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 eles pode ser complicado, dependendo de como o autor o 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.

2- As Tabelas dos Caracteres na Screen 0

Toda letra, número, símbolo ou caractere gráfico é representado por um número, de 0 a 255, chamado de código ASCII. Por exemplo, a palavra “CARRINHO” tem suas letras representadas pelos códigos 67, 65, 82, 82, 73, 78, 72 e 79.

Os valores da tabela de 0 a 127 são padrões para maioria dos micros, onde estão as letras, números e símbolos comuns, como !, #, @ etc. Varia-se então, de micro a micro, os valores compreendidos entre 128 a 255.

A tabela 2.1 mostra os principais caracteres gráficos e seus códigos correspondentes.

Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car	Cód	Car
000		001		002		003		004		005		006		007		008		009		010		011	
012		013		014		015		016		017		018		019		020		021		022		023	
024		025		026		027		028		029		030		031		032		033	!	034	"	035	#
036	$	037	%	038	&	039	'	040	(	041	)	042	*	043	+	044	,	045		046	.	047	/
048	0	049	1	050	2	051	3	052	4	053	5	054	6	055	7	056	8	057	9	058	:	059	;
060	<	061	=	062	>	063	?	064	@	065	A	066	B	067	C	068	D	069	E	070	F	071	G
072	H	073	I	074	J	075	K	076	L	077	M	078	N	079	O	080	P	081	Q	082	R	083	S
084	T	085	U	086	V	087	W	088	X	089	Y	090	Z	091	[	092	\	093	]	094	^	095	_
096	`	097	a	098	b	099	c	100	d	101	e	102	f	103	g	104	h	105	i	106	j	107	k
108	l	109	m	110	n	111	o	112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127									

Tabela 2.1. Códigos ASCII padrão.

Os modos de texto do MSX possuem duas tabelas para representar os caracteres na tela. Uma tabela contendo o código ASCII de cada posição da tela e outra tabela contendo o desenho de cada caractere. A primeira é a tabela de nomes, e a segunda a tabela de caracteres.

A screen 0 do MSX possui 40 colunas e 24 linhas (quando no estado de key off). Portanto, esta tela ocupa 960 bytes (40x24) na memória de vídeo (VRAM).

A tabela de nomes contém os códigos ASCII de todos os 960 caracteres que estão sendo exibidos na tela naquele exato momento. A alteração do valor de qualquer posição desta memória, irá alterar instantaneamente a exibição de um caractere na tela.

Essa tabela começa na posição 0 e termina na posição 959 da VRAM. O valor 0 corresponde ao canto superior esquerdo, enquanto que o valor 959 corresponde ao canto inferior direito. Para testar isso, utilize o comando em Basic:

VPOKE endereço, código_ascii

Por exemplo, VPOKE 0,67 desenha o caractere “a” no canto superior esquerdo da tela.

A relação entre coordenada coluna, linha e endereço de memória e expressa por:

$$E = X + (Y \times 40)$$

Onde X varia de 0 a 39 e Y varia de 0 a 23.

A tabela de caracteres, por sua vez, contém o desenho de cada letra, numero, símbolo ou caracter gráfico que será exibido na tela.

Ela começa na posição 2048 (&H800) e termina na posição 4095 (&HFFF), onde cada desenho de caractere ocupa 8 bytes na memória. Assim, ela ocupa 2048 (256x8) bytes no total. A posição de cada desenho de caractere nessa tabela possui relação direta com o valor do código ASCII.

Para se calcular a posição inicial de cada caractere na tabela de caracteres, devemos utilizar a expressão:

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

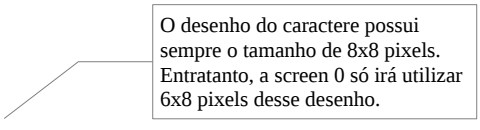
Onde P é a posição na tabela e C é o código ASCII do caractere em questão.

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

```
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
```



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

Ao alterarmos o desenho da letra “A” nessa tabela, estaremos alterando o desenho de todos caracteres da tela que posuam o valor ASCII igual a 65.

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

Para inverter as cores dos caracteres de uma letra, basta aplicar a operação booleana NOT aos bits do desenho do caractere. Por exemplo, podemos inverter os caracteres “A” da tela, 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, quando alteramos o desenho de um caractere, isso irá afetar todos os caracteres que possuirem o mesmo código na tela. Assim, todas as letra “A” da tela serão invertidas, conforme mostra a figura 3.1.

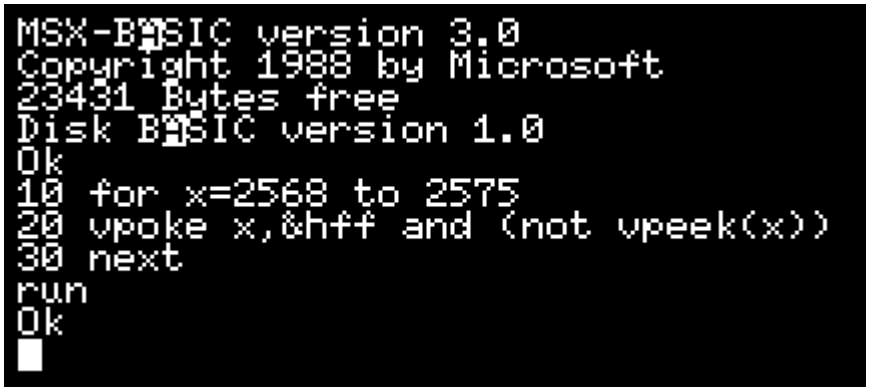


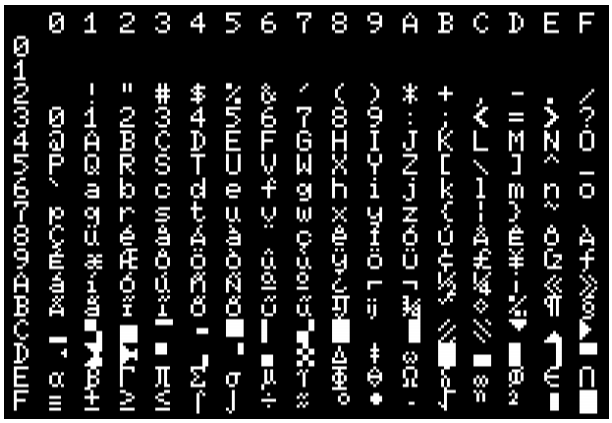
Figura 3.1. Inversão das cores do caractere “A”.

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 da barra. 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, invertendo em tempo real os caracteres da linha atual do menu. Isto consumiria a largura do texto do menu e poderia ser utilizado em áreas da tabela ASCII que tivessem desenho de figuras.

A figura 4.1 apresenta a tabela ASCII completa do Expert MSX (Brasil), bem como o programa em Basic para gerá-la.



```
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 ASCII 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 protuguesa.

Uma vez analisada a tabela ASCII do MSX, vamos 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 ASCII 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).

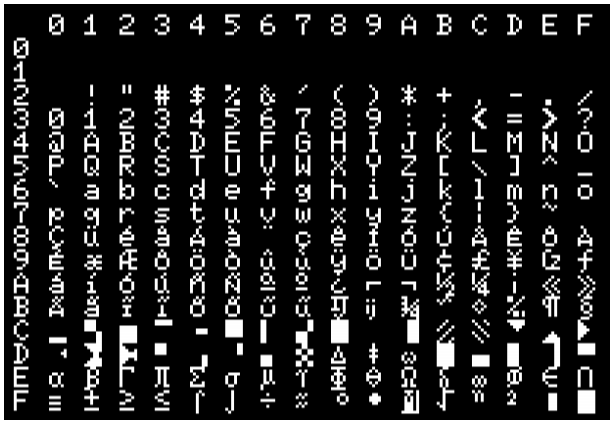


Figura 4.2. O caractere “M” invertido.

Não basta só criar esta inversão, pois a letra “M” na posição 0,0 da tela ainda possui código ASCII igual a 77 na tabela de nomes. Dessa forma, temos que alterar o código ASCII 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.

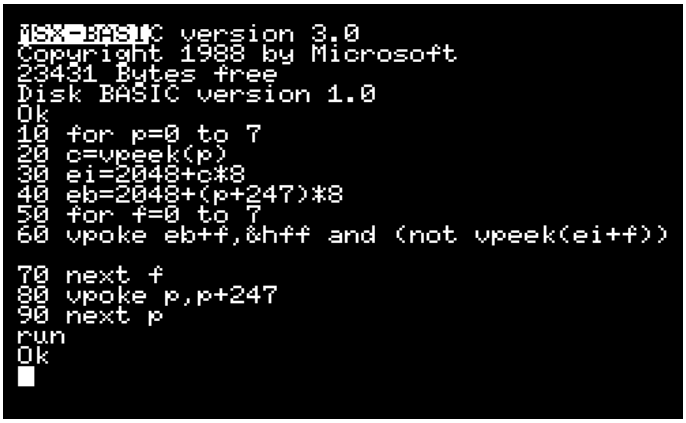


Figura 4.3. Resultado da inversão para 8 caracteres.

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 isso seria armazenar todas as opções do menu em um vetor. Daí, somente é necessário reecrever a posição atual, consultando o 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.

```
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 é apresentado 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

<table>
<tr><th>Endereço</th><th>Screen 0</th><th>Screen 1</th><th>Screen 2</th><th>Screen 3</th></tr>
<tr><td>0</td><td rowspan="2">Nomes
0</td><td rowspan="3">Padrões
7</td><td rowspan="5">Padrões
12</td><td rowspan="3">Padrões
17</td></tr>
<tr><td>959</td></tr>
<tr><td>2947</td><td></td></tr>
<tr><td>2048</td><td rowspan="2">Padrões
2</td><td rowspan="3"></td><td rowspan="2">Nomes
15</td></tr>
<tr><td>4095</td></tr>
<tr><td>6143</td><td rowspan="10"></td><td rowspan="3"></td></tr>
<tr><td>6144</td><td rowspan="2">Nomes
5</td><td rowspan="2">Nomes
10</td></tr>
<tr><td>6911</td></tr>
<tr><td>6912</td><td rowspan="2">Atributos Sprites
8</td><td rowspan="2">Atributos Sprites
13</td><td rowspan="2">Atributos Sprites
18</td></tr>
<tr><td>7039</td></tr>
<tr><td>8192</td><td rowspan="3">Cores
6</td><td rowspan="3">Cores
11</td><td rowspan="3"></td></tr>
<tr><td>8223</td></tr>
<tr><td>14335</td></tr>
<tr><td>14336</td><td rowspan="2">Padrões Sprites
9</td><td rowspan="2">Padrões Sprites
14</td><td rowspan="2">Padrões Sprites
19</td></tr>
<tr><td>16383</td></tr>
</table>

Fonte: Revista CPU-MSX, número 9.

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: julho de 2017.

e-mail: flamar98@hotmail.com
homepage: marmsx.msxall.com

Referências bibliográficas:

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