu

Como a alteração do desenho de um caractere afeta todos os caracteres com o mesmo código ASCII, é necessário que se adote alguma estratégia que afete somente os caracteres utilizados no menu. Assim, o primeira parte da tabela ASCII não poderá ser afetada.

Podem ser usadas duas estratégias para tingir tal fim: a primeira é repetir os caracteres de texto na segunda metade da tabela ASCII com as cores invertidas. Porém, essa parte da tabela possui alguns caracteres especiais como as letras acentuadas do português, que seriam perdidos nessa operação. A segunda estratégia seria criar uma pequena área na segunda parte da tabela ASCII, de preferência em cima de figuras, invertendo em tempo real os caracteres da linha atual do menu. Isto consumiria um espaço da largura máxima do texto do menu.

A figura 4.1 apresenta os caracteres do Expert MSX (Brasil), bem como o programa em Basic para gerá-los.



```
10 SCREEN 0
20 FOR F=0 TO 15
30 LOCATE F*2+2,0
40 PRINT HEX$(F)
50 LOCATE 0,F+1
60 PRINT HEX$(F)
70 NEXT F
80 FOR X=0 TO 15
90 FOR Y=0 TO 15
100 LOCATE X*2+2,Y+1
110 A = X+(Y*16)
120 IF A < 32 THEN 140
130 PRINT CHR$(A)
140 NEXT Y,X
```

Figura 4.1 – Tabela ASCII completa do Expert MSX 1 da Gradiente.

A composição do código de caractere em hexadecimal na figura 4.1 é feita, pegando-se a linha seguida da coluna. Por exemplo, a letra “a” está localizada na linha 6 e coluna 1, formando o número hexadecimal 61 (97 em decimal).

Observe que da posição &H80 (128) até &HBF (191) estão localizados os caracteres acentuados da língua portuguesa. Essa área não poderá ser afetada, caso deseje-se utilizar a acentuação em português.

Uma vez analisados os caracteres do MSX, vejamos ver como a segunda estratégia pode ser implementada.

Tomando-se por base a figura 3.1, imagine que desejamos inverter a primeira letra da primeira linha da tela, ou seja, a letra “M”. Em vez de inverter diretamente o desenho do caractere “M” na tabela de caracteres, pegamos outra posição qualquer da tabela de caracteres que não esteja sendo utilizada, e copiamos o “M” invertido para lá. Podemos utilizar, por exemplo, a posição 250.

```
10 E1 = 2048 + 8*ASC("M") : E2 = 2048 + 8*250
20 FOR I=0 TO 7
30 VPOKE E2+I,&HFF AND (NOT VPEEK(E1+I))
40 NEXT I
```

O resultado da estratégia anterior pode ser observado na figura 4.2. Observe o “M” invertido na posição &HFA (250).

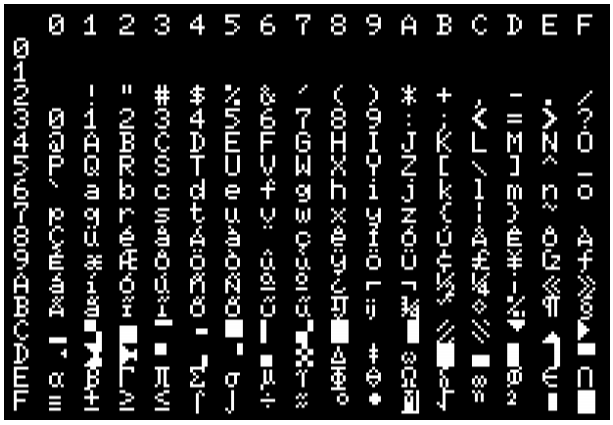


Figura 4.2. O caractere “M” invertido.

Não basta só criar esta inversão, pois a letra “M” na posição 0,0 da tela ainda possui o código igual a 77 na tabela de nomes. Dessa forma, temos que alterar o valor da posição 0 da tabela de nomes de 77 para 250. Procedendo desta maneira, garantimos que somente os caracteres sob a barra terão suas cores invertidas.

Podemos também utilizar essa estratégia para *n* caracteres. A figura 4.3. mostra o programa e o resultado obtido para os 8 caracteres da primeira linha. Observe que somente as letras dessa região foram afetadas.

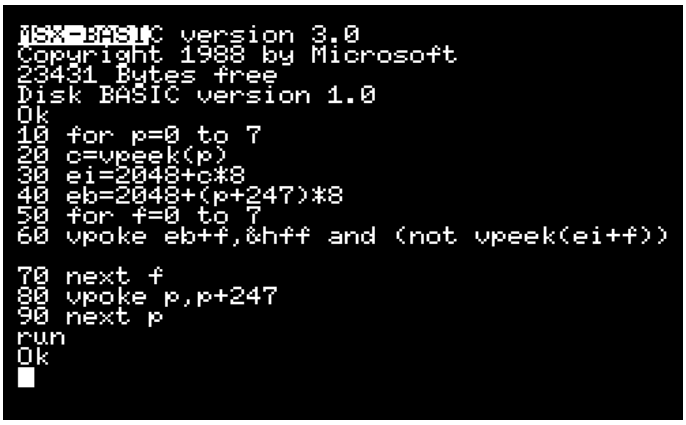


Figura 4.3. Resultado da inversão para 8 caracteres.

Obs: a instrução “SCREEN 0” reinicia todas as tabelas, fazendo com que as alterações sejam perdidas.

5- A construção do menu de barras

Foi visto na seção anterior, que ambas as estratégias alteram o código ASCII do texto sob a barra do menu na tabela de nomes. Uma vez que a barra serve para navegar sobre diversas opções do menu na tela, ele sai da posição atual e vai para outra. Assim, é necessário armazenar os códigos ASCII da posição atual, de forma que seja possível restaurar o texto original, quando o menu for movimentado.

A solução mais simples para resolver esse problema seria armazenar todos os códigos de caractere das opções do menu em um vetor. Daí, somente é necessário reescrever a posição atual, consultando esse vetor.

O programa a seguir cria um menu simples, utilizando a segunda estratégia.

```
05 ' Desenha menu
10 DIM OP$(5)
20 OP$(1) = "Editar  "
30 OP$(2) = "Cortar  "
40 OP$(3) = "Carregar"
50 OP$(4) = "Salvar  "
60 OP$(5) = "Sair    "
70 SCREEN 0:COLOR 15,0,0:WIDTH 40
80 PRINT"Menu":PRINT
90 FOR F=1 TO 5
100 PRINT OP$(F)
110 NEXT F
120 OP=1
130 GOSUB 500
200 ' Controle
210 A$=INKEY$:IF A$="" THEN 210
220 A = ASC(A$)
230 IF A<>30 AND A<>31 THEN 210
240 IF A=30 AND OP=1 THEN 210
250 IF A=31 AND OP=5 THEN 210
260 LOCATE 0,OP+1:PRINT OP$(OP)
270 IF A$=CHR$(30) THEN OP=OP-1
280 IF A$=CHR$(31) THEN OP=OP+1
290 GOSUB 500
300 GOTO 210
500 ' Desenha barra
510 E = (OP+1)*40
520 FOR P=0 TO 7
530 C = VPEEK(E+P)
540 EI = 2048 + C*8
550 EB = 2048 + (P+240)*8
560 FOR F=0 TO 7
570 VPOKE EB+F, &HFF AND (NOT VPEEK(EI+F))
580 NEXT F
590 VPOKE P+E,P+240
600 NEXT P
610 RETURN
```

A modificação em tempo real da tabela ASCII é lenta para a linguagem Basic do MSX.

O programa a seguir irá utilizar a primeira estratégia, criando um “clone” da primeira metade da tabela ASCII na segunda metade, invertendo as cores. Além disso, já cria uma lista anexa com os caracteres modificados para as cores invertidas.

```
05 ' Desenha menu
06 SCREEN 0:COLOR 15,0,0:WIDTH 40
07 GOSUB 500
10 DIM OP$(5,2)
20 OP$(1,1) = "Editar  "
30 OP$(2,1) = "Cortar  "
40 OP$(3,1) = "Carregar"
50 OP$(4,1) = "Salvar  "
60 OP$(5,1) = "Sair    "
70 FOR F=1 TO 5
80 FOR C=1 TO 8
90 OP$(F,2) = OP$(F,2) + CHR$(ASC(MID$(OP$(F,1),C,1)) + 128)
100 NEXT C,F
110 PRINT"Menu":PRINT
120 FOR F=1 TO 5
130 PRINT OP$(F,1)
140 NEXT F
150 OP=1
160 GOTO 290
200 ' Controle
210 A$=INKEY$:IF A$="" THEN 210
220 A = ASC(A$)
230 IF A<>30 AND A<>31 THEN 210
240 IF A=30 AND OP=1 THEN 210
250 IF A=31 AND OP=5 THEN 210
260 LOCATE 0,OP+1:PRINT OP$(OP,1)
270 IF A$=CHR$(30) THEN OP=OP-1
280 IF A$=CHR$(31) THEN OP=OP+1
290 LOCATE 0,OP+1:PRINT OP$(OP,2)
300 GOTO 210
500 ' Desenha tabela
510 FOR C=32 TO 127
520 EI = 2048 + C*8
530 ED = 2048 + (C+128)*8
540 FOR F=0 TO 7
550 VPOKE ED+F, &HFF AND (NOT VPEEK(EI+F))
560 NEXT F,C
570 RETURN
```

Há um tempo maior para a criação da tabela invertida, mas a execução do menu é muito mais rápida que na estratégia anterior.

Os códigos de cada evento esperado para o menu são apresentados a seguir.

```
if a=30 then <tratamento para cima>
if a=31 then <tratamento para baixo>
if a=29 then <tratamento para esquerda>
if a=28 then <tratamento para direita>
if a=32 then <tratamento para o espaço>
if a=27 then <tratamento para o ESC>
if a=13 then <tratamento para o enter>
```

6 - Extra: o mapa da VRAM do MSX 1

Endereço	Screen 0	Screen 1	Screen 2	Screen 3
0	Nomes 0	Caracteres 7	Caracteres 12	Caracteres 17
959				
2947				
2048	Caracteres 2			Nomes 15
4095				
6143				
6144		Nomes 5	Nomes 10	
6911				
6912		Atributos Sprites 8	Atributos Sprites 13	Atributos Sprites 18
7039				
8192		Cores 6	Cores 11	
8223				
14335				
14336		Padrões Sprites 9	Padrões Sprites 14	Padrões Sprites 19
16383				

Fonte: Revista CPU-MSX, número 15.

Obs: os números que acompanham a descrição de cada trecho de memória é o valor da instrução em Basic BASE(n), que obtém o endereço inicial dessas tabelas.

7- Créditos e bibliografia

O artigo foi escrito por Marcelo Silveira, Engenheiro de Sistemas e Computação, formado pela Universidade do Estado do Rio de Janeiro.

Data: outubro de 2003.
Revisão 1: julho de 2017.
Revisão 2: novembro de 2017.

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

Referências bibliográficas:

- O Livro Vermelho do MSX, Avalon Software, editora Mc Grall Hill.