Apêndice

Estrutura da Memória e do Arquivo de Imagem

A memória de vídeo do MSX (VRAM) é uma memória separada da memória principal (RAM). A VRAM depende do modelo do MSX, variando de 16 Kb nos MSX 1 até 128 Kb nos MSX 2 em diante.

Esta memória possui as informações no qual o processador de vídeo do MSX varre para poder desenhar o conteúdo da tela em um ciclo de 60 Hz ou 50 Hz, dependendo do país em que o MSX foi desenvolvido. Os MSXs 2 já possuem uma seleção para escolher tal freqüência de varredura.

Normalmente no MSX quando salvamos um arquivo em disco, fazemos um *dump* de memória de vídeo, ou seja, copiamos diretamente a informação desta memória para o arquivo, sem fazer nenhum tipo de compressão de dados ou outra operação qualquer. Os arquivos binários de MSX possuem os 7 primeiros bytes reservados para alguma informação, o que chamamos de *header* do arquivo.

A estrutura do *header* é a seguinte:

Byte	Descrição
00	Descrição do tipo de arquivo: FF = Programa Basic, FE = Binario
01	Endereço inicial - LSB
02	Endereço inicial - MSB
03	Endereço final - LSB
04	Endereço final - MSB
05	Endereço de execução - LSB
06	Endereço de execução - MSB

Nota: LSB é a menor parte de um número de 16 bits e MSB é a maior parte de um número de 16 bits. Por exemplo, o número hexadecimal D400:

MSB = D4
LSB = 00

Ele se apresenta realmente invertido na memória: 00 D4.

Características das Telas do MSX

Screen 2 / 4

Característica	Descrição						
Dimensões	256 x 192						
Cores	16						
Endereço Inicial	Base(11) para cores e base(12) para preenchimento cores: 2000H, padrão: 0						
Header	FE	0	0	FF	37	0	0
Tamanho de Cada Área	6144 (1800H)						
Tamanho Geral	12288 (&H3000): 256 x 192 x 2 / 8						
Pixel x Memória	2 bytes = 1 linha de 8 pixels						
Representação da Cor	Valor puro de 0 a 15, de acordo com a tabela de cores do MSX 1.						
Para salvar a tela	*10 screen 2* *20 rem código do desenho* ... *100 bsave "nome.src",base(12),base(11)+6143,s*						
Para carregar a tela	*10 screen 2* *20 bload "nome.src",s*						

Tabela de cores do MSX 1:

Fig 1 – Cores do MSX

A Screen 2 tem um modo particular de salvar as telas gráficas. Isto se deve ao fato da limitação da memória de vídeo de 16 K, para o MSX 1.

A tela é dividida em blocos de 8 x 8, onde cada byte representa uma linha deste bloco (vide fig 2).

Na Screen 2 existem 2 tipos de arquivos desenvolvidos: o *dump* da memória, contendo somente o header e os dados da imagem e a tela dentro de um programa.

A imagem do *dump* é carregada e salva conforme todos os modos de tela restantes do MSX. Ela é carregada com a opção ,s. A única coisa em que se diferem estes modos é a área de memória a salvar, que será visto adiante.

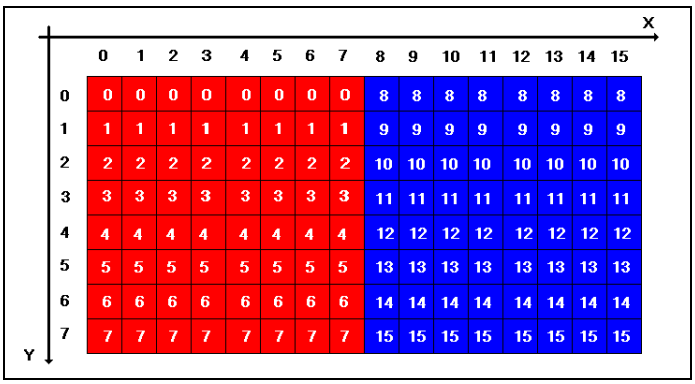


Fig 2 - Padrão de preenchimento

Conforme a figura ao lado, verificamos que a varredura de memória é feita por uma linha de 8 x 8 pixels. Depois, pula-se para o próximo bloco, ao lado.

No eixo x, temos 32 (256/8) posições, enquanto no eixo y, temos 192 posições.

Dois bytes são necessários para cada linha de 8 pixels. O primeiro byte representa a cor de frente e a cor de fundo. A primeira metade do byte indica um valor de 0 a 15 referente a cor de frente, enquanto que a metade restante, a cor de fundo. O segundo byte representa o padrão de exibição dos pixels de frente ou fundo. Quando o bit for 1, será a cor de frente e quando for 0, a cor de fundo. Portanto, se quiser colocar em uma linha, metade com azul escuro e metade com vermelho, na coordenada 0,0 deverá fazer:

Código de cor do azul: 4, ou, 100 em binário
Código de cor do vermelho: 8, ou, 1000 em binário

VRAM (0000H) = 11110000
VRAM (2000H) = 01001000


Fig 3.

O arquivo gerado no disco repete a estrutura gerada na VRAM: A primeira metade do arquivo guarda dados relativos ao preenchimento, enquanto que a segunda metade guarda os dados relativos às cores. Existe uma parte entre as duas áreas de memória, precisamente entre 1800H e 1FFFH, que correspondem à tabelas da screen 2 e devem ser preservadas.

Normalmente, a tabela de preenchimento começa em 0 e tem tamanho de 6144 bytes. A tabela de cores normalmente começa em 8192 (&h2000) e também tem tamanho de 6144 bytes. Estes valores são obtidos pelo comando do basic BASE(12) e BASE (11), respectivamente. Como o comando bsave só define o endereço inicial e final, teremos que absorver o “lixo” entre 1800H e 1FFFH.

O outro tipo de imagem é uma imagem precedida de um programa em seu corpo. Este tipo de tela é carregada com a opção ,r. Com este tipo de tela, é possível criar vinhetas de apresentação de tela, como aparição da esquerda para a direita, aparição por blocos, etc. O modo em que se apresentam os dados podem ser diversos. Como há um programa antes gerenciando os dados, ele pode até utilizar uma tela compacta.

Se houver desejo de converter este tipo de imagem para o de *dump*, é relativamente simples. O usuário deverá carregar esta tela e salvar o conteúdo de memória. Um programa em basic para isto:

```
10 screen 2
20 bload "tela.tel",r
30 bsave "tela.scr", base(12),base(11)+6143,s
```

Screen 3

Característica	Descrição						
Dimensões	256 x 192.						
Cores	16.						
Endereço Inicial	0						
Header	FE	00	00	00	06	00	00
Tamanho	1536 (32*48)						
Pixel x Memória	1 byte						
Representação da Cor	Valor puro de 0 a 15, de acordo com a tabela de cores do MSX 1.						
Para salvar a tela	*10 screen 3* *20 rem código do desenho* ... *100 bsave "nome.src",0 ,1535 ,s*						
Para carregar a tela	*10 screen 3* *20 bload "nome.src",s*						

A Screen 3 é também mapeada por blocos. Segue o esquema de distribuição dos pontos pela memória:

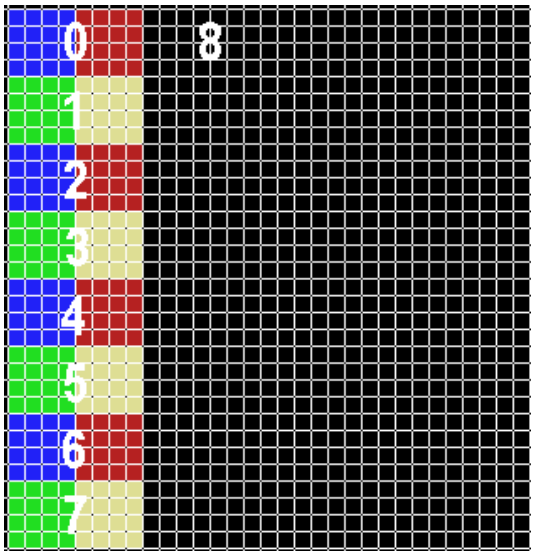


fig 4.

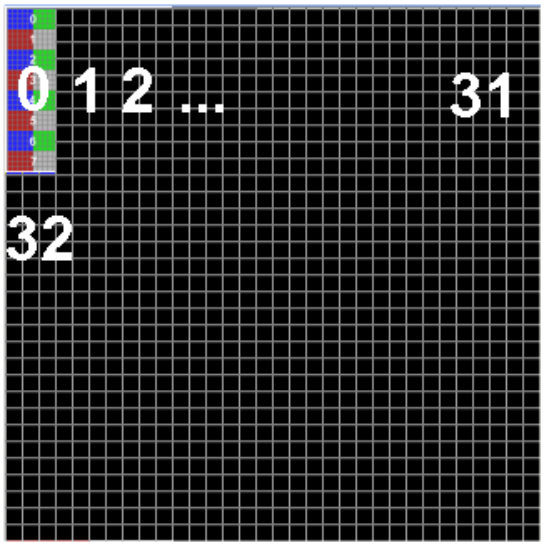


fig 5.

Cada byte representa um bloco na tela, de 8x4. A primeira parte do byte (bits mais significativos) contém a cor da esquerda do bloco, e o resto a segunda parte do bloco. Para o endereço 0, conforme a figura, teríamos o valor &B01000110 (46 Hexa). O 0100 representa a cor 4 (azul escuro) e o 0110 representa o 6 (vermelho escuro).

No eixo y, usamos 32 linhas. Para passar para as próximas 32 linhas, x deve atingir 256.

Screen 5

Característica	Descrição
Dimensões	256 x 212.
Modo Entrelaçado	256 x 412. Utiliza 2 páginas de vídeo.
Cores	16.
Header	FE 00 00 00 6A 00 00
Endereço Inicial	0.
Endereço Final	27135 (&H69FF) => 256 x 212 / 2.
Pixel x Memória	1 byte = 2 pixels ou 1 pixel = 4 bits.
Representação da Cor	Valor puro de 0 a 15, de acordo com a tabela de cores do MSX 1. Ex: A cor cinza em binário é 1110.
Para salvar a tela	*10 screen 5* *20 rem código do desenho* ... *100 bsave "nome.sr5",0,27135,s*
Para carregar a tela	*10 screen 5* *20 bload "nome.sr5",s*

Neste modo de vídeo, cada byte da memória de vídeo controla 2 pixels consecutivos. A parte mais significativa do byte controla o pixel mais a esquerda, enquanto que a outra parte controla o pixel da esquerda.

Rastreamento da tela:

0	1	2	...	127
128	129	130	...	255
...	...	...	...	...
27008	27009	27010	...	27135

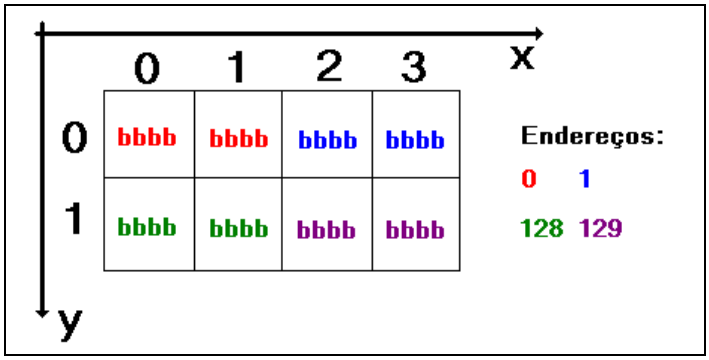


fig 6.

A partir de um pixel, podemos encontrar o endereço de memória correspondente:

*Endereço = int(x/2) + y*128*

Por controlar diretamente o pixel, este modo de tela não borra, como na Screen 2.
A partir desta tela, o arquivo de tela tem os seguintes bytes:
Header - FE 00 00 FF D3 00 00
Dados - Informação direta

Modo Entrelaçado:

Podemos enganar a vista humana, trocando rapidamente a exibição de 2 telas gráficas. Deste modo, uma imagem irá parecer ter a resolução de 256 x 424 pixels.

Para este fim, pega-se a imagem original, com reais 256 x 424, e salva-se em dois arquivos de tela. O primeiro arquivo terá as linhas pares da imagem, enquanto o segundo, as linhas ímpares.

Para exibir no modo entrelaçado:

```
10 screen 5,,,,,3
20 bload "telapar.sr5",s
30 set page 1,1
40 bload "telaimp.sr5",s
50 goto 50
```

A paleta de cores nada mais significa alterar o valor R, G, B das cores originais do MSX 1. Esta informação não é gravada nas telas convencionais, porém pode ser gravada adicionalmente. Cada informação de vermelho, verde e azul (Red, Green e Blue) varia de 0 a 7, ou seja, temos 8 variações para cada elemento, num total de 512 combinações.

Cor	R	G	B
0	0	0	0
1	0	0	0
2	1	6	1
3	3	7	3
4	1	1	7
5	2	3	7
6	5	1	1
7	2	6	7
8	7	1	1
9	7	3	3
10	6	6	1
11	6	6	4
12	1	4	1
13	6	2	5
14	5	5	5
15	7	7	7

Deste modo, é possível exibir uma imagem preta e branca com 16 níveis de cinza.

Screen 6

Característica	Descrição
Dimensões	512 x 212.
Modo Entrelaçado	512 x 412. Utiliza 4 páginas de vídeo.
Cores	4.
Header	FE 00 00 00 6A 00 00
Endereço Inicial	0.
Endereço Final	27135 (&H69FF) => 512 x 212 / 4.
Pixel x Memória	1 byte = 4 pixels ou 1 pixel = 2 bits.
Representação da Cor	Valor de 0 a 3. Pode-se mudar a palheta de cores.
Para salvar a tela	*10 screen 6* *20 rem código do desenho* ... *100 bsave "nome.sr6",0,27135,s*
Para carregar a tela	*10 screen 6* *20 bload "nome.sr6",s*

A Screen 6 tem a mesma resolução que a Screen 7, porém tem menos cores. Tem somente 4 cores. Com isto, ela pode usar 4 páginas de vídeo.

Cada byte representa 4 pixels da seguinte forma:

7	6	5	4	3	2	1	0
pixel		pixel + 1		pixel + 2		pixel +3	

Exemplo para um conjunto de pixels 4x1 para o byte (00011011) binário:

fig 7.

Screen 7

Característica	Descrição
Dimensões	512 x 212.
Modo Entrelaçado	512 x 412. Utiliza 2 páginas de vídeo.
Cores	16.
Header	FE 00 00 00 D4 00 00
Endereço Inicial	0.
Endereço Final	54271 (&HD3FF) => 512 x 212 / 2.
Pixel x Memória	1 byte = 2 pixels ou 1 pixel = 4 bits.
Representação da Cor	Valor puro de 0 a 15, de acordo com a tabela de cores do MSX 1. Ex: A cor cinza em binário é 1110.
Para salvar a tela	*10 screen 7* *20 rem código do desenho* ... *100 bsave "nome.sr7",0,54271,s*
Para carregar a tela	*10 screen 7* *20 bload "nome.sr7",s*

Esta tela é idêntica à Screen 5. As únicas diferenças são o número de páginas e o total de linhas no eixo x. A Screen 5 tem 4 páginas de vídeo, enquanto esta tem apenas 2.

A partir de um pixel, podemos encontrar o endereço de memória correspondente:

*Endereço = int(x/2) + y*256*

Para o modo entrelaçado, consulte a screen 5.

Screen 8

Característica	Descrição
Dimensões	256 x 212.
Modo Entrelaçado	256 x 412. Utiliza 2 páginas de vídeo.
Cores	256.
Header	FE 00 00 00 D4 00 00
Endereço Inicial	0.
Endereço Final	54271 (&HD3FF) => 256 x 212.
Pixel x Memória	1 byte = 1 pixel.
Representação da Cor	1 byte contém a informação dos elementos R, G e B de um pixel, neste formato: gggrrbb. R e G variam de 0 a 7 e B varia de 0 a 3.
Para salvar a tela	*10 screen 8* *20 rem código do desenho* ... *100 bsave "nome.pic",0,54271,s*
Para carregar a tela	*10 screen 8* *20 bload "nome.pic",s*

O rastreamento é feito de forma idêntica a das Screens 5 e 7, notando que agora 1 pixel equivale a 1 byte. A partir de um pixel, podemos encontrar o endereço de memória correspondente:

*Endereço = int(x/2) + y*256*

Exemplos de representação de cores para um pixel:

Exemplo 1: R=5(&b101), G=6(&b110), B=2(&b10)
110101**10** = 214
Exemplo 2: R=0(&b000), G=2(&b010), B=3(&b11)
000010**11** = 11

Este modo de tela é excelente para conversão de telas no formato 24 bits. Isto é feito simplesmente com uma regra de três:

$$\frac{\text{Maior cor do canal 24 bits}}{\text{Cor 24 bit}} \qquad \frac{\text{Maior cor do canal MSX}}{\text{Cor MSX}}$$

Por exemplo, para a cor 100 do canal vermelho (R):

255 --- 7
100 --- C

C = 3

Screen 10,11 e 12

Característica	Descrição
Dimensões	256 x 212.
Modo Entrelaçado	256 x 412. Utiliza 2 páginas de vídeo.
Cores	19268.
Header	FE 00 00 00 D4 00 00
Endereço Inicial	0.
Endereço Final	54271 (&HD3FF) => 256 x 212.
Pixel x Memória	4 bytes consecutivos = 4 pixels consecutivos.
Representação da Cor	Grupos de 4 bytes, onde: Byte 1: y1 y1 y1 y1 y1 kl kl kl Byte 2: y2 y2 y2 y2 y2 kh kh kh Byte 3: y3 y3 y3 y3 y3 jl jl jl Byte 4: y4 y4 y4 y4 y4 jh jh jh O elemento y é individual, k e j é comum a todos.
Para salvar a tela	*10 screen 12* *20 rem código do desenho* ... *100 bsave "nome.pic",0,54271,s*
Para carregar a tela	*10 screen 12* *20 bload "nome.pic",s*

O que quer dizer y, j e k?

Os componentes j e k estão diretamente ligados à palheta de cor do MSX 2, de 512 cores. Quando j e k são negativos, referem-se ao azul. Quando positivas, k refere-se à cor verde e j à cor vermelha. Ambos referem-se aos quatro pixels. Já o componente y refere-se à intensidade da cor em cada pixel.

A conversão do sistema YJK para o RGB é feito da seguinte forma:

$R = Y_n + J$
$G = Y_n + K$
$B = 5/4*Y_n - J/2 - K/4$

Obviamente, o cálculo do RGB é feito para cada um dos quatro pixels.
Devemos ter consciência que:

if J > 31 then J=J-64 : if K > 31 then K=K-64

if R < 0 then R = 0 else if R > 31 then R = 31
if G < 0 then G = 0 else if G > 31 then G = 31
if B < 0 then B = 0 else if B > 31 then B = 31

Já a conversão do sistema RGB para o sistema YJK:
- Calcula-se a média para cada elemento RGB dos 4 pontos.
- Faz-se:

Y = B/2 + R/4 + G/8
J = R - Y
K = G - Y

- Certifica-se que Y varia de 0 a 31, J e K variam de -32 a 31.

A conversão do sistema RGB para o YJK é feito da seguinte forma:

Para calcular j e k:
Y = B/2 + R/4 + G/8
J = R-Y
K = G-Y
Onde:
$R = (R_1 + R_2 + R_3 + R_4)/4$
$G = (G_1 + G_2 + G_3 + G_4)/4$
$B = (B_1 + B_2 + B_3 + B_4)/4$

Para calcular os y´s individuais:
Esquece-se Y
$Y_1 = B_1/2 + R_1/4 + G_1/8$
$Y_2 = B_2/2 + R_2/4 + G_2/8$
$Y_3 = B_3/2 + R_3/4 + G_3/8$
$Y_4 = B_4/2 + R_4/4 + G_4/8$

Para testar o sistema YJK, rode e altere os valores binários do programa abaixo:

```
10 SCREEN 12
20 FOR Y=0 TO 10
30 K=Y*255
40 FOR X=0 TO 255 STEP 4
50 VPOKE X+K+0,&B00111111
60 VPOKE X+K+1,&B00111111
70 VPOKE X+K+2,&B00111111
80 VPOKE X+K+3,&B00111111
90 NEXT X,Y
```

Altere somente os valores em verde e vermelho.

Conforme observado, a manipulação desta tela é razoavelmente complicada! É muito mais simples trabalharmos em um lugar onde possamos disfrutar de pelo menos 512 cores simultâneas na tela.

Diferenças entre Screens 10, 11 e 12

Para converter uma tela da Screen 12 para 10 ou 11, basta zerar o bit 3, ou seja aplicar o filtro AND &B11110111.

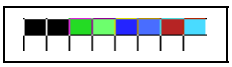
A Screen 11 possui 256 cores, como a Screen 8.

A Screen 10 possui 15 cores, como a Screen 5

O MSX Viewer 4 retira este bit para você. Basta salvar a tela como Screen 10/11.

Telas do COPY

Conforme explicado no manual do MSX Viewer 3, o comando copy do MSX 2 é capaz de salvar apenas trechos de tela. Por exemplo, da tela da Screen 5:



Esta imagem de 8 x 2 pixels pode ser salva em um arquivo e carregada em qualquer posição da tela, apontada pelas coordenadas (x,y). Esta coordenada carrega o restante da imagem para a direita e para baixo.

O header desta imagem NÃO TEM os bytes usuais, como FE, tamanho do arquivo, etc e é composto de 4 bytes: os dois primeiros são relativos à quantidade de pixels no eixo x e os dois restantes à quantidade de pixels no eixo y. A imagem acima tem 8 pixel no eixo x e 2 no eixo y.

Como de costume, o primeiro número é a parte menos significativa do número e o segundo, a mais significativa. Por exemplo, 67 4D é na verdade o número hexadecimal 4D67 ou CD 40 é o número 40CD. Basta inverter a ordem para se obter o valor correto.

O restante dos bytes é o conteúdo da tela, linha a linha.

A imagem acima ficaria então:

08 00 02 00 01 23 45 67 FF FF FF FF

Códigos ASCII

A tabela ASCII é composta de índices que referenciam alguma função, letra, número, expressão ou símbolo gráfico. Por exemplo, o código 65 (41H) é o índice para a posição da tabela gráfica respectiva à letra ‘a’ minúscula.

O formato padrão de ASCII varia de 0 a 127 (7 bits), ficando os valores restantes (128 a 255) a cargo do fabricante ou desenvolvedor do sistema operacional.

Portanto, os valores de 0 a 127 da tabela ASCII são PADRÕES para maioria dos sistemas. Por exemplo, 65 é ‘a’ tanto no PC como no MSX , tanto no Linux como no DOS, tanto no Master System como no Intellivision.

Estes primeiros 128 índices dizem respeito à funções como LF (Line Feed), CR (Carriage Return), BS (Back Space), etc, simbolos comuns (¡, @, #, %, $, “), letras maiúsculas, letras minúsculas e números.

As letras acentuadas requerem novos desenhos gráficos, diferentes dos usuais, como por exemplo o desenho de ‘a’ é diferente de ‘ã’. Portanto, o espaço adicional deverá ser utilizado.

De acordo com cada língua, cada espaço será indexado a uma letra presente ao alfabeto de interesse do desenvolvedor do sistema.

Os Experts e Hotbits se utilizam de acentuação da língua portuguesa, assim como símbolos relacionados ao povo brasiliero, como o ‘Cz’ (Cruzado).

Tabela ASCII padrão

ASCII	Hex	Símb.	ASCII	Hex	Símb.	ASCII	Hex	Símb.	ASCII	Hex	Símb.
0	0	NUL	16	10	DLE	32	20	SPC	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	“	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	‘	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	´	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Símb.	ASCII	Hex	Símb.	ASCII	Hex	Símb.	ASCII	Hex	Símb.
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	x
72	48	H	88	58	X	104	68	h	120	78	w
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	(
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	)
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	⌂

Além de termos coincidência em relação às letras, números e símbolos, temos coincidência também quanto às funções. Por exemplo, o comando print chr$(7) fará o computador soar um “beep” tanto no MSX como no PC.

Obs: o símbolo 5F (_) é o sublinhado.

A seguir, será mostrada uma tabela descrevendo as funções de 0 a 31.

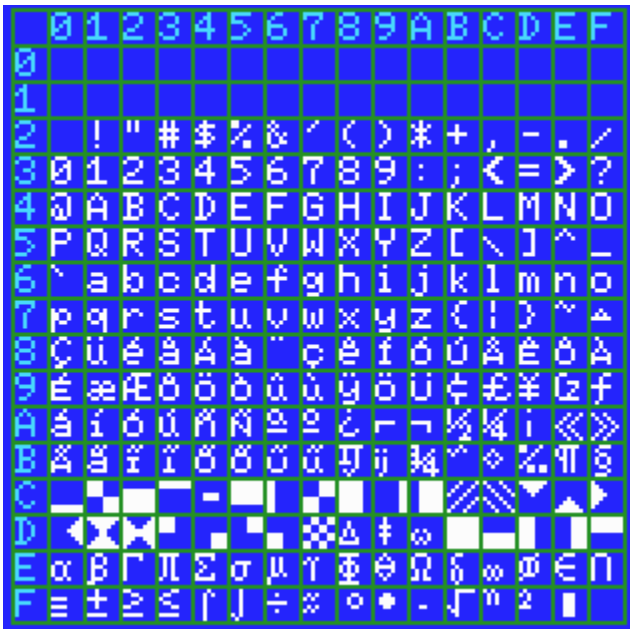
Descrição das funções ASCII

ASCII	Hex	Símbolo	Descrição
0	0	NUL	Null (nulo)
1	1	SOH	Start of Heading (inicio do cabeçário)
2	2	STX	Start of Text (começo do texto)
3	3	ETX	End of Text (fim do texto)
4	4	EOT	End of Tansmission (fim da transmissão)
5	5	ENQ	Enquiry ()
6	6	ACK	Acknowledge (reconhecimento)
7	7	BEL	Bell (campainha)
8	8	BS	Backspace (espaço para trás)
9	9	TAB	Horizontal Tabulation (tabulação horizontal)
10	A	LF	Line Feed (próxima linha)
11	B	VT	Vertical Tabulation (tabulação vertical)
12	C	FF	Form Feed (alimentação do formulário)
13	D	CR	Carriage Return (inicio da linha)
14	E	SO	Shift Out (shift solto)
15	F	SI	Shift In (shift apertado)
16	10	DLE	Data Link Escape (interrupção de comunicação)
17	11	DC1	Device Control 1 (dispositivo de controle 1)
18	12	DC2	Device Control 2 (dispositivo de controle 2)
19	13	DC3	Device Control 3 (dispositivo de controle 3)
20	14	DC4	Device Control 4 (dispositivo de controle 4)
21	15	NAK	Negative Acknowledge (não reconhecido)
22	16	SYN	Synchronous Idle (idle síncrono)
23	17	ETB	End of Transmission Block (fim de transmissão de bloco)
24	18	CAN	Cancel (cancelamento)
25	19	EM	End of Medium (fim do meio)
26	1A	SUB	Substitute (substituto)
27	1B	ESC	Escape (escape)
28	1C	FS	File Separator (separador de arquivo)
29	1D	GS	Group Separator (separador de grupo)
30	1E	RS	Record Separator (separador de registro)
31	1F	US	Unit Separator (separador de unidade)

O que difere o MSX do PC está de 128 a 255. Existe um problema maior ainda: os distintos sistemas operacionais do PC também possuem esta parte da tabela diferente. Todo cuidado é pouco.

Serão mostradas a seguir, as tabelas do MSX, Windows, DOS e Unix.

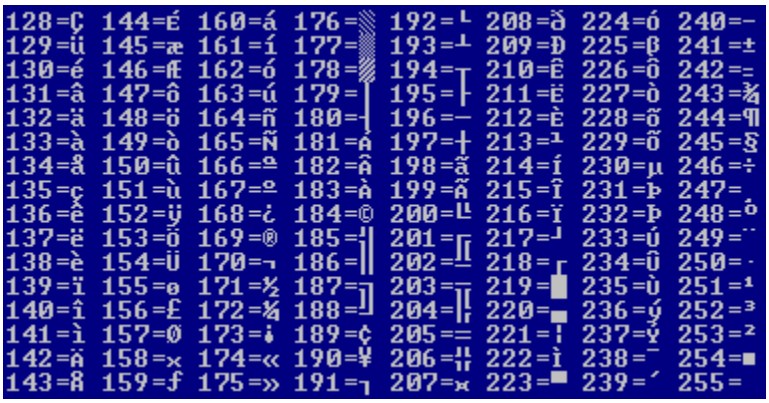
MSX Brasileiro (Hotbit)



Windows (Língua portuguesa, Brasil)

121=y	136=ˆ	151=–	166=¦	181=µ	196=Ä	211=Ó	226=â	241=ñ
122=z	137=‰	152=˜	167=§	182=¶	197=Å	212=Ô	227=ã	242=ò
123={	138=Š	153=™	168=¨	183=·	198=Æ	213=Õ	228=ä	243=ó
124=|	139=‹	154=š	169=©	184=¸	199=Ç	214=Ö	229=å	244=ô
125=}	140=Œ	155=›	170=ª	185=¹	200=È	215=×	230=æ	245=õ
126=~	141=▯	156=œ	171=«	186=º	201=É	216=Ø	231=ç	246=ö
127=▯	142=Ž	157=▯	172=¬	187=»	202=Ê	217=Ù	232=è	247=÷
128=€	143=▯	158=ž	173=-	188=¼	203=Ë	218=Ú	233=é	248=ø
129=▯	144=▯	159=Ÿ	174=®	189=½	204=Ì	219=Û	234=ê	249=ù
130=‚	145=‘	160=	175=¯	190=¾	205=Í	220=Ü	235=ë	250=ú
131=ƒ	146=’	161=¡	176=°	191=¿	206=Î	221=Ý	236=ì	251=û
132=„	147=“	162=¢	177=±	192=À	207=Ï	222=Þ	237=í	252=ü
133=…	148=”	163=£	178=²	193=Á	208=Ð	223=ß	238=î	253=ý
134=†	149=•	164=¤	179=³	194=Â	209=Ñ	224=à	239=ï	254=þ
135=‡	150=–	165=¥	180=´	195=Ã	210=Ò	225=á	240=ð	255=ÿ

DOS



UNIX

O Linux tem a mesma tabela de caracteres do Windows.