

100 DICAS PARA MSX



N Editora
Aleph

**TÉCNICAS E
TRUQUES DE
PROGRAMAÇÃO**

RENATO DA SILVA OLIVEIRA

100 DICAS

PARA

MSX

TÉCNICAS E TRUQUES DE PROGRAMAÇÃO

AUTORES:

Henrique de Figueredo Luz
Luis Tarcísio de Carvalho Jr.
Milton Maldonado Jr. (The Pilot)
Pierluigi Piazzi
Renato da Silva Oliveira
Rubens Pereira Silva Jr.

**3ª EDIÇÃO
1989**



© 1988 - EDITORA ALEPH

EXPEDIENTE:

Coordenação Editorial: PIERLUIGI PIAZZI
Coordenação Didática: BETTY FROMER PIAZZI
Produção Editorial: ROSA KOGAN FROMER
Editoração: RENATO DA SILVA OLIVEIRA
Ilustrações: DURVALY ODILON NICOLETTI



ALEPH Publicações e
Assessoria Pedagógica Ltda
R. Dr. Luiz Migliano 1110 cj.301
05711 São Paulo SP
Caixa Postal 20707 CEP 01498
Tel: (011) 843-3202

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Oliveira, Renato da Silva, 1960-
051c 100 dicas para MSX : técnicas e truques de programação / Renato da Silva Oliveira. -- 1. ed. -- São Paulo : Aleph, 1988.
(Coleção MSX)

1. MSX (Computadores) - Programação I. Título.
II. Série.

88-0118

CDD-001.642

Índices para catálogo sistemático:

1. MSX : Computadores : Programação : Processamento de dados 001.642

CEM DICAS PARA O MSX

SUMARIO

APRESENTAÇÃO	6
1 - DICAS SOBRE O TECLADO	7
2 - DICAS PARA USAR O VIDEO	21
3 - DICAS SONORAS	88
4 - DICAS PARA CASSETE	105
5 - DICAS PARA IMPRESSORA	117
6 - DICAS PARA O DRIVE	134
7 - DICAS DE PROCESSAMENTO	153
8 - NOTAS SOBRE A BIBLIOGRAFIA RECOMENDADA ..	191

APRESENTAÇÃO

A grande maioria dos usuários de microcomputadores MSX nunca teve outro micro antes. Apesar de terem entrado no mundo da computação pela porta da frente, é provável que esses usuários tenham alguma dificuldade inicial para programar suas máquinas. Poucos deles têm, de imediato, uma visão muito clara dos poderosos recursos de que dispõem.

Este livro contém mais de cem dicas de programação já prontas para serem usadas. Elas permitem um aproveitamento muito maior dos recursos dos MSX mesmo pelos usuários iniciantes e, certamente, acrescentam conhecimentos valiosos aos programadores mais experientes.

Apesar de serem geralmente independentes umas das outras, as dicas estão agrupadas em 7 capítulos, de acordo com a função a que elas se destinam.

O leitor poderá usá-las em outros programas para otimizá-los ou obter resultados específicos.

Cada dica está apresentada de forma bem prática e resumida de modo a tornar seu uso imediato muito fácil. As explicações nem sempre são detalhadas mas, sempre que possível, faz-se referência a textos de outros livros onde o assunto é comentado mais extensamente.

Apesar de muitas dicas serem programas em Linguagem de Máquina, optamos pelo uso exclusivo da linguagem BASIC para gerá-las. Isso torna a digitação mais fácil e o uso mais imediato. Todos os programas estão em BASIC, mas mesmo assim, para diminuir a ocorrência de erros de digitação, junto a cada listagem pode-se encontrar em "vídeo inverso" sua SOMA SINTÁTICA. As explicações sobre a produção e uso dessa "soma" são dadas na dica 7.1 (página 178) e ela é, portanto, a primeira dica que deve ser lida.

Lembre-se que a SOMA TOTAL de um programa, mesmo coincidindo com a apresentada no livro, não elimina totalmente a ocorrência de erros de digitação, mas apenas a reduz. Deve-se também considerar que a soma que apresentamos foi obtida com um EXPERT 1.1 e há casos em que outros tipos de MSX produzem somas diferentes.

Esperamos que este livro possa abrir novos horizontes aos programadores MSX; tanto aos iniciantes quanto aos que não têm tempo suficiente para descobrir sozinhos os incontáveis "macêtes" dessas máquinas maravilhosas.



DICAS SOBRE O TECLADO

As dicas deste capítulo abordam predominantemente assuntos relativos a manipulação do teclado nos micros MSX.

Uma vez que o principal meio de entrada de dados para a UCP do micro é o teclado, a utilidade destas dicas é evidente.

1.1 - Carregando o Buffer do Teclado	8
1.2 - Limpando o Buffer do Teclado	9
1.3 - Programando as teclas de funções	10
1.4 - Restabelecendo as teclas de funções ..	11
1.5 - Checando as teclas especiais	12
1.6 - Travando a tecla CAPS LOCK	13
1.7 - Usando o click do teclado	14
1.8 - Reprogramando todo o teclado	15
1.9 - Usando a barra de espaços	18
1.A - Usando as teclas de setas	19
1.B - Uso da instrução ON KEY GOSUB	20

1.1 - CARREGANDO O BUFFER DO TECLADO

O MSX reserva uma área (buffer) de 40 bytes na memória RAM para armazenar temporariamente os dados digitados através do teclado. Se esses dados forem comandos, serão executados na sequência em que estiverem no buffer. Pode-se carregar comandos no buffer do teclado através de programas em BASIC ou em ASSEMBLY a fim de gerar efeitos especiais. Por exemplo, para fazer com que um programa em BASIC seja carregado de fita cassete e seja automaticamente executado, basta usar o programa apresentado a seguir.

```
100 SCREEN 1: X$=CHR$(34): Y$=CHR$(13)      800
110 A$="CLOAD"+X$+"CAS:"+X$+"=RUN"+Y$     159E
120 EN=&HFBF0                               1997
130 FOR F=1 TO LEN(A$)                     203E
140 CH=ASC(MID$(A$,F,1))                   250F
150 POKE EN,CH                             2008
160 EN=EN+1                                3017
170 NEXT F                                  3407
180 X=65536!+&HFBF0+LEN(A$)               4088
190 Y=X-256*INT(X/256)                     4A0F
200 POKE &HF3F8,Y                          4F80
210 POKE &HF3F9,INT(X/256)                 5890
220 POKE &HF3FA,&HF0                        5E73
230 POKE &HF3FB,&HF0                        6407
```

TOTAL = 6407

Os dados a serem inseridos no buffer estão na variável A\$. A variável EN armazena o endereço do buffer a ser preenchido. A variável CH armazena o caractere de A\$ a ser inserido no endereço EN do buffer. O endereço &HFBF0 é o início do buffer do teclado (KEYBUFF).

Os endereços &HF3F8 e &HF3F9 armazenam o próximo endereço a ser preenchido no buffer do teclado (PUTPNT) e são preenchidos de modo a apontarem para o endereço subsequente ao do fim da mensagem inserida.

Os endereços &HF3FA e &HF3FB armazenam o último endereço do buffer lido pelo micro (GETPNT) e são posicionados de modo a apontarem para o endereço do primeiro caractere inserido (&HFBF0).

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - páginas 40, 41, 42 e 143.

1.2 - LIMPANDO O BUFFER DO TECLADO

Observe o programa a seguir:

```
10 SCREEN 0,,0:LOCATE,,0:KEY OFF          502
20 PRINT " Entre uma escolha:":PRINT      083
30 PRINT " [ 1 ] Opção 1 ...."          141F
40 PRINT " [ 2 ] Opção 2 ...."          1EAA
50 PRINT " [ 3 ] Opção 3 ...."          2778
60 PRINT:PRINT:PRINT                     2884
70 B$=INKEY$:IF LEN(B$)<1 THEN 70        3802
80 FOR F=1 TO 500 : NEXT F : BEEP        4407
90 PRINT "OPÇÃO ";B$;" ESCOLHIDA !"      5800
100 A$=INPUT$(1)                          5F13
110 PRINT A$                               6378
```

TOTAL = 6378

Note que ao fazer a primeira opção, se a tecla da escolha for pressionada por muito tempo ou se mais de uma tecla for pressionada, a segunda opção também será feita. Isso ocorre porque o buffer do teclado fica carregado com os caracteres digitados até que eles sejam usados.

No BIOS do MSX existe uma rotina que pode ser útil nessas situações: a KILLBUFF. Sempre que é executada, ela limpa o buffer do teclado. Para chamá-la, basta usar as instruções:

```
DEFUSR0 = &H0156 : POKE 0,USR0(0)
```

Experimente inserir a linha a seguir no programa anterior e depois execute-o novamente.

```
95 DEFUSR0 = &H0156 : POKE 0,USR0(0)
```

Você notará que a segunda opção não mais será atrapalhada por digitações acidentais.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX vol. 2 - página 110.
Aprofundando-se no MSX - página 159.

1.3 - PROGRAMANDO AS TECLAS DE FUNÇÕES

As teclas de funções podem ser facilmente redefinidas para atender às necessidades específicas de um programa.

A forma mais direta de reprogramá-las é usando a instrução KEY. Entretanto pode-se também redefinir as teclas de funções diretamente na memória RAM.

Experimente executar o seguinte programa e depois pressione a tecla F1.

```
10 FOR F=&HF87F TO &HF87F+38  
20 POKE F,ASC("A")  
30 NEXT F
```

715
B3D
CAB

Com isso a tecla F1 passou a ter uma sequência de 39 caracteres "A". Lembre-se que com a instrução KEY podemos inserir no máximo 15 caracteres em cada tecla de função. Com este recurso, podemos atribuir a uma única tecla até 39 caracteres, entretanto deve-se tomar alguns cuidados, pois o conteúdo das demais teclas podem ser alterados. Experimente digitar a tecla F2 após ter executado o programa acima. Você verá que seu conteúdo foi alterado pelo programa.

O que acontece é que existem 160 bytes da RAM, reservados a partir do endereço &HF87F (FNKSTR), para armazenar os textos das teclas de funções. Cada tecla tem seu texto começando sempre num mesmo endereço e o número máximo de caracteres atribuíveis a uma única tecla de função é 39. O 40º caractere da área de texto de uma tecla é sempre um 0.

Você pode também atribuir às teclas de funções sequências de caracteres de controle. Por exemplo, digite a instrução a seguir e depois pressione a tecla F1.

```
KEY 1,CHR$(7)+CHR$(28)+CHR$(8)+CHR$(7)+C  
HR$(127)+CHR$(9)+CHR$(11)
```

BIBLIOGRAFIA RECOMENDADA:

- Linguagem Basic MSX - página 81.
- Curso de Basic v.1 - páginas 21 e 22.
- Aprofundando-se no MSX - página 50.
- Programação Avançada em MSX - página 145

1.4 - RESTABELECENDO AS TECLAS DE FUNÇÕES

As teclas de funções podem ser facilmente programadas. Entretanto, após usar um programa que as redefine, pode ser necessário reinicializá-las com as funções originais. Para isso pode-se simplesmente programar tecla por tecla novamente ou chamar uma rotina do BIOS (INITFNK em &H003E) que se encarrega de fazer isso automaticamente. Para executar essa rotina do BIOS basta digitar a seguinte instrução:

```
DEFUSR0 = &H3E : POKE 0,USR0(0)
```

Observe o programa exemplo a seguir. Ele redefine as teclas de funções e logo a seguir restabelece seus conteúdos originais.

```
10 SCREEN 0 : KEY ON
20 FOR F=1 TO 10
30 KEY F,"NOVA !"
40 NEXT F
50 SCREEN 0
60 PRINT,," TECLAS REDEFINIDAS !"
70 PRINT,," PRESSIONE RETURN !"
80 A$=INPUT$(1)
90 DEFUSR0 = &H3E : X = USR0(0)
100 SCREEN 0
110 PRINT,," TECLAS RESTABELECIDAS !"

```

0000
0002
0007
00FE
0055
1000
100F
1000
20F0
20FF
4046

```
TOTAL = 4646
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 156.

1.5 - CHECANDO AS TECLAS ESPECIAIS

Nos micros MSX existem várias maneiras de verificarmos se alguma tecla especial está ou não pressionada. Uma delas é consultando as variáveis do sistema OLDKEY (&HFBDA) ou NEWKEY (&HFBE5). O programa listado a seguir ilustra esse procedimento, checando especificamente se a tecla LGRA (ou GRAPH) está pressionada.

Ao ser executado o programa lê o conteúdo da região da memória RAM que vai do endereço &HFBDA até o endereço &HFBEF, mostrando na tela seus conteúdos em binário. Experimente pressionar algumas teclas (menos ^STOP!). Você verá que a cada tecla pressionada corresponde um bit de algum dos bytes dessa região. Para checar qualquer tecla, portanto, basta verificar se o bit está ou não em "0". Experimente pressionar a tecla LGRA (ou GRAPH). A configuração que corresponde à ela é o valor 251 (ou 11111011, em binário) no byte &HFBE0. Observe como a linha 250 faz o teste para ver se ela está ou não pressionada e tente alterar o programa para que ele teste se a tecla RGRA (ou CODE) está pressionada.

```
100 REM F3
110 REM LE OLDKEY E NEWKEY 73E
120 REM 84F
130 SCREEN 0 423
140 LOCATE 0,5,0 082
150 FOR F=&HFBDA TO &HFBE4 148B
160 PRINT "&H";HEX$(F);"=">"; 1001
170 PRINT RIGHT$("00000000"+BIN$(PEEK(2031
F)),8);
180 PRINT " &H";HEX$(F+11);"=">"; 3037
190 PRINT RIGHT$("00000000"+BIN$(PEEK(4230
F+11)),8)
200 NEXT F 5100
210 PRINT:PRINT:PRINT 537E
220 REM 55E7
230 REM TESTA A TECLA LGRA 6360
240 REM 6436
250 IF PEEK(&HFBE0)=(PEEK(&HFBE0)AND251) 8037
THEN PRINT "LGRA PRESSIONADA"ELSEPRINT"
"
260 GOTO 140 8236
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 83 a 86.

1.6 - TRAVANDO A TECLA CAPS LOCK

Existe uma variável do sistema (CAPST, em &HFCAB) que indica o estado da tecla CAPS LOCK. Se CAPS LOCK está ativa, essa variável contém 255 ou algum outro valor maior que zero, e se inativa, contém 0. Cada vez que a tecla CAPS LOCK é pressionada os bits da variável CAPST são complementados de modo que seu valor é sempre 0 ou 255.

Para simular o pressionamento de CAPS LOCK basta "pokear" 255 ou 0 em CAPST.

Para travar a CAPS LOCK, deixando-a ativa, basta pokear qualquer valor maior que 0 e menor que 255 em CAPST, pois assim, mesmo que CAPS LOCK seja pressionada, a complementação dos bits de CAPST produzirá um valor também maior que 0 e menor que 255.

Experimente digitar e usar o programa a seguir. Após executá-lo, digite algumas letras usando as teclas CAPS LOCK e SHIFT. Você notará que elas ficaram praticamente inoperantes para as letras.

```
10 SCREEN 0 : WIDTH 38
20 POKE &HFCAB,1
30 PRINT,"      DIGITE ALGUMAS LETRAS"
40 PRINT,"      (com ou sem SHIFT):"
50 PRINT : PRINT
60 A$=INPUT$(1)
70 PRINT A$;
80 GOTO 60
```

257
589
194
1544
1937
1042
2023
2313

TOTAL = 2313

Para fazer com que a tecla CAPS LOCK volte a funcionar normalmente, comande:

```
POKE &HFCAB,0
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 50.
Programação Avançada em MSX - página 146.
Coleção de Programas para MSX vol 2 - páginas 51 e 55.

1.7 - USANDO O CLICK DO TECLADO

O click do teclado pode ser usado para muitas e diferentes aplicações, deixando livre para outros usos o PSG.

Para gerar o click do teclado por software, é necessário acessar o hardware da máquina com comandos OUT. Veja o programa a seguir.

10 SCREEN 1	10E
20 KEY OFF	2F5
30 COLOR 1,4	451
40 LOCATE 32*RND(1),24	941
50 PRINT "*"	A44
60 OUT &HAA,&HFF	156
70 OUT &HAA,&H7F	1091
80 GOTO 40	1363

TOTAL = 1363

Cada vez que as linhas 60 e 70 são executadas um "click" é gerado. Você pode usar isto para sonorizar seus programas mesmo sem usar o PSG.

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX vol 2 - página 15.
Aprofundando-se no MSX - página 87.

1.8 - REPROGRAMANDO TODO O TECLADO

Agora seu MSX também pode ter um "teclado inteligente" como nos velhos tempos do Apple e do Sinclair. O programa a seguir instala em seu teclado as principais palavras reservadas do Basic sem perder sua função original.

A redefinição atinge as letras maiúsculas e minúsculas, totalizando 52 palavras reservadas. No que é possível, as teclas seguem o padrão Sinclair (A=NEW, K=LIST, etc). Palavras do Sinclair que não existem no MSX (PAUSE, SCROLL, etc) são trocadas por outras do último.

Para usar o programa, digite-o e comande RUN. Após a execução deverão aparecer os códigos das teclas e as palavras correspondentes. Experimente então teclar SELECT e em seguida a tecla A. Você deverá obter "NEW". Tecle agora SELECT+"a" para obter "FRE(". Tente então outras combinações. O que acontece quando a tecla após o SELECT não é uma letra?

```
1000 REM -----
1010 REM  TOKEN V. 1.0 BY THE PILOT
1020 REM          JANEIRO DE 1988
1030 REM -----
1040 REM
1050 DATA 21,9B,D1,CD,CE,D1,21,16
1060 DATA D0,22,A5,FD,3E,C3,32,A4
1070 DATA FD,AF,32,D7,D1,C9,4F,3A
1080 DATA D7,D1,A7,79,20,05,FE,18
1090 DATA 28,49,C9,FE,41,38,3D,FE
1100 DATA 5B,30,08,D6,40,47,21,72
1110 DATA D0,18,0E,FE,61,38,2D,FE
1120 DATA 7B,30,29,D6,60,47,21,08
1130 DATA D1,7E,A7,23,20,FB,10,F9
1140 DATA 3E,C9,32,A4,FD,7E,A7,28
1150 DATA 06,CD,A2,00,23,18,F6,3E
1160 DATA C3,32,A4,FD,AF,32,D7,D1
1170 DATA C1,C3,DA,08,4F,AF,32,D7
1180 DATA D1,79,C9,3E,FF,32,D7,D1
1190 DATA AF,C9,00,4E,45,57,00,42
1200 DATA 45,45,50,00,43,4F,4E,54
1210 DATA 00,44,49,4D,20,00,52,45
1220 DATA 4D,20,00,46,4F,52,20,00
1230 DATA 47,4F,54,4F,20,00,47,4F
1240 DATA 53,55,42,20,00,49,4E,50
1250 DATA 55,54,20,00,4C,4F,41,44
1260 DATA 20,00,4C,49,53,54,20,00
```

```
64E
FRE
1509
2050
222E
30E9
4304
5522
6B02
8056
88E0
8E10
9602
9EEE
A74F
B9F9
C882
DEFC
E30F
F96
94E
C60
17FE
2173
3003
4146
5216
```

```

1270 DATA 4C,4C,49,53,54,20,00,4D
1280 DATA 4F,54,4F,52,20,00,4E,45
1290 DATA 58,54,20,00,50,4F,4B,45
1300 DATA 20,00,50,52,49,4E,54,00
1310 DATA 50,53,45,54,20,28,00,52
1320 DATA 55,4E,00,53,41,56,45,20
1330 DATA 00,54,52,4F,4E,00,49,46
1340 DATA 20,00,43,4C,53,00,50,52
1350 DATA 45,53,45,54,20,28,00,43
1360 DATA 4C,45,41,52,00,52,45,54
1370 DATA 55,52,4E,00,45,4E,44,00
1380 DATA 00,46,52,45,28,00,49,4E
1390 DATA 4B,45,59,24,00,44,53,4B
1400 DATA 46,28,00,41,54,4E,28,00
1410 DATA 54,41,4E,28,00,53,47,4E
1420 DATA 28,00,41,42,53,28,00,53
1430 DATA 51,52,28,00,41,53,43,28
1440 DATA 00,56,41,4C,28,00,4C,45
1450 DATA 4E,28,00,55,53,52,00,33
1460 DATA 2E,31,34,31,35,39,32,37
1470 DATA 21,00,4E,4F,54,00,50,45
1480 DATA 45,4B,28,00,54,41,42,28
1490 DATA 00,53,49,4E,28,00,49,4E
1500 DATA 54,28,00,53,54,52,49,4E
1510 DATA 47,24,28,00,52,4E,44,28
1520 DATA 00,43,48,52,24,28,00,56
1530 DATA 41,52,50,54,52,28,00,43
1540 DATA 4F,53,28,00,45,58,50,28
1550 DATA 00,53,54,52,24,28,00,4C
1560 DATA 4E,28,00,0C,50,72,6F,67
1570 DATA 72,61,6D,61,20,65,73,63
1580 DATA 72,69,74,6F,20,70,6F,72
1590 DATA 3A,0D,0A,54,48,45,20,50
1600 DATA 49,4C,4F,54,20,65,6D,20
1610 DATA 4A,61,6E,65,69,72,6F,2F
1620 DATA 31,39,38,38,2E,00,7E,A7
1630 DATA C8,CD,A2,00,23,18,F7,00
1640 DATA FIM
1650 CLS:PRINT "CARREGANDO TOKEN"
1660 FOR I=&HD000 TO &HD1D8:READ A$:POKE
I,VAL("&H"+A$):NEXT I
1670 DEFUSR=&HD000:A=USR(0):PRINT:PRINT
1680 FOR I=65 TO 90:PRINT "<SELECT>+"CHR
$(I);" = "";CHR$(24);CHR$(I):FOR T=0 TO 1
00:NEXT T:NEXT I
1690 FOR I=97 TO 122:PRINT "<SELECT>+"CH
R$(I);" = "";CHR$(24);CHR$(I):FOR T=0 TO
100:NEXT T:NEXT I
1700 END

```

É recomendável salvar o código binário para facilitar o uso do programa. Para isto digite:

```
BSAVE "TOKEN.BIN",&HD000,&HD1DB
```

ou

```
BSAVE "CAS:TOKEN",&HD000,&HD1DB
```

Para executar o programa, use o comando:

```
BLOAD "TOKEN.BIN",R
```

ou

```
BLOAD "CAS:",R
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 66 a 71 e capítulo 3.
Programação Avançado em MSX - capítulo 1 e apêndice 3.

1.9 - USANDO A BARRA DE ESPAÇOS

A barra de espaços no MSX pode ser usada de muitas e diferentes maneiras. No programa abaixo apresentamos um exemplo em que a barra é usada para controlar o fluxo do processamento usando a função STRIG.

```
10 SCREEN 0 : PRINT : PRINT SPC(6); 487
20 PRINT "DIGITE A BARRA DE ESPAÇO !" 555
30 BEEP 1688
40 IF NOT STRIG(0) THEN 30 1914
50 PLAY "ABCDEFGF" 1914
```

TOTAL = 1914

Uma outra forma de se controlar o fluxo do processamento é através das interrupções. Essa maneira é ilustrada pelo programa a seguir.

```
10 SCREEN 0 : PRINT : PRINT SPC(6); 487
20 PRINT "DIGITE A BARRA DE ESPAÇO !" 555
30 STRIG(0) ON : ON STRIG GOSUB 60 1688
40 PLAY"V15ABCDEFGF07L32" 1914
50 GOTO 50 1914
60 PLAY"C#" : RETURN 2318
```

TOTAL = 2318

Observe que os dois programas apresentados são fundamentalmente diferentes. O primeiro apenas interrompe o processamento normal do programa até que a barra de espaços seja pressionada, enquanto o segundo desvia o processamento para uma sub-rotina, esteja ele em que linha estiver, sempre que a barra de espaços for pressionada.

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 112, 159, 160, 178 e 179.
Coleção de Programas para MSX v.2 - páginas 75 a 82.

1.A - USANDO AS TECLAS DE SETAS

As teclas de setas podem ser testadas através de uma função do BASIC MSX. O programa apresentado a seguir ilustra um uso típico dessa função, associada ao controle de um ponto plotante na SCREEN 2.

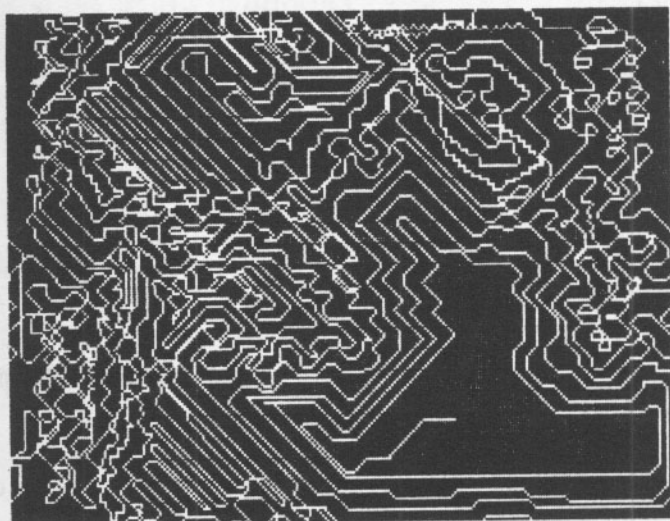
Note que usamos "parênteses lógicos" para tornar a tela ilimitada. Se o ponto tentar sair da tela por qualquer um de seus 4 lados, automaticamente será transportado para a primeira posição do lado oposto.

```
100 SCREEN 2,,0 : DEFINT A-Z
110 X=128 : Y=96
120 PSET (X,Y),15
130 A=STICK(0) : IF A=0 THEN 130
140 Y=Y+(A<3 OR A=8)-(A>3 AND A<7)
150 X=X-(A>1 AND A<5)+(A>5)
160 X = -255*(X=-1) - X*(X<>-1)
170 X = -X*(X<>256)
180 Y = -191*(Y=-1) - Y*(Y<>-1)
190 Y = -Y*(Y<>192)
200 GOTO 120
```

4BC
99C
C74
14F6
2178
2ADA
3A22
430B
5103
5984
5E2E

TOTAL = 5E2E

Para entender o programa, lembre-se que quando o conteúdo dentro dos parênteses for verdadeiro, ele pode ser substituído pelo valor numérico "-1", e quando for falso, pelo valor "0".



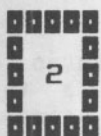
1.B - USO DA INSTRUÇÃO ON KEY GOSUB

As teclas de funções podem ser usadas para chamar sub-rotinas automaticamente durante as interrupções. Observe o programa a seguir. Digite-o e rode-o.

10 SCREEN 0,,0 : PLAY "L64"	3E1
20 FOR F=1 TO 10 : KEY(F) ON : NEXT F	3E7
30 ON KEY GOSUB 100,200,300,400,500,600,700,800,900,1000	40E6
40 GOTO 40	41E8
100 PRINT " F1 PRESSIONADA !!! "	27AC
110 PLAY "C"	282E
120 RETURN	2089
200 PRINT " F2 PRESSIONADA !!! "	3EE0
210 PLAY "C#"	4378
220 RETURN	4488
300 PRINT " F3 PRESSIONADA !!! "	5875
310 PLAY "D"	5D9E
320 RETURN	5F5F
400 PRINT " F4 PRESSIONADA !!! "	72FE
410 PLAY "D#"	7891
420 RETURN	7A54
500 PRINT " F5 PRESSIONADA !!! "	839A
510 PLAY "E"	867D
520 RETURN	874D
600 PRINT " F6 PRESSIONADA !!! "	8E5F
610 PLAY "F"	9101
620 RETURN	9238
700 PRINT " F7 PRESSIONADA !!! "	9A84
710 PLAY "F#"	9D7E
720 RETURN	9F86
800 PRINT " F8 PRESSIONADA !!! "	ABE9
810 PLAY "G"	A020
820 RETURN	AE72
900 PRINT " F9 PRESSIONADA !!! "	0088
910 PLAY "G#"	04E8
920 RETURN	05F7
1000 PRINT " F10 PRESSIONADA !!! "	0798
1010 PLAY "A"	0C10
1020 RETURN	0060

TOTAL = 0060

Observe que, independentemente da linha do programa que estiver sendo executada, sempre que alguma tecla de função for pressionada uma sub-rotina será chamada.



DICAS PARA USAR O VÍDEO

Este capítulo aborda os recursos do vídeo do MSX. Existe um circuito dedicado ao controle do vídeo (VDP) que tem à sua disposição 16 Kbytes de memória RAM (VRAM) para armazenar os dados da tela. O controle do vídeo pode ser feito através do BASIC, com os vários comandos dedicados a isso, ou diretamente em Linguagem de Máquina.

2.1 - Cor de frente igual a cor de fundo	22
2.2 - Pseudo-borda na SCREEN 0	23
2.3 - Textos na SCREEN 2	24
2.4 - Caracteres menores que 32	26
2.5 - Caracteres de controle do vídeo	27
2.6 - Usando o STEP em comandos gráficos	29
2.7 - Posicionando o cursor na SCREEN 2	31
2.8 - Movimentos na tela	32
2.9 - SCREEN 4	35
2.A - Setores com o CIRCLE	39
2.B - Redefinindo caracteres	40
2.C - SCROLL UP para SCREEN 0	52
2.D - SCROLL DOWN para SCREEN 0	53
2.E - SCROLL LEFT para SCREEN 0	54
2.F - SCROLL RIGHT para SCREEN 0	55
2.G - SCROLL UP para SCREEN 1	56
2.H - SCROLL DOWN para SCREEN 1	57
2.I - SCROLL LEFT para SCREEN 1	58
2.J - SCROLL RIGHT para SCREEN 1	59
2.K - Centralizando caracteres	60
2.L - Animação com SPRITES	61
2.M - Letras ampliadas	63
2.N - Entendendo o DRAW	66
2.O - "WARP 8" na SCREEN 2	67
2.P - "SPRITEANDO" a tabela de caracteres ...	68
2.Q - Arlequim bêbado	69
2.R - Usando 40 ou 64 colunas na SCREEN 2 ...	71
2.S - Carimbador de SPRITES 8x8 na SCREEN 2 .	73
2.T - Carimbador de SPRITES 16x16 na SCREEN 2	75
2.U - Armazenando telas na RAM	76
2.V - Usando a VRAM para dados	77
2.W - Pesquisador de desenhos	79
2.X - Imagens instantâneas	83
2.Y - Impressão em tamanho duplo na SCREEN 2	84

2.1 - COR DE FRENTE IGUAL A COR DE FUNDO

Muitos programas ao terminarem a execução ou serem interrompidos por CONTROL + STOP deixam a tela com a cor de frente igual a cor de fundo e a primeira impressão que se tem é que o micro quebrou. Digite e rode o programinha listado a seguir:

```
10 SCREEN 2
20 CIRCLE (128,86),50
30 COLOR 1
40 GOTO 40
```

```
100
3E4
4EF
570
```

Após o desenho do círculo terminar, digite CONTROL + STOP. A tela deverá ficar totalmente escura. Para verificar o que está acontecendo basta digitar:

```
SHIFT + HOME/CLS   e
SHIFT + F1
```

Com isso, a tela será limpa e as cores normais do vídeo serão restabelecidas, desde que as teclas de funções não tenham sido redefinidas pelo programa que foi interrompido.

Se o procedimento descrito acima não funcionar, tente digitar o comando abaixo, mesmo sem vê-lo na tela:

```
COLOR 15,1 : SCREEN 0 (e RETURN)
```

2.2 - PSEUDO-BORDA NA SCREEN 0

O comando COLOR, quando usado com a SCREEN 0, não permite a especificação da cor da borda. Com um pequeno programa em BASIC podemos resolver parcialmente o problema, gerando uma PSEUDO-BORDA para a SCREEN 0. Digite e execute o programa a seguir e depois verifique as novas características da tela.

```
10 COLOR 1,15 : SCREEN 0          208
20 FOR F=2048 TO 4095             784
30   X = NOT(VPEEK(F)) AND 255    008
40   VPOKE F, X                  105F
50 NEXT F                          110F
60 INPUT "Qual a cor da borda (0-15)";B 118F6
70 IF B<0 OR B>15 THEN 60        2420
80 COLOR ,B                       267F
90 GOTO 60                         2A8A
```

TOTAL = 248A

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.3 - TEXTOS NA SCREEN 2

O MSX permite a impressão de letras e gráficos na SCREEN 2. Para isso é necessário abrir um arquivo na tela (GRP:) e usar a instrução PRINT #. Observe o programa a seguir:

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 CIRCLE (128,86),50
40 PRESET (110,84)
50 PRINT #1, "EDITORA ALEPH"
60 GOTO 60
```

100
409
786
887
127
1219

TOTAL = 1219

A linha 10 seleciona o uso da tela gráfica de alta resolução (GRP:).

A linha 20 abre um arquivo nessa tela. Não é necessário especificar o tipo de arquivo (...FOR OUTPUT...), uma vez que ele só pode ser de saída!

A linha 30 desenha um círculo de centro na posição (128,86) e raio de 50 pixels.

A linha 40 "marca" um ponto na posição (110,84), a partir do qual a mensagem será impressa.

A linha 50 imprime a mensagem na tela a partir do ponto marcado pela linha 40. Cada caractere é definido dentro de uma matriz de 8x8 pontos. A posição marcada pela linha 40 posiciona o vértice superior esquerdo do primeiro caractere da mensagem.

Observe que podemos "criar" novos tipos de letras na SCREEN 2, usando uma dupla impressão dos caracteres normais. Experimente inserir no programa anterior as seguinte linhas:

```
55 PRESET (109,84)
56 PRINT #1, "EDITORA ALEPH"
```

Isso deve ter produzido uma mensagem em "bold"!

Um outro recurso é o uso de espaçamento menor entre as letras a serem impressas. Experimente executar o programa a seguir. Ele imprime na SCREEN 2 uma mensagem com espaçamento reduzido.

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 A$="EDITORA ALEPH"
```

100
409
858

```

40 FOR F=1 TO LEN(A$)
50 PRESET (110+(F-1)*6,84)
60 PRINT #1,MID$(A$,F,1)
70 NEXT F
80 GOTO 80

```

```

100=
1786
1187
2150
2439

```

TOTAL = 2439

Para entender melhor o funcionamento do programa, experimente substituir a linha 50 por:

```
50 PRESET (110+(F-1)*12,84)
```



BIBLIOGRAFIA RECOMENDADA:

Curso de BASIC v.1 - páginas 65 e 66.
 Coleção de Programas para MSX - páginas 32, 33, 59 e 60.

2.4 - CARACTERES MENORES QUE 32

Os micros MSX dispõem de 256 caracteres, todos apresentáveis no vídeo. Entretanto, os primeiros 32 caracteres correspondem a códigos de controle de periféricos (0 a 31) e para serem apresentados através de seus códigos necessitam de uma sintaxe peculiar da instrução PRINT CHR\$. O programa a seguir apresenta os 32 caracteres de controle através de seus códigos.

```
10 SCREEN 1
20 WIDTH 16
30 FOR F=0 TO 31
40 PRINT CHR$(1)+CHR$(64+F);
50 NEXT F
```

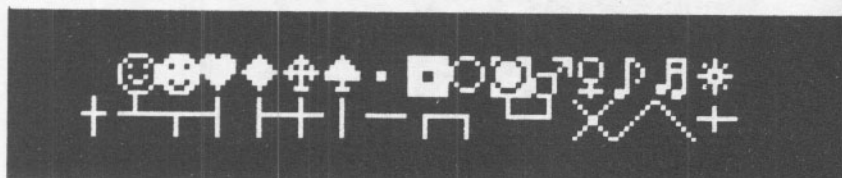
```
10E
20E
30E
40E
50E
```

```
TOTAL = 50E
```

Note que a apresentação é feita pela linha 40. Para se mostrar um caractere de controle através de seu código é necessário usar a sintaxe:

```
PRINT CHR$(1)+CHR$(64+ nq )
```

Onde "nq" é o código (de 0 a 31) do caractere a ser apresentado.



BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.1 - páginas 26 e 27.

2.5 - CARACTERES DE CONTROLE DE VÍDEO

O padrão de vídeo dos MSX é um dos mais utilizados em microcomputadores. Alguns caracteres podem ser usados para controlar o vídeo em funções como o posicionamento do cursor, limpeza da tela, formato do cursor, etc.

A seguir relacionamos os caracteres de controle do vídeo do MSX. Eles devem ser usados com o comando PRINT. Por exemplo, para deixar o cursor posicionado na linha 10 e coluna 20 basta comandar:

```
PRINT CHR$(27)CHR$(32+10)CHR$(32+20);
```

Apaga a tela

```
CHR$(27)"J"      ou  
CHR$(27)"E"
```

Apaga do cursor ao fim da linha

```
CHR$(27)"K"
```

Apaga do cursor ao fim da tela

```
CHR$(27)"J"
```

Apaga a linha inteira

```
CHR$(27)"I"
```

Insere uma linha em branco

```
CHR$(27)"L"
```

Elimina uma linha

```
CHR$(27)"M"
```

Posiciona cursor

```
CHR$(27)"Y";  
CHR$(32+nº da linha);  
CHR$(32+nº da coluna)
```

Cursor linha acima

```
CHR$(27)"A"
```

Cursor linha abaixo

```
CHR$(27)"B"
```

Cursor coluna a direita

```
CHR$(27)"C"
```

Cursor coluna a esquerda

```
CHR$(27)"D"
```

Cursor em HOME
CHR\$(27)"H"

Cursor inteiro
CHR\$(27)"x4"

Cursor pela metade
CHR\$(27)"y4"

Cursor apagado
CHR\$(27)"x5"

Cursor aceso
CHR\$(27)"y5"

O programa a seguir ilustra o uso de alguns dos recursos descritos acima. Digite-o, rode-o e estude-o.

10 SCREEN 0 : WIDTH 38 : KEY OFF	020
11 PRINT "Digite qualquer coisa";	056
12 PRINT "e use as teclas de setas";	106
13 PRINT " e as teclas HOME e CLS !"	156
14 PRINT : PRINT : PRINT	206
20 A\$=INKEY\$: IF A\$="" THEN 20	256
30 IF A\$(<)CHR\$(12) THEN 50	306
40 PRINT CHR\$(27)"j"	356
50 IF A\$(<)CHR\$(11) THEN 70	406
60 PRINT CHR\$(27)"k"	456
70 PRINT A\$;	506
90 GOTO 20	556

TOTAL = 6050

BIBLIOGRAFIA RECOMENDADA.

Sistema de Disco para MSX - página 106.

2.6 - USANDO O STEP EM COMANDOS GRÁFICOS

A instrução auxiliar STEP pode ser muito útil quando usada com comandos gráficos.

Veja o programa a seguir:

```
10 SCREEN 2 100  
20 PSET (20,20) 200  
30 PSET STEP (20,20) 300  
40 GOTO 40 400
```

TOTAL = 735

Ao ser executado ele irá plotar dois pontos na diagonal que sai do canto superior esquerdo da tela.

Isso acontece porque a instrução STEP permite a definição de um novo sistema de coordenadas na tela.

O sistema de coordenadas normal é um sistema ABSOLUTO, isto é, suas coordenadas são sempre correspondentes a uma mesma origem, fixa no canto superior esquerdo da tela.

O sistema usado pelo STEP é um sistema MÓVEL, em que as coordenadas correspondem a uma origem móvel, definida pelo último ponto "marcado" na tela. Note que PONTO "MARCADO" não é necessariamente PONTO "PLOTADO". Por exemplo, ao se usar um comando CIRCLE o ponto "marcado" é o seu centro, enquanto que os pontos plotados são os do círculo.

Vamos tentar entender isso melhor.

Observe novamente o programa anterior.

A linha 10 apenas seleciona a SCREEN 2 do micro. Essa tela tem uma resolução de 256 colunas x 192 linhas, num total de 49152 pontos.

A linha 20 marca o ponto de coordenadas $x=20$ e $y=20$ na tela. Note que essas coordenadas são referentes ao sistema ABSOLUTO, pois NÃO existe a instrução STEP precedendo as coordenadas.

A linha 30 também marca um ponto na tela (o de coordenadas $x'=20$ e $y'=20$), porém as coordenadas desse ponto são precedidas pela instrução STEP e, portanto, referem-se ao sistema de coordenadas MÓVEL. Como o último ponto marcado na tela foi o de coordenadas ABSOLUTAS $x=20$ e $y=20$, esse ponto foi tomado como ORIGEM do sistema MÓVEL. Portanto, as coordenadas $x'=20$ e $y'=20$ do sistema MÓVEL correspondem às coordenadas $x=40$ e $y=40$ do sistema ABSOLUTO.

Agora, esquite um pouco a cabeça tentando entender o funcionamento dos três programinhas listados na próxima página.

PROGRAMA 1

```
10 SCREEN 2
20 PSET (0,0)
30 CIRCLE STEP(6,6),4
40 GOTO 30
```

100

288

574

705

PROGRAMA 2

```
10 SCREEN 2
20 PSET (0,0)
30 LINE STEP(3,3)-STEP(3,3)
40 GOTO 30
```

100

288

700

814

PROGRAMA 3

```
10 SCREEN 2
20 FOR F=0 TO 6.28 STEP .2
30   X=80*SIN(F) : Y=80*COS(F)
40   LINE STEP(X,Y)-(128,86)
50 NEXT F
60 GOTO 60
```

100

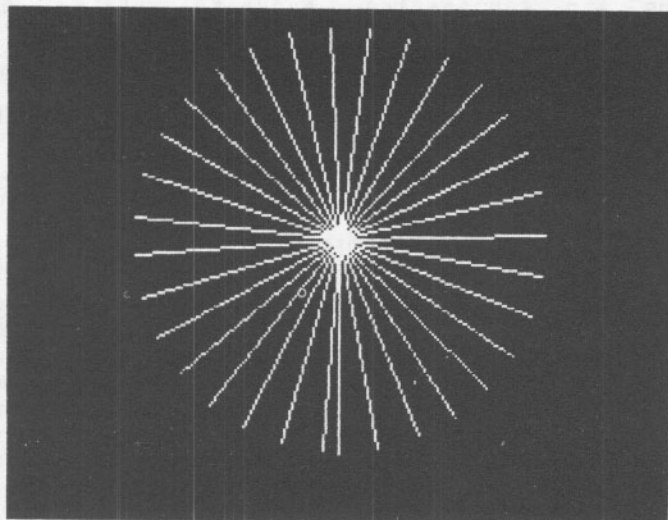
702

1100

1868

1785

104E



2.7 - POSICIONANDO O CURSOR NA SCREEN 2

A forma mais genérica de se posicionar o cursor na SCREEN 2 é usando o seguinte comando:

```
PSET (coluna,linha),POINT (coluna,linha)
```

Por exemplo, para posicionar o cursor na coluna 89 e linha 190 devemos fazer:

```
PSET (89,190),POINT (89,190)
```

Se quisermos posicionar o cursor para impressão de textos na SCREEN 2 podemos considerá-la dividida em 32 colunas e 24 linhas, como a SCREEN 1. Desse modo, o comando PSET acima pode ser substituído por:

```
PSET (8*col,8*lin),POINT (8*col,8*lin)
```

Para imprimir na coluna 10 e linha 5 devemos usar então:

```
PSET (8*10,8*5),POINT (8*10,8*5)
```

Agora, digite e rode o programa a seguir:

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 C = 10 : L = 10
40 PSET(8*C,8*L),POINT(8*C,8*L)
50 PRINT #1, "ALEPH"
60 GOTO 50
```

```
100
409
80E
1108
1500
1813
```

```
TOTAL = 1813
```

Na linha 30 define-se a coluna (de 0 a 31) e a linha (de 0 a 23) onde desejamos imprimir. A linha 40 posiciona o cursor e a linha 50 imprime na tela.

Veja a dica 2.3 para entender melhor o programa.

2.8 - MOVIMENTOS NA TELA

Muitas vezes pode-se desejar que a tela do MSX apresente algum movimento global. A seguir, apresentamos seis programas que produzem movimentos globais na SCREEN 1.

PROGRAMA 1

```
10 ' TESTE DE MOVIMENTO
20 SCREEN 1 : WIDTH 32 : COLOR 1,7,4
30 FOR F=0 TO 22
40 PRINT STRING$(32,"0"); : NEXT F
50 FOR F=8*ASC("0") TO 8*ASC("0")+7
60 IF B=0 THEN VPOKE F,VPEEK(F)/8
70 IF B=1 THEN VPOKE F,8*VPEEK(F)
80 NEXT F : B=(B+1) MOD 2 : GOTO 50
```

1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

TOTAL = 4950

PROGRAMA 2

```
100 ' TESTE DE MOVIMENTO 2
110 SCREEN 1 : WIDTH 32 : COLOR 1,7,4
120 FOR F=0 TO 22
130 PRINT STRING$(32,"."); : NEXT F
140 A=VPEEK(8*ASC("."))
150 FOR F=8*ASC(".") TO 8*ASC(".")+6
160 VPOKE(F),VPEEK(F+1)
170 NEXT F
180 VPOKE F,A
190 GOTO 140
```

81E
007
1029
1901
2103
2753
377E
3906
3DEA
4080

TOTAL = 4080

PROGRAMA 3

```
200 SCREEN1 : WIDTH 32
210 FOR F=48 TO 69
220 A$=A$+CHR$(F)
230 NEXT F
240 FOR F=1 TO 31
250 A=INT(1+RND(1)*21)
260 A$=RIGHT$(A$,A)+LEFT$(A$,21-A)
270 FOR G=1 TO 21:LOCATE F,G
280 PRINT MID$(A$,G,1):NEXT G
290 NEXT F
300 FOR F=48*8 TO 69*8+7
310 A=INT(8*RND(1))
```

330
815
0EE
FF5
1497
1007
2865
2507
387F
3829
459A
40E7

320 IF RND(1)>.01 THEN VPOKE F,0 ELSE VP	6105
330 OKE F,2^A	
330 NEXT F	64F4
340 AZ=VPEEK(69*8+7)	6096
350 FOR FZ=69*8+6 TO 48*8 STEP-1	7003
360 VPOKE FZ+1,VPEEK(FZ)	8300
370 NEXT FZ	8480
380 VPOKE FZ+1,AZ	8879
390 GOTO 340	8AAA

TOTAL = 8AAA

PROGRAMA 4

400 SCREEN1 : WIDTH 32	2F9
410 FOR F=48 TO 69	795
420 A\$=A\$+CHR\$(F)	067
430 NEXT F	F35
440 FOR F=1 TO 31	1390
450 A=INT(1+RND(1)*21)	10F9
460 A\$=RIGHT\$(A\$,A)+LEFT\$(A\$,21-A)	2787
470 FOR G=1 TO 21:LOCATE F,G	2769
480 PRINT MID\$(A\$,G,1):NEXT G	3849
490 NEXT F	3A88
500 FOR F=48*8 TO 69*8+7	4504
510 A=INT(8*RND(1))	4009
520 IF RND(1)>.01 THEN VPOKE F,0 ELSE VP	6090
530 OKE F,2^A	
530 NEXT F	6424
540 AZ=VPEEK(48*8)	605A
550 FOR FZ=48*8 TO 69*8+6	7894
560 VPOKE FZ,VPEEK(FZ+1)	8111
570 NEXT FZ	8270
580 VPOKE FZ,AZ	84FF
590 GOTO 540	8670

TOTAL = 8670

PROGRAMA 5

600 ' PSEUDO-SCROLL A ESQUERDA	991
610 SCREEN 1 : WIDTH 32 : COLOR 1,7,4	0A8
620 X=8*ASC(" ")	12E8
630 B=0 : VPOKE X,&B00000001	1AA5
640 VPOKE X,VPEEK(X)*2	209F
650 B=B+1 : IF B=7 THEN 630	2098
660 GOTO 640	2E42

TOTAL = 2E42

PROGRAMA 6

670 ' PSEUDO-SCROLL A DIREITA	983
680 SCREEN 1 : WIDTH 32 : COLOR 1,7,4	00A
690 X=8*ASC(" ")	1312
700 B=0 : VPOKE X,&B10000000	1380
710 VPOKE X,VPEEK(X)/2	218C
720 B=B+1 : IF B=7 THEN 700	2AFC
730 GOTO 710	2F0D

TOTAL = 2F0D

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.9 - SCREEN 4

Quando comandamos SCREEN 1 ou SCREEN 2, o interpretador BASIC prepara a VRAM com as tabelas próprias para a tela selecionada e o VDP para agir sobre essas tabelas.

Analisando os procedimentos executados pelo micro quando usamos um comando SCREEN podemos idealizar uma forma de fazer a VRAM ser preparada com as tabelas da SCREEN 2 e as variáveis do sistema serem preparadas para operar sobre a SCREEN 1! Desse modo, poderemos conciliar a rapidez de operação da SCREEN 1 com todos os recursos de cores e formas da SCREEN 2.

A idéia é "enganar" o interpretador, fazendo-o pensar que está operando sobre a SCREEN 1, enquanto a VRAM e o VDP estarão preparados para a SCREEN 2. Note que as variáveis do sistema, preparadas pelo interpretador para uma dada SCREEN, funcionam como um "mecanismo inibitório" para o VDP, fazendo-o comportar-se sob rígido controle. Se usarmos diretamente as rotinas do BIOS, sem que o interpretador "saiba", poderemos evitar os "mecanismos inibitórios" e gerar algo parecido com uma "ESQUIZOFRENIA INDUZIDA" no micro.

Observe atentamente o programa a seguir. Digite-o e, antes de executá-lo, leia os comentários que o seguem.

PROGRAMA SCREEN 4

100 COLOR 15,1,15	10F
110 SCREEN 2	33E
120 DEFINT A-Z	62E
130 SCREEN 1	80F
140 DEFUSR = &H7E	B8F
150 X = USR(0)	F28
160 FOR F = 0 TO 2047	1488
170 X = PEEK (&H1BBF + F)	11FC
180 VPOKE (X, X)	2615
190 VPOKE (&H800 + F), X	20F4
200 VPOKE (&H1000 + F), X	3602
210 NEXT F	4830
220 FOR F=0 TO 7	3EE2
230 VPOKE 2048+255*8+F,255	4811
240 VPOKE 4096+255*8+F,255	511F
250 NEXT F	5807

TOTAL = 5807

Antes de mais nada, lembre-se de como a VRAM fica dividida quando se usa a SCREEN 1 ou a SCREEN 2.

Uma vez estudada a estrutura da VRAM quando no modo SCREEN 1 ou SCREEN 2, podemos analisar o programa.

A linha 100 serve apenas para que a cor da borda da tela seja selecionada quando o comando SCREEN na linha 110 for executado.

A linha 110 seleciona a SCREEN 2 para que as tabelas da VRAM sejam preparadas para ela.

A linha 120 serve apenas para aumentar um pouco a velocidade de execução do programa, afinal ele necessita apenas de valores inteiros. Este recurso dos micros MSX é extremamente útil para esta finalidade.

A linha 130 seleciona a SCREEN 1 para que as variáveis do sistema indiquem ao interpretador o modo SCREEN 1. Note que até aqui não houve nenhum "truque" de programação digno de maiores explicações. O interpretador ainda não foi "enganado"! Isso só ocorrerá após a execução das próximas linhas.

As linhas 140 e 150 executam a rotina do BIOS SETGRP (em &H007E). Ela prepara o VDP para acessar as tabelas da SCREEN 2. Note que, com isso, já temos a VRAM preparada como SCREEN 1 (e residualmente como SCREEN 2) e o VDP preparado para acessar a SCREEN 2.

As linhas de 160 a 210 carregam as tabelas de caracteres da VRAM com os caracteres da ROM (de 0 a 255). Os desenhos dos 255 caracteres são definidos três vezes, uma para cada terço da tela. A tabela correspondente às linhas de 0 a 7 da tela ocupa os primeiros 2 Kbytes da VRAM. De 2 a 4 Kbytes está a tabela correspondente às linhas de 8 a 15. As linhas de 16 a 23 usam a tabela de caracteres entre 4 e 6 Kbytes.

As linhas de 220 a 250 apenas redefinem os desenhos do caractere 255 (cursor) para serem usados no segundo e terceiro terço da tela.

Note que a linha 180 poderia ter sido omitida sem nenhum problema, pois o comando SCREEN 1 da linha 130 já havia carregado os desenhos dos 255 caracteres no primeiro terço da tela.

ALTERANDO AS CORES DOS CARACTERES

Na SCREEN 1 podemos alterar a cor de grupos de 8 caracteres. Por exemplo, se alterarmos a cor do caractere 3, as cores dos caracteres de 0 a 7 serão simultaneamente alteradas. Na SCREEN 2, podemos alterar a cor de cada um dos caracteres

individualmente. Podemos ainda ir mais além e definir, para cada caractere, 16 diferentes cores! Isso mesmo, 16 cores em cada caractere!

Acrescente ao programa anterior as linhas mostradas a seguir e rode-o novamente. Com isso, os caracteres dos números e dos parênteses terão suas cores redefinidas no 1º terço da tela!

```

260 X=8192+8*ASC("(")
270 FOR F=X TO X+7
280   VPOKE F,&B1000001
290 NEXT F
300 X=8192+8*ASC(")")
310 FOR F=X TO X+7
320   VPOKE F,&B1000001
330 NEXT F
340 X=8192+8*ASC("0")
350 Y=8192+8*ASC("9")+7
360 FOR F=X TO Y
370   VPOKE F,&B11010001
380 NEXT F

```

```

5F25
66E0
73F5
77B2
8055
8534
8905
8AB6
9084
9995
9DA4
A3BC
A5A5

```

TOTAL = A5A5

Agora vamos fazer com que o cursor seja redefinido com as 16 cores. Acrescente também ao programa as linhas mostradas a seguir. Depois, execute-o.

```

390 F=8192+8*255
400 VPOKE F+0,&B00001000
410 VPOKE F+1,&B00011001
420 VPOKE F+2,&B00101010
430 VPOKE F+3,&B00111011
440 VPOKE F+4,&B01001100
450 VPOKE F+5,&B01011101
460 VPOKE F+6,&B01101110
470 VPOKE F+7,&B01111111
480 F=10240+8*255
490 VPOKE F+0,&B00001000
500 VPOKE F+1,&B00011001
510 VPOKE F+2,&B00101010
520 VPOKE F+3,&B00111011
530 VPOKE F+4,&B01001100
540 VPOKE F+5,&B01011101
550 VPOKE F+6,&B01101110
560 VPOKE F+7,&B01111111
570 F=12288+8*255
580 VPOKE F+0,&B00001000

```

```

A678
B7A5
C2A3
CCCC
D8AA
E621
F201
FF0C
474
9C1
E80
12C4
1A87
2172
2706
2FB3
3B13
4610
4CCA
5A66

```

```
590 VPOKE F+1,&B00011001
600 VPOKE F+2,&B00101010
610 VPOKE F+3,&B00111011
620 VPOKE F+4,&B01001100
630 VPOKE F+5,&B01011101
640 VPOKE F+6,&B01101110
650 VPOKE F+7,&B01111111
```

```
218F
740F
8180
870E
E081
E0F3
960E
```

TOTAL = 960E

Agora, o cursor está colorido!

Os exemplos apresentados são bem simples para facilitar a compreensão. Os recursos oferecidos pela SCREEN híbrida que apresentamos são, entretanto, muito mais vastos.

Você deve ter percebido que o programa demora vários segundos para ser executado. Isso é aceitável quando levamos em conta que o BASIC tem que acessar quase 16 Kbytes, entretanto para os programadores mais exigentes a demora pode ser um fator muito negativo.

Podemos pensar, então, em transformar o programa numa rotina em Linguagem de Máquina. Indo além, podemos pensar numa rotina que permita a implementação do comando SCREEN 4 no BASIC, de modo que ao ser executado ele gere a tela híbrida com a mesma velocidade que as outras SCREEN's. Para facilitar a alteração das cores dos caracteres podemos imaginar um novo comando do BASIC ou ainda o aproveitamento de comandos não implementados como o IPL ou o CMD. Isto, entretanto, já é assunto para um texto mais extenso.

Se você não quiser esperar, poderá encontrar estas e muitas outras idéias já executadas e analisadas no livro PROGRAMAÇÃO AVANÇADA EM MSX. Para estudar mais detalhadamente a estrutura da VRAM nas várias SCREEN's, as rotinas do BIOS e as Variáveis do Sistema, sugerimos a leitura atenta do livro APROFUNDANDO-SE NO MSX. Nesses dois livros os assuntos são tratados de forma bastante completa.

Exemplos e aplicações práticas comentadas passo a passo podem ser encontradas nos livros COLEÇÃO DE PROGRAMAS PARA MSX, volumes 1 e 2.

Para completar seu conhecimento sobre a SCREEN 1, veja a dica 2.Q (Arlequim Bêbado).

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.A - SETORES COM CIRCLE

O comando CIRCLE do BASIC permite o traçado de arcos de circunferências e de perímetros de setores circulares.

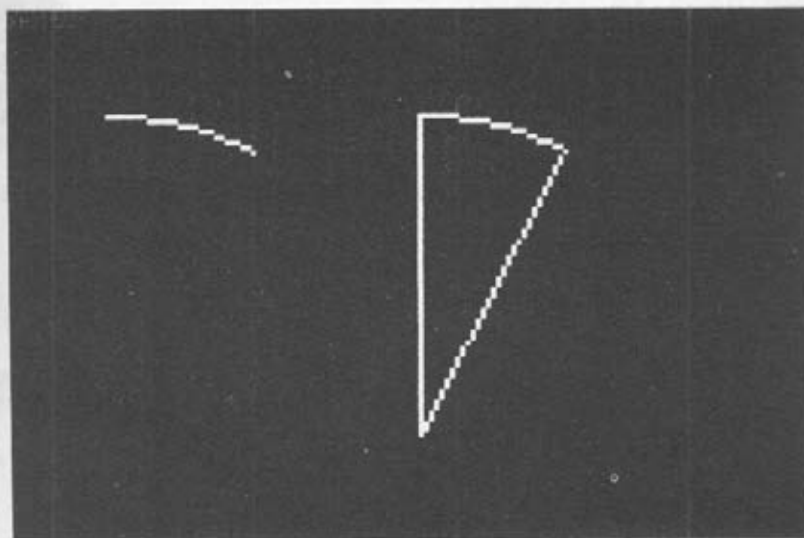
Para traçar setores, basta usar valores negativos para os ângulos inicial e final. Veja o programa a seguir:

```
10 SCREEN 2
20 PI = 4*ATN(1)
30 CIRCLE (70,80),60,15,PI/3,PI/2,1
40 CIRCLE (130,80),60,15,-PI/3,-PI/2,1
50 GOTO 50
```

A linha 20 define a variável PI, atribuindo-lhe o valor da constante matemática π .

A linha 30 traça um arco de circunferência normal.

A linha 40 traça o perímetro de um setor, pois os ângulos são negativos.



2.8 - REDEFININDO CARACTERES

Uma maneira rápida de redefinir caracteres é acessar a tabela de definição diretamente na VRAM. Essa tabela ocupa posições diferentes em cada tipo de tela (SCREEN's 0, 1, 2 ou 3). A seguir apresentamos 3 programas bem curtos que redefinem, apenas como exemplo, o formato do caractere de código 65 (ou &H41), originalmente a letra "A" nas SCREEN's 0, 1 e 2.

PARA A SCREEN 0

```
100 SCREEN 0
110 E=2048+8*65
120 FOR F=E TO E+7
130 READ A$
140 VPOKE F,VAL("&B"+A$)
150 NEXT F
160 PRINT : PRINT "CHR$(65)=";CHR$(65)
170 PRINT : LIST
180 DATA 00111100
190 DATA 01000010
200 DATA 10011001
210 DATA 10111101
220 DATA 10111101
230 DATA 10011001
240 DATA 01000010
250 DATA 00111100
```

TOTAL = 55lin

PARA A SCREEN 1

Apenas substitua as linhas 100 e 110 do programa anterior por:

```
100 SCREEN 1
110 E=8*65
```

PARA A SCREEN 2

```
100 SCREEN 2
110 FOR F=6144 TO 6144+24*32-1
120 VPOKE F,65
130 NEXT F
140 FOR F=0 TO 4096 STEP 2048
```

150	FOR G=F+8*65 TO F+8*65+7	106A
160	READ A% : VPOKE G,VAL("&B"+A%)	2800
170	VPOKE G+8192,&B00011111	833A
180	NEXT G	8A7E
190	RESTORE	6030
200	NEXT F	8758
210	GOTO 210	4904
220	DATA 00111100	4866
230	DATA 01000010	4E72
240	DATA 10011001	5770
250	DATA 10111101	61A0
260	DATA 10111101	68E9
270	DATA 10011001	7590
280	DATA 01000010	7E01
290	DATA 00111100	85F8

TOTAL = 85F8

Essa forma de redefinir caracteres, entretanto, nem sempre é a melhor, pois quando um comando SCREEN é executado as tabelas da VRAM são automaticamente recarregadas. Para entender melhor esse problema, rode novamente o programa feito para a SCREEN 0 e depois comande SCREEN 0. Você verá que o caractere de código 65 voltou a ser a letra "A". A redefinição feita na VRAM foi destruída pelo comando SCREEN.

Para contornar esse problema podemos redefinir toda a tabela de caracteres na própria RAM e fazer com que o comando SCREEN carregue essa nova tabela para a VRAM, ao invés de usar a tabela da ROM. O programa apresentado a seguir se presta a facilitar a redefinição dos caracteres de forma mais permanente. Digite e grave o programa. Depois veremos como usá-lo.

1000	KEYOFF : SCREEN 0 : WIDTH 40	604
1010	FOR E=&HB000 TO &HBAEF	08A
1020	READ C% : POKE E,VAL("&H"+C%)	1800
1030	NEXT E	15A8
1040	DEFUSR=&HB000 : X=USR(0)	10A9
1050	DATA 3E,01,21,AB,FC,77,CD,1E	2642
1060	DATA B1,CD,E5,B0,CD,26,B2,21	3153
1070	DATA 08,00,22,B7,FC,21,B8,00	4880
1080	DATA 22,B9,FC,21,CB,BA,7E,2F	5304
1090	DATA FE,00,28,06,CD,8D,00,23	6017
1100	DATA 18,F4,16,08,CD,57,B2,FE	6708
1110	DATA 53,20,06,AF,21,AB,FC,77	8E6A
1120	DATA C9,21,09,B0,E5,FE,43,CA	9488
1130	DATA 96,B2,FE,0D,28,1F,0E,01	9720

1140 DATA FE,1C,28,11,0E,FF,FE,1D
1150 DATA 28,0B,0E,F0,FE,1E,28,05
1160 DATA 0E,10,FE,1F,C0,3A,C9,B2
1170 DATA 81,32,C9,B2,C9,CD,0E,B2
1180 DATA 16,02,CD,57,B2,FE,0D,C8
1190 DATA 21,65,B0,E5,01,00,FE,FE
1200 DATA 20,28,24,0C,FE,4D,28,1F
1210 DATA FE,1C,28,11,0E,FF,FE,1D
1220 DATA 28,0B,0E,F8,FE,1E,28,05
1230 DATA 0E,08,FE,1F,C0,3A,CA,B2
1240 DATA 81,E6,3F,32,CA,B2,C9,CD
1250 DATA 46,B2,3A,CA,B2,F5,0F,0F
1260 DATA 0F,E6,07,5F,16,00,FD,19
1270 DATA F1,E6,07,3C,CB,08,CB,09
1280 DATA 3D,20,F9,FD,7E,00,A0,B1
1290 DATA FD,77,00,CD,E5,B0,CD,46
1300 DATA B2,CD,26,B2,CD,CB,B1,06
1310 DATA 08,D5,E5,3E,08,FD,5E,00
1320 DATA CD,EC,B1,E1,D1,CD,E0,B1
1330 DATA FD,23,10,ED,C9,CD,46,B2
1340 DATA 0E,BF,1E,07,CD,CB,B1,06
1350 DATA 08,0E,05,C5,D5,E5,06,08
1360 DATA FD,7E,00,07,F5,9F,5F,3E
1370 DATA 05,CD,EC,B1,CD,D6,B1,CD
1380 DATA D6,B1,F1,10,EE,E1,D1,C1
1390 DATA CD,E0,B1,0D,20,DD,CD,E0
1400 DATA B1,FD,23,10,D4,C9,01,00
1410 DATA 08,11,CB,B2,2A,20,F9,C5
1420 DATA D5,3A,1F,F9,CD,0C,00,FB
1430 DATA D1,C1,12,13,23,0B,78,B1
1440 DATA 20,ED,CD,72,00,3A,E9,F3
1450 DATA 07,07,07,07,4F,3A,EA,F3
1460 DATA B1,01,00,18,2A,C9,F3,CD
1470 DATA 56,00,21,0B,B1,01,0A,FF
1480 DATA 1E,06,3E,11,CD,8A,B1,21
1490 DATA 06,31,01,BE,AA,1E,06,3E
1500 DATA 09,CD,8A,B1,21,30,31,01
1510 DATA BE,FF,1E,06,3E,02,CD,8A
1520 DATA B1,AF,32,CA,B2,21,C9,B2
1530 DATA 77,E5,CD,C6,B0,E1,34,20
1540 DATA F8,C9,F5,C5,E5,CD,CB,B1
1550 DATA C1,F1,5F,F1,F5,D5,E5,F5
1560 DATA C5,D5,E5,78,CD,EC,B1,E1
1570 DATA D1,CD,E0,B1,0D,20,FA,C1
1580 DATA F1,3D,20,EB,E1,D1,F1,F5
1590 DATA C5,D5,E5,3E,01,CD,EC,B1
1600 DATA CD,E0,B1,10,F6,E1,D1,CD
1610 DATA D6,B1,0D,20,FA,C1,F1,3D
1620 DATA 20,E5,C9,06,00,50,CD,11

A297
A8A0
B004
C072
D038
E58C
9AE
10BE
15F6
1D30
2606
3229
43D2
5506
6A68
7E7B
888F
90FD
97FB
A144
A92D
B801
C00E
E03C
F518
850
E62
133C
1AE7
2495
3173
432F
53F5
69A9
70A1
8938
8E7D
94F9
9DFF
AB71
B984
C8A0
D245
E22A
E4D
CAB
1288
1A7E
2838

1630 DATA 01,CD,14,01,57,C9,CB,0A
1640 DATA D0,C5,01,08,00,09,C1,C9
1650 DATA 23,7D,E6,07,C0,C5,01,FB
1660 DATA 00,09,C1,C9,C5,47,CD,4A
1670 DATA 00,4F,7A,2F,A1,CB,03,30
1680 DATA 01,B2,05,28,0C,CB,0A,30
1690 DATA F0,CD,4D,00,CD,D9,B1,18
1700 DATA E5,CD,4D,00,C1,C9,3A,CA
1710 DATA B2,F5,E6,07,07,4F,07,81
1720 DATA C6,BF,4F,F1,E6,38,0F,5F
1730 DATA 0F,83,C6,07,5F,C9,3A,C9
1740 DATA B2,F5,CD,3C,B2,C6,0C,4F
1750 DATA F1,0F,0F,0F,0F,CD,3C,B2
1760 DATA C6,08,5F,C9,E6,0F,57,07
1770 DATA 47,07,07,80,82,C9,3A,C9
1780 DATA B2,6F,26,00,29,29,29,EB
1790 DATA FD,21,CB,B2,FD,19,C9,06
1800 DATA 00,C5,D5,CD,78,B2,D1,C1
1810 DATA 04,21,40,1F,CD,9C,00,20
1820 DATA 07,2B,7C,B5,20,F6,18,E9
1830 DATA CB,40,C4,78,B2,C3,9F,00
1840 DATA D5,CD,CB,B1,F1,47,5F,D5
1850 DATA E5,CD,4A,00,AA,CD,4D,00
1860 DATA CD,D6,B1,1D,20,F3,E1,D1
1870 DATA CD,E0,B1,10,EA,C9,01,00
1880 DATA 08,11,80,8B,ED,53,20,F9
1890 DATA 21,CB,B2,ED,80,CD,38,01
1900 DATA 07,07,E6,03,4F,06,00,21
1910 DATA C1,FC,09,CB,7E,28,0E,21
1920 DATA C5,FC,09,7E,07,07,07,07
1930 DATA E6,0C,B1,CB,FF,32,1F,F9
1940 DATA C9,00,00,01,24,79,1F,1F
1950 DATA 1F,E6,07,E9,E5,2A,5E,EC
1960 DATA 7E,23,22,5E,EC,E1,4F,C9
1970 DATA AF,BE,28,0D,79,BE,23,3E
1980 DATA 00,C8,CB,7E,23,28,FB,18
1990 DATA EF,23,7E,23,A1,BE,23,7E
2000 DATA 23,28,0D,CB,7E,20,F3,7E
2010 DATA 87,23,30,FB,CB,7E,20,EA
2020 DATA CB,7E,C8,23,23,23,18,F8
2030 DATA E5,C5,21,33,63,01,05,00
2040 DATA ED,B1,C1,E1,28,0D,C9,FE
2050 DATA E9,28,08,C9,FE,45,28,03
2060 DATA FE,4D,C0,F5,3E,FF,32,47
2070 DATA EC,F1,C9,18,C3,76,C9,E9
2080 DATA CD,D4,62,18,0B,CD,D4,62
2090 DATA 47,CD,D4,62,CD,31,64,78
2100 DATA CD,31,64,3E,48,C3,41,64
2110 DATA 79,E6,38,18,F3,3E,49,CB

3038
4264
534E
6941
709C
888F
8E0A
9560
A022
A892
B8EE
C0F7
E089
F5D8
80D
E64
144C
18F4
2548
3298
44A8
5746
6C88
808D
8827
986D
97C6
A27B
AA59
BC87
CE63
E216
F6A1
848
EF9
1385
1063
2672
34AF
4641
5638
6A63
7F14
8974
8E8E
9495
9E4F
A6E8
B8CA

2120 DATA 59,28,02,3E,52,18,22,C3
 2130 DATA 3A,64,3E,30,11,3E,31,11
 2140 DATA 3E,32,18,15,CD,29,64,CD
 2150 DATA 84,63,18,3F,CD,29,64,CD
 2160 DATA 9B,63,18,37,CD,87,63,3E
 2170 DATA 27,C3,41,64,3E,0E,11,3E
 2180 DATA 20,11,3E,26,11,3E,0A,11
 2190 DATA 3E,0C,11,3E,07,18,77,FE
 2200 DATA 06,28,EC,E6,3E,C6,08,FE
 2210 DATA 0C,20,6B,3A,55,EC,C6,0C
 2220 DATA 18,64,CD,29,64,CD,D4,62
 2230 DATA CD,48,63,3E,29,18,CA,CD
 2240 DATA 29,64,C5,CD,3D,63,C1,18
 2250 DATA F2,CD,29,64,CD,A3,63,18
 2260 DATA EA,E6,03,87,C6,10,18,3E
 2270 DATA CD,D4,62,3A,5E,EC,81,47
 2280 DATA 3A,5F,EC,CE,00,CB,79,28
 2290 DATA 01,3D,C3,44,63,79,E6,07
 2300 DATA FE,06,20,22,CD,29,64,CD
 2310 DATA A3,63,3A,55,EC,87,28,BB
 2320 DATA C5,CD,D4,62,C1,87,28,B3
 2330 DATA 16,2B,F2,09,64,16,2D,ED
 2340 DATA 44,CD,2B,64,18,A2,21,AF
 2350 DATA 64,85,6F,30,01,24,7E,E6
 2360 DATA 7F,FE,20,C4,41,64,BE,F8
 2370 DATA 23,18,F3,FD,21,D1,EC,18
 2380 DATA ED,16,28,FD,72,00,FD,23
 2390 DATA C9,F5,0F,0F,0F,0F,CD,3A
 2400 DATA 64,F1,E6,0F,FE,0A,DE,69
 2410 DATA 27,FD,77,00,FD,23,C9,3E
 2420 DATA 0D,18,F6,FD,21,13,ED,3E
 2430 DATA 50,FD,2B,FD,36,00,20,3D
 2440 DATA 20,F7,C9,3B,3B,E5,3B,E8
 2450 DATA 8A,8A,E8,9B,3B,5F,E5,E8
 2460 DATA 3B,90,9B,A3,9B,E8,E5,87
 2470 DATA 7C,50,3B,93,E5,62,3B,65
 2480 DATA 3B,68,3B,84,A3,74,93,93
 2490 DATA 74,93,55,55,93,C1,3B,97
 2500 DATA 3B,CB,3B,8D,90,6C,A3,93
 2510 DATA AA,AA,93,3B,3B,93,3B,E8
 2520 DATA 3B,9B,3D,C9,D0,D0,3B,CB
 2530 DATA 3D,3D,3B,93,B7,B7,93,A3
 2540 DATA B7,B7,A3,9B,B7,B7,9B,C2
 2550 DATA C3,C4,C5,C8,CC,CD,C1,42
 2560 DATA C3,44,C5,48,CC,53,D0,4E
 2570 DATA DA,5A,A0,4E,C3,43,A0,50
 2580 DATA CF,50,C5,50,A0,4D,A0,41
 2590 DATA C6,49,D8,49,D9,28,43,A9
 2600 DATA 3F,43,43,C6,2F,43,50,CC

C8DF
 DDF0
 F3A6
 512
 803
 F4E
 1A0E
 2352
 3206
 430F
 5566
 6AD9
 7ED9
 87DA
 8D35
 9486
 9DB1
 A68E
 BBCE
 CA60
 DF89
 F562
 637
 DDA
 10B5
 1A60
 2280
 3100
 43A3
 55E4
 6BAD
 8003
 88FC
 8EF6
 962A
 A08F
 AA16
 BBAF
 C0C4
 E36F
 F7B0
 74E
 E21
 14AB
 1DF3
 25ED
 34FE
 4631
 5720

2610	DATA	27,44,41,C1,F3,44,C9,FB	60E9
2620	DATA	45,C9,D9,45,58,D8,76,48	82E1
2630	DATA	41,4C,D4,17,52,4C,C1,07	8B46
2640	DATA	52,4C,43,C1,1F,52,52,C1	8FCF
2650	DATA	0F,52,52,43,C1,37,53,43	9729
2660	DATA	C6,00,FF,2A,26,FF,22,27	A003
2670	DATA	FF,3A,24,FF,32,25,FF,F9	A808
2680	DATA	10,EF,02,11,EF,0A,12,C0	8883
2690	DATA	40,09,C7,06,1E,CF,01,1F	C808
2700	DATA	4C,C4,FF,CE,1D,F8,88,0C	0F9E
2710	DATA	41,44,C3,FF,C6,1D,F8,80	F4F1
2720	DATA	0C,CF,09,08,41,44,C4,FF	3F1
2730	DATA	E6,1C,F8,A0,01,41,4E,C4	A85
2740	DATA	FF,CD,23,C7,C4,22,43,41	F2F
2750	DATA	4C,CC,FF,FE,1C,F8,88,01	1740
2760	DATA	43,D0,C7,05,06,CF,08,04	2000
2770	DATA	44,45,C3,FF,10,21,44,4A	2FDC
2780	DATA	4E,DA,FF,EB,18,FF,E3,19	4215
2790	DATA	FF,08,0A,45,D8,FF,DB,1A	5403
2800	DATA	49,CE,C7,04,06,CF,03,04	6981
2810	DATA	49,4E,C3,C7,C2,22,FF,C3	70CE
2820	DATA	23,FF,E9,15,4A,D0,FF,18	861B
2830	DATA	21,E7,20,20,4A,D2,FF,D3	8AF1
2840	DATA	1B,4F,55,D4,FF,00,00,4E	9260
2850	DATA	4F,D0,FF,F6,1C,F8,80,01	9813
2860	DATA	4F,D2,CF,C1,16,50,4F,D0	A496
2870	DATA	CF,C5,16,50,55,53,C8,FF	B600
2880	DATA	C9,00,C7,C0,17,52,45,D4	C750
2890	DATA	C7,C7,0B,52,53,D4,FF,DE	DC05
2900	DATA	1D,F8,98,0C,53,42,C3,FF	F0EC
2910	DATA	D6,1C,F8,90,01,53,55,C2	C
2920	DATA	FF,EE,1C,F8,A8,01,58,4F	627
2930	DATA	D2,00,74,66,6F,53,20,79	967
2940	DATA	6E,69,54,00,C0,40,05,42	143C
2950	DATA	49,D4,C0,80,05,52,45,D3	1000
2960	DATA	C0,C0,05,53,45,D4,F8,10	2E08
2970	DATA	01,52,CC,F8,18,01,52,D2	3EA2
2980	DATA	F8,00,01,52,4C,C3,F8,08	505F
2990	DATA	01,52,52,C3,F8,20,01,53	6768
3000	DATA	4C,C1,F8,28,01,53,52,C1	7BC7
3010	DATA	F8,38,01,53,52,CC,00,A9	828F
3020	DATA	43,50,C4,B9,43,50,44,D2	876B
3030	DATA	A1,43,50,C9,B1,43,50,49	8F94
3040	DATA	D2,AA,49,4E,C4,BA,49,4E	987A
3050	DATA	44,D2,A2,49,4E,C9,B2,49	A200
3060	DATA	4E,49,D2,A8,4C,44,C4,88	B534
3070	DATA	4C,44,44,D2,A0,4C,44,C9	C594
3080	DATA	B0,4C,44,49,D2,44,4E,45	DBB0
3090	DATA	C7,BB,4F,54,44,D2,B3,4F	F0B1

3100 DATA 54,49,D2,AB,4F,55,54,C4
 3110 DATA A3,4F,55,54,C9,4D,52,45
 3120 DATA 54,C9,45,52,45,54,CE,6F
 3130 DATA 52,4C,C4,67,52,52,C4,00
 3140 DATA CF,4A,07,41,44,C3,FF,46
 3150 DATA 0D,FF,56,0E,FF,5E,0F,49
 3160 DATA CD,C7,40,02,49,CE,CF,4B
 3170 DATA 28,CF,43,29,F7,57,13,F7
 3180 DATA 47,14,4C,C4,C7,41,03,4F
 3190 DATA 55,D4,CF,42,07,53,42,C3
 3200 DATA 00,2A,77,7E,21,96,22,9E
 3210 DATA 09,86,19,8E,23,A6,29,AE
 3220 DATA 34,86,35,BE,36,2B,39,E1
 3230 DATA 46,E3,4E,E5,56,E9,5E,F9
 3240 DATA 66,6E,70,71,72,73,74,75
 3250 DATA 3F,8F,CD,F3,43,28,07,FE
 3260 DATA 3F,28,14,C3,47,43,21,0B
 3270 DATA 67,CD,02,67,CD,9F,00,F5
 3280 DATA CD,84,42,F1,FE,03,C8,21
 3290 DATA E3,67,7E,B7,C8,CD,B6,42
 3300 DATA 23,18,F7,45,44,49,54,4F
 3310 DATA 52,20,43,4F,4D,4D,41,4E
 3320 DATA 44,0D,0A,4E,45,57,0D,4C
 3330 DATA 49,53,54,20,5B,6E,31,5B
 3340 DATA 2D,6E,32,5D,5D,0D,4C,4C
 3350 DATA 49,53,54,20,5B,6E,31,5B
 3360 DATA 2D,6E,32,5D,5D,0D,41,55
 3370 DATA 54,4F,20,5B,6E,31,5B,2C
 3380 DATA 6E,32,5D,5D,0D,52,45,4E
 3390 DATA 55,4D,20,5B,6E,31,5B,2C
 3400 DATA 6E,32,5B,2C,6E,33,5D,5D
 3410 DATA 5D,0D,44,45,4C,45,54,45
 3420 DATA 20,6E,31,5B,2D,6E,32,5D
 3430 DATA 0D,46,49,4E,44,73,0D,53
 3440 DATA 45,41,52,43,48,73,0D,4C
 3450 DATA 53,45,41,52,43,48,73,0D
 3460 DATA 43,48,41,4E,47,45,64,73
 3470 DATA 31,64,5B,73,32,5D,0D,53
 3480 DATA 41,56,45,20,22,66,69,6C
 3490 DATA 65,20,6E,61,6D,65,22,0D
 3500 DATA 4C,4F,41,44,20,22,66,69
 3510 DATA 6C,65,20,6E,61,6D,65,22
 3520 DATA 0D,4D,45,52,47,45,20,22
 3530 DATA 66,69,6C,65,20,6E,61,6D
 3540 DATA 65,22,0D,4D,41,50,0D,41
 3550 DATA 5B,4E,55,50,4F,49,52,53
 3560 DATA 44,48,2F,78,78,5D,0D,42
 3570 DATA 41,0D,00,0D,4D,4F,4E,49
 3580 DATA 54,4F,52,20,43,4F,4D,4D

F785
 46A
 702
 1290
 1AA6
 2AA5
 3038
 5006
 66A7
 783D
 8225
 874A
 8E84
 982D
 A384
 B59C
 C4F2
 D9AA
 E09F
 F857
 84
 572
 F07
 186F
 2877
 3A51
 408E
 6486
 7989
 7F9E
 846F
 800E
 967D
 A174
 B475
 C8EE
 DA09
 E008
 F045
 297
 72C
 1280
 157C
 2000
 3076
 507E
 66A7
 7AFF
 8184

3590 DATA 41,4E,44,0D,0A,43,78,20
 3600 DATA 20,20,20,20,20,20,20,20
 3610 DATA 20,20,20,43,68,61,6E,67
 3620 DATA 65,20,64,75,6D,70,0D,44
 3630 DATA 78,78,58,2C,79,79,5D,20
 3640 DATA 20,20,20,20,20,44,75,6D
 3650 DATA 70,20,6D,65,6D,6F,72,79
 3660 DATA 0D,50,78,78,58,2C,79,79
 3670 DATA 5D,20,20,20,20,20,20,50
 3680 DATA 72,69,6E,74,20,6D,65,6D
 3690 DATA 6F,72,79,0D,56,78,78,58
 3700 DATA 2C,79,79,5D,20,20,20,20
 3710 DATA 20,20,56,2D,52,41,4D,20
 3720 DATA 70,72,69,6E,74,0D,4D,58
 3730 DATA 78,78,5D,20,20,20,20,20
 3740 DATA 20,20,20,20,4D,65,6D,6F
 3750 DATA 72,79,20,73,65,74,0D,53
 3760 DATA 58,78,78,5D,20,20,20,20
 3770 DATA 20,20,20,20,20,53,65,74
 3780 DATA 20,6D,65,6D,6F,72,79,0D
 3790 DATA 46,78,78,2C,79,79,2C,7A
 3800 DATA 20,20,20,20,20,20,46,69
 3810 DATA 6C,6C,20,6D,65,6D,6F,72
 3820 DATA 79,0D,54,78,78,2C,79,79
 3830 DATA 2C,7A,7A,20,20,20,20,20
 3840 DATA 54,72,61,6E,73,66,65,72
 3850 DATA 0D,52,58,78,78,5D,20,20
 3860 DATA 20,20,20,20,20,20,20,52
 3870 DATA 65,61,64,20,74,61,70,65
 3880 DATA 0D,58,58,72,5D,20,20,20
 3890 DATA 20,20,20,20,20,20,20,65
 3900 DATA 78,61,6D,69,6E,65,20,72
 3910 DATA 65,67,2E,0D,47,78,78,58
 3920 DATA 2C,79,79,58,2C,7A,7A,5D
 3930 DATA 5D,20,47,6F,0D,4C,58,50
 3940 DATA 5D,58,78,78,58,2C,79,79
 3950 DATA 5D,5D,20,64,69,73,61,73
 3960 DATA 73,65,6D,62,6C,65,0D,3F
 3970 DATA 58,3F,5D,20,20,20,20,20
 3980 DATA 20,20,20,20,20,64,69,73
 3990 DATA 70,6C,61,79,20,74,68,69
 4000 DATA 73,0D,0D,00,DB,A8,32,32
 4010 DATA FA,3A,C1,FC,32,30,FA,32
 4020 DATA 31,FA,0E,00,CD,95,69,38
 4030 DATA 03,32,30,FA,0E,40,CD,95
 4040 DATA 69,38,03,32,31,FA,21,C9
 4050 DATA FC,06,40,7E,87,38,2B,23
 4060 DATA 10,F9,2A,48,FC,11,00,80
 4070 DATA B7,ED,52,20,1D,21,30,FA

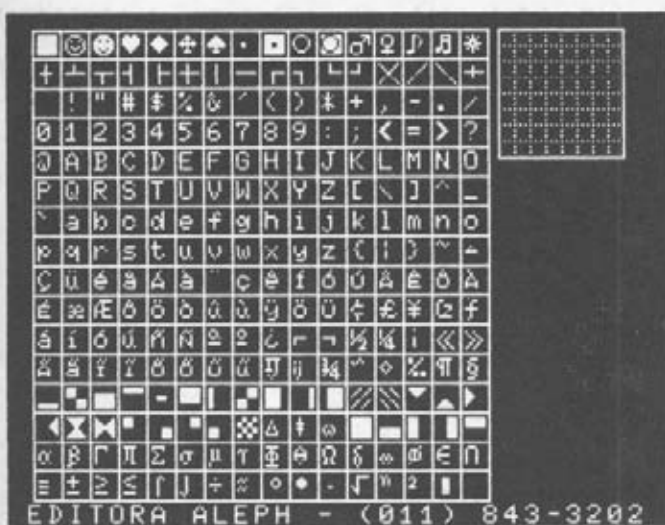
8280
 8281
 8282
 8283
 8284
 8285
 8286
 8287
 8288
 8289
 8290
 8291
 8292
 8293
 8294
 8295
 8296
 8297
 8298
 8299
 8300
 8301
 8302
 8303
 8304
 8305
 8306
 8307
 8308
 8309
 8310
 8311
 8312
 8313
 8314
 8315
 8316
 8317
 8318
 8319
 8320
 8321
 8322
 8323
 8324
 8325
 8326
 8327
 8328
 8329
 8330
 8331
 8332
 8333
 8334
 8335
 8336
 8337
 8338
 8339
 8340
 8341
 8342
 8343
 8344
 8345
 8346
 8347
 8348
 8349
 8350
 8351
 8352
 8353
 8354
 8355
 8356
 8357
 8358
 8359
 8360
 8361
 8362
 8363
 8364
 8365
 8366
 8367
 8368
 8369
 8370
 8371
 8372
 8373
 8374
 8375
 8376
 8377
 8378
 8379
 8380
 8381
 8382
 8383
 8384
 8385
 8386
 8387
 8388
 8389
 8390
 8391
 8392
 8393
 8394
 8395
 8396
 8397
 8398
 8399
 8400
 8401
 8402
 8403
 8404
 8405

4080 DATA 3A,C1,FC,BE,28,14,23,BE
 4090 DATA 28,10,21,F0,69,11,00,C1
 4100 DATA 01,C4,00,ED,80,CD,00,C1
 4110 DATA B7,C9,FB,37,C9,21,C1,FC
 4120 DATA 06,04,AF,E6,03,B6,C5,E5
 4130 DATA 61,2E,10,F5,CD,0C,00,2F
 4140 DATA 5F,F1,D5,F5,CD,14,00,F1
 4150 DATA D1,F5,D5,CD,0C,00,C1,47
 4160 DATA 79,2F,5F,F1,F5,C5,CD,14
 4170 DATA 00,C1,79,B8,20,17,F1,2D
 4180 DATA 20,D9,24,24,24,24,4F,7C
 4190 DATA FE,40,28,05,FE,80,79,20
 4200 DATA C8,79,E1,E1,C9,F1,E1,C1
 4210 DATA B7,F2,EA,69,C6,04,FE,90
 4220 DATA 38,B4,23,3C,10,AD,37,C9
 4230 DATA 3A,30,FA,26,00,CD,1C,C1
 4240 DATA 3A,31,FA,26,40,CD,1C,C1
 4250 DATA DB,A8,F5,3A,32,FA,D3,AB
 4260 DATA 47,F1,FB,C9,CD,49,C1,FA
 4270 DATA 29,C1,DB,A8,A1,80,D3,AB
 4280 DATA C9,CD,98,C1,28,13,E5,CD
 4290 DATA 6E,C1,4F,06,00,7D,A4,B2
 4300 DATA 21,C5,FC,09,77,E1,79,18
 4310 DATA DB,CD,A1,C1,21,C5,FC,72
 4320 DATA C9,F3,F5,7C,07,07,E6,03
 4330 DATA 5F,1C,3E,C0,07,07,1D,20
 4340 DATA FB,5F,2F,4F,F1,F5,E6,03
 4350 DATA 47,04,3E,AB,C6,55,10,FC
 4360 DATA 57,A3,47,F1,B7,C9,F5,7A
 4370 DATA E6,C0,4F,F1,F5,57,DB,AB
 4380 DATA 47,E6,3F,B1,F5,7A,0F,0F
 4390 DATA E6,03,57,14,3E,AB,C6,55
 4400 DATA 15,20,FB,A3,57,7B,2F,67
 4410 DATA F1,CD,B4,C1,F1,E6,03,C9
 4420 DATA 14,15,C0,47,7B,FE,03,78
 4430 DATA C9,0F,0F,E6,03,57,3A,FF
 4440 DATA FF,2F,47,E6,FC,B2,57,32
 4450 DATA FF,FF,7B,C9,D3,AB,3A,FF
 4460 DATA FF,2F,6F,A4,B2,32,FF,FF
 4470 DATA 78,D3,AB,C9,0D,2A,2A,20
 4480 DATA 43,4F,4D,50,49,4C,41,44
 4490 DATA 4F,52,20,41,53,4D,43,4F
 4500 DATA 43,41,52,8A,8B,86,AB,B0
 4510 DATA AD,BE,DF,BE,83,BA,AF,B7
 4520 DATA DF,D2,DF,D7,CF,CE,CE,D6
 4530 DATA DF,C7,CB,CC,D2,CC,CD,CF
 4540 DATA CD,FF,20,00,00,00,00,00

4440
 4450
 4460
 4470
 4480
 4490
 4500
 4510
 4520
 4530
 4540
 4550
 4560
 4570
 4580
 4590
 4600
 4610
 4620
 4630
 4640
 4650
 4660
 4670
 4680
 4690
 4700
 4710
 4720
 4730
 4740
 4750
 4760
 4770
 4780
 4790
 4800
 4810
 4820
 4830
 4840
 4850
 4860
 4870
 4880
 4890
 4900
 4910
 4920
 4930
 4940
 4950
 4960
 4970
 4980
 4990
 5000

TOTAL = 8284

Com o programa digitado e gravado corretamente, comande RUN para executá-lo. A tela deverá estar como mostra a figura a seguir:



Você tem agora em seu micro um poderoso editor de caracteres com dois modos de operação, SELEÇÃO e EDIÇÃO. O modo SELEÇÃO permite a escolha do caractere a ser editado. O modo EDIÇÃO permite a alteração do seu "desenho". Logo após ser carregado, o programa opera no modo SELEÇÃO. Experimente usar as teclas de setas e observe o que acontece com o cursor (na tabela de caracteres) e com o quadrado no canto superior direito do vídeo (CARACTERE AMPLIADO).

Para alterar ou redesenhar completamente um dado caractere, deve-se inicialmente levar o cursor até ele com a ajuda das teclas de setas.

Feito isso, deve-se entrar no modo de EDIÇÃO, pressionando a tecla RETURN. Assim procedendo, o cursor desaparecerá da tela e um pequeno ponto será visível no quadrado do CARACTERE AMPLIADO, onde poderá ser feita a edição.

Para apagar os pontos marcados do caractere a ser editado, basta pressionar a BARRA DE ESPAÇOS.

Para marcar um ponto no caractere, basta pressionar a tecla da letra "M" (de Marcar).

Após redesenhar o caractere, para voltar ao modo de SELEÇÃO, basta pressionar RETURN novamente.

Uma vez alterados ou redesenhados os caracteres, deve-se avisar ao programa que essa nova tabela deve ser usada. Para isso basta pressionar a tecla da letra "C" (de Confirmar).

Por fim, para sair do programa e retornar ao BASIC, pressiona-se a tecla da letra "S" (de Sair).

Resumindo, temos os seguintes comandos à nossa disposição:

- SETAS - Movem o cursor;
- C - Confirma o uso da tabela redefinida;
- S - Sai do programa e retorna ao BASIC;
- RETURN - Passa do modo SELEÇÃO para o EDIÇÃO e vice-versa;
- ESPAÇO - Apaga pontos no caractere em edição;
- M - Marca pontos no caractere em edição.

Até agora vimos como usar o programa para gerar e assumir uma nova tabela de caracteres. Vamos aprender como usar essa nova tabela.

Após ter retornado ao BASIC é conveniente, antes de mais nada, salvar a nova tabela em fita ou em disco. Para isso, comande:

```
BSAVE "TABELA.DAT",&HBB80,&HC380
```

A seguir, apague o programa em BASIC que está na memória do micro comandando NEW e SCREEN 1.

Caso você queira carregar uma tabela já salva em fita ou em disco e assumí-la, basta comandar:

```
BLOAD "TABELA.DAT"  
POKE &HF920,&H80  
POKE &HF921,&HBB
```

Só para o EXPERT:

```
POKE &HF91F,2
```

Só para o HOTBIT:

```
POKE &HF91F,3
```

A seguir, deve-se usar o comando SCREEN.

Para voltar a operar com a tabela original da ROM, rode o programinha mostrado a seguir.

```

10 POKE &HF91F,0
20 POKE &HF920,&HBF
30 POKE &HF921,&H1B
40 SCREEN 1

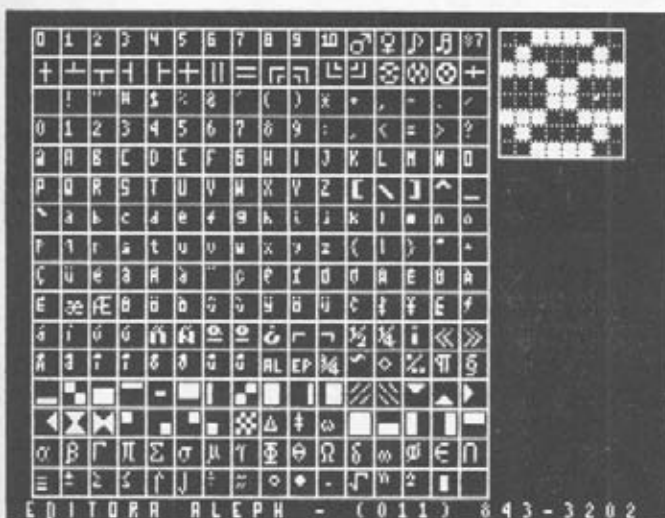
```

```

228
547
106
30F

```

Agora experimente redefinir a tabela de caracteres como mostrado abaixo:



BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.1 - páginas 83 a 86.
 Coleção de Programas para MSX v.2 - páginas 10 a 24,
 73 e 83 a 85.
 Aprofundando-se no MSX - capítulo 4.
 Programação Avançada em MSX - página 145.

2.C - SCROLL UP PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para cima. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

100	REM -----	65E
110	REM SCROLL SCREEN 0 UP - Rubens Jr.	1006
120	REM -----	1EC6
130	FOR F=&HE000 TO &HE046	2613
140	READ A\$: POKE F,VAL("&H"+A\$)	3554
150	NEXT F : DEFUSR0=&HE000	3050
160	DATA 21,00,00,22,44,E0,06,17	4AFE
170	DATA F3,21,28,00,ED,5B,44,E0	6060
180	DATA 19,C5,CD,2F,E0,C1,2A,44	74E1
190	DATA E0,11,28,00,19,22,44,E0	8161
200	DATA 10,E7,21,98,03,01,28,00	8629
210	DATA 3E,20,CD,56,00,FB,C9,E5	8008
220	DATA D5,01,28,00,C5,11,18,FC	9784
230	DATA D5,CD,59,00,E1,C1,D1,CD	9FE0
240	DATA 5C,00,E1,C9,00,00,00,00	8161
250	REM -----	C541
260	REM Exemplo de uso	0752
270	REM -----	F4A8
280	SCREEN 0 : WIDTH 40 : KEY OFF	FF9E
290	X = 40 * RND(1)	40C
300	LOCATE X,20 : PRINT ".";	7E8
310	X = USR(0)	802
320	GOTO 290	007
TOTAL =		007

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.D - SCROLL DOWN PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para baixo. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

100 REM -----	63E
110 REM SCROLL SCREEN 0 DOWN-Rubens Jr.	1189
120 REM -----	1789
130 FOR F=&HE000 TO &HE049	2828
140 READ A\$: POKE F,VAL("&H"+A\$)	8824
150 NEXT F : DEFUSR0=&HE000	9100
160 DATA 21,70,03,22,47,E0,06,17	41801
170 DATA F3,21,28,00,ED,58,47,E0	6148
180 DATA 19,EB,C5,CD,32,E0,C1,2A	75A8
190 DATA 47,E0,11,28,00,87,ED,52	8284
200 DATA 22,47,E0,10,E4,21,00,00	8750
210 DATA 01,2B,00,3E,20,CD,56,00	8048
220 DATA FB,C9,E5,D5,01,28,00,C5	96F4
230 DATA 11,18,FC,D5,CD,59,00,E1	9F90
240 DATA C1,D1,CD,5C,00,E1,C9,00	8158
250 DATA 00,00,00,00,00,00,00,00	0118
260 REM -----	D8C1
270 REM Exemplo de uso	EE0A
280 REM -----	101
290 SCREEN 0 : WIDTH 40 : KEY OFF	661
300 X = 40 * RND(1)	881
310 LOCATE X,0 : PRINT ".";	EA1
320 X = USR(0)	1284
330 GOTO 300	153F

TOTAL = 153F

BIBLIOGRAFIA RECOMENDADA:

- Aprofundando-se no MSX - capítulo 4.
- Programação Avançada em MSX - capítulo 2.
- Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.E - SCROLL LEFT PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para a esquerda. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

100 REM -----	65E
110 REM SCROLL SCREEN 0 LEFT-Rubens Jr.	1190
120 REM -----	1F90
130 FOR F=&HE000 TO &HE03F	26F4
140 READ A\$: POKE F,VAL("&H"+A\$)	3635
150 NEXT F : DEFUSR0=&HE000	3D3E
160 DATA 21,00,00,22,3D,E0,06,18	4043
170 DATA F3,C5,CD,1C,E0,C1,2A,3D	6014
180 DATA E0,11,28,00,19,22,3D,E0	7560
190 DATA 10,EF,FB,C9,E5,01,28,00	81C7
200 DATA C5,11,18,FC,D5,CD,59,00	87D0
210 DATA 21,19,FC,11,18,FC,01,27	8E3A
220 DATA 00,ED,B0,3E,20,12,E1,C1	981D
230 DATA D1,CD,5C,00,C9,00,00,E1	A08F
240 REM -----	B764
250 REM Exemplo de uso	C7F8
260 REM -----	E54D
270 SCREEN 0 : WIDTH 40 : KEY OFF	F191
280 X = 23 * RND(1)	FC7A
290 LOCATE 39,X : PRINT ".:"	20D
300 X = USR0(0)	549
310 GOTO 280	693
TOTAL = 693	

BIBLIOGRAFIA RECOMENDADA:

- Aprofundando-se no MSX - capítulo 4.
- Programação Avançada em MSX - capítulo 2.
- Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.F - SCROLL RIGHT PARA A SCREEN 0

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 0 para a direita. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instruçãoUSR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

100 REM -----	65E
110 REM SCROLL SCREEN 0 RIGHT-Rubens Jr	1139
120 REM -----	1F39
130 FOR F=&HE000 TO &HE03F	269D
140 READ A\$: POKE F,VAL("&H"+A\$)	350E
150 NEXT F : DEFUSR0=&HE000	3CE7
160 DATA 21,00,00,22,3D,E0,06,18	48EC
170 DATA F3,C5,CD,1C,E0,C1,2A,3D	5F8D
180 DATA E0,11,28,00,19,22,3D,E0	7509
190 DATA 10,EF,FB,C9,E5,01,28,00	8170
200 DATA C5,11,18,FC,D5,CD,59,00	8779
210 DATA 21,3E,FC,11,3F,FC,01,27	8EC0
220 DATA 00,ED,B8,3E,20,12,E1,C1	98A8
230 DATA D1,CD,5C,00,C9,00,00,E1	A11A
240 REM -----	87EF
250 REM Exemplo de uso	C883
260 REM -----	E5D8
270 SCREEN 0 : WIDTH 40 : KEY OFF	F21C
280 X = 23 * RND(1)	FD05
290 LOCATE 0,X : PRINT ". ";	269
300 X = USR0(0)	59A
310 GOTO 280	603

TOTAL = 603

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 130 e 131.

2.6 - SCROLL UP PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para cima. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

100 REM -----	65E
110 REM SCROLL SCREEN 1 UP - Rubens Jr.	10C7
120 REM -----	1EC7
130 FOR F=&HE000 TO &HE046	2614
140 READ A\$: POKE F,VAL("&H"+A\$)	3553
150 NEXT F : DEFUSR0=&HE000	303E
160 DATA 21,00,18,22,44,E0,06,17	4AF8
170 DATA F3,21,20,00,ED,5B,44,E0	606F
180 DATA 19,C5,CD,2F,E0,C1,2A,44	74E3
190 DATA E0,11,20,00,19,22,44,E0	816B
200 DATA 10,E7,21,E0,1A,01,20,00	86A2
210 DATA 3E,20,CD,56,00,FB,C9,E5	8D54
220 DATA D5,01,20,00,C5,11,18,FC	9805
230 DATA D5,CD,59,00,E1,C1,D1,CD	A061
240 DATA 5C,00,E1,C9,00,00,C9,00	B1EC
250 REM -----	C5FC
260 REM Exemplo de uso	D80D
270 REM -----	F563
280 SCREEN 1 : WIDTH 32 : KEY OFF	54
290 X = 32 * RND(1)	58A
300 LOCATE X,20 : PRINT ".";	8A9
310 X = USR0(0)	800
320 GOTO 290	DC5

TOTAL = DC5

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.H - SCROLL DOWN PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para baixo. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

100	REM -----	65E
110	REM SCROLL SCREEN 1 DOWN-Rubens Jr.	118A
120	REM -----	1F8A
130	FOR F=&HE000 TO &HE049	28E4
140	READ A\$: POKE F,VAL("&H"+A\$)	3625
150	NEXT F : DEFUSR0=&HE000	302E
160	DATA 21,C0,1A,22,47,E0,06,17	4060
170	DATA F3,21,20,00,ED,58,47,E0	61E7
180	DATA 19,EB,C5,CD,32,E0,C1,2A	7647
190	DATA 47,E0,11,20,00,87,ED,52	82F0
200	DATA 22,47,E0,10,E4,21,00,18	87E3
210	DATA 01,20,00,3E,20,CD,56,00	8D06
220	DATA FB,C9,E5,D5,01,20,00,C5	9787
230	DATA 11,18,FC,D5,CD,59,00,E1	A023
240	DATA C1,D1,CD,5C,00,E1,C9,00	B1EE
250	DATA 00,0E,00,00,00,00,00,00	C109
260	REM -----	DC7F
270	REM Exemplo de uso	EF98
280	REM -----	18F
290	SCREEN 1 : WIDTH 32 : KEY OFF	728
300	X = 40 * RND(1)	B78
310	LOCATE X,0 : PRINT ".";	F68
320	X = USR0(0)	134B
330	GOTO 300	1606

TOTAL = 1606

BIBLIOGRAFIA RECOMENDADA:

- Aprofundando-se no MSX - capítulo 4.
- Programação Avançada em MSX - capítulo 2.
- Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.1 - SCROLL LEFT PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para a esquerda. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

100 REM -----	65E
110 REM SCROLL SCREEN 1 LEFT-Rubens Jr.	1191
120 REM -----	1F91
130 FOR F=&HE000 TO &HE03F	28F5
140 READ A% : POKE F,VAL("&H"+A%)	3636
150 NEXT F : DEFUSR0=&HE000	3D3F
160 DATA 21,00,18,22,3D,E0,06,18	4C3D
170 DATA F3,C5,CD,1C,E0,C1,2A,3D	600E
180 DATA E0,11,20,00,19,22,3D,E0	7552
190 DATA 10,EF,FB,C9,E5,01,20,00	81C1
200 DATA C5,11,18,FC,D5,CD,59,00	87CA
210 DATA 21,19,FC,11,18,FC,01,1F	8DC8
220 DATA 00,ED,B0,3E,20,12,E1,C1	97A8
230 DATA D1,CD,5C,00,C9,00,00,E1	A01D
240 REM -----	B6F2
250 REM Exemplo de uso	C786
260 REM -----	E4D8
270 SCREEN 1 : WIDTH 32 : KEY OFF	F114
280 X = 23 * RND(1)	F8FD
290 LOCATE 31,X : PRINT ".";	188
300 X = USR0(0)	4C4
310 GOTO 280	60E
TOTAL = 60E	

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

Programação Avançada em MSX - capítulo 2.

Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

2.J - SCROLL RIGHT PARA A SCREEN 1

O programa apresentado a seguir gera uma rotina em L.M. para girar a SCREEN 1 para a direita. A rotina em L.M. é carregada a partir de &HE000 e pode ser chamada pela instrução USR.

As linhas finais do programa em BASIC ilustram uma utilização da rotina em L.M. em conjunto com o BASIC.

Experimente usar esta dica em conjunto com as apresentadas no item 2.8 (Movimentos na tela). Você conseguirá na SCREEN 1 movimentos globais de SCROLL suaves e bastante rápidos!

100 REM -----	63E
110 REM SCROLL SCREEN 1 RIGHT-Rubens Jr	113A
120 REM -----	1F3A
130 FOR F=&HE000 TO &HE03F	269E
140 READ A\$: POKE F,VAL("&H"+A\$)	350F
150 NEXT F : DEFUSR0=&HE000	3CE8
160 DATA 21,00,18,22,3D,E0,06,18	48E6
170 DATA F3,C5,CD,1C,E0,C1,2A,3D	5FB7
180 DATA E0,11,20,00,19,22,3D,E0	74FB
190 DATA 10,EF,FB,C9,E5,01,20,00	816A
200 DATA C5,11,18,FC,D5,CD,59,00	8773
210 DATA 21,36,FC,11,37,FC,01,1F	807F
220 DATA 00,ED,B8,3E,20,12,E1,C1	975A
230 DATA D1,CD,5C,00,C9,00,00,E1	9F0C
240 REM -----	8541
250 REM Exemplo de uso	0785
260 REM -----	138A
270 SCREEN 1 : WIDTH 32 : KEY OFF	F0C3
280 X = 23 * RND(1)	F8AC
290 LOCATE 0,X : PRINT ".";	110
300 X = USR0(0)	421
310 GOTO 280	57A
TOTAL = 57A	

BIBLIOGRAFIA RECOMENDADA:

- Aprofundando-se no MSX - capítulo 4.
- Programação Avançada em MSX - capítulo 2.
- Coleção de Programas para MSX v.2 - páginas 12, 130 e 131.

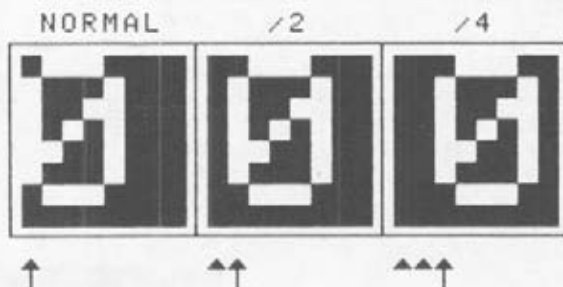
2.K - CENTRALIZANDO CARACTERES

Os caracteres são desenhados dentro de uma matriz de 8x8 posições. Muitos deles usam apenas o lado esquerdo dessa matriz e há casos em que ao serem impressos na SCREEN 1 ou SCREEN 2 ficam fora de alinhamento. Podemos evitar isso de uma forma bem simples e rápida, redefinindo os caracteres. A título de exemplo, vamos 'centralizar' os caracteres dos números e das letras maiúsculas na SCREEN 1.

```
100 SCREEN 1
110 FOR F=6144 TO 6144+24*32-1 STEP 2
120   VPOKE F,ASC("█")
130 NEXT F : LOCATE 10,10
140 PRINT "0123456789" : LOCATE 2,12
150 PRINT "ABCDEFGHIJKLMN O PQRSTU VWXYZ"
160 FOR F=ASC("0") TO 8*ASC("9")+7
170   VPOKE F,VPEEK(F)/2
180 NEXT F
190 FOR F=ASC("A") TO 8*ASC("Z")+7
200   VPOKE F,VPEEK(F)/2
210 NEXT F
```

```
168
9F2
E70
1296
10E8
2553
3510
3043
3FE8
4046
5672
591E
```

TOTAL = 591E



BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - páginas 72 e 73.
Aprofundando-se no MSX - página 90.

2.1 - ANIMAÇÃO COM SPRITES

Se você definir vários SPRITE's com uma figura em posições sucessivas e os colocar na tela sempre na mesma camada, em sequência, elas darão o efeito de animação.

Se você quiser alterar o desenho do "boneco" no programa listado a seguir, uma boa idéia é eliminar a linha 90, digitando seu conteúdo manualmente no modo direto. A seguir você interrompe a listagem na sequência de linhas DATA que definem o SPRITE a ser alterado, modifica o desenho e comanda RUN. Desta forma você pode congelar o boneco com STOP e alterar os 0's e 1's da linha DATA, visualizando o efeito anterior à alteração.

Obviamente você pode definir SPRITE's de 16x16 pixels, obtendo figuras muito mais detalhadas.

Esse programa é apenas um exemplo (parece um bêbado tentando dançar BREAK!) mas pode ser alterado e melhorado conforme sua imaginação.

Pegue papel quadriculado e lápis e monte sua história!

```
90 SCREEN 1
100 GOSUB 220:SPRITE$(0)=S$
110 GOSUB 220:SPRITE$(1)=S$
120 GOSUB 220:SPRITE$(2)=S$
130 GOSUB 220:SPRITE$(3)=S$
140 GOSUB 220:SPRITE$(4)=S$
150 FOR I=1 TO 250 STEP 10
160 FOR S=0 TO 4
170 PUT SPRITE 0,(I+S*2,30),8,S
180 FORT=0T050:NEXTT
190 NEXT S
200 NEXT I
210 GOTO 150
220 S$=""
230 FOR C=1 TO 8
240 READ K$
250 S$=S$+CHR$(VAL("&B"+K$))
260 NEXT C
270 RETURN
280 DATA 00011000
290 DATA 00011000
```

```
150
66E
881
1095
1120
1168
2887
28E2
3189
3180
3788
4219
4588
4929
4F10
5287
6117
6488
6881
7048
7680
```

300	DATA	00111100	831E
310	DATA	00111100	831F
320	DATA	00111100	81E6
330	DATA	00011000	8A25
340	DATA	00011000	803E
350	DATA	00011100	80E0
360	REM	-----	92F4
370	DATA	00011000	9B19
380	DATA	00011000	9D28
390	DATA	00111100	A261
400	DATA	01011010	A120
410	DATA	01011010	AE49
420	DATA	00010100	B562
430	DATA	00100110	B042
440	DATA	00110000	C245
450	REM	-----	C920
460	DATA	00011000	D050
470	DATA	00011000	DA24
480	DATA	00111100	E331
490	DATA	01011010	E080
500	DATA	10011001	F595
510	DATA	00100100	FE24
520	DATA	01000010	268
530	DATA	01100011	48A
540	REM	-----	60F
550	DATA	00011000	875
560	DATA	00011000	901
570	DATA	00111100	E08
580	DATA	01011010	1323
590	DATA	10011010	182F
600	DATA	01100100	1074
610	DATA	01000100	2058
620	DATA	00000110	2508
630	REM	-----	20CE
640	DATA	00011000	3476
650	DATA	00011000	3AA0
660	DATA	00111100	408C
670	DATA	00111100	4602
680	DATA	01011010	5002
690	DATA	00101000	5A76
700	DATA	00101100	6408
710	DATA	00111110	601A
720	REM	-----	7715

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
 Programação Avançada em MSX - capítulo 2.

2.M - LETRAS AMPLIADAS

O programa apresentado a seguir gera uma tela padrão na SCREEN 2 onde podem ser inseridas 4 mensagens. Digite-o e execute-o. Depois, experimente alterar o conteúdo das linhas 135, 140, 145 e 150. Experimente também alterar o caractere entre as aspas na linha 130. A tela gerada será gravada em disco ou em fita pela linha 705. Se desejar, altere o nome do arquivo.

100 COLOR 15,1,1	110E
105 SCREEN 2	35E
110 OPEN"GRP:" AS #1	769
115 REM	8A3
120 REM define mensagens	E24
125 REM -----	1A43
130 X\$(0)="0"	1E8E
135 X\$(1)="XXXXX"	2311
140 X\$(2)="XXXXXXXXXXXXX"	2E01
145 X\$(3)="XXXXXXXXXXXXX"	3070
150 X\$(4)="XXXXXXXXXXXXXXXXXXXXXXXXXXXXX"	5078
155 REM	5DE8
160 REM desenha mensagem	688A
165 REM -----	8301
170 X=6.5+(245-8*LEN(X\$(4)))/2	9204
175 PSET(X,170),POINT(X,170)	97E3
180 PRINT #1, X\$(4)	9AC9
185 PSET(X+1,170),POINT(X+1,170)	A42C
190 PRINT #1, X\$(4)	ADDE
195 REM	ABD6
200 REM desenha linhas horizontais	AFD5
205 REM -----	0626
210 COLOR 9	08AD
215 YY = 1	0F08
220 FOR F=6 TO 191 STEP .2	0919
225 F = F + YY	E1D5
230 LINE(8,F)-(247,F),9	EF8F
235 YY = YY + .4	FCA6
240 NEXT F	FF9D
245 REM	101
250 REM desenha a moldura	A68
255 REM -----	FAF
260 FOR F=0 TO 255 STEP 8	1594
265 PSET(F,0),4	198D
270 PRINT #1,X\$(0)	1F38
275 PSET(F,191-7),4	25D0
280 PRINT #1,X\$(0)	2838
285 NEXT F	2DAC

290	FOR F=0 TO 191 STEP 8	8041
295	PSET(0,F),4	8042
300	PRINT #1,X\$(0)	8043
305	PSET(255-7,F),4	8044
310	PRINT #1,X\$(0)	8045
315	NEXT F	8046
320	REM	8047
325	REM desenha texto	8048
330	REM -----	8049
335	COLOR 14	8050
340	EN=PEEK(&HF920)+256*PEEK(&HF921)	8051
345	FOR TX=1 TO 3	8052
350	A\$=X\$(TX)	8053
355	Q=0	8054
360	FOR F=1 TO LEN(A\$)	8055
365	A=EN+8*ASC(MID\$(A\$,F,1))	8056
370	FOR G=0 TO 7	8057
375	B\$=BIN\$(PEEK(A+G))	8058
380	B\$=RIGHT\$("00000000"+B\$,8)	8059
385	FOR H=1 TO 8	8060
390	IF MID\$(B\$,H,1)="1" THEN Q=Q+1	8061
395	NEXT H	8062
400	NEXT G	8063
405	NEXT F	8064
410	P=2*Q-1	8065
415	DIM X(P),Y(P),S(P),T(P)	8066
420	E=0	8067
425	I=5	8068
430	FOR F=1 TO LEN(A\$)	8069
435	A=EN+8*ASC(MID\$(A\$,F,1))	8070
440	FOR G=0 TO 7	8071
445	B\$=BIN\$(PEEK(A+G))	8072
450	B\$=RIGHT\$("00000000"+B\$,8)	8073
455	FOR H=1 TO 8	8074
460	IF MID\$(B\$,H,1)="0" THEN 490	8075
465	X(E)=I+H-1	8076
470	Y(E)=8-G	8077
475	X(E+Q)=X(E)	8078
480	Y(E+Q)=Y(E)	8079
485	E=E+1	8080
490	NEXT H	8081
495	NEXT G	8082
500	REM	8083
505	REM passo horizontal	8084
510	REM -----	8085
515	PS=6	8086
520	I=I + PS	8087
525	NEXT F	8088
530	REM	8089

535	REM parâmetro de escala X	0339
540	REM -----	0339
545	PX=3	0339
550	IF TX=1 THEN PX=5	0339
555	REM	0339
560	REM parâmetro de escala Y	0339
565	REM -----	0339
570	PY=5	0339
575	IF TX=1 THEN PY=7	0339
580	REM	0339
585	REM acha posições na tela	0339
590	REM -----	0339
595	XI=INT((230-LEN(A\$)*PS*PX)/2)	0339
600	IF TX=1 THEN XI=XI-8	0339
605	YI=30+45*TX	0339
610	IF TX=1 THEN YI=YI-1	0339
615	IF TX=2 THEN YI=YI-1	0339
620	IF TX=3 THEN YI=YI+5	0339
625	DEFFN A(F)=XI+PX*X(F)	0339
630	DEFFN B(F)=YI-PY*Y(F)	0339
635	REM	0339
640	REM "plota" círculos das letras	0339
645	REM -----	0339
650	FOR F=0 TO P-1	0339
655	CX=FN A(F) :REM x do centro	0339
660	CY=FN B(F) :REM y do centro	0339
665	AX=1.7 :REM achatamento	0339
670	RD=4 :REM raio	0339
675	IF TX=1 THEN AX=1.4 : RD=6	0339
680	CIRCLE(CX,CY),RD,, , , AX	0339
685	PAINT(CX,CY)	0339
690	NEXT F	0339
695	ERASE X,Y,S,T	0339
700	NEXT TX	0339
705	BSAVE "TELAXXX.SCR",0,&H3FFF,S	0339
710	GOTO 710	0339

TOTAL = 243A

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
 Programação Avançada em MSX - capítulo 2.

2.N - ENTENDENDO O COMANDO DRAW

O comando DRAW do BASIC MSX é, na realidade, uma macro-linguagem gráfica que permite a confecção de desenhos nas telas 2 e 3.

Sua sintaxe é simples:

DRAW <expressão string>

A montagem do desenho é feita através da expressão string segundo regras bem determinadas e de fácil memorização. Para conhecê-los basta consultar o verbete DRAW do dicionário de comandos do livro LINGUAGEM BASIC MSX.

A melhor maneira de se familiarizar com esse comando é gerar uma variável string contendo os comandos desejados e ver o efeito na tela.

Para facilitar a visualização do desenho obtido é conveniente "reticular" a tela (de 10 em 10 pontos, por exemplo) antes da execução do desenho. Experimente digitar o programa a seguir:

```
10 SCREEN 2
20 FOR L=0 TO 191 STEP 10
30   FOR C=0 TO 255 STEP 10
40     PSET (C,L)
50   NEXT C
60 NEXT L
70 A$="BM60,60U10F10D10L5U5L5D5L10"
80 DRAW A$
90 GOTO 90
```

10)
68B
CR2
F42
1104
1277
1021
1153
211E

TOTAL = 211E

Agora vá alterando a variável A\$ definida na linha 80 com novos sub-comandos do DRAW e veja os efeitos gerados por suas experiências até se familiarizar com esse poderoso comando.

BIBLIOGRAFIA RECOMENDADA:

- Linguagem Basic MSX - páginas 56 a 58.
- Coleção de Programas para MSX v.1 - página 54.
- Coleção de Programas para MSX v.2 - página 52.
- Curso de BASIC v.1 - páginas 61 e 62.

2.0 - "WARP 8" NA SCREEN 2

Você está indo para o planeta da Princesa Vespa para salvá-la do terrível "Capacete Preto"! Para chegar lá você passa por um aglomerado de estrelas, com velocidade hiperfotônica. Como simular este efeito? Digite o programa a seguir e boa viagem!

```
100 SCREEN 2:DEFINT I-S,X-Y
110 SPRITE$(1)=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)
120 DIM X(31),Y(31),C(31),D(31),VX(31),VY(31),X1(31),Y1(31)
130 FOR I=0 TO 31
140 X(I)=80+INT(80*RND(-TIME))
150 FOR T=0 TO INT(300*RND(3)):NEXT T
160 Y(I)=40+INT(80*RND(-TIME))
170 IF X(I)=125 THEN GOTO 140
180 X(0)=126:Y(0)=86
190 PUT SPRITE I,(X(I),Y(I)),15,1
200 D(I)=(Y(I)-85)/(X(I)-125)
210 NEXT I
220 FOR T=1 TO 1000
230 FOR I=0 TO 31
240 VX(I)=SGN(X(I)-125)*I*T^3/1000
250 VY(I)=SGN(Y(I)-85)*ABS(VX(I)*D(I))
260 X1(I)=X(I)+VX(I):Y1(I)=Y(I)+VY(I)
270 ON ERROR GOTO 330
280 IF X1(I)<0 OR X1(I)>225 OR Y1(I)<0 OR Y1(I)>191 THEN C=14
290 PUT SPRITE I,(X1(I),Y1(I)),15-C,1
300 C=0
310 NEXT I
320 NEXT T
330 GOTO 330
```

500
1E30
1894
3745
4330
47F8
5004
6994
7556
7E33
876A
89A6
8F30
948E
A06E
AF00
C748
C079
1032
115
284
381
520
6E8

TOTAL = 6E8

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.P - "SPRITEANDO" A TABELA DE CARACTERES

Muitas vezes sentimos a necessidade de fazer letras ou símbolos se movimentarem na tela como se fossem SPRITES. Isso é simples de ser feito; como podemos definir 256 SPRITES (8x8) e o MSX dispõe de 256 caracteres, podemos transformar cada caractere num SPRITE!

O programa a seguir procura o endereço do começo da tabela de caracteres na variável EN (contida nos endereços &HF920 e &HF921) e transfere sua configuração para a região da VRAM reservada para os desenhos dos SPRITES.

```
10 SCREEN 1 : DEFINT A-Z : KEY OFF      6A1
20 EN!=PEEK(&HF920)+256*PEEK(&HF921)    1030
30 FOR F=0 TO 2047                      150A
40 VPOKE 14336+F,PEEK(EN!+F)          1C02
50 NEXT                                 1D98
```

Após rodar este programa, cada SPRITE corresponderá a um dos 256 caracteres do MSX, sendo seu número o próprio código ASC do caractere.

Para ver um dos mil efeitos possíveis a serem empregados com esse recurso, digite a complementação do programa, a seguir:

```
60 INPUT A$                             2026
70 L=LEN(A$) : C=3                       2640
80 FOR I=1 TO L                          2AC4
90 CH%=MID$(A$,I,1)                      3111
100 CH=ASC(CH%)                          3B79
110 FOR Y=0 TO 96                         3FB4
120 PUT SPRITE I,((C+I)*8,Y-1),,CH      4F59
130 NEXT                                  5013
140 LOCATE C+I-2,12,0 : PRINT CH%       5F68
150 NEXT                                  600F
```

Para rodá-lo, se você já rodou o programa anterior, basta digitar GOTO 60. Desta forma não perdemos o tempo de transferência da tabela de caracteres para a área de SPRITES da VRAM. O programa pede a entrada de uma string (seu nome, por exemplo). Evite os caracteres gráficos de código entre 0 e 31.

Implemente agora seu programa de maneira a aceitar qualquer caractere.

2.0 - ARLEQUIM BÊBADO

Quando ativamos a SCREEN 1, os 32 bytes da VRAM compreendidos entre os endereços 8192 e 8223 ficam reservados para atributos de cores. Cada um desses bytes define a cor de frente e a cor de fundo de um conjunto de 8 caracteres (8*32=256).

Digite o programa a seguir para entender melhor este mecanismo.

```
100 SCREEN 1:KEY OFF:DEFINT A-Z
110 FOR L=0 TO 15
120 FOR C=0 TO 15
130 VPOKE 6182+32*L+C,16*L+C
140 NEXT C
150 NEXT L
160 LOCATE 3,20,0:PRINT"BYTE ALTERADO=";
170 GOTO 190
180 A=STICK(0):IF A=0 THEN GOTO 180
190 I=I-(A=3)+(A=7)-2*(A=5)+2*(A=1)
200 COLOR 15,1,1
210 VPOKE 8192+I,&B10111000
220 LOCATE 17,20:PRINT 8192+I
230 IF STICK(0)<>0 THEN GOTO 230
240 GOTO 180
```

619
41E
E34
171B
1928
1375
2370
2730
3208
45F4
4312
5732
6330
6E03
7321

A linha 100 configura o VDP para SCREEN 1, apaga as teclas de função e define todas as variáveis como inteiras para tornar o programa mais rápido.

As linhas de 110 a 150 colocam todos os caracteres na tela através do VPOKE para evitar problemas na impressão dos caracteres de controle (0 a 31 e 127) através da instrução PRINT.

A linha 210 insere, num dos 32 bytes citados, um valor que define a cor de frente e a cor de fundo segundo o seguinte critério: os 4 bits da esquerda definem a cor de frente (no nosso exemplo &B1011 = 11 = amarelo) e os 4 bits da direita a cor de fundo (&B1000 = 8 = vermelho).

Se você quiser "vpokear" um número em decimal, basta calculá-lo segundo a regra:

VALOR DECIMAL=(COR DE FRENTE)*16+(COR DE FUNDO)

No nosso exemplo, o valor decimal seria:

$$11*16 + 8 = 184$$

Experimente substituir esse valor na linha 210 e

depois invente outras combinações de cores.

A linha 190 permite alterar o endereço do VPOKE usando o recurso do "parênteses lógico".

Se a afirmação for verdadeira seu valor será -1, se for falsa, será 0. Dessa forma, pressionando as teclas de setas (STICK(0)), você pode alterar o valor de I e, portanto, o endereço do byte da VRAM a ser alterado.

Assim, por meio desse programa, você pode visualizar quais são os grupos de 8 caracteres que têm sua cor alterada em função do byte de atributo que você mudou.

Se você, por exemplo, alterar os bytes 8197, 8198 e 8199 com os comandos

```
VPOKE8197,184:VPOKE8198,184:VPOKE8199,184
```

e comandar LIST para um programa em BASIC na SCREEN 1, verá uma listagem colorida apenas nos algarismos e símbolos aritméticos.

Note que a alteração no byte que inclui o caractere "espaço vazio" colore toda a tela ao redor da tabela de caracteres (pois ela está cheia de espaços vazios) e que uma alteração no último dos 32 bytes de atributos de cor (8233) altera a cor do cursor (caractere de código 255), permitindo uma visualização mais fácil do mesmo na hora de editar um programa.

Apague as linhas de 160 a 240 com "DELETE 160-240" e acrescente estas linhas ao programa:

```
160 FOR I=1 TO 32  
170 X=INT(RND(-TIME)*14+1)  
171 Y=INT(RND(-TIME)*14+1)  
180 IF X=Y THEN GOTO 170  
190 VPOKE 8191+I,X*16+Y  
200 NEXT I
```

Desta forma você estará sorteando um número de 1 a 15 para a cor de frente e fundo (a cor transparente foi eliminada) e verificando se elas são diferentes.

Rode o programa várias vezes para ver seu efeito e observe como fica uma listagem na SCREEN 1: seu programa fica parecendo um arlequim bêbado!

BIBLIOGRAFIA RECOMENDADA.

Coleção de Programas para MSX v.2 - páginas 18 e 19.

2.R - USANDO 40 OU 64 COLUNAS NA SCREEN 2

Normalmente a impressão de caracteres na SCREEN 2 é semelhante à da SCREEN 1, em 32 colunas. Entretanto, com um pequeno programa em Linguagem de Máquina pode-se fazer com que a impressão assemelhe-se à da SCREEN 0, em 40 colunas. Com ligeiras alterações nessa mesma rotina, podemos fazê-la imprimir em 64 colunas. É exatamente isso que faz o programa listado a seguir. Digite-o e, antes de mais nada, grave-o em disco ou em fita. Depois rode-o.

```

100 SCREEN 0 : WIDTH 40
110 CLEAR 200,&HE000
120 FOR F=&HE000 TO &HE0D1
130 READ A% : POKE F,VAL("&H"+A%)
140 NEXT F
150 PRINT,,,,,,,,,
160 PRINT SPC(10);"[ 1 ] 40 COLUNAS"
170 PRINT SPC(10);"[ 2 ] 64 COLUNAS"
180 A%=INKEY%
190 IF A%="1" THEN 230
200 IF A%(">")"2" THEN 180
210 POKE &HE0AD,4
220 POKE &HE0CA,255
230 SCREEN 0 : NEW
240 DATA FE,03,C0,3A,AF,FC,FE,02
250 DATA C0,EB,46,23,5E,23,56,04
260 DATA 05,C8,1A,CD,19,E0,13,18
270 DATA F7,F5,C5,D5,E5,FD,E5,ED
280 DATA 4B,B7,FC,ED,5B,B9,FC,CD
290 DATA 39,E0,ED,43,B7,FC,ED,53
300 DATA B9,FC,FD,E1,E1,D1,C1,F1
310 DATA C9,CD,AB,00,D0,20,07,FE
320 DATA 0D,28,73,FE,20,D8,6F,26
330 DATA 00,29,29,29,C5,D5,ED,5B
340 DATA 20,F9,19,11,40,FC,06,08
350 DATA C5,D5,3A,1F,F9,CD,0C,00
360 DATA FB,D1,C1,12,13,23,10,F0
370 DATA D1,C1,3A,E9,F3,32,F2,F3
380 DATA FD,21,40,FC,D5,26,08,CB
390 DATA 7A,20,2A,CD,8F,E0,38,2B
400 DATA C5,2E,06,FD,7E,00,CB,78
410 DATA 20,15,CD,C8,E0,38,15,CB
420 DATA 7F,28,0C,F5,D5,E5,CD,11
430 DATA 01,CD,20,01,E1,D1,F1,07
440 DATA 03,2D,20,E2,C1,FD,23,13
450 DATA 25,20,CC,D1,21,06,00,09
460 DATA 44,4D,CD,C8,E0,D0,01,00

```

```

2EB
683
D40
1476
1692
1083
2596
3311
3606
3EEA
4703
4BE2
5103
5790
6BA3
7EA2
8413
89FE
9087
98FE
A530
B800
C932
DHE5
E511
11D
65D
C5A
158A
1E21
3162
427A
555F
6A0C
70A5
8380
8890

```

```

470 DATA 00,21,08,00,19,EB,C9,E5
480 DATA 21,BF,00,B7,ED,52,E1,C9
490 DATA E5,21,EF,00,B7,ED,42,E1
500 DATA C9,00,00,00,00,00,00,00

```

```

91F7
9AEF
A800
88EE

```

Com isso, a rotina de impressão não standard já estará carregada na memória do micro e pronta para ser usada.

Note que a impressão em 40 ou 64 colunas deve ser feita na SCREEN 2 usando-se a sintaxe:

```
A%=USR0("string a ser impressa")
```

O programa a seguir ilustra o uso da impressão em 40 colunas.

```

10 SCREEN 2
20 PRESET (0,10)
30 A%=USR0("Teste de 40 colunas")
40 OPEN "GRP:" AS #1
50 PRESET (0,20)
60 PRINT #1,"Teste de 32 colunas"
70 GOTO 70

```

```

100
24E
801
E99
11F6
1412
10CE

```

A impressão normal, usando o PRINT # , continua a funcionar sem alterações, em 32 colunas.

Observe também que quando se usa a opção de 64 colunas os caracteres normais se sobrepõem. Isso pode ser evitado se você redefinir uma fonte de caracteres (veja a dica 2.8) apropriadamente e rodar o programa a seguir.

```

100 DEFUSR=&HE000
110 POKE &HF91F,2
120 POKE &HF920,&H80
130 POKE &HF921,&HBB
140 SCREEN 2
150 PRESET(0,10)
160 A%=USR("123456789_123456789_123456789_123456789_123456789_123456789_1234")
170 LINE (0,18)-(255,18),15
180 GOTO 180

```

```

387
620
89F
0E2
EE8
11A9
28A2
3724
3211

```

TOTAL = 3417

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.

2.5 - CARIMBANDO SPRITES 8 x 8 NA SCREEN 2

O recurso dos SPRITES é muito útil na apresentação visual dos programas, mas eles apresentam algumas limitações, como por exemplo o fato de 5 sprites não serem impressos pelo VDP se estiverem na mesma linha.

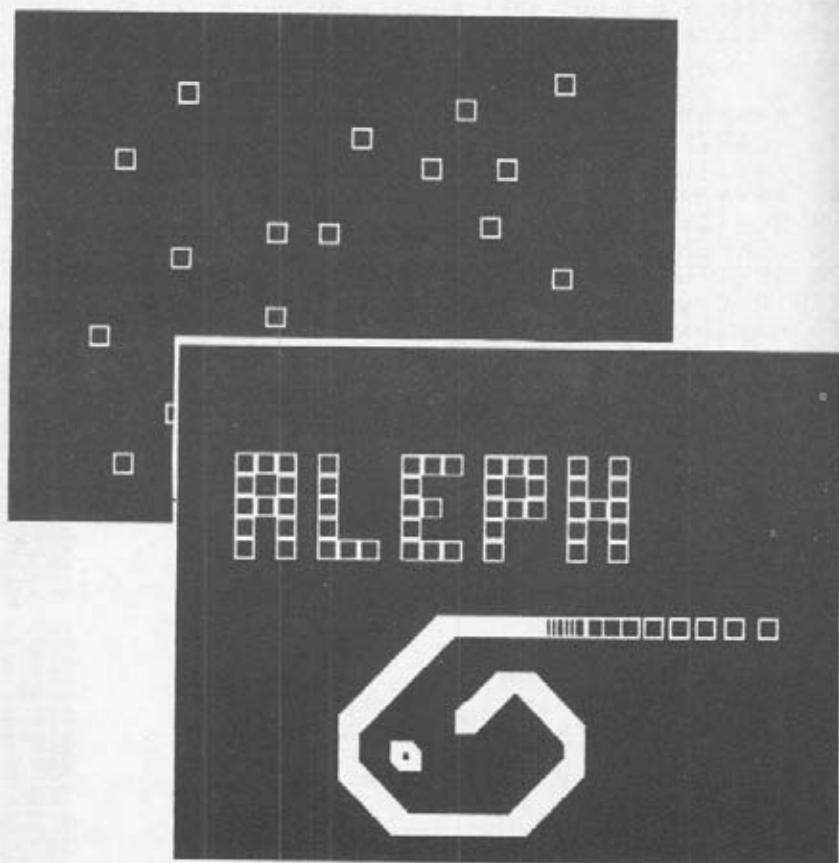
A rotina apresentada a partir da linha 120 na listagem abaixo imprime o SPRITE 8 x 8 cujo padrão é definido pela variável SC% na coordenada dada pelas variáveis X e Y. O canal #1 para arquivos também deve estar livre para ser utilizado pela rotina.

1	******	520
2	*CARIMBADOR 8x8 *	021
3	*BY THE DOCTOR LUZ*	1230
4	******	1031
10	SL=(INP(&HAB)&AND&HC0)/64	12013
20	SCREEN 2,0:SC%=0	12085
30	SPRITE\$(SC%)=CHR\$(255)+STRING\$(6,CHR\$(14488
	(&B10000001))+CHR\$(255)	
40	A%=INKEY%	4107
50	PUTSPRITE 10,(X,Y),15,SC%	5509
60	IF A%=CHR\$(28)THEN X=X+1	6011
70	IF A%=CHR\$(29)THEN X=X-1	6097
80	IF A%=CHR\$(30)THEN Y=Y-1	7504
90	IF A%=CHR\$(31)THEN Y=Y+1	8010
100	IF A%="F"THEN SCREEN 0:END	8159
110	IF A%=" " THEN GOSUB 130	8012
120	GOTO 40	8110
130	FORLZ =0 TO 7	9475
140	POKE &HC200+LZ,VPEEK(BASE(14)+LZ+8*S	1112
	CX)	
150	NEXT	A369
160	POKE &HF91F,SL	A80F
170	POKE &HF920,0	ACCF
180	POKE &HF921,&HC0	B1F1
190	OPEN "GRP:" AS #1	B05F
200	PRESET (X,Y+1):PRINT#1,"a"	0735
210	POKE &HF91F,&H0	0E14
220	POKE &HF920,&HBF	1081
230	POKE &HF921,&H1B	1008
240	CLOSE #1	1010
250	RETURN	1039

TOTAL = 1139

A idéia desse carimbador de SPRITES é muito simples. A variável SC% ajuda a encontrar na VRAM o início da tabela de formação do SPRITE, que é transferido para a RAM a partir do endereço &HC200.

Em seguida, muda-se o conteúdo da variável do sistema CGPNT (em &HF91F) para que ao mandarmos imprimir o caracter "e" na tela gráfica, não seja impressa a matriz de pontos da ROM, mas o padrão do SPRITE que foi transferido a partir do endereço &HC200.



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.T - CARIMBADOR DE SPRITES 16 x 16 NA SCREEN 2

O princípio de funcionamento do programa a seguir é o mesmo que o do carimbador de SPRITES 8 x 8, apresentado na dica anterior. Contudo, agora a transferência da VRAM para RAM é um pouco mais demorada, pois os SPRITES 16 x 16 são definidos por 32 bytes cada um.

```
10 *****
20 *CARIMBADOR 16x16 *
30 *BY THE DOCTOR LUZ*
40 *****
50 SL=(INP(&HAB)AND&HC0)/64
60 SCREEN 2,2:SC%=1
70 SPRITE$(SC%)=CHR$(255)+STRING$(14,CHR
$(8B10000000))+CHR$(255)+CHR$(255)+STRIN
G$(14,CHR$(8B00000001))+CHR$(255)
80 A$=INKEY$
90 PUTSPRITE 10,(X,Y),15,SC%
100 IF A$=CHR$(28)THEN X=X+1
110 IF A$=CHR$(29)THEN X=X-1
120 IF A$=CHR$(30)THEN Y=Y-1
130 IF A$=CHR$(31)THEN Y=Y+1
140 IF A$="F"THEN SCREEN 0:END
150 IF A$=" " THEN GOSUB 170
160 GOTO 80
170 FORLZ=0 TO 31
180 POKE &HC200+LZ,VPEEK(BASE(14)+LZ+32*
SC%)
190 NEXT
200 POKE &HF91F,SL
210 POKE &HF920,0
220 POKE &HF921,&HC0
230 OPEN "GRP:" AS #1
240 PRESET (X,Y+1):PRINT#1,"aB"
250 PRESET (X,Y+9):PRINT#1,"aC"
260 POKE &HF91F,&H0
270 POKE &HF920,&HBF
280 POKE &HF921,&H1B
290 CLOSE #1
300 RETURN
```

101HL = 85F

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
Programação Avançada em MSX - capítulo 2.

2.U - ARMAZENANDO TELAS NA RAM

O programa em BASIC listado a seguir gera na memória do micro, a partir do endereço &HE000, uma rotina em Linguagem de Máquina que permite o armazenamento da tela na RAM e sua posterior recuperação, de volta para a VRAM.

A primeira parte do programa é a responsável pela geração da rotina de transferência. A segunda parte é um exemplo de como a rotina em L.M. deve ser usada.

Observe a incrível velocidade com que a rotina de transferência consegue armazenar (e recuperar!) os 16 Kbytes da VRAM !!! Em BASIC, essa mesma transferência, se fosse possível, certamente demoraria algo em torno de 100 vezes mais!

100 FOR F=&HE000 TO &HE03E	57C
110 READ A\$: POKE F,VAL("&H"+A\$)	D01
120 NEXT F	EA2
130 DEFUSR0=&HE000 : DEFUSR1=&HE011	173E
140 DATA CD,22,E0,21,00,00,11,00	224E
150 DATA 40,01,FF,3F,CD,59,00,18	2AA0
160 DATA 24,CD,22,E0,21,00,40,11	3882
170 DATA 00,00,01,FF,3F,CD,5C,00	4045
180 DATA 18,13,F3,DB,A8,47,CB,3F	5F8E
190 DATA CB,3F,CB,3F,CB,3F,80,E6	76A0
200 DATA FC,D3,A8,FB,C9,F3,DB,A8	8778
210 DATA E6,F0,D3,A8,FB,C9,A2,00	8D63
220 REM -----Exemplo de utilização-----	96D8
230 COLOR 1,7,5 : SCREEN 2	9D46
240 FOR F=80 TO 1 STEP -10	A543
250 CIRCLE(128,80),80,1,,,80/F	AE5C
260 CIRCLE(128,80),80,1,,,F/80	BCA2
270 NEXT F	BF48
280 LINE (128,160)-(128,0)	C801
290 LINE (48,80)-(208,80)	D387
300 POKE 0,USR0(0) : ' VRAM para RAM	E871
310 SCREEN 0 : WIDTH 38 : LOCATE 10,10	F9A2
320 PRINT "Digite RETURN!":A\$=INPUT\$(1)	986
330 SCREEN 2	AE3
340 POKE 0,USR1(0) : ' RAM para VRAM	141D
350 GOTO 350	1663

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulos 0, 3 e 4.
Programação Avançada em MSX - capítulo 2.
Coleção de Programas para MSX v.1 - páginas 31 a 54.

2.V - USANDO A VRAM PARA DADOS

O vídeo do MSX é controlado por um circuito especial chamado VDP, que tem a sua disposição 16 Kbytes de memória chamada Vídeo-RAM (VRAM).

O VDP pode usar a VRAM de quatro formas diferentes, correspondentes às SCREEN 0, 1, 2 e 3 do MSX.

Para cada SCREEN, a VRAM é dividida de forma diferente e sempre sobram algumas áreas não ocupadas. A seguir, apresentamos uma tabela onde relacionamos a área não usada da VRAM para cada SCREEN do MSX.

SCREEN	ÁREA NÃO USADA
0	960 até 2047 4096 até 16383
1	2048 até 6143 7040 até 8191 8224 até 14335
2	7040 até 8191
3	3584 até 6912 7940 até 14335

Todas as áreas livres da VRAM podem ser usadas para armazenamento de dados. Isso pode ser feito tanto com rotinas em Linguagem de Máquina como com o comando VPOKE do BASIC.

Vamos ilustrar de forma bem simples como se pode usar a VRAM para dados. Digite e execute o programa a seguir.

```
10 SCREEN 0 : WIDTH 38
20 A$="DADOS EXEMPLOS PARA A VRAM"
30 VPOKE 4096,LEN(A$)
40 FOR F=1 TO LEN(A$)
50   VPOKE 4096+F,ASC(MID$(A$,F,1))
60 NEXT F
```

```
291
CE3
1100
1768
2260
248E
```

Com isso os dados da string A\$ estarão na VRAM. Agora você pode comandar NEW, apagando o programa e as

variáveis da memória RAM normal do micro.

Para recuperar os dados que foram passados para a VRAM, digite e rode o programa abaixo.

```
10 SCREEN 0 : WIDTH 39
20 X=VPEEK(4096)
30 A$=""
40 FOR F=1 TO X
50   A$=A$+CHR$(VPEEK(4096+F))
60 NEXT F
70 PRINT A$
```

298
629
850
C30
1482
1668
1868

Com isso, A\$ estará novamente com os dados que havíamos gravado na VRAM.



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 105 e 106.

2.M - PESQUISADOR DE DESENHOS

O programa apresentado adiante é um editor de caracteres para telas gráficas. Digite-o, confira-o e grave-o. Depois, rode o programa.

Ao ser executado ele permite que o usuário "vasculhe" a memória do micro a procura de um desenho para tela gráfica. Os dois terços superiores da tela são usados para mostrar os desenhos dos caracteres da região da memória pesquisada tal qual eles apareceriam na figura da tela gráfica. O terço inferior da tela é usado para mostrar um único caractere em destaque. Esse caractere pode ser redesenhado e corresponde à ampliação das posições indicadas por um pequeno quadrado no centro da tela:

Assim que o programa é carregado, as teclas de setas permitem mover a região da memória pesquisada. Lembre-se que de &H0000 a &H7FFF existe ROM e a partir de &H8000 existe RAM. As funções das teclas de setas estão resumidas abaixo.

- ▶ : pesquisa um byte para a frente;
- SHIFT + ▶ : pesquisa 8 bytes para frente
(1 caractere);
- ◀ : pesquisa um byte para trás;
- SHIFT + ◀ : pesquisa 8 bytes para trás
(1 caractere);
- ▲ : pesquisa 256 bytes para frente
(1 linha);
- SHIFT + ▲ : pesquisa 2048 bytes para frente
(1/3 de tela);
- ▼ : pesquisa 256 bytes para trás (1 linha);
- SHIFT + ▼ : pesquisa 2048 bytes para trás
(1/3 de tela).

Uma vez escolhida a região de memória a ser pesquisada, pode-se editar o caractere em destaque. Para isso deve-se digitar a tecla SELECT. Ela permite a passagem do modo "seleção" para o modo "edição". Pressionando SELECT a primeira vez um pequeno cursor surgirá no caractere em destaque na parte inferior da tela. Pode-se movê-lo com as teclas de setas. Para alterar um ponto, basta pressionar a barra de espaços; se ele estiver apagado, será aceso e, se estiver aceso, será apagado.

Com o caractere já editado, pode-se retornar do modo de edição digitando mais uma vez a tecla SELECT.

Para sair do programa e retornar ao BASIC, basta estar fora do modo de edição e pressionar a barra de espaços.

Se, ao retornar ao BASIC, a tela ficar toda com uma só cor, use um comando:

```
color 15,1,1
```

Isso deverá resolver o problema.

100	CLEAR 200,&HD000	306
105	FOR FZ=&HD000 TO &HD32A	3E0
110	READ A\$: POKE FZ,VAL("&H"+A\$)	FF4
115	NEXT FZ	110E
120	BSAVE"ACHADES.BIN",&HD000,&HD32A	2100
125	DEFUSR0=&HD000 : END	1E18
130	DATA 3E,0F,32,EB,F3,CD,72,00	2600
135	DATA 3E,01,32,B0,FB,21,FA,D2	308E
140	DATA ED,5B,CF,F3,01,10,00,CD	418E
145	DATA 5C,00,21,0A,D3,ED,5B,CD	508E
150	DATA F3,01,08,00,CD,5C,00,CD	618E
155	DATA B3,D1,E5,CD,07,D2,E1,2A	8464
160	DATA F8,D2,11,00,00,01,00,10	8E70
165	DATA CD,5C,00,06,01,21,E0,FB	8E74
170	DATA 7E,CB,47,20,02,06,08,23	954F
175	DATA 23,7E,CB,7F,20,0C,2A,F8	A276
180	DATA D2,23,10,FD,22,F8,D2,C3	A7E0
185	DATA 27,D0,CB,67,20,0C,2A,F8	01E7
190	DATA D2,2B,10,FD,22,F8,D2,C3	030E
195	DATA 27,D0,CB,77,20,0F,2A,F8	E818
200	DATA D2,11,00,01,19,10,FD,22	F00F
205	DATA F8,D2,C3,27,D0,CB,6F,20	787
210	DATA 11,2A,F8,D2,AF,11,00,01	018
215	DATA ED,52,10,F8,22,F8,D2,C3	1807
220	DATA 27,D0,21,E2,FB,7E,CB,47	1064
225	DATA 20,01,C9,2B,7E,CB,77,C2	2618
230	DATA 3B,D0,CD,C0,00,21,11,D3	38FC
235	DATA 36,04,2B,2B,3A,E2,FB,07	4049
240	DATA 30,03,CD,DB,D2,07,2B,30	5060
245	DATA 03,CD,DB,D2,07,30,03,CD	7210
250	DATA E1,D2,07,23,30,03,CD,E1	841F
255	DATA D2,7E,FE,60,30,02,36,60	8A50
260	DATA FE,83,38,02,36,83,2B,7E	8E86
265	DATA FE,8B,30,02,36,8B,FE,AE	976F
270	DATA 38,02,36,AE,01,08,00,ED	A000
275	DATA 5B,CD,F3,21,0A,D3,CD,5C	AE3E
280	DATA 00,CD,E7,D2,21,E2,FB,7E	000E
285	DATA 0F,DA,7E,D1,2A,F8,D2,01	0848
290	DATA 80,08,09,3A,0E,D3,FE,90	E671

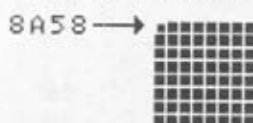
295 DATA 38,01,23,FE,95,38,01,23
300 DATA FE,9A,38,01,23,FE,9F,38
305 DATA 01,23,FE,A4,38,01,23,FE
310 DATA A9,38,01,23,FE,AE,38,01
315 DATA 23,FE,B3,38,01,23,06,00
320 DATA 3A,0F,D3,FE,60,20,04,CB
325 DATA F8,18,36,FE,65,20,04,CB
330 DATA F0,18,2E,FE,6A,20,04,CB
335 DATA E8,18,26,FE,6F,20,04,CB
340 DATA E0,18,1E,FE,74,20,04,CB
345 DATA D8,18,16,FE,79,20,04,CB
350 DATA D0,18,0E,FE,7E,20,04,CB
355 DATA C8,18,06,FE,83,20,02,CB
360 DATA C0,7E,A8,77,2A,F8,D2,01
365 DATA 80,08,09,CD,07,D2,21,E1
370 DATA FB,7E,CB,77,C2,A5,D0,DD
375 DATA 21,0E,D3,DD,36,00,8B,DD
380 DATA 23,DD,36,00,60,DD,23,DD
385 DATA 23,DD,36,00,00,CD,C0,00
390 DATA 01,08,00,ED,5B,CD,F3,21
395 DATA 0A,D3,CD,5C,00,CD,E7,D2
400 DATA C3,3B,D0,21,12,D3,CD,0E
405 DATA 4B,3E,0F,32,E9,F3,3E,28
410 DATA 32,B7,FC,3E,8A,32,B9,FC
415 DATA 2A,F8,D2,01,80,08,09,E5
420 DATA 7C,CD,C8,D2,CD,CD,D2,E1
425 DATA E5,7C,CD,CD,D2,E1,E5,7D
430 DATA CD,C8,D2,CD,CD,D2,E1,E5
435 DATA 7D,CD,CD,D2,3E,01,CD,8D
440 DATA 00,3E,57,CD,8D,00,3E,01
445 DATA CD,8D,00,3E,57,CD,8D,00
450 DATA 3E,CF,CD,8D,00,E1,C9,01
455 DATA 08,00,09,E5,AF,ED,42,C5
460 DATA 7E,21,E9,F3,36,01,07,30
465 DATA 02,36,0F,21,87,FC,36,60
470 DATA 21,B4,00,41,2B,2B,2B,2B
475 DATA 2B,10,F9,22,B9,FC,08,3E
480 DATA D3,CD,8D,00,08,21,E9,F3
485 DATA 36,01,07,30,02,36,0F,21
490 DATA B7,FC,36,65,08,CD,8D,00
495 DATA 08,21,E9,F3,36,01,07,30
500 DATA 02,36,0F,21,87,FC,36,6A
505 DATA 08,CD,8D,00,08,21,E9,F3
510 DATA 36,01,07,30,02,36,0F,21
515 DATA B7,FC,36,6F,08,CD,8D,00
520 DATA 08,21,E9,F3,36,01,07,30
525 DATA 02,36,0F,21,87,FC,36,74
530 DATA 08,CD,8D,00,08,21,E9,F3
535 DATA 36,01,07,30,02,36,0F,21

FF75
9FF
EE2
14B1
1F2A
2720
38F8
44E6
5E33
73B1
85A6
8C1F
904B
950C
A2E1
B04B
C2E0
D87D
E9BE
F09F
7A4
0E1
1490
1E16
2714
3A2B
4C21
5F7E
736E
847F
8A0C
909E
9A6C
A30B
B25D
C3FE
D56F
EAFE
FF6D
851
D1E
1399
1DF3
273A
3960
433A
5D0E
72A7
8401

540 DATA B7,FC,36,79,08,CD,8D,00
 545 DATA 08,21,E9,F3,36,01,07,30
 550 DATA 02,36,0F,21,B7,FC,36,7E
 555 DATA 08,CD,8D,00,08,21,E9,F3
 560 DATA 36,01,07,30,02,36,0F,21
 565 DATA B7,FC,36,83,08,CD,8D,00
 570 DATA 08,C1,E1,0D,C2,08,D2,C9
 575 DATA 0F,0F,0F,0F,C9,E6,0F,C6
 580 DATA 30,FE,3A,38,02,C6,07,CD
 585 DATA 8D,00,C9,35,35,35,35,35
 590 DATA C9,34,34,34,34,34,C9,D9
 595 DATA 21,29,D3,46,C5,06,00,C5
 600 DATA C1,10,FC,C1,10,F6,D9,C9
 605 DATA 00,B0,FF,FF,C3,C3,C3,C3
 610 DATA FF,FF,F8,88,88,88,F8,00
 615 DATA 00,00,3F,80,00,06,8B,60
 620 DATA 01,00,28,34,30,2C,31,33
 625 DATA 38,29,F2,28,37,32,2C,31
 630 DATA 34,36,29,2C,31,2C,42,46
 635 DATA 00,32,55

8000
 8068
 95F4
 9E65
 A060
 B08A
 D1F8
 E6F7
 F016
 491
 8F5
 F5A
 181C
 2036
 3020
 4031
 5809
 6F00
 7F50
 820A

TOTAL = 820A



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 4.
 Programação Avançada em MSX - capítulo 2.

2.X - IMAGENS INSTANTÂNEAS

Muitas vezes a produção de uma tela por um programa em BASIC é demorada demais, principalmente se estamos trabalhando com a SCREEN 2.

Podemos fazer com que a imagem não seja mostrada durante a confecção do desenho, mas somente após ele estar pronto.

Uma forma bem simples de conseguir isso é usando o comando COLOR como exemplificamos a seguir.

100 COLOR 0,9,9 : SCREEN 2	374
110 REM ----- tela exemplo -----	4F6
120 FOR F=0 TO 125 STEP 10	1071
130 G=80-F*80/125	1604
140 LINE (F,80)-(125,G)	1E82
150 LINE (125,G)-(250-F,80)	26E8
160 LINE (F,80)-(125,160-G)	3874
170 LINE (125,160-G)-(250-F,80)	426A
180 NEXT F	4451
190 COLOR ,,1	4816
200 BEEP : GOTO 200	4100

Na linha 100 fazemos com que a cor de frente, de fundo e da borda da tela sejam iguais, impossibilitando a visão do que está sendo feito na tela. A linha 190 só é executada após o desenho estar terminado e coloca as cores na tela de uma só vez. Dessa forma, não vemos o desenho ser feito.

Experimente executar o programa novamente eliminando o comando COLOR da linha 100. Você verá o desenho sendo elaborado na tela.

Uma outra forma de conseguir o mesmo efeito é usando as rotinas do BIOS DISSCR (&H41) e ENASCR (&H44) como no programa a seguir.

10 DEFUSR0=&H41 : DEFUSR1=&H44	66H
20 POKE 0,USR0(0)	80H
30 FOR F=1 TO 24	C5B
40 PRINT "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"	14AB
50 NEXT F	16C2
60 POKE 0,USR1(0)	1880

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - página 147.
Coleção de Programas para MSX v.2 - páginas 42 e 68.

2.Y - IMPRESSOR EM TAMANHO DUPLO NA SCREEN 2

Aqui vai um programinha em linguagem de máquina que permite ao seu programa em BASIC imprimir qualquer mensagem em tamanho dobrado, na SCREEN 2. A grande utilidade dele é a de preencher a lacuna existente entre as SCREEN's 2 e 3, já que na primeira impressão é em tamanho natural, e na segunda em letras tiranossáuricas.

A utilização do programa é simples: após carregar a rotina em L.M. na memória, basta chamá-la pela função USR("texto") ou USR(var\$). O posicionamento na tela é feito pelo comando PSET, PRESET ou outro comando gráfico, e as cores de frente e fundo são dadas pelo comando COLOR.

1000	DATA	3A,63,F6,FE,03,C0,21,F8	88H
1010	DATA	F7,7E,23,66,6F,7E,32,E0	8Ab
1020	DATA	D0,23,7E,32,E1,D0,23,7E	148A
1030	DATA	32,E2,D0,3A,B5,FC,32,DE	1029
1040	DATA	D0,06,08,C5,CD,2B,D0,C1	2942
1050	DATA	10,F9,C9,0E,02,3A,E0,D0	0897
1060	DATA	32,E3,D0,C5,3A,B3,FC,32	4005
1070	DATA	DD,D0,2A,E1,D0,7E,E5,26	610F
1080	DATA	00,6F,29,29,29,11,BF,1B	77A9
1090	DATA	19,3E,08,90,C5,4F,06,00	8877
1100	DATA	09,7E,06,06,17,CD,6F,D0	897F
1110	DATA	10,FA,C1,E1,23,3A,E3,D0	8F87
1120	DATA	3D,32,E3,D0,20,D7,21,DE	9906
1130	DATA	D0,34,C1,0D,20,BF,C9,21	A0E9
1140	DATA	DF,D0,36,FF,38,02,36,00	B27C
1150	DATA	F5,C5,CD,8B,D0,21,DD,D0	057A
1160	DATA	34,CD,8B,D0,21,DD,D0,34	0894
1170	DATA	C1,F1,C9,3A,DE,D0,4F,E6	EC08
1180	DATA	07,6F,79,CB,3F,CB,3F,CB	8A
1190	DATA	3F,67,3A,DD,D0,E6,F8,06	705
1200	DATA	00,4F,09,3A,DD,D0,E6,07	8C7
1210	DATA	47,3E,08,90,47,AF,37,17	148E
1220	DATA	10,FD,47,3A,DF,D0,A7,28	1DAE
1230	DATA	06,CD,4A,00,B0,18,07,78	2A22
1240	DATA	2F,47,CD,4A,00,A0,CD,4D	3C83
1250	DATA	00,3A,E9,F3,87,87,87,87	5008
1260	DATA	47,3A,EA,F3,B0,01,00,20	6172
1270	DATA	09,CD,4D,00,C9,FA,6A,00	7598
1280	DATA	14,11,80,00,FIM	8033
1290	CLS:FOR I=&HD000 TO &HD0E3:READ A\$=&E71		
	POKE I,VAL("&H"+A\$):NEXT I:END		

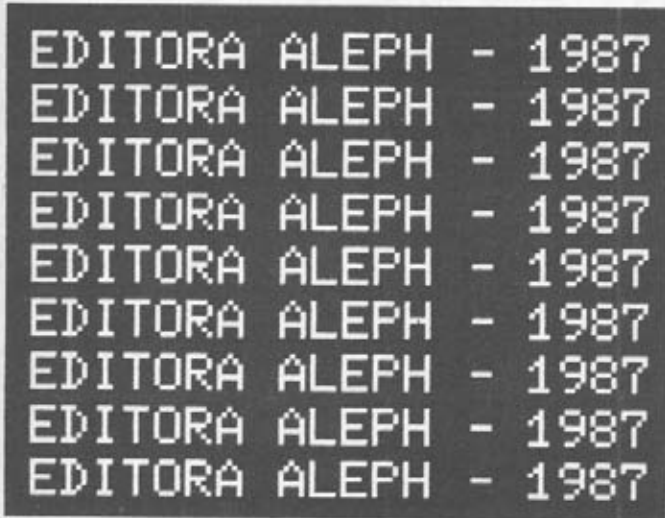
TOTAL = 8E71

O próximo programa é um exemplo de utilização da rotina:

```
10 SCREEN 2
20 PRESET (10,80):COLOR 11,4
30 DEFUSR=&HD000
40 A$=USR("Milton Maldonado Jr.")
50 COLOR 15,1,1
60 GOTO 60
```

100
482
YES
1000
1420
1780

Dica especial: para mudar a altura das letras, use POKE &HD02C,n com n variando de 1 a 8. Que acontece quando n vale 0?



EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987
EDITORA ALEPH - 1987



DICAS SONORAS

Neste capítulo abordamos vários recursos sonoros disponíveis nos micros MSX.

Além dos métodos normais, usando o PLAY e o SOUND, comentam-se alguns métodos não standard de geração de sons.

3.1 - Produzindo sons com ecos	87
3.2 - Percussão com o PLAY	88
3.3 - PLAY com SOUND	89
3.4 - Percussão com o click do teclado	90
3.5 - Teclado piano	91
3.6 - Editor de sons	95
3.7 - Músicas com programas	97
3.8 - Música aleatória	99
3.9 - Partitura sonora	100
3.A - Despertador	102
3.B - Digitalizador de voz	103

3.1 - PRODUZINDO SONS COM ECOS

A produção de sons com ecos nos micros MSX é extremamente simples através do comando PLAY. Para isso basta gerar um mesmo som nos três canais, porém com um pequeno atraso um em relação aos outros.

Veja o programa a seguir.

```
10 PLAY "S0M5000"  
20 PLAY "C#32", "R8C#32", "R16C#32"
```

```
301  
888
```

Ele gera um som com eco.

Uma outra forma é usar um envelope periódico. Por exemplo, digite e rode o programinha listado a seguir.

```
10 PLAY "T240S0M1000"  
20 PLAY "02CEGAA+AGE"  
30 PLAY "02CEGAA+AGE"  
40 PLAY "FA03CDD+DC02A"  
50 PLAY "02CEGAA+AGE"  
60 PLAY "02DEFF+GFED"  
70 GOTO 20
```

```
338  
881  
104  
1444  
2133  
11877  
11180
```

Agora altere a linha 10, deixando-a como mostramos abaixo, e execute o programa novamente.

```
10 PLAY "T240S8M1000"
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.2 - PERCUSSÃO COM PLAY

A versatilidade do comando PLAY permite facilmente a geração de sons de percussão. Veja o exemplo a seguir:

```
10 PLAY "T120S0M9004L4AA06L16AAAAA4"  
20 GOTO 10
```



Note que usamos o envelope 0 com um período de modulação muito curto. Experimente agora substituir T120 por T140 e M90 por M200.



BIBLIOGRAFIA RECOMENDADA:

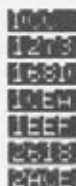
Linguagem Basic MSX - páginas 119 a 121.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.3 - PLAY COM SOUND

Uma outra forma de gerar sons de percussão com o comando PLAY é selecionando ruído para um ou mais canais através de um comando SOUND.

Experimente rodar o seguinte programa.

```
100 C$="0250M1500L8T136CCM3500CM1500CCCM"SH
3500CM1500C
110 SOUND 6,32
120 FOR I=1 TO 2
130   SOUND 7,28
140   PLAY """, """, C$
150 NEXT
160 C$="S0M10000C1...."
170 PLAY """, """, C$
```



Altere o valor do registro 6 (sempre entre 1 e 32) e também a duração das notas e o tempo de execução. Você verá que pode produzir os sons mais variados. Um último recurso é mudar o formato da envoltória. Experimente alterá-lo, por exemplo, para S13. Agora, ao invés de uma "bateria", você deverá obter um "chocalho".



BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121 e 144 a 150.
Curso de Basic MSX v.1 - aula 8.
Aprofundando-se no MSX - capítulo 5.

3.4 - PERCUSSÃO COM O CLICK DO TECLADO

Uma terceira e última forma de produzir sons de percussão no MSX é usando o click do teclado. Digite e execute o programa a seguir.

```
10 KEY(1)ON=KEY(2)ON
20 ON KEY GOSUB 70,80
30 X%=1
40 OUT 170,127
50 OUT 170,255
60 FOR F%=1 TO X%:NEXT F%:GOTO 40
70 X%=X%+1 : RETURN
80 X%=X%-1 AND X%>0 : RETURN
```

```
30E
710
928
E3E
E49
1130
1071
2334
```

TOTAL = 2334

Use as teclas F1 e F2 para perceber o funcionamento do programa.

Esse recurso é usado de forma magistral no conhecido "joguinho" PITFALL. Além de música nos três canais ele usa o click do teclado para fazer a percussão.



BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - página 15.
Aprofundando-se no MSX - página 87.

3.5 - TECLADO PIANO

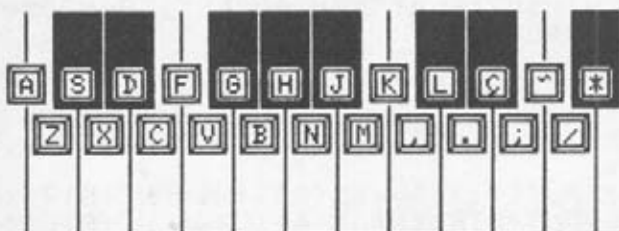
Existe uma maneira rápida e não muito sofisticada de se transformar um trecho do teclado de seu MSX num teclado musical.

No exemplo listado a seguir, as duas filas inferiores de teclas do MSX serão utilizadas.



Estas teclas correspondem ao Expert 1.1. No Hotbit haverá uma pequena modificação que será comentada adiante.

Vamos atribuir a cada uma das letras (ou símbolos) dessas duas filas o papel de uma tecla branca ou preta, conforme o esquema a seguir:



Para isto basta digitar o programa listado adiante.

```

10 SCREEN0:WIDTH39:KEY OFF:POKE&HFCAB,1 606
20 DIM N(255):PLAY"S0M2000L8" 663
30 FOR I=0 TO 18 1036
40 READ D$:D=ASC(D$):N(D)=I+41 1997
50 NEXT I. 11302
60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",",L,, 2898
C,;,/,*
70 A$=INPUT$(1):A=ASC(A$):N$="N"+STR$(N( 3000
A))
80 PRINT A$,=PRINT N$:PLAY N$ 4728
90 GOTO 70 4007

```

TOTAL = 4007

Os que tiverem um Hotbit devem alterar a linha 60 para:

```
60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",L,,  
C,/,>,L
```

Nesta linha são definidas as teclas que fazem, em seqüência, o papel das teclas de um piano. Você pode alterar esta linha para usar outro trecho do teclado do MSX. Lembre-se, porém, de ajustar o valor final de I no laço que vai de 30 a 50.

Na linha 40 o comando N(D)=I+36 ajusta a escala musical de maneira a fazer a primeira tecla (no nosso caso o Z) tocar o Dó central (N36).

Você também pode ajustar este valor de maneira a pegar outros trechos da escala musical em seu teclado.

Se você quiser alterar o timbre do seu "piano", modifique o PLAY da linha 20.

Como alteração final, sugerimos alguns acréscimos nas linhas 20, 70 e 80 de maneira a ficarem assim:

```
10 SCREEN0=WIDTH39:KEY OFF:POKE&HFCAB,1 606  
20 DIM N(255):PLAY"S0M2000L8","S0M2000L8" 1008  
  ", "S0M2000L8"  
30 FOR I=0 TO 18 178E  
40 READ D$:D=ASC(D$):N(D)=I+36 199A  
50 NEXT I 1805  
60 DATA Z,S,X,D,C,V,G,B,H,N,J,M,",",L,, 288E  
C,/,*,*  
70 A$=INPUT$(1):A=ASC(A$):N$="N"+STR$(N( 3003  
A)):NA$="N"+STR$(N(A)+4):NB$="N"+STR$(N(  
A)+7)  
80 PRINT A$,:PRINT N$:IF N(A)=0 THEN GOT 4723  
0 70 ELSE PLAY N$,NA$,NB$  
90 GOTO 70 4A85
```

TOTAL = 4A85

Experimente, ainda, alterar o comando SCREEN na linha 10 para: SCREEN0,,0

Você deve estar curioso para saber como foram geradas as figuras que ilustram o começo desta dica. Para fazer isto, basta digitar este complemento do programa:

```
1000 OPEN"GRP:"AS #1  
1010 COLOR 1,15,15:SCREEN 2  
1020 T$="C1S4R10E2D14H2L10G2U14F2D10G2R1  
4H2U10E2L14"
```

```

1030 FOR I= 1 TO 12
1040 READ A$
1050 PRESET(20*I,100):DRAW T$
1060 PRESET(20*I+3,102):PRINT#1,A$
1070 NEXT I
1080 DATA A,S,D,F,G,H,J,K,L,C,~,*
1090 FOR I= 1 TO 11
1100 READ A$
1110 PRESET(20*I+10,119):DRAW T$
1120 PRESET(20*I+13,121):PRINT#1,A$
1130 NEXT I
1140 DATA Z,X,C,V,B,N,M,"",";",";",";"/
1150 T1$="C1L20U48R9U36L18D36R9"/
1160 T2$="C1L20U48BU18U18"
1170 FOR I=0 TO 11
1180 PRESET (45+20*I,162):IF I=0 OR I=3
OR I=7 OR I=10 THEN DRAW T2$ ELSE DRAW T
1$=PAINT(25+20*I,90),1,1
1190 NEXT I
1200 RETURN

```

Além disso você deve alterar as linhas 10 e 80 do programa original para:

```

10 POKE &HFCAB,1
80 IF N(A)=0 THEN GOTO 70 ELSE PLAY N$,N
A$,NB$

```

e acrescentar a linha 68:

```
68 GOSUB 1000
```

Obviamente, quem tiver um Hotbit deve fazer as correspondentes alterações nas linhas 1080 e 1140.

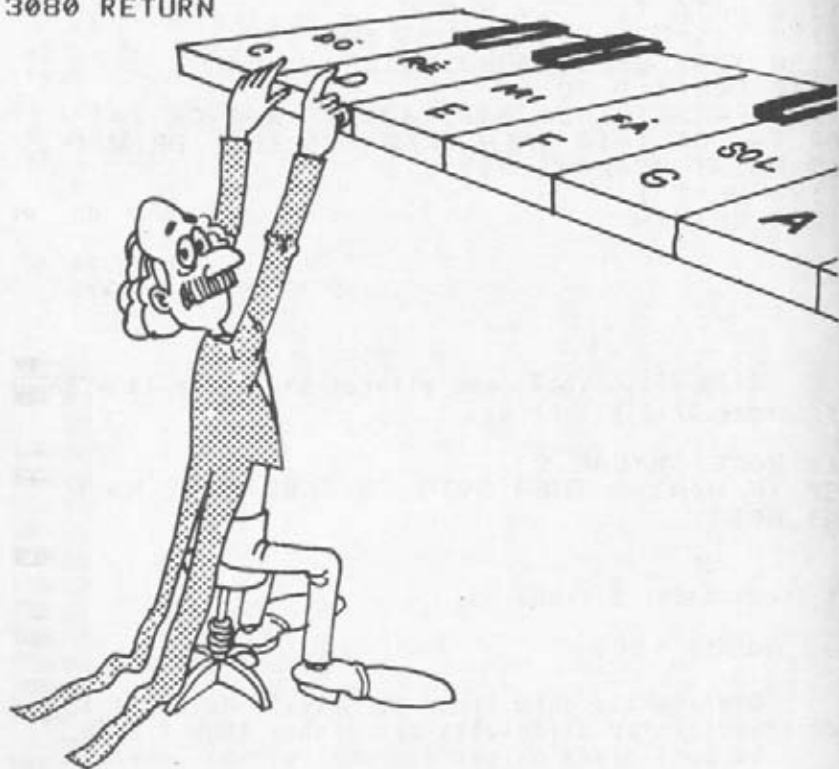
Se você ainda quiser dar uma última sofisticada em seu programa, acrescente-lhe as linhas a seguir:

```

25 GOSUB 3000
80 IF N(A)=0 THEN GOTO 70 ELSE PLAY N$,N
A$,NB$:PUT SPRITE 0,(21+10*X(P),139-19*Y
(P)),8,1
1015 FOR S=1 TO 8
1016 IF S=8 THEN S$=S$+CHR$(127) ELSE S$
=S$+CHR$(65)

```

```
1017 NEXT S
1018 SPRITE$(1)=S$
3000 DIM X(19),Y(19)
3010 X(0)=1
3020 FOR P=0 TO 18
3030 R=P MOD 12
3040 X(P+1)=X(P)+1:IF R=4 OR R=11 THEN X
(P+1)=X(P)+2
3050 IF X(P) MOD 2=1 THEN Y(P)=1 ELSE Y(P)=
2
3070 NEXT P
3080 RETURN
```



BIBLIOGRAFIA RECOMENDADA:

Curso de Basic MSX v.1 - aula 8.
Linguagem Basic MSX - páginas 119 a 121.

3.6 - EDITOR DE SONS

O programa a seguir é de grande utilidade na procura de valores corretos para serem usados com o comando SOUND.

Ao ser executado, ele apresenta uma tabela com os números dos registros do PSG à esquerda. Para selecionar um dado registro deve-se usar a tecla de seta para cima. O número do registro selecionado aparecerá na parte inferior da tela, logo abaixo da tabela. Para alterar seu valor deve-se usar as teclas de seta para a esquerda e seta para a direita. O registro 7 (de mixagem) é alterado de forma diferente. Inicialmente deve ser selecionado com a tecla de seta para cima; depois deve-se pressionar a barra de espaços; a seguir, basta pressionar 0 teclas, 0 ou 1, para programar em binário o registro; após ter inserido os valores 0 e 1 nos bits do registro 7, pode-se pressionar RETURN para voltar ao modo de seleção normal do programa.

É recomendável que se conheça o funcionamento dos registros do PSG para programá-los conscientemente.

10	'*****'	59A
20	'* BY THE DOCTOR LUZ *	000
30	'*****'	164E
40	FOR L=&HC000 TO &HC145	1CE8
50	READ A\$:POKE L,VAL("&H"+A\$)	2599
60	NEXT	214E
70	DEFUSR=&HC000	2028
80	PRINT"RODAR PROGRAMA ?"	387E
90	A\$=INPUT\$(1)	4103
100	IF A\$="S" THEN A=USR(0) ELSE END	4E9A
1000	DATA CD,C3,00,21,3C,C1,11,00	6480
1010	DATA 00,01,0A,00,CD,5C,00,3E	7890
1020	DATA 07,1E,38,CD,93,00,3E,08	8920
1030	DATA 1E,0F,CD,93,00,3E,09,CD	8F4E
1040	DATA 93,00,3E,0A,CD,93,00,26	9330
1050	DATA 01,2E,02,CD,C6,00,CD,76	9068
1060	DATA C0,CD,9F,00,FE,5A,C8,FE	9488
1070	DATA 1C,28,0F,FE,1D,28,21,FE	9398
1080	DATA 1E,28,65,FE,20,CA,C8,C0	05FE
1090	DATA 18,DD,DD,21,1B,C1,3A,2A	080F
1100	DATA C1,16,00,5F,DD,19,DD,7E	EE68
1110	DATA 00,C6,01,DD,77,00,18,C7	327
1120	DATA DD,21,1B,C1,3A,2A,C1,16	81E
1130	DATA 00,5F,DD,19,DD,7E,00,D6	112F
1140	DATA 01,DD,77,00,18,B1,3A,2B	1170

```

1150 DATA C1,DD,21,1B,C1,DD,77,07
1160 DATA 06,0F,0E,00,59,CD,F6,C0
1170 DATA 3E,20,CD,A2,00,DD,7E,00
1180 DATA 5F,79,DD,23,0C,CD,93,00
1190 DATA CD,F6,C0,3E,0A,CD,A2,00
1200 DATA 3E,0D,CD,A2,00,10,DD,C9
1210 DATA 3E,0A,CD,A2,00,3E,0D,CD
1220 DATA A2,00,3A,2A,C1,3C,FE,0E
1230 DATA D4,C5,C0,32,2A,C1,5F,CD
1240 DATA F6,C0,C3,27,C0,3E,00,C9
1250 DATA 21,09,09,CD,C6,00,3E,4D
1260 DATA CD,A2,00,3E,49,CD,A2,00
1270 DATA 3E,58,CD,A2,00,06,08,CD
1280 DATA 9F,00,CD,A2,00,CB,22,FE
1290 DATA 30,28,02,CB,C2,10,F0,7A
1300 DATA 32,28,C1,C3,27,C0,7B,CB
1310 DATA 3F,CB,3F,CB,3F,CB,3F,21
1320 DATA 2C,C1,D5,5F,16,00,19,7E
1330 DATA CD,A2,00,D1,21,2C,C1,7B
1340 DATA E6,0F,5F,16,00,19,7E,CD
1350 DATA A2,00,C9,3D,C1,49,C1,FA
1360 DATA 29,C1,00,0F,0F,0F,00,00
1370 DATA 00,00,00,00,30,31,32,33
1380 DATA 34,35,36,37,38,39,41,42
1390 DATA 43,44,45,46,52,45,47,2E
1400 DATA 20,56,41,4C,4F,52

```

```

2096
2445
3012
4036
5132
758F
8500
880A
8222
9011
H53D
8504
0687
0937
E10F
40F
02F
110A
1990
2237
2089
3E71
4079
6500
0982
8920

```

TOTAL = 8920



BIBLIOGRAFIA RECOMENDADA:

- Linguagem Basic MSX - páginas 144 a 150.
- Aprofundando-se no MSX - capítulo 5.
- Linguagem de Máquina MSX - páginas 144 a 146.

3.7 - MÚSICAS COM PROGRAMAS

O programa abaixo ilustra um típico caso de multiprocessamento das máquinas MSX. Enquanto executa as tarefas de processamento normais de um micro, o MSX pode, simultaneamente, executar músicas. Apenas a título de exemplo, usamos um processamento bem simples, que consiste apenas em mostrar na região central da tela alguns números.

Note que a cada 1/54 segundos o processamento normal é interrompido para que o gerador de sons possa ser programado. Enquanto o gerador de sons executa a programação, o processamento normal continua.

```
100 SCREEN 0,,0:WIDTH 39:KEY OFF
110 INTERVAL ON : DEFINIT F,G
120 DIM P$(34),Q$(34)
130 GOSUB 250:K=1
140 P$(1)="t200s0m7000"
150 Q$(1)="t200s0m7000"
160 PLAY P$(1),Q$(1)
170 ON INTERVAL=54 GOSUB 620
180 FOR F=1 TO 1000
190 LOCATE 8,10,0
200 PRINT USING"#####";F^2;K
210 FOR G=1 TO 100 : NEXT G
220 NEXT F
230 GOTO 180
240
250 P$(2)="r4o3":Q$(2)="r4o3"
260 P$(3)="a4o4c2d4":Q$(3)="r4L64ao4cea"
270 P$(4)="e4.f8e4":Q$(4)="o3ao4cea"
280 P$(5)="d2o3b4":Q$(5)="o3gbo4dg"
290 P$(6)="g4.a8b4":Q$(6)="o3gbo4dg"
300 P$(7)="o4c2o3a4":Q$(7)="o3ao4cea"
310 P$(8)="a4.g#8a4":Q$(8)="dfao5d"
320 P$(9)="b2g#4":Q$(9)="o4eg#bo5d"
330
340 P$(10)="e2a4":Q$(10)="o4eg#bo5d"
350 P$(11)="o4c2d4":Q$(11)="o3ao4cea"
360 P$(12)="e4.f8e4":Q$(12)="o3ao4cea"
370 P$(13)="d2o3b4":Q$(13)="o3gbo4dg"
380 P$(14)="g4.a8b4":Q$(14)="o3gbo4dg"
390 P$(15)="o4c4.o3b8a4":Q$(15)="o3ao4cea"
400 P$(16)="g#4.f#8g#4":Q$(16)="o4eg#bo5d"
410 P$(17)="a2.":Q$(17)="o3ao4cea"
420 P$(18)="a2r4":Q$(18)="o3ao4cea"
```



```

430 '
440 P$(19)="o4g2.":Q$(19)="14o4ceg"
450 P$(20)="g4.f8e4":Q$(20)="ceg"
460 P$(21)="d2o3b4":Q$(21)="o3gbo4d"
470 P$(22)="g4.a8b4":Q$(22)="o3gbo4d"
480 P$(23)="o4c2o3a4":Q$(23)="o3ao4ce"
490 P$(24)="a4.g#8a4":Q$(24)="o3ao4ce"
500 P$(25)="b2g#4":Q$(25)="eg#b"
510 P$(26)="e2.":Q$(26)="eg#b"
520 '
530 P$(27)="o4g2.":Q$(27)="04CEG"
540 P$(28)="g4.f8e4":Q$(28)="ceg"
550 P$(29)="d2o3b4":Q$(29)="o3gbo4d"
560 P$(30)="g4.a8b4":Q$(30)="o3gbo4d"
570 P$(31)="o4c4.o3b8a4":Q$(31)="o3ao4ce"
"
580 P$(32)="g#4.f#8g#4":Q$(32)="eg#b"
590 P$(33)="a2.a2R4":Q$(33)="o3ao4ceo3a2"
"
600 RETURN
610 '
620 K=K+1
630 IF K=34 THEN K=2
640 PLAY P$(K),Q$(K)
650 RETURN

```



BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 108, 178 e 179.

3.8 - MÚSICA ALEATÓRIA

Se você tiver curiosidade em saber como será a música do ano 3000, basta mandar seu MSX soltar a imaginação.

Digite o programa a seguir e ouça seu MSX "compondo". Quando você não aguentar mais, digite CONTROL + STOP.

Se você, porém, tiver paciência e curiosidade, ouça atentamente e verifique que há um certo padrão e algumas passagens geniais. Afinal a geração dos números aleatórios que estão sob a música é feita pelo micro segundo uma rígida regra matemática.

```
100 PLAY "S0M8000","S0M8000","S0M8000" 700
110 L$="L"+STR$(INT(RND(-TIME)*31)*2+2) 1000
120 X$=L$+"N"+STR$(INT(RND(-TIME)*60)) 2000
130 Y$=L$+"N"+STR$(INT(RND(-TIME)*30+50)) 3000
)
140 Z$=L$+"N"+STR$(INT(RND(-TIME)*16+80)) 4000
)
150 PLAY X$,Y$,Z$ 4000
160 GOTO 110 4000
```

A composição do exemplo intitula-se "A ORDEM DO CAOS" e o andamento é "ALLEGRO VIVACE CON UN PIZZICO DI PAZZIA". Se você quiser alterar as regras mude a linha 100 (tentando outros envelopes) e as linhas 120 a 140, mudando a distribuição das notas. Lembre-se, porém, que o argumento do N no PLAY não pode ultrapassar 96.

Lembrando o famoso exemplo do físico JAMES JEANS (2 macacos imortais acorrentados a um piano por toda a eternidade acabarão tocando uma sonata de BEETHOVEN), tenha paciência e fique esperando a obra prima do seu MSX!



3.9 - PARTITURA SONORA

Excetuando-se alguns raros privilegiados que têm o chamado "ouvido absoluto", a maioria das pessoas tem uma certa dificuldade em identificar uma nota musical tocada individualmente.

Associar o som da nota a uma tecla do piano ou a uma posição na partitura musical se torna ainda mais difícil.

Para treinar seu ouvido e sua leitura de partitura, digite o programa a seguir. Para fazer a nota "subir" ou "descer" pela escala musical basta usar as teclas de seta para cima e seta para baixo. Uma vez escolhida a nota, basta apertar a barra de espaços para ouvir seu som.

```

100 P$(1)="DO           C"=P$(2)="RE           D"
110 P$(3)="MI           E"=P$(4)="FA           F"
120 P$(5)="SOL          G"=P$(6)="LA           A"
130 P$(7)="SI           B"
140 COLOR 15,1,1:SCREEN 2,,0
150 OPEN"GRP":"AS #1
160 PRESET(31,170)
170 PRINT#1,"OITAVA  NOTA  CIFRA"
180 FOR C=0 TO 1
190 FOR L=63+48*C TO 95+48*C STEP 8
200 LINE(0,L)-(255,L)
210 NEXT L,C
220 FOR I=1 TO 26
230 D=23+8*I
240 PRESET(D,152-4*I):PRINT#1,CHR$(1)+
    CHR$(73)
250 LINE(D,5)-(D+6,21),,B
260 IF IMOD7=2 OR IMOD7=6 THEN 280
270 LINE(D-3,5)-(D+1,15),,BF
280 NEXT I
290 LINE(30,151)-(38,151):LINE(126,103)
    -(134,103):LINE(222,55)-(237,55)
300 D1$="C15S12LHUERM+2,+1M+1,+2M-1,+2M-
    2,+1M-2,-1H2U2M+1,-2E4U2HD16GH"
310 PRESET(14,86):DRAW D1$
320 PRESET(14,87):DRAW D1$
330 D2$="C15S12LHUERM+2,+1M+1,+2M-1,+362"
    "
340 PRESET(14,120):DRAW D2$
350 PRESET(14,121):DRAW D2$:PSET(27,115)
    :PSET(27,123)

```

```

360 FOR T=1 TO 8:C%=C%+CHR$(PEEK(7222+T))
):D%=D%+CHR$(PEEK(7206+T)):NEXT T:SPRITE
$(1)=C%:SPRITE$(2)=D%
370 I=13:PLAY"S0M7000"
380 A=STICK(0)
390 I=I+(A=5)-(A=1):X=23+8*I:Y=152-4*I
400 IF I<1 THEN I=1
410 IF I>26 THEN I=26
420 O=(I+15)\7:W=(I+15)MOD7+1
430 IF A=0 THEN 460
440 LINE (40,180)-(240,190),1,BF
450 PRESET(41,181):PRINT#1,0;" "P$(
W)
460 PUT SPRITE 3,(X,Y-1),,1
470 PUT SPRITE 4,(X,22),,2
480 N=12*(I\7)+14+2*(IMOD7)+(IMOD7>1)+(I
MOD7>5)
490 A$="N"+STR$(N)
500 IF STRIG(0) THEN GOSUB 520
510 GOTO 380
520 PLAY A$
530 IF STRIG(0) THEN 530
540 RETURN

```

OITAVA	NOTA	CIFRA
4	DO	C

BIBLIOGRAFIA RECOMENDADA:

Linguagem Basic MSX - páginas 119 a 121.
 Curso de Basic MSX v.1 - aula 8.
 Aprofundando-se no MSX - capítulo 5.

3.A - DESPERTADOR

Se você quiser usar seu MSX como relógio despertador, digite o programa a seguir e rode-o. Forneça a hora, minuto e segundo do momento de despertar e depois atualize o horário. Ao digitar o valor do SEGUNDO ATUAL digite um número um pouco maior e fique esperando até seu relógio de pulso indicar o mesmo número. Nesse instante pressione RETURN.

```
100 SCREEN 1:D=0:KEY OFF
110 INPUT"HORA DO DESPERTAR";HD
120 INPUT"MINUTO DO DESPERTAR";MD
130 INPUT"SEGUNDO DO DESPERTAR";SD
140 INPUT"HORA ATUAL";H0
150 INPUT"MINUTO ATUAL";M0
160 INPUT"SEGUNDO ATUAL";S0
170 TIME=0
180 CLS:LOCATE 3,10,0:PRINT"DESPERTADOR
PARA ";
190 PRINT USING "##:##:##";HD;MD;SD
200 LOCATE 3,12:PRINT" HORA CERTA
";
210 T=TIME\60:S1=(S0+T)MOD 60
220 M=(S0+T)\60:M1=(M0+M) MOD 60
230 H=(M+M0)\60:H1=(H0+H) MOD 24
240 PRINT USING "##:##:##";H1;M1;S1
250 D=(H1=HD)+(M1=MD)+(S1=SD)
260 IF D=-3 THEN PLAY "V15N40N42"
270 IF STRIG(0) THEN D=0
280 GOTO 200
```

TOTAL = 0597

Não esqueça de deixar o volume do micro ou da TV no máximo e siga o seguinte procedimento. vá dormir. Ao tocar o alarme, acorde! Levante e, rastejando, aproxime-se do micro. Pressione a barra de espaços para acabar com o maldito barulho e não volte para a cama! Se quiser um barulho mais irritante altere o PLAY da linha 260. Se chegar atrasado no serviço, leve a listagem do programa para seu chefe e tentem descobrir, juntos, onde você errou na digitação!

BIBLIOGRAFIA RECOMENDADA:

LINGUAGEM BASIC MSX - página 166.

3.B - DIGITALIZADOR DE VOZ

Com este programa você pode digitalizar qualquer som e depois reproduzi-lo através do canal de áudio do micro. É muito útil em entradas de programas, que podem ser o trecho de uma música ou mesmo sua própria voz.

A entrada do som é feita pelo cabo de entrada (EAR) do cassete, e o som reproduzido sai pelo canal de áudio. Se o seu micro é um Expert, o som sairá pelo alto-falante interno. No Hot-Bit este sairá pela televisão.

A duração do som gravado é determinada pelos endereços &HC007 e &HC035. Os valores destes dois bytes devem ser iguais e compreendidos entre 0 e 170, e a maior duração é obtida com o valor 0. Para mudar a duração, basta dar POKes nestes endereços com os valores acima especificados.

Você pode salvar uma informação digitalizada em disco ou fita com o comando:

```
BSAVE "NOME",ST,&HB000
```

onde

```
ST=256*PEEK(&HC007)
```

Naturalmente se o valor dado a &HC007 for menor que 128 o BSAVE não terá o efeito desejado, pois este salvará uma parte da ROM em vez de salvar a RAM abaixo de &HB000.

1000	REM	-----	544
1010	REM	Digitalizador de Voz	019
1020	REM	By The Pilot 1988	1530
1030	REM	-----	2006
1040	DATA	F3,3E,AA,D3,A8,21,00,80	2800
1050	DATA	11,00,B0,AF,06,08,4F,DB	3A82
1060	DATA	A2,E6,80,B1,CB,3F,00,00	48EE
1070	DATA	00,00,00,00,00,00,10,EE	5EAF
1080	DATA	77,23,E5,ED,52,E1,38,E3	74A8
1090	DATA	3E,A0,D3,A8,FB,C9,F3,3E	882A
1100	DATA	AA,D3,A8,21,00,80,11,00	8DAA
1110	DATA	B0,7E,06,08,4F,1F,1F,D3	922F
1120	DATA	AA,79,CB,3F,00,00,00,00	98CC
1130	DATA	00,00,00,00,10,EE,23,E5	A556
1140	DATA	ED,52,E1,38,E4,3E,A0,D3	B179
1150	DATA	A8,FB,C9,FIM	BCFE

```

1160 CLS:FOR I=&HC000 TO &HC05A:READ A$:
POKE I,VAL("&H"+A$):NEXT I
1170 PRINT "GRAVAR:DEFUSR=&HC000:PRINT U
SR(0)
1180 PRINT "TOCAR: DEFUSR=&HC02E:PRINT U
SR(0)
1190 END

```

Se o seu micro for um Hotbit, mude a linha 1190 para:

```
1190 POKE &HC002,255:POKE &HC030,255:END
```



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 3.



DICAS PARA O CASSETE

Neste capítulo abordaremos o gravador cassete, seus recursos e suas utilizações.

Além dos recursos disponíveis normalmente no BASIC, várias aplicações somente possíveis através da Linguagem de Máquina são abordadas. O gravador pode ser usado com mais segurança e maior rapidez.

- 4.1 - Auto-execução de programas em cassete . 106
- 4.2 - Gravando programas BASIC em binário ... 108
- 4.3 - Gravando textos 112
- 4.4 - Lendo textos 113
- 4.5 - Verificando gravações 114
- 4.6 - Gravador de SCREEN 2 116

4.1 - AUTO-EXECUÇÃO DE PROGRAMAS EM CASSETTE

O programa apresentado a seguir gera e grava em fita uma pequena rotina em Linguagem de Máquina capaz de carregar e auto-executar programas em BASIC gravados por CSAVE.

O procedimento para usá-lo é o seguinte:

- ▶ digite e execute o programa;
- ▶ prepare o gravador cassete com a fita em que será gravado o programa em BASIC;
- ▶ digite RETURN para gravar a rotina em L.M. (ela será gravada com o nome AUTOMA);
- ▶ digite NEW;
- ▶ digite o programa em BASIC a ser gravado em fita cassete;
- ▶ grave-o com o comando:

CSAVE"MATRIZ"

Agora, a fita deve conter, em sequência, o programa AUTOMA (gravado por BSAVE) e o programa MATRIZ (gravado por CSAVE). O programa AUTOMA pode ter qualquer outro nome, porém o programa MATRIZ deve ser gravado exatamente com esse nome (com as letras maiúsculas!).

Para carregar e auto-executar o programa MATRIZ, basta posicionar a fita no programa AUTOMA e comandar:

BLOAD"CAS:",R

Quando for carregado para a memória do micro o programa AUTOMA carregará e executará o programa MATRIZ!

PROGRAMA GERADOR DO AUTOMA

```
100 REM
110 REM AUTOCAS.BAS
120 REM
130 SCREEN 0
140 WIDTH 39
150 CLEAR 200,&HC000
160 FOR F=&HC000 TO &HC041
170   READ A$
180   A=VAL("&H"+A$)
190   POKE F,A
200 NEXT F
210 PRINT,,,, "PREPARE O GRAVADOR E ";
```

```

220 PRINT "PRESSIONE RETURN!"
230 A$=INPUT$(1)
240 BSAVE"CAS:AUTOMA",&HC000,&HC041
250 END
260 REM
270 REM DATA's para L.M.
280 REM
290 DATA F3,FD,21,B0,FB,FD,36,00
300 DATA 00,FD,36,01,01,21,34,C0
310 DATA 11,F0,FB,01,0D,00,ED,B0
320 DATA 21,F0,FB,22,FA,F3,11,0E
330 DATA 00,19,22,F8,F3,21,2B,C0
340 DATA CD,3F,70,22,4D,41,54,52
350 DATA 49,58,22,00,43,4F,4C,4F
360 DATA 52,37,2C,31,3A,52,55,4E
370 DATA 0D,49,00,00,00,00,00,00

```

```

3740
3750
3760
3770
3780
3790
3800
3810
3820
3830
3840
3850
3860
3870
3880
3890
3900
3910
3920
3930
3940
3950
3960
3970
3980
3990

```



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6.
 Programação Avançada em MSX - capítulo 4.

4.2 - GRAVANDO PROGRAMAS BASIC EM BINÁRIO

Quando dispomos apenas do gravador cassete para armazenar dados de um programa, o processo de gravação e leitura se torna lento e pouco confiável. Uma forma de torná-lo mais prático é armazenando o programa e seus dados conjuntamente em formato binário. Para isso devemos dispor de um pequeno programa em Linguagem de Máquina que deve preparar a memória do micro para poder gravá-la sem perder os dados.

Vamos exemplificar o que acabamos de descrever. Imagine que você queira armazenar a matriz gerada pelo programa da figura a seguir.

100 DIM A\$(5000,1)	280
120 FOR F=0 TO 5000	732
130 FOR G=0 TO 1	840
140 A\$(F,G)=STR\$(2*F+G^2)	1424
150 NEXT G	1690
160 NEXT F	1873
	TOTAL = 1873

Uma das maneiras de fazer isso seria abrir um arquivo no cassete e gravar cada um dos dados nesse arquivo. Esse processo é extremamente demorado e portanto mais sujeito a falhas. Se pudermos gravar em binário (com BSAVE) a região da memória que contém as variáveis, e até o próprio programa, economizaremos tempo e aumentaremos a confiabilidade dos dados.

A seguir vamos apresentar um programa em BASIC que, ao ser executado, gera em fita ou em disco um programa em Linguagem de Máquina com o nome "BASBIN" (em fita) ou "BASBIN.BIN" (em disco). Antes de prosseguir, digite-o e execute-o. Depois, verifique se o programa BASBIN foi realmente gerado.

100 REM	173
110 REM BINARIZADOR DE BASIC	800
120 REM	918
130 E=&HD000	954
140 READ A\$: A=VAL("&H"+A\$)	1394
150 IF A\$="XX" THEN 170	1430
160 POKE E,A : E=E+1 : GOTO 140	2487
170 E=&HD100	2825
180 READ A\$: A=VAL("&H"+A\$)	3183
190 IF A\$="XX" THEN 210	3225
200 POKE E,A : E=E+1 : GOTO 180	4305
210 E=&HD200	4630

```

220 READ A$ : A=VAL("&H"+A$)
230 IF A$="XX" THEN 250
240 POKE E,A : E=E+1 : GOTO 220
250 BSAVE"BASBIN.BIN",&HD000,&HD28C
260 END
270 REM DADOS EM &HD000
280 DATA F3,21,07,80,ED,4B,80,F3
290 DATA CD,4C,D0,ED,4B,77,F1,CD
300 DATA 4C,D0,ED,4B,6A,F1,CD,4C
310 DATA D0,ED,4B,68,F1,CD,4C,D0
320 DATA ED,4B,68,F1,CD,4C,D0,ED
330 DATA 4B,A0,F0,CD,4C,D0,ED,4B
340 DATA C6,F6,CD,4C,D0,ED,4B,C4
350 DATA F6,CD,4C,D0,ED,4B,C2,F6
360 DATA CD,4C,D0,ED,4B,76,F6,CD
370 DATA 4C,D0,FB,C9,71,23,70,23
380 DATA C9,4B,50,45,4C,41,20,41
390 DATA 52,4F,54,49,44,45,4F,54
400 DATA 41,4E,45,52,XX
410 REM DADOS EM &HD100
420 DATA F3,21,07,80,01,80,F3,CD
430 DATA 4D,D1,01,77,F1,CD,4D,D1
440 DATA 01,6A,F1,CD,4D,D1,01,68
450 DATA F1,CD,4D,D1,01,68,F1,CD
460 DATA 4D,D1,01,A0,F0,CD,4D,D1
470 DATA 01,C6,F6,CD,4D,D1,01,C4
480 DATA F6,CD,4D,D1,01,C2,F6,CD
490 DATA 4D,D1,01,76,F6,CD,4D,D1
500 DATA 21,55,D1,11,07,80,01,14
510 DATA 00,ED,80,FB,C9,7E,02,23
520 DATA 03,7E,02,23,C9,45,44,49
530 DATA 54,4F,52,41,20,41,4C,45
540 DATA 50,48,20,2D,20,31,39,38
550 DATA 37,XX
560 REM DADOS EM &HD200
570 DATA F3,21,07,80,01,80,F3,CD
580 DATA 65,D2,01,77,F1,CD,65,D2
590 DATA 01,6A,F1,CD,65,D2,01,68
600 DATA F1,CD,65,D2,01,68,F1,CD
610 DATA 65,D2,01,A0,F0,CD,65,D2
620 DATA 01,C6,F6,CD,65,D2,01,C4
630 DATA F6,CD,65,D2,01,C2,F6,CD
640 DATA 65,D2,01,76,F6,CD,65,D2
650 DATA 21,78,D2,11,07,80,01,14
660 DATA 00,ED,80,21,6D,D2,11,F0
670 DATA FB,01,0B,00,ED,80,21,F0
680 DATA FB,22,FA,F3,11,0C,00,19
690 DATA 22,F8,F3,FB,C9,7E,02,23
700 DATA 03,7E,02,23,C9,47,4F,54

```

```

5805
634E
724F
8186
8214
8701
8826
9240
980A
A705
BA54
C05F
E174
F5EF
258
789
00E
1842
1E82
282E
3908
4AFA
5F7C
7323
808A
86E3
80EA
9688
9FB7
B130
C133
D478
E87E
F237
88
672
AD7
125A
1344
2764
3A02
4C30
6209
7720
8238
88AB
9028
9A60
A805

```

```

710 DATA 4F,20,30,30,31,32,30,0D
720 DATA 45,44,49,54,4F,52,41,20
730 DATA 41,4C,45,50,48,20,2D,20
740 DATA 31,39,38,37,XX

```

```

8504
8505
8506
8507

```

Uma vez com o programa BASBIN gravado, vamos testá-lo, ao mesmo tempo em que aproveitamos para aprender como ele deve ser usado. Para isso, limpe a memória com um NEW e digite o programa a seguir exatamente como ele está listado, sem nenhum espaço a menos ou a mais!

```

100 REM EDITORA ALEPH - 1987
110 CLEAR 200,&HD000
120 X$="EDITORA ALEPH - 1987"
130 PLAY"T240S8M1000"
140 PLAY"02CEGAA+AGE"
150 PLAY"02CEGAA+AGE"
160 PLAY"FA03CDD+DC02A"
170 PLAY"02CEGAA+AGE"
180 PLAY"02DEFF+GFED"
190 SCREEN 0:PRINT ";;;X$"
200 GOTO 140

```

```

8508
8509
8510
8511
8512
8513
8514
8515
8516
8517
8518
8519
8520

```

TOTAL = 4085

Após tê-lo digitado, execute-o, espere alguns segundos e pressione CONTROL+STOP para interrompê-lo. Vamos agora preparar a memória para poder ser gravada em binário. Digite o seguinte comando:

```
BLOAD"BASBIN.BIN",R
```

A seguir, vamos gravar a memória com o programa e com os dados. Para isso, comande:

```
BSAVE"TESTES.BIN",&H8000,&HD300,&HD200
```

Com isso o programa em BASIC e seus dados serão gravados em formato binário.

Para tornar a carregá-lo, basta comandar:

```
BLOAD"TESTES.BIN
```

E a seguir, comandar:

```
DEFUSR=&HD100:?USR(0)
```

Experimente fazer isso e depois comande:

PRINT X\$

Você verá que o conteúdo de X\$ ainda está presente.

Se tivéssemos comandado:

```
BLOAD"BASBIN.BIN",R
```

O programa seria carregado e começaria a rodar automaticamente a partir da linha 120. Essa é uma outra forma de fazer programas em BASIC se auto-executarem logo após a carga a partir de fitas cassete.

Você pode usar o BASBIN.BIN com qualquer programa em BASIC, desde que ele não ocupe a memória acima de &HD000. O programa em BASIC deverá começar sempre com as duas primeiras como mostradas a seguir:

```
100 REM EDITORA ALEPH - 1987  
110 CLEAR 200,&HD000
```

A rigor, o primeiro parâmetro do CLEAR da linha 110 pode ser alterado, mas o segundo deve ser necessariamente menor ou igual a &HD000 !



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

4.3 -GRAVANDO TEXTOS

Este programinha transfere uma mensagem (em ASCII) para a fita cassete. Após a chamada, ele transfere para a fita os dados definidos após &HC016 até o primeiro byte &H00, retornando então ao interpretador BASIC.

```
1000 DATA CD,EA,00,21,16,C0,7E,A7      603
1010 DATA 28,08,E5,CD,ED,00,E1,23      01E
1020 DATA 18,F4,CD,F0,00,C9            1230
1025 REM a seguir está a mensagem       1924
1030 DATA 4F,53,20,43,59,4C,4F,4E      2328
1040 DATA 49,4F,53,20,49,4E,56,41      3405
1050 DATA 44,49,52,41,4F,20,41,20     48E2
1060 DATA 54,45,52,52,41,20,21,00     5490
1070 CLS:FOR I=&HC000 TO &HC035         6537
1080 READ A$:POKE I,VAL("&H"+A$):NEXT I: 7838
END
```

TOTAL = 7838



BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

4.4 - LENDO TEXTOS

Este programa faz a função oposta à da dica 4.3, lê a fita e transfere para a tela a informação lida. Devido à simplicidade deste sistema, caracteres de controle (CR, LF, etc) não são reconhecidos como tais e aparecem na tela como símbolos gráficos. A leitura termina na primeira ocorrência do byte &H00.

Para posicionar a informação na tela use os comandos

```
LOCATE C,L:PRINT ;
```

onde C e L são as coordenadas de impressão desejada.

```
1000 DATA CD,E1,00,CD,E4,00,A7,28
1010 DATA 04,D3,98,18,F6,CD,E7,00
1020 DATA C9,FIM
1030 CLS:FOR I=&HC100 TO &HC110:READ A$:
POKE I,VAL("&H"+A$):NEXT I:END
```

```
000
001
002
003
```

```
100000 = 224H
```



BIBLIOGRAFIA RECOMENDADA.

Programação Avançada em MSX - capítulo 4.

4.5 - VERIFICANDO GRAVAÇÕES

Quantas vezes você já não deu socos no seu gravador por causa de um programa que não carregou? Para programas em BASIC ainda existe o comando "CLOAD?", mas para programas em binário nada foi implementado no BASIC MSX visando a verificação de gravações.

O programa a seguir instala na RAM uma sub-rotina em L.M. que faz a função de verificar um arquivo gravado no formato binário. Eis o programa:

1000	REM	-----	6A8
1010	REM	VERIFICADOR DE BSAVE V. 1.0	10F5
1020	REM	(C) 1987/88 BY THE PILOT	19F0
1030	REM	-----	2447
1040	DATA	CD,6C,00,AF,D3,99,D3,99	3684
1050	DATA	CD,E1,00,38,76,06,0A,C5	4758
1060	DATA	CD,E4,00,C1,10,F9,FE,D0	5AE8
1070	DATA	20,EE,06,06,C5,CD,E4,00	70AB
1080	DATA	38,61,D3,98,C1,10,F5,3E	84FD
1090	DATA	3E,D3,98,D3,98,CD,E1,00	8AE8
1100	DATA	21,57,F8,06,06,E5,C5,CD	8F41
1110	DATA	E4,00,C1,E1,77,23,10,F5	9886
1120	DATA	3E,2C,2A,57,F8,CD,A9,D0	A0FD
1130	DATA	D3,98,2A,59,F8,CD,A9,D0	A038
1140	DATA	D3,98,2A,5B,F8,CD,A9,D0	C031
1150	DATA	3E,20,06,12,D3,98,10,FC	D0DB
1160	DATA	2A,57,F8,ED,5B,59,F8,13	E620
1170	DATA	AF,32,5D,F8,E5,D5,CD,E4	F908
1180	DATA	00,D1,E1,38,0E,BE,C4,A2	574
1190	DATA	D0,23,A7,E5,ED,52,E1,20	865
1200	DATA	EB,18,05,3E,01,32,5D,F8	1005
1210	DATA	21,10,00,CD,C6,00,CD,E7	181F
1220	DATA	00,3A,5D,F8,A7,C0,21,FC	2387
1230	DATA	D0,7E,A7,C8,23,CD,A2,00	36AC
1240	DATA	18,F7,CD,A9,D0,CD,B4,D0	4884
1250	DATA	C9,F5,7C,CD,C7,D0,7D,CD	5885
1260	DATA	C7,D0,F1,C9,F5,3E,2D,D3	6F08
1270	DATA	98,F1,CD,C7,D0,3E,20,D3	8248
1280	DATA	98,3E,01,32,5D,F8,C9,F5	87E8
1290	DATA	CB,3F,CB,3F,CB,3F,CB,3F	8E7D
1300	DATA	01,EC,D0,81,4F,3E,00,88	96AE
1310	DATA	47,0A,D3,98,F1,E6,0F,01	9785
1320	DATA	EC,D0,81,4F,3E,00,88,47	AC6E
1330	DATA	0A,D3,98,C9,30,31,32,33	BE13
1340	DATA	34,35,36,37,38,39,41,42	CC22
1350	DATA	43,44,45,46,0D,0A,53,45	E424
1360	DATA	4D,20,45,52,52,4F,53,0D	F948

```

1370 DATA 0A,00,FIM
1380 CLS:PRINT "CARREGANDO ROTINA V-BSAV
E"
1390 FOR I=&HD000 TO &HD109:READ A$:POKE
I,VAL("&H"+A$):NEXT I:PRINT "ROTINA CAR
REGADA.":END

```

Uso: após salvar o seu programa em binário, rebobine a fita e comande:

```
DEFUSR=&HD000:#?USR(0)
```

Naturalmente, o seu programa não pode estar entre as posições &HD000 e &HD110 (por que?). A qualquer instante a verificação pode ser interrompida por CONTROL+STOP.



BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.

4.6 - GRAVADOR DE SCREEN 2

Desta vez é um programinha que guarda a tela gráfica (SCREEN 2) em fita cassete, deixando de ser privilégio dos possuidores de disk drives essa operação.

Após carregar a rotina em L.M., basta chamar a rotina desejada pela funçãoUSR. Naturalmente, a função deve ser pré-programada, pois o computador apaga a SCREEN 2 no modo de comando.

```
1000 REM -----
1010 REM COPIADOR SCREEN 2<-->FITA
1020 REM JANEIRO 1988 - THE PILOT
1030 REM -----
1040 DATA F3,D8,98,AF,D3,99,D3,99
1050 DATA CD,EA,00,D8,CD,ED,00,CD
1060 DATA ED,00,CD,ED,00,01,00,38
1070 DATA DB,98,C5,CD,ED,00,C1,D8
1080 DATA 0B,78,B1,20,F3,CD,F0,00
1090 DATA C9,21,00,20,01,00,18,3E
1100 DATA 1F,CD,56,00,F3,DB,98,AF
1110 DATA D3,99,D3,99,CD,E1,00,D8
1120 DATA CD,E4,00,CD,E4,00,CD,E4
1130 DATA 00,CD,E4,00,01,00,38,C5
1140 DATA CD,E4,00,C1,D8,D3,98,0B
1150 DATA 78,B1,20,F3,CD,E7,00,C9
1160 DATA FIM
1170 CLS:FOR I=&HD000 TO &HD05F:READ A$:
POKE I,VAL("&H"+A$):NEXT
1180 PRINT "SCREEN 2<-->FITA":PRINT:PRIN
T "SALVAR: &HD000":PRINT"LER: &HD029"
:PRINT:END
```

```
645
1030
1129
2158
3040
4350
5676
6871
8031
8878
8826
9479
904E
A5EE
B89D
C9C5
CE6C
E748
```

TOTAL = 3A5

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 4.



DICAS PARA A IMPRESSORA

O acesso à impressora nos micros MSX pode ser realizado de várias maneiras diferentes, sendo que o usuário pode interferir em algumas delas.

As dicas deste capítulo desvendam aos usuários comuns recursos disponíveis somente através da Linguagem de Máquina ou de programas em BASIC bem construídos.

5.1 - Eco na impressora	118
5.2 - Transformando PRINT em LPRINT	119
5.3 - Impressão dupla	120
5.4 - HEXA-PRINTER e HEXA-SCREEN	122
5.5 - Filtro genérico	123
5.6 - Impedindo o uso da impressora	126
5.7 - Cópia gráfica	127
5.8 - Caracteres deitados	130
5.9 - Strings em modo gráfico	131
5.A - Máquina de escrever	132
5.B - Impressor de programas	133

5.1 - ECO NA IMPRESSORA

O MSX possui duas rotinas do BIOS para o envio de dados à impressora. Uma delas, a LPTOUT, pode ser facilmente usada para reproduzir o que for enviado para a tela diretamente na impressora. Uma aplicação desse tipo de recurso pode ser facilmente entendida se pensarmos em programas que apresentam resultados apenas na tela. Para fazê-los enviar os resultados para a impressora teríamos normalmente que alterá-los por inteiro. Obviamente, a maneira mais fácil é usar o programa apresentado a seguir.

```
100 CLEAR 200,&HE000
110 FOR F=&HE000 TO &HE0B1
120 READ A$: POKE F,VAL("&H"+A$)
130 NEXT F : DEFUSR0=&HE000
140 POKE 0,USR0(0) : END
150 REM
160 REM DADOS
170 REM
1000 DATA FD,21,A4,FD,FD,36,00,C3
1010 DATA FD,36,01,11,FD,36,02,E0
1020 DATA C9,FE,7A,30,31,47,3A,80
1030 DATA E0,FE,01,78,28,3F,47,3A
1040 DATA 80,E0,FE,02,78,28,19,47
1050 DATA 3A,80,E0,FE,03,78,28,49
1060 DATA 47,3A,80,E0,FE,04,78,28
1070 DATA 07,FE,20,38,09,CD,A5,00
1080 DATA 21,80,E0,36,00,C9,FE,1B
1090 DATA 28,0C,FE,0D,28,EF,FE,0A
1100 DATA 28,EB,3E,20,18,E7,21,80
1110 DATA E0,36,01,18,E8,FE,59,28
1120 DATA 0A,FE,4E,28,0D,FE,4F,28
1130 DATA 09,18,D5,21,80,E0,36,03
1140 DATA 18,D3,21,80,E0,36,02,18
1150 DATA CC,21,80,E0,36,04,18,C5
1160 DATA 00,00,52,45,4E,41,54,4F
```

816
991
1065
105A
1090
1192
22A0
2405
2002
3E55
4192
63AA
7A08
822E
93A3
9700
9F40
A787
B868
C558
D054
E5E7
45A
FE0
140A

TOTAL = 140A

Após executá-lo, pode-se apagá-lo da memória com o comando NEW. Entretanto é conveniente salvá-lo previamente em disco ou em fita.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6 e página 159.
Programação Avançada em MSX - capítulo 3 e página 149.

5.2 - TRANSFORMANDO PRINT EM LPRINT

O pequeno programa apresentado a seguir deve ser inserido no final de programas maiores. Para acioná-lo, deve-se comandar:

```
RUN 65000 (e RETURN)
```

Ao ser executado ele "varre" o programa principal a procura de instruções PRINT e as substitui por instruções LPRINT. Isso pode ser muito útil para redirecionar a saída de dados da tela para a impressora, porém o programa não deve ter instruções PRINT #, pois serão transformadas em LPRINT #, ocasionando erros de sintaxe. O programa também não deverá conter números cujo código compactado seja igual a token do PRINT (145), pois nesse caso eles terão seus valores alterados (observe como procedemos para evitar isso na linha 65010 do programa, ao invés de 145, escrevemos 100+45!).

```
65000 REM 274
65001 REM Muda PRINT p/ LPRINT 853
65002 REM DBA
65003 EI = 32769! 12F0
65004 B1 = PEEK(EI) : B2 = PEEK(EI+1) 115A
65005 B3 = PEEK(EI+2) : B4 = PEEK(EI+3) 28F7
65006 PL = B1 + 256*B2 328A
65007 NL = B3 + 256*B4 3046
65008 IF B1=0 AND B2=0 THEN END 4690
65009 FOR F=EI+4 TO PL-2 5059
65010 IF PEEK(F)<>(100+45) THEN 65012 507F
65011 POKE F,157 66FE
65012 NEXT F 6930
65013 EI=PL 6039
65014 GOTO 65004 6E98
```

```
TOTAL = 6E98
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 2.

5.3 - IMPRESSÃO DUPLA

O programa apresentado a seguir gera uma rotina em Linguagem de Máquina a partir do endereço &HE000 capaz de fazer com que a impressão de listagens e outros dados seja feita com dupla passagem da cabeça de impressão. Com isso os textos ficarão mais legíveis e com maior contraste.

Após digitar e gravar o programa, você pode apagá-lo da memória com o comando NEW.

Caso você deseje desativar a dupla impressão, basta comandar:

```
POKE &HFFB6,&HC9
```

Para ativá-la novamente, uma vez que tenha sido desativada, basta comandar:

```
POKE &HFFB6,&HC3
```

Note que o programa em BASIC, ao ser rodado pede a quantidade de caracteres a serem impressos em cada linha. Uma vez especificado esse parâmetro, sempre que a dupla impressão estiver ativa, as linhas serão impressas com essa largura.

A rotina em Linguagem de Máquina não funcionará como esperado se o número de colunas especificado for maior que o número de colunas da impressora.

```
100 SCREEN 0 : CLEAR 200,&HE000
110 PRINT : PRINT : PRINT
120 PRINT" Caracteres por linha ";
130 INPUT C
140 IF C<1 OR C>255 THEN RUN
150 FOR F=&HE000 TO &HE06F
160 READ A$:POKE F,VAL("&H"+A$)
170 NEXT F
180 FOR F=&HE070 TO &HE173
190 POKE F,0
200 NEXT F
210 POKE &HE02A,C
220 DEFUSR=&HE000 : POKE 0,USR(0)
230 END
240 DATA F3,21,0E,E0,22,B7,FF,3E
250 DATA C3,32,B6,FF,FB,C9,F5,C5
260 DATA D5,E5,FE,0D,28,52,FE,0A
270 DATA 28,13,ED,4B,6F,E1,21,70
280 DATA E0,09,03,ED,43,6F,E1,77
290 DATA 79,FE,FF,20,3B,3E,C9,32
```

```
300 DATA B6,FF,CD,50,E0,CD,50,E0
310 DATA ED,4B,6F,E1,78,B1,3E,0A
320 DATA C4,A5,00,21,00,00,22,6F
330 DATA E1,3E,C3,32,B6,FF,18,18
340 DATA ED,4B,6F,E1,78,B1,C8,21
350 DATA 70,E0,7E,CD,A5,00,23,0D
360 DATA 20,F8,3E,0D,CD,A5,00,C9
370 DATA E1,D1,C1,F1,33,33,B7,C9
```

```
0200
0200
0200
0200
0200
0200
0200
0200
```

```
TOTAL = 0EE
```



BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - capítulo 3.

5.4 - HEXA-PRINTER E HEXA-SCREEN

Muitas vezes precisamos saber quais caracteres (normalmente de controle) um dado programa (normalmente em Linguagem de Máquina) está enviando para a impressora. Nesses casos, seria muito cômodo se ao invés de imprimir os caracteres propriamente ditos pudessémos imprimir seus códigos hexadecimais. É exatamente isso que o programa listado abaixo faz, com uma opção ainda mais útil para quem não possui impressora: a possibilidade de simular na tela do vídeo a saída de impressora. Digite e grave o programa. Depois rode-o e use algumas vezes o comando LPRINT.

100 FOR F=&HD000 TO &HD071	5A8
110 READ A\$:POKE F,VAL("&H"+A\$)	0B8
120 NEXT F	E52
130 SCREEN 0:LOCATE 0,7,0:KEY OFF	13E2
140 PRINT SPC(10);"[1] HEXAPRINTER"?"	1B38
150 PRINT SPC(10);"[2] HEXASCREEN" "	2304
160 A\$=INKEY\$	2608
170 IF A\$="1" THEN 260	3002
180 IF A\$(">"2" THEN 160	3916
190 POKE &HD03B,&HA2	3FA4
200 POKE &HD04A,&HA2	46A4
210 POKE &HD04F,&HA2	4E40
220 POKE &HD058,&HC	56E8
230 POKE &HD05C,&HA2	5F9F
240 POKE &HD061,&HA2	6888
250 POKE &HD070,&HC	7478
260 DEFUSR=&HD000 : S=USR(0)	79E8
270 SCREEN 0 : NEW	701E
280 DATA F3,DD,21,B6,FF,DD,36,00	8400
290 DATA C3,DD,36,01,13,DD,36,02	87E8
300 DATA D0,FB,C9,E5,D5,C5,F5,FE	9060
310 DATA 0A,20,05,21,70,D0,36,01	9928
320 DATA 21,B6,FF,36,C9,47,3E,F0	A808
330 DATA A0,CB,3F,CB,3F,CB,3F,CB	B760
340 DATA 3F,C6,30,FE,3A,38,02,C6	C620
350 DATA 07,C5,CD,A5,00,C1,3E,0F	E858
360 DATA A0,C6,30,FE,3A,38,02,C6	F498
370 DATA 07,CD,A5,00,3E,20,CD,A5	F4
380 DATA 00,21,70,D0,35,20,0C,36	448
390 DATA 10,3E,0D,CD,A5,00,3E,0A	090
400 DATA CD,A5,00,21,B6,FF,36,C3	1574
410 DATA F1,C1,D1,E1,33,33,B7,C9	2828
420 DATA 10,52,52,45,4E,41,54,4F	340F

5.5 - FILTRO GENÉRICO

O programa apresentado a seguir permite compatibilizar os caracteres acentuados de seu MSX (Expert 1.1 ou Hotbit) com a sua impressora, desde que ela os possua.

Digite e grave o programa a seguir e depois execute-o comandando:

RUN

Ligue a impressora e pressione a tecla RETURN.

Serão impressos os caracteres correspondentes aos códigos de 128 (&H80) a 255 (&HFF) e seus respectivos códigos em hexadecimal. Não se assuste se durante essa impressão sua impressora "agir" de forma estranha pois pode ser que alguns dos caracteres enviados para ela correspondam a alguns de seus controles, como avanço de linha, beep, etc.

Quando a impressão terminar veja o resultado e pressione RETURN novamente. Serão listadas na tela as linhas de 620 a 710 do programa para que você as altere conforme o resultado obtido na listagem da impressora.

Procure cada caractere das linhas DATA na lista impressa e substitua o código 20 (da listagem original) pelo código que foi impresso.

Feitas todas as alterações (não se esqueça de pressionar RETURN após cada uma delas!), comande:

GOTO 360

Com isso será gravado um arquivo de nome "FILTRO.BIN". Ele é seu programa "filtro" e para rodá-lo, basta comandar:

BLOAD "FILTRO.BIN",R

Utilize-o toda vez que você pretender imprimir um texto com acentuação.

```
100 POKE &HF417,1:SCREEN 0
110 WIDTH 38:KEYOFF
120 LOCATE 0,10:PRINT" PREPARE SUA";
130 PRINT" IMPRESSORA E QUANDO ES-";
140 PRINT" TIVER PRONTO APERTE A TECLA";
150 PRINT" <RETURN>."
160 A$=INPUT$(1)
170 IF ASC(A$)<>13 THEN 160
```

```
098
105
117
1192
2017
2550
2840
3550
```

180 FOR F= 128 TO 255	8376
190 LPRINT CHR\$(F);" = ";HEX\$(F)	8381
200 NEXT	8386
210 CLS	8391
220 LOCATE0,7:PRINT" VEJA O QUE SAIU";	8396
230 PRINT" NA IMPRESSORA E "	8401
240 PRINT" ALTERE AS LINHAS DATA."	8406
250 PRINT:PRINT" DEPOIS DE ALTERADAS";	8411
260 PRINT", COMANDE : "	8416
270 PRINT"GOTO 300":PRINT	8421
280 PRINT" APERTE A TECLA <RETURN>";	8426
290 PRINT" PARA COMEÇAR"	8431
300 A\$=INPUT\$(1)	8436
310 IF ASC(A\$)<>13 THEN 300	8441
320 CLS:PRINT " NÃO SE ESQUEÇA DO";	8446
330 PRINT " <RETURN> APÓS AS ALTER";	8451
340 PRINT "AÇÕES E DE COMANDAR GOTO ";	8456
350 PRINT "360.":PRINT:LIST 620-710:END	8461
360 CLS	8466
370 LOCATE 0,10:PRINT"PREPARE O DISCO";	8471
380 PRINT" PARA GRAVAR O PROGRAMA"	8476
390 PRINT"FILTRO E TECLE RETURN ";	8481
400 PRINT" QUANDO PRONTO"	8486
410 A\$=INPUT\$(1)	8491
420 IF ASC(A\$)<>13 THEN 410	8496
430 RESTORE 620	8501
440 FOR L=0 TO 55	8506
450 READ Z\$	8511
460 Z\$="&H"+RIGHT\$(Z\$,2)	8516
470 POKE &HD026+L,VAL(Z\$)	8521
480 NEXT	8526
490 RESTORE 720	8531
500 FOR E=&HD000 TO &HD025	8536
510 READ Z\$	8541
520 POKE E,VAL("&H"+Z\$)	8546
530 NEXT	8551
540 BSAVE"FILTRO.BIN",&HD000,&HD05D	8556
550 CLS	8561
560 LOCATE0,8:PRINT" O PROGRAMA";	8566
570 PRINT" ESTÁ GRAVADO.":PRINT	8571
580 PRINT" PARA RODÁ-LO, COMANDE:"	8576
590 PRINT:PRINT" BLOAD";CHR\$(34);	8581
600 PRINT"FILTRO.BIN";CHR\$(34);",R"	8586
610 PRINT:PRINT:END	8591
620 DATA Ç=20,ü=20,é=20,â=20,À=20,à=20	8596
630 DATA `=20,ç=20,ê=20,f=20,ó=20,ú=20	8601
640 DATA Æ=20,ê=20,ô=20,À=20,é=20,æ=20	8606
650 DATA Æ=20,ô=20,ö=20,ò=20,û=20,ù=20	8611
660 DATA ý=20,ö=20,ü=20,ç=20,œ=20,¥=20	8616

```

670 DATA C=20, f=20, á=20, í=20, ó=20, ú=20
680 DATA ã=20, ñ=20, ò=20, ô=20, ç=20, r=20
690 DATA ı=20, ı̇=20, ı̈=20, i=20, «=20, »=20
700 DATA ž=20, š=20, ſ=20, ŷ=20, ð=20, ò=20
710 DATA ů=20, ů=20
720 DATA 21, B6, FF, 3E, C3, 77, 23, 11
730 DATA 13, D0, 73, 23, 72, 21, 17, F4
740 DATA 3E, FF, 77, FE, 80, D8, FE, B8
750 DATA D0, E5, D5, 21, A6, CF, 16, 00
760 DATA 5F, 19, 7E, D1, E1, C9

```

```

1000
2000
3000
4000
5000
6000
7000
8000
9000

```

TOTAL = 808E

Como exemplo, apresentamos a seguir as linhas DATA de 620 a 710 preenchidas para compatibilizar um Expert 1.1 com uma impressora Mônica El6030.

```

620 DATA C=A6, u=D9, é=C8, á=C3, A=A2, à=C1
630 DATA =BB, c=C6, ê=C9, f=AC, ó=D1, ú=B7
640 DATA Á=A3, Ê=A9, Ò=B2, À=A1, É=A8, Æ=D5
650 DATA Ĥ=E5, ð=D2, ò=D4, ô=D0, ů=D8, ù=D6
660 DATA ŷ=DA, o=D4, ú=B9, ç=BF, ž=BC, ž=BE
670 DATA C=20, f=20, á=C2, í=CC, ó=D1, ú=D7
680 DATA ã=20, ñ=20, ò=DC, ô=DD, ç=DE, r=20
690 DATA ı=20, ı̇=20, ı̈=20, i=20, «=20, »=20
700 DATA ž=A4, š=C4, ŷ=20, ŷ=20, ð=D3, ò=D3
710 DATA ů=20, ů=20

```

5.6 - IMPEDINDO O USO DA IMPRESSORA

Muitos programadores tentam proteger ao menos a originalidade de seus programas BASIC dos milhares de piratas amadores que farteiam por este país simplesmente desativando a listagem do mesmo na tela. Ledo engano o daqueles que pensam que isso é eficaz! Basta comandar LLIST e a listagem será enviada para a impressora. Se, entretanto, o programa ao ser carregado, desativar o uso da impressora o problema estará resolvido. Para isso, basta inserir o código &HC3 na posição de memória &HFFB6. Experimente rodar o programinha exemplo listado abaixo e depois tente enviá-lo para a impressora.

```
10 POKE &HFFB6,&HC3  
20 REM Tente me listar numa impressora !  
30 PRINT "Comande LLIST !"  
40 END
```

TOTAL = 1635

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - página 149.

5.7 - C6PIA GRÁFICA

O programa listado a seguir imprime uma cópia fiel da tela presente no vídeo do micro mediante o simples pressionamento da tecla ESC.

Digite-o e grave-o. Depois, execute-o. Com isso a rotina em Linguagem de Máquina estará pronta para ser usada na memória do micro.

100 FOR F=&HE000 TO &HE25F	598
110 READ A% : POKE F,VAL("&H"+A%)	0AF
120 NEXT F	E49
130 DEFUSR=&HE000 : S=USR(0)	11593
140 SCREEN 0 : NEW	1150A
150 DATA 3A,CC,FD,FE,C9,C0,21,12	2283
160 DATA E0,22,CD,FD,3E,CD,32,CC	2301
170 DATA FD,C9,FE,3A,C0,F5,C5,D5	3078
180 DATA E5,ED,73,59,E2,0E,00,3A	4101
190 DATA AF,FC,B7,21,F0,00,11,28	6488
200 DATA 06,28,06,21,00,01,11,20	7514
210 DATA 08,3E,1B,CD,97,E0,3E,4B	800A
220 DATA CD,97,E0,7D,CD,97,E0,7C	8377
230 DATA CD,97,E0,06,00,CD,A1,E0	8380
240 DATA D5,C5,21,5B,E2,42,11,08	8410
250 DATA 00,C5,E5,06,08,7E,FE,08	8466
260 DATA 3F,CB,11,19,10,F7,79,CD	8820
270 DATA 97,E0,E1,C1,23,10,EA,C1	0073
280 DATA D1,04,78,BB,20,D7,3E,0D	10001
290 DATA CD,97,E0,3E,1B,CD,97,E0	1422
300 DATA 3E,41,CD,97,E0,3E,08,CD	085
310 DATA 97,E0,3E,0A,CD,97,E0,3E	1610
320 DATA 18,CD,97,E0,0C,79,FE,18	1880
330 DATA 20,8D,E1,D1,C1,F1,C9,CD	11A1
340 DATA A5,00,D0,ED,7B,59,E2,18	219E
350 DATA F1,C5,D5,E5,FD,E5,21,5B	2174
360 DATA E2,3E,40,36,00,23,3D,20	4077
370 DATA FA,3A,AF,FC,B7,F5,C5,C4	5309
380 DATA 73,E1,C1,69,26,00,29,29	6973
390 DATA 29,5D,54,29,29,F1,F5,20	770E
400 DATA 01,19,58,19,EB,D6,02,79	9122
410 DATA 01,00,00,2A,24,F9,E5,2A	974E
420 DATA 22,F9,38,19,20,0C,2A,CB	9016
430 DATA F3,E3,2A,C7,F3,E6,18,47	A584
440 DATA 18,0B,2A,D5,F3,E3,2A,D1	A835
450 DATA F3,07,E6,06,4F,19,CD,4A	80E4
460 DATA 00,6F,26,00,29,29,29,09	0059
470 DATA EB,FD,E1,FD,19,2A,C9,F3	E08A
480 DATA 19,0F,0F,0F,E6,1F,4F,06	F580
490 DATA 00,3A,E6,F3,57,E6,0F,5F	9CF

500	DATA	F1,E5,3D,20,08,2A,BF,F3	1E20
510	DATA	09,CD,4A,00,57,21,5B,E2	1E48
520	DATA	06,08,FD,E5,E3,CD,4A,00	1E74
530	DATA	4F,E1,FD,23,3A,AF,FC,D6	204A
540	DATA	02,38,15,28,0C,51,0E,F0	303F
550	DATA	78,FE,05,28,0B,FD,2B,1B	4294
560	DATA	07,E3,CD,4A,00,57,23,E3	5384
570	DATA	C5,06,08,CB,11,34,35,20	6885
580	DATA	0D,7A,30,04,0F,0F,0F,0F	70E4
590	DATA	E6,0F,20,01,7B,77,23,10	8E94
600	DATA	EA,C1,10,BE,E1,FD,E1,E1	9E88
610	DATA	D1,C1,C9,78,07,07,07,C6	9925
620	DATA	07,47,79,07,07,07,C6,07	AA58
630	DATA	4F,AF,CD,87,00,57,CD,4A	AB66
640	DATA	00,FE,D0,CB,D5,C5,CD,99	98A0
650	DATA	E1,C1,F1,3C,FE,20,20,EA	0D51
660	DATA	C9,91,2F,FE,27,D0,4F,23	0E06
670	DATA	CD,4A,00,5F,78,93,5F,9F	1479
680	DATA	57,23,CD,4A,00,47,23,CD	618
690	DATA	4A,00,CB,7F,28,05,21,20	1110
700	DATA	00,19,EB,14,15,C0,E6,0F	1169
710	DATA	C8,57,3A,E0,F3,CB,4F,0F	1100
720	DATA	3E,08,30,01,87,28,05,CB	27AF
730	DATA	80,CB,88,87,6F,C6,06,B9	30E6
740	DATA	D8,BB,D8,79,D6,07,4F,7D	4324
750	DATA	26,08,38,08,91,FE,09,38	5200
760	DATA	02,3E,08,67,7B,D6,07,5F	6791
770	DATA	7D,2E,08,38,08,93,FE,09	7074
780	DATA	38,02,3E,08,6F,FD,21,5B	8E16
790	DATA	E2,D5,CB,79,20,48,E5,FD	9417
800	DATA	E5,CB,7B,20,38,FD,7E,00	9809
810	DATA	B7,20,32,C5,D5,E5,3A,E0	A219
820	DATA	F3,0F,30,04,CB,39,CB,3B	AA8F
830	DATA	C8,5B,28,04,CB,9B,CB,E1	B984
840	DATA	68,26,00,44,29,29,29,09	0A58
850	DATA	ED,4B,26,F9,09,CD,4A,00	1048
860	DATA	1C,07,1D,20,FC,30,03,FD	1346
870	DATA	72,00,E1,D1,C1,FD,23,1C	7BF
880	DATA	2D,20,BE,FD,E1,E1,11,08	1026
890	DATA	00,FD,19,D1,0C,25,20,A9	1514
900	DATA	C9,3E,41,4C,45,50,48,C9	108F

TOTAL = 108F

Agora, teste a rotina executando o programinha a seguir e pressionando a tecla ESC.

10	SCREEN	2	100
20	FOR	F=80 TO 1 STEP -10	6F2

```

30 CIRCLE (128,80),80,15,,,80/F
40 CIRCLE (128,80),80,15,,,F/80
50 NEXT F
60 LINE (128,160)-(128,0)
70 LINE (48,80)-(208,80)
80 GOTO 80

```

```

007
13A5
15AC
1118
263D
2945

```

TOTAL = 2945

O programa copia para a impressora todas a regiões da tela (qualquer uma das SCREEN's) cuja cor possua um código superior a 7. inclusive os SPRITES.



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 6.
 Programação Avançada em MSX - capítulo 3.

5.8 - CARACTERES DEITADOS

O programa listado adiante gera uma rotina em L.M. que imprime uma string em modo gráfico, porém gira, previamente, seus caracteres de 90°.

Isso pode ser de grande utilidade para produzir mensagens que devem ser lidas "em pé", de cima para baixo.

A string a ser impressa deve ser passada como parâmetro na funçãoUSR.

```
10 '*****
20 '* BY THE DOCTOR LUZ *
30 '*****
40 FOR L=&HC000 TO &HC08A
50 READ A$:POKE L,VAL("&H"+A$)
60 NEXT
70 DEF USR=&HC000
80 PRINT"RODAR PROGRAMA ?"
90 A$=INPUT$(1)
100 IF A$="S" THEN GOTO 200 ELSE END
200 PRINT"DIGITE A STRING A SER IMPRESSA"
"
210 INPUT A$
220 A$=USR(A$)
230 GOTO 80
1000 DATA 2A,F8,F7,46,05,CD,3F,C0
1010 DATA 23,7E,5F,23,7E,57,EB,E5
1020 DATA E1,23,E5,7E,C5,21,BF,1B
1030 DATA 5F,16,00,CB,23,CB,12,CB
1040 DATA 23,CB,12,CB,23,CB,12,AF
1050 DATA ED,5A,EB,CD,61,C0,06,08
1060 DATA 21,00,C2,7E,CD,A5,00,23
1070 DATA 10,F9,C1,10,D3,E1,C9,3E
1080 DATA 1B,CD,A5,00,3E,4B,CD,A5
1090 DATA 00,58,16,00,CB,23,CB,12
1100 DATA CB,23,CB,12,CB,23,CB,12
1110 DATA 7B,CD,A5,00,7A,CD,A5,00
1120 DATA C9,06,08,C5,21,00,C2,06
1130 DATA 08,1A,17,CB,16,23,10,FA
1140 DATA C1,13,10,EF,C9,CD,ED,00
1150 DATA D1,E1,38,07,23,E7,20,F2
1160 DATA C3,F0,00,21,30,71,C3,1A
1170 DATA 70,7D,C5
```

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 140 a 143.

5.9 - STRINGS EM MODO GRÁFICO

O programa a seguir é quase igual ao da dica 5.9, porém a impressão da string é feita sem que os caracteres seja girados.

Lembre-se que tanto esta dica quanto a anterior só funcionarão se a impressora conectada ao micro entrar em modo gráfico segundo o padrão EPSON e tiver 9 agulhas em sua cabeça de impressão.

```
10 '*****
20 '* BY THE DOCTOR LUZ *
30 '*****
40 FOR L=&HC000 TO &HC072
50 READ A$:POKE L,VAL("&H"+A$)
60 NEXT
70 DEFUSR=&HC000
80 PRINT"RODAR PROGRAMA ?"
90 A$=INPUT$(1)
100 IF A$="S" THEN GOTO 200 ELSE END
200 PRINT"DIGITE A STRING A SER IMPRESSA"
"
210 INPUT A$
220 A$=USR(A$)
230 GOTO 80
5000 DATA 2A,F8,F7,46,05,CD,3C,C0
5010 DATA 23,7E,5F,23,7E,57,68,26
5020 DATA 00,ED,5A,E5,E1,2B,E5,7E
5030 DATA C5,21,C6,1B,5F,16,00,CB
5040 DATA 23,CB,12,CB,23,CB,12,CB
5050 DATA 23,CB,12,AF,ED,5A,06,08
5060 DATA 7E,CD,A5,00,2B,10,F9,C1
5070 DATA 10,DA,E1,C9,3E,1B,CD,A5
5080 DATA 00,3E,4B,CD,A5,00,58,16
5090 DATA 00,CB,23,CB,12,CB,23,CB
5100 DATA 12,CB,23,CB,12,7B,CD,A5
5110 DATA 00,7A,CD,A5,00,C9,06,08
5120 DATA C5,21,00,C2,06,08,1A,17
5130 DATA CB,16,23,10,FA,C1,13,10
5140 DATA EF,C9,10
```

TOTAL = 1E09

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - página 143.

5.A - MÁQUINA DE ESCREVER

O pequeno programa listado a seguir, após ser executado, faz com que todos os caracteres digitados no teclado apareçam na tela e também na impressora. Desse modo pode-se usar a impressora quase como uma máquina de escrever.

```
100 KEY1,CHR$(13)+CHR$(10)
110 SCREEN 0 : WIDTH 40
120 PRINT "F1 avança 1 linha!"
130 PRINT "HOME/CLS avança uma página!"
140 FOR F=&HE000 TO &HE00E
150   READ A$:POKE F,VAL("&H"+A$)
160 NEXT F : DEFUSR0=&HE000
170 POKE 0,USR0(0) : END
180 DATA CD,9F,00,CD,A2,00,CD,4D
190 DATA 01,CD,B7,00,30,F2,C9,00
```

```
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
```

TOTAL = 6088



BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 139 e 140.

5.8 - IMPRESSOR DE PROGRAMAS

O programa apresentado a seguir permite a impressão de listagens de programas sem que eles passem pelo "picote" do formulário contínuo. Ele só funcionará se o formulário for o de uso padrão, com 86 linhas de impressão, e se o entrelinhamento da impressora não tiver sido alterado.

Para usar o programa, digite-o e rode-o. Ele irá gerar e gravar o programa em Linguagem de Máquina, responsável pelo "salto" sobre o "picote". Antes de comandar LLIST para listar um programa de mais de 86 linhas, carregue a rotina em Linguagem de Máquina comandando:

```
BLOAD"IMPRE.BIN",R
```

Depois, pode-se usar o LLIST normalmente.

```
100 FOR FZ=&HE000 TO &HE03A
110   READ A$:POKE FZ,VAL("&H"+A$)
120 NEXT FZ:DEFUSR0=&HE000
130 POKE 0,USR0(0)
140 BSAVE"IMPRE.BIN",&HE000,&HE03A
150 END
160 DATA F3,3E,C3,32,B6,FF,21,0E
170 DATA E0,22,B7,FF,FB,C9,FE,0A
180 DATA C0,3A,3A,E0,3C,32,3A,E0
190 DATA FE,3F,3E,0A,C0,3E,C9,32
200 DATA B6,FF,3E,0A,CD,A5,00,CD
210 DATA A5,00,CD,A5,00,CD,A5,00
220 DATA 3E,C3,32,B6,FF,AF,32,3A
230 DATA E0,C9,00,00,00,00,00,00
```

57A5
07C9
128C
164E
110C8
110C8
208F
079E
430C
551F
1221
861E
80E8
9168

TOTAL = 9168

BIBLIOGRAFIA RECOMENDADA:

Programação Avançada em MSX - página 149.



DICAS PARA O DRIVE

O disk drive é sem dúvida um dos mais importantes periféricos no sistema MSX. Dentre as máquinas de 8 bits, provavelmente a que consegue um melhor aproveitamento dos drives são as do padrão MSX. Vamos ver alguns recursos do drive a partir do BASIC e do MSXDOS.

Aconselhamos que a numeração dos programas seja seguida à risca, pois em alguns casos pode-se tirar proveito disso. A sequência de programas das dicas entre a 6.2 e a 6.6 pode ser emendada através de comandos "MERGE", constituindo-se num único programa utilitário de disco. Leia mais sobre isso nas explicações da dica 6.2.

A bibliografia recomendada para todas estas dicas é a mesma:

Usando o Disk Drive no MSX.
Sistema de Disco para MSX e
Drives Leopard de 3 1/2"

6.1 - Personalizando disquetes	135
6.2 - Cabeçalhos de arquivos	136
6.3 - Características de discos	139
6.4 - Mapa de discos	141
6.5 - Leitor de setores	145
6.6 - Leitor de arquivos	147
6.7 - Programador de funções	149
6.8 - CLS em MSXDOS	151
6.9 - BEEP em MSXDOS	152
6.A - "SWAP" de arquivos em MSXDOS	152

6.1 - PERSONALIZANDO DISQUETES

O pequeno programa apresentado a seguir grava uma mensagem de até 512 bytes no último setor de um disco.

Você pode usá-lo para "marcar" seus discos com seu nome ou com qualquer outra mensagem.

O texto da mensagem deve ser inserido nas linhas DATA's ao fim do programa e deve terminar sempre com o caractere "0".

100 X%=DSKI\$(0,0)	362
110 EN=PEEK(&HF351)+256*PEEK(&HF352)	086
120 LS=PEEK(EN+19)+256*PEEK(EN+20)-1	1457
130 READ X%	1003
140 IF X%="" THEN 190	2473
150 FOR F=EN TO EN+LEN(X%)-1	2207
160 POKE F,ASC(MID\$(X%,F-EN+1,1))	3330
170 NEXT F : EN=F	4230
180 GOTO 130	4504
190 DSK0% 0,LS	4770
200 END	4855
210 REM -----	507E
220 REM Texto a ser gravado	6517
230 REM -----	8023
240 DATA Este disquete é de	8003
250 DATA propriedade exclusiva	9702
260 DATA de seu próprio dono !	9E69
270 DATA 0	A094

TOTAL = A094

BIBLIOGRAFIA RECOMENDADA:

Usando o Disk Drive no MSX - páginas 115 e 116.
Sistema de Disco para MSX - páginas 64 e 65.
Drives Leopard de 3 " - páginas 98 e 99.

6.2 - CABEÇALHOS DE ARQUIVOS

O programa apresentado a seguir lê o cabeçalho de programas gravados em disco no formato binário. Após digitá-lo e conferi-lo, grave-o no formato ASCII.

2000	REM		162
2010	REM	cabeçalho de arquivos	010
2020	REM	-----	FAA
2030	PLAY	"S0M500001CH"	1505
2040	SCREEN	0 : WIDTH 38 : KEY OFF	1193
2050	COLOR	15,1,1 : PRINT : PRINT	2071
2060	PRINT	SPC(9);"CABEÇALHO DE ARQUIVOS"	391A
2070	PRINT	SPC(9);"-----"	49A6
2080	PRINT	:FILES:PRINT:PRINT:X\$=""	5085
2090	PRINT	"NOME DO ARQUIVO " : INPUT X\$	6320
2100	IF	X\$="" THEN 1000	60A0
2110	OPEN	X\$ AS #1 LEN=1	7806
2120	FIELD	#1,1 AS A\$	7058
2130	IF	LOF(1)<>0 THEN 2140	88E0
2131	CLOSE	#1 : KILL X\$: GOTO 2000	8940
2140	GET	#1,1 : IF ASC(A\$)=254 THEN 2160	9284
2150	PRINT	"NÃO É BINÁRIO":GOTO 2260	987E
2160	RESTORE	2250 : PRINT	A0EA
2170	FOR	F= 1 TO 6 STEP 2	A7F4
2180	READ	B\$: PRINT "--> " ; B\$; " : "	B6F6
2190	GET	#1,F+2	B048
2200	PRINT	"&H"	C1DA
2210	PRINT	RIGHT\$("0"+HEX\$(ASC(A\$)),2);	C6AC
2220	GET	#1,F+1	D591
2230	PRINT	RIGHT\$("0"+HEX\$(ASC(A\$)),2)	E4A5
2240	NEXT	F : CLOSE #1	E008
2250	DATA	INÍCIO ,FIM ,EXECUÇÃO	FA0E
TOTAL =			FA0E

Para usar o programa acima e os programas das dicas de 6.3 a 6.6 será necessário digitar também o programa MENU, listado a seguir. Digite-o e grave-o em ASCII com o nome MENU.

240	FOR	F=1 TO 10 : KEY F,"" : NEXT F	86E
1000	REM		A11
1010	REM	MENU PRINCIPAL	117B
1020	REM	-----	1AAA
1030	PLAY	"S8M40005G#"	202F
1040	COLOR	15,1 : SCREEN 0 : WIDTH 40	2903
1050	KEY	OFF : CLOSE	2049
1060	FOR	F=1 TO 8 : KEY (F) ON : NEXT F	3886

1070 LOCATE 9,2,0	810E
1080 PRINT ">>> MENU PRINCIPAL <<<"	8268
1090 PRINT SPC(9);STRING\$(22,"-")	8270
1100 PRINT ,,,	8300
1110 PRINT SPC(7);" [F1] MENU PRINCIPA	8380
L": PRINT	
1120 PRINT SPC(7);" [F2] CABEÇALHO DE	8478
ARQUIVOS": PRINT	
1130 PRINT SPC(7);" [F3] CARACTERÍSTIC	8578
AS": PRINT	
1140 PRINT SPC(7);" [F4] MAPA DO DISQU	8680
ETE": PRINT	
1150 PRINT SPC(7);" [F5] DUMP DE SETOR	8778
ES": PRINT	
1160 PRINT SPC(7);" [F6] DUMP DE ARQUI	8878
VOS": PRINT	
1170 PRINT SPC(7);" [F7] EDITAR DISCO	8978
": PRINT	
1180 PRINT SPC(7);" [F8] TERMINAR": PR	9078
INT	
1190 ON KEY GOSUB 1000,2000,3000,4000,50	918
00,6000,7000,8000	
1200 GOTO 1200	096
2260 PRINT : LOCATE 8,23,0	1060
2270 PRINT">>> TECLE ESPAÇO <<<";	1160
2280 IF STRIG(0)=0 THEN 2280 ELSE RETURN	1260
1000	
3460 PRINT">>> TECLE ESPAÇO <<<"	2130
3470 IF STRIG(0)=0 THEN 3470 ELSE RETURN	3050
1000	
4920 LOCATE8,23,0	4268
4930 PRINT ">>> TECLE ESPAÇO <<<";	5408
4940 IF STRIG(0)=0 THEN 4940 ELSE RETURN	6548
1000	
5500 LOCATE8,22,0	6640
5510 PRINT">>> TECLE ESPAÇO <<<";	7782
5520 IF STRIG(0)=0 THEN 5520 ELSE GOTO 1	8920
000	
6430 LOCATE8,22,0	889E
6440 PRINT ">>> TECLE ESPAÇO <<<"	9250
6450 IF STRIG(0)=0 THEN 6450 ELSE GOTO 1	9359
000	
7000 REM	9582
7010 REM Editar disco	9687
7020 REM -----	9780
7030 SCREEN 0	9802
7040 PRINT : PRINT : PRINT SPC(10);	9928
7050 PRINT "NÃO IMPLEMENTADA !!!"	0080
7060 PRINT : PRINT : PRINT SPC(10);	0190

7070 PRINT " DIGITE RETURN !!!"	8022
7080 COLOR (F) MOD 15 : F=F+1	8024
7090 IF INKEY%=CHR\$(13) THEN RUN	8026
7100 BEEP : GOTO 7080	8028
8000 REM	8030
8010 REM terminar	8032
8020 REM -----	8034
8030 SCREEN 0 : WIDTH 40 : KEY ON	8036
8040 DEFUSR0=&H3E : X=USR0(0)	8038
8050 FOR F=1 TO 8 : KEY(F) OFF : NEXT F	8040
8060 END	8042

TOTAL = 2920

Quando você já tiver todos os programas (6.2, 6.3, 6.4, 6.5 e 6.6) digitados e gravados, digite a seguinte sequência de comandos:

```
LOAD "MENU"
MERGE "programa 6.2"
MERGE "programa 6.3"
MERGE "programa 6.4"
MERGE "programa 6.5"
MERGE "programa 6.6"
```

Depois, grave o programa presente na memória do micro com o nome "DISCOUT1.BAS". Rode-o para ver como as dicas operam em conjunto.

A "SOMA TOTAL" do programa completo deverá ser:

TOTAL = 9042

6.3 - CARACTERÍSTICAS DE DISCOS

A rotina apresentada adiante lê o setor 0 de um disco e fornece as características de formatação do mesmo. Após digitar e conferir a listagem, grave-a no formato ASCII para posterior utilização com o comando MERGE.

3000	REM		14E
3010	REM	Características do disco	01F
3020	REM	-----	1241
3030	PLAY	"S0M500002DH"	1884
3040	SCREEN	0 : WIDTH 38 : KEY OFF	20A6
3050	PRINT	"CARACTERÍSTICAS DO DISQUETE"	2062
3060	PRINT	STRING\$(27,"-") : PRINT	36D7
3070	PRINT	"Fornecedor O&M :"	4940
3080	PRINT	"Bytes por setor :"	5E9A
3090	PRINT	"Setores por bloco .. :"	7236
3100	PRINT	"Setores reservados . :"	810E
3110	PRINT	"Nº de F.A.T.'s :"	870E
3120	PRINT	"Entr. no diretório . :"	8E19
3130	PRINT	"Setores no disco ... :"	9756
3140	PRINT	"Tipo de disco :"	9100
3150	PRINT	"Setores por F.A.T. . :"	B016
3160	PRINT	"Setores por trilha . :"	C465
3170	PRINT	"Faces :"	D9FE
3180	PRINT	"Setores ocultos :"	EE58
3190	A\$=DSKI\$(0,0) : C = 256		F080
3200	E=C*PEEK(&HF352)+PEEK(&HF351)		938
3210	DEFFNA(X)=C*PEEK(E+X)+PEEK(E+X-1)		171B
3220	DEFFNB(X)=PEEK(E+X) : LOCATE 23,3		219B
3230	FOR I=E+3 TO E+10		288F
3240	PRINT CHR\$(PEEK(I));		31EE
3250	NEXT I		3487
3260	LOCATE 22,4 : PRINT FNA(12)		3050
3270	LOCATE 22,5 : PRINT FNB(13)		4821
3280	LOCATE 22,6 : PRINT FNA(15)		52EB
3290	LOCATE 22,7 : PRINT FNB(16)		50EB
3300	LOCATE 22,8 : PRINT FNA(18)		681E
3310	LOCATE 22,9 : PRINT FNA(20)		7887
3320	LOCATE 23,10: PRINT HEX\$(FNB(21))		88AE
3330	LOCATE 22,11: PRINT FNA(23)		920B
3340	LOCATE 22,12: PRINT FNA(25)		9720
3350	LOCATE 22,13: PRINT FNA(27)		9860
3360	LOCATE 22,14: PRINT FNA(29)		A10E
3370	PRINT ,,"Capacidade total ... :"		AA4A
3380	BS = FNA(12) : SE = FNA(20)		B396
3390	LOCATE 22,16,0		B981
3400	PRINT USING "#####";SE*BS;		C95C

```

3410 PRINT " bytes"
3420 PRINT "Área ainda disponível:"
3430 D=DSKF(0):S=FNB(13):LOCATE 22,17
3440 PRINT USING "#####";D*S*BS;
3450 PRINT " bytes":PRINT:LOCATE 8,21,0

```

0135

E545

F603

753

1090

TOTAL = 1090

CARACTERÍSTICAS DO DISQUETE

```

-----
Fornecedor O&M ..... : MSX-02
Bytes por setor .... : 512
Setores por bloco .. : 2
Setores reservados . : 1
Nº de F.A.T.'s ..... : 2
Entr. no diretório . : 112
Setores no disco ... : 720
Tipo de disco ..... : FD
Setores por F.A.T. . : 2
Setores por trilha . : 2
Fases ..... : 2
Setores ocultos .... : 0

Capacidade total ... : 368640 bytes
Área ainda disponível: 205824 bytes

```

Ok



6.4 - MAPA DE DISCOS

O programa apresentado a seguir lê a FAT do disco presente no drive A e apresenta na tela o conteúdo de cada uma de suas posições. Em seguida o diretório do disco é lido e cada um de seus arquivos são mostrados na tela com todos os seus parâmetros especificados, incluindo os blocos do disco em que ele está armazenado.

Após digitar e conferir o programa, grave-o no formato ASCII para posterior utilização com o comando MERGE.

Caso um arquivo tenha sido apagado com o comando KILL do DISK BASIC ou ERASE ou DEL do MSXDOS, o programa permitirá a sua recuperação, mas APENAS NO DIRETÓRIO. Se o arquivo apagado ocupar menos que um bloco do disco, ele será automaticamente recuperado também na FAT, mas se ele ocupar mais de um bloco, apesar de ser recuperado no diretório, não será mapeado por completo na FAT.

4000	REM		13A
4010	REM	mapa do disquete	80C
4020	REM	-----	C8D
4030	PLAY	"S0M500003E#"	1228
4040	SCREEN	0 : WIDTH 36 : KEY OFF	1A08
4050	PRINT	SPC(6);">>> MAPA DO DISQUETE	2506
	<<<"		
4060	PRINT	SPC(6);STRING\$(24,"-")	2E98
4070	PRINT	:PRINT:PRINT	30FC
4080	PRINT	SPC(6);">>> LENDO A F.A.T. <	4A74
	<<<":LOCATE	0,3,0	
4090	GOSUB	8260	4E0D
4100	FOR	F=5 TO 11	56A8
4110	LOCATE	0,2,0:PRINTCHR\$(27)"J"	6606
4120	PRINT	SPC(9);">>> SETOR";	750A
4130	PRINT	USING "###";F;	7C4A
4140	PRINT	" <<<"	7E0F
4150	PRINT	SPC(9);"=====	8297
4160	A\$	= DSKEI\$(0,F)	86E0
4170	P	= PEEK(&HF351)+256*PEEK(&HF352)	92AD
4180	FOR	G=0 TO 15	9860
4190	LOCATE	0,5,0:PRINTCHR\$(27)"J";	A1A2
4200	IF	INKEY\$=CHR\$(27) THEN 1000	AC40
4210	PRINT	"ARQUIVO";(F-5)*16+G+1	0052
4220	PRINT	"-----"	0E8A
4230	REM	-----	EB56
4240	PRINT	"NOME : ";	FA55
4250	C\$	= ""	FF91

4260		FOR H=0 TO 10	447
4270		A = P + 32*G + H	852
4280		IF PEEK (A) <> 0 THEN 4310	1338
4290		PRINTSTRING\$(95,127)	1852
4300		GOTO 4920	2409
4310		C\$ = C\$ + CHR\$(PEEK(A))	2949
4320		IF H=7 THEN C\$=C\$+"."	3067
4330		NEXT H	4237
4340		PRINT C\$	4780
4350	REM	-----	5024
4360		PRINT"ATRIBUTOS :";	6090
4370		AT = PEEK (P + 32*G + 11)	7179
4380		AT\$ = RIGHT\$("00000000"+BIN\$(AT	8272
4390),8)	
4390		PRINT " ";AT\$	8839
4400	REM	-----	9540
4410		PRINT"HORA :";	9609
4420		X=(P+32*G+22)	10389
4430		X1=PEEK(X)	11476
4440		X2=PEEK(X+1)	12494
4450		X1\$=RIGHT\$("00000000"+BIN\$(X1),	13511
4460		8)	
4460		X2\$=RIGHT\$("00000000"+BIN\$(X2),	14029
4470		8)	
4470		H\$=LEFT\$(X2\$,5)	15007
4480		M\$=RIGHT\$(X2\$,3)+LEFT\$(X1\$,3)	16099
4490		H=VAL("&B"+H\$)	173
4500		M=VAL("&B"+M\$)	978
4510		PRINT " ";H;M	1084
4520	REM	-----	1195
4530		PRINT"DATA :";	2148
4540		H2=PEEK(P + 32*G + 24)	2277
4550		H1=PEEK (P + 32*G + 25)	2376
4560		H1\$=RIGHT\$("00000000"+BIN\$(H1),	2508
4570		8)	
4570		H2\$=RIGHT\$("00000000"+BIN\$(H2),	2611
4580		8)	
4580		D=VAL("&B"+RIGHT\$(H2\$,5))	2725
4590		M=VAL("&B"+RIGHT\$(H1\$,1)+LEFT\$(2878
4600		H2\$,3))	
4600		A=1980+VAL("&B"+LEFT\$(H1\$,7))	3023
4610		PRINT D;M;A	3100
4620	REM	-----	3170
4630		PRINT"1º BLOCO : ";	3271
4640		H=(P+32*G+26)	3343
4650		H=PEEK(H)+256*PEEK(H+1)	3426
4660		PB=H	3501
4670		PRINT RIGHT\$("000"+HEX\$(H),3)	3583
4680	REM	-----	3632

4690	PRINT "Nº DE BYTES: ";	92
4700	H=(P+32*G+28)	74C
4710	H=PEEK(H)+256*PEEK(H+1)+	1107
	4096*PEEK(H+2)+65536!*PEEK(H+3)	
4720	PRINT RIGHT\$("0000"+HEX\$(H),4)	2750
4730	REM -----	3002
4740	PRINT "MAPA DA FAT: ";	4092
4750	H=PB	521A
4760	IF H>359 THEN 4830	5251
4770	PRINT RIGHT\$("000"+HEX\$(B%(H)),	7871
3); " "		
4780	IF H>359 THEN 4830	8475
4790	H=B%(H)	8803
4800	IF H>359 THEN 4830	8E18
4810	PRINT RIGHT\$("000"+HEX\$(B%(H)),	9887
3); " "		
4820	GOTO 4780	9076
4830	REM -----	998C
4840	PRINT	A08A
4850	PRINT	A1F1
4860	IF LEFT\$(C\$,1)=CHR\$(&HE5) THEN	B788
GOSUB 8070		
4870	LOCATE 10,23,0	C151
4880	PRINT "DIGITE RETURN:";	D807
4890	A%=INPUT\$(1)	E078
4900	NEXT G	E5EA
4910	NEXT F	E981
4920	LOCATE 8,23,0	F048
4930	PRINT "))) TECLE ESPAÇO <<<";	490
4940	IF STRIG(0)=0 THEN 4940 ELSE RUN	B1F
8070	REM	D40
8080	REM Recupera deletados	14EF
8090	REM -----	1C29
8100	PRINT,,") Arquivo deletado ! <"	245A
8110	PRINT,"RECUPERAR (S/N)?:";	2C2C
8120	BEEP : BEEP : BEEP	310C
8130	X = PEEK(&HFCAB) : POKE &HFCAB,1	3083
8140	X% = INPUT\$(1) : PRINT X%	4807
8150	IF X%<"S" THEN 8220	5298
8160	PRINT,"1º caractere do nome:";	6883
8170	Z% = INPUT\$(1) : PRINT Z%	7718
8180	IF Z%<"A" OR Z%>"Z" THEN 8160	8275
8190	POKE (A-10),ASC(Z%)	8917
8200	DSK0% 0,F	8B16
8210	GOTO 8230	8CA1
8220	IF X%<"N" THEN 8110	9366
8230	POKE &HFCAB,X : PRINT : PRINT	9C11
8240	RETURN	9D88
8250	REM	9F90

8260	REM leitor de F.A.T.	A305
8270	REM -----	B600
8280	AZ(0)=BZ(0):ERASE AZ,BZ	C428
8290	DIM AZ(539),BZ(359)	D08A
8300	A%=DSKI\$(0,1)	D817
8310	P = PEEK(&HF351)+256*PEEK(&HF352)	E683
8320	FOR F=0 TO 511	EE50
8330	AZ(F) = PEEK(P+F)	F958
8340	NEXT F	FAD0
8350	A%=DSKI\$(1,2)	FE90
8360	P = PEEK(&HF351)+256*PEEK(&HF352)	98E
8370	FOR F=0 TO 27	0FF
8380	AZ(F+512) = PEEK(P+F)	1688
8390	NEXT F	1907
8400	G=0	1B0F
8410	FOR F=0 TO 539 STEP 3	280F
8420	IF INKEY%=CHR\$(27) THEN 1000	2C5A
8430	BZ(G) =AZ(F)+256*(AZ(F+1) AND &HF	4226
8440	BZ(G+1)=(AZ(F+1) AND &HF0)/16 +	6A26
	AZ(F+2)*16	
8450	G=G+2	6F6A
8460	NEXT F	72E8
8470	FOR F=1 TO 360	7808
8480	IF INKEY%=CHR\$(27) THEN 1000	845F
8490	PRINT RIGHT\$("000"+HEX\$(F-1),3);"	9280
8500	W";RIGHT\$("000"+HEX\$(BZ(F-1)),3);"	
8510	IF F/72 (> F)\72 THEN 8550	A088
8520	LOCATE 10,23,1	A680
8530	PRINT "DIGITE RETURN:";	A000
8540	A%=INPUT\$(1)	B5AE
8550	LOCATE 0,3,0	B057
8560	NEXT F	C006
8570	PRINT CHR\$(27);"J"	C852
	RETURN	C9F2

TOTAL = C9F2

```

>>> MAPA DO DISQUETE <<<
-----
>>> SETOR 5 <<<
=====
ARQUIVO 6
-----
NOME          : FILTRO2 .ASM
ATRIBUTOS    : 00000000
HORA         : 0 0
DATA        : 25 4 1986
1º BLOCO    : 009
Nº DE BYTES : 0480
MAPA DA FAT : 00A 00B FFF

```

6.5 - LEITOR DE SETORES

O programa a seguir gera na tela um "DUMP" do setor de disco especificado pelo usuário. Após digitá-lo e conferi-lo, grave-o em formato ASCII para posterior utilização com o comando MERGE.

100 CLEAR 1000,&HC000	325
105 ONERROR GOTO 100	677
110 RESTORE 150	830
120 FOR F%=&HC000 TO &HC02D	E36
130 READ X% : POKE F%,VAL("&H"+X%)	1580
140 NEXT F% : DEFUSR0=&HC000	106A
150 DATA 21,00,00,11,2E,C0,01,C0	2030
160 DATA 03,CD,59,00,01,58,02,21	3551
170 DATA F6,C0,11,CE,C0,ED,B0,06	413F
180 DATA 28,21,4E,C3,36,20,23,10	5742
190 DATA FB,21,2E,C0,11,00,00,01	605F
200 DATA C0,03,CD,5C,00,C9,F3,21	8136
210 DATA 52,45,4E,41,54,4F,20,44	8A00
220 DATA 41,20,53,49,4C,56,41,20	8765
230 DATA 4F,4C,49,56,45,49,52,41	96AD
5000 REM	9888
5010 REM dump de setores	9F28
5020 REM -----	AB93
5030 PLAY"S0M500005G#"	B610
5040 SCREEN 0 : WIDTH 39	BC9F
5050 X%=DSKI\$(0,0)	C263
5060 E=PEEK(&HF351)+256*PEEK(&HF352)	0E70
5070 NS=256*PEEK(E+20)+PEEK(E+20-1)-1	1A01
5080 S%=""	E2AF
5090 PRINT "ENTRE O SETOR : &H ";	F638
5100 PRINT CHR\$(8);CHR\$(8);	FC3E
5110 X%=INPUT\$(1) : PRINT X%;	1A3
5120 IF X%=CHR\$(8) THEN 5000	7F2
5130 S%=S%+X%	B6F
5140 IF LEN(S%)<3 THEN 5110	13A3
5150 S%=RIGHT\$("00"+S%,3)	1898
5160 S=VAL("&H"+S%)	2163
5170 IF S<0 OR S>NS THEN 5000	294C
5180 CLS	2AAE
5190 PRINT : PRINT " SETOR : &H";S%	3AF6
5200 PRINT : PRINT STRING\$(39,"-")	43E7
5210 LOCATE 0,21:PRINT STRING\$(39,"-")	5120
5220 X%=DSKI\$(0,S)	5894
5230 EN=PEEK(&HF351)+256*PEEK(&HF352)	6604
5240 FOR F=0 TO 63	6E01
5250 IF INKEY%=CHR\$(27) THEN 1000	770E
5260 IF F/16<>F\16 THEN 5320	770C


```

5270     LOCATE 10,23,0
5280     PRINT "DIGITE RETURN:";
5290     Y%=INPUT$(1)
5300     LOCATE 10,23,0
5310     PRINT " ";
5320     X=USR(0)
5330     LOCATE 0,19,0
5340     PRINT RIGHT$("00"+HEX$(8*F),3);
5350     FOR G=0 TO 7
5360         X=PEEK(EN+8*F+G)
5370         Y%=RIGHT$("0"+HEX$(X),2)
5380         PRINT " ";Y%;
5390     NEXT G
5400     PRINT " ! ";
5410     FOR G=0 TO 7
5420         X=PEEK(EN+8*F+G)
5430         Y%=CHR$(X)
5440         IF X>31 THEN 5460
5450         Y%=CHR$(1)+CHR$(64+X)
5460         PRINT Y%;
5470     NEXT G
5480     PRINT
5490 NEXT F

```

```

8247
893A
902F
96CE
A09C
A68F
AD7F
BA87
C4A0
C900
DC61
E7EB
E088
F303
F68E
FF07
315
809
1304
1781
1776
1088
1191

```

TOTAL = 1E91

SETOR : 8HQ05

```

-----
080 43 45 4D 31 2D 33 20 20 CEM1-3
088 42 41 53 00 00 00 00 00 BAS
090 00 00 00 00 00 00 00 00
098 99 0C 06 00 80 00 00 00 00 C
0A0 46 49 4C 54 52 4F 32 20 FILTR02
0A8 41 53 4D 00 00 00 00 00 00 ASM
0B0 00 00 00 00 00 00 00 00 00
0B8 99 0C 09 00 80 0A 00 00 00 00 C
0C0 46 49 4C 54 52 4F 32 20 FILTR02
0C8 42 49 4E 00 00 00 00 00 00 00 BIN
0D0 00 00 00 00 00 00 00 00 00 00
0D8 99 0C 0C 00 F3 00 00 00 00 00 00 C
0E0 43 45 4D 31 2D 34 20 20 CEM1-4
0E8 42 41 53 00 00 00 00 00 00 00 00 BAS
0F0 00 00 00 00 00 00 00 00 00 00 00
0F8 99 0C 07 00 80 01 00 00 00 00 00 C
-----

```

DIGITE RETURN: █

6.6 - LEITOR DE ARQUIVOS

O programa a seguir gera na tela um "DUMP" do arquivo especificado pelo usuário.

Note que as linhas entre 100 e 230 são idênticas às do programa leitor de setores.

Após digitar e conferir o programa, grave-o em formato ASCII para posterior utilização com o comando MERGE.

100 CLEAR 1000,&HC000	325
105 ONERROR GOTO 100	677
110 RESTORE 150	890
120 FOR FZ=&HC000 TO &HC02D	E36
130 READ X% : POKE FZ,VAL("&H"+X%)	1530
140 NEXT FZ : DEFUSR0=&HC000	1106A
150 DATA 21,00,00,11,2E,C0,01,C0	2100
160 DATA 03,CD,59,00,01,58,02,21	3551
170 DATA F6,C0,11,CE,C0,ED,B0,06	478F
180 DATA 28,21,4E,C3,36,20,23,10	5742
190 DATA FB,21,2E,C0,11,00,00,01	605F
200 DATA C0,03,CD,5C,00,C9,F3,21	8136
210 DATA 52,45,4E,41,54,4F,20,44	8ACC
220 DATA 41,20,53,49,4C,56,41,20	8F65
230 DATA 4F,4C,49,56,45,49,52,41	96AD
6000 REM	9807
6010 REM dump de arquivos	9DF8
6020 REM -----	AA86
6030 PLAY"S0M500003D#"	B584
6040 SCREEN 0 : WIDTH 39	B8A5
6050 PRINT" ARQUIVOS:" : PRINT : FILES	C977
6060 PRINT : PRINT	CC88
6070 PRINT "Entre o nome do arquivo:";	E25A
6080 INPUT X%	E617
6090 IF X%="" THEN 1000	EF2A
6100 CLS	F0A0
6110 PRINT : PRINT "ARQUIVO :";X%	F83C
6120 PRINT : PRINT STRING\$(39,"-")	A4
6130 LOCATE 0,21:PRINT STRING\$(39,"-")	51C
6140 OPEN X% AS #1 LEN=1	AF9
6150 FIELD #1,1 AS Y%	FDD
6160 LF=INT(LOF(1)/8)+1	1081
6170 FOR F=0 TO LF	107E
6180 IF INKEY%=CHR\$(27) THEN 1000	2781
6190 IF F/16<>F\16 THEN 6250	8008
6200 LOCATE 10,23,0	3050
6210 PRINT "DIGITE RETURN:";	4FC9
6220 X%=INPUT\$(1)	5A57
6230 LOCATE 10,23,0	669C

```

6240     PRINT "                               ";
6250     X=USR0(0)
6260     LOCATE 0,19,0
6270     PRINT RIGHT$("000"+HEX$(1+8*F),4)
?
6280     X$=""
6290     FOR G=F*8+1 TO F*8+8
6300         IF G<=LOF(1) THEN 6340
6310             Z$=""
6320             X%=X$+" "
6330             GOTO 6390
6340             GET #1,G
6350             X=ASC(Y$)
6360             IF X>31 THEN X%=X%+CHR$(X)
6370             IF X<32 THEN X%=X%+CHR$(1)+CHR$(
(64+X)
6380             Z%=RIGHT$("0"+HEX$(X),2)
6390             PRINT " ";Z%;
6400     NEXT G
6410     PRINT "! ";X$
6420 NEXT F

```

TOTAL = 405

ARQUIVO : CEM6-A. BAS

```

-----
0001 31 30 30 20 43 4C 45 41 100 CLEA
0009 52 20 31 30 30 30 2C 26 R 1000, &
0011 48 43 30 30 30 0D 0A HC000 J01
0019 30 35 20 4F 4E 45 50 05 ONERR
0021 4F 52 20 47 4F 44 50 DR GOTO
0029 31 30 30 0D 0A 31 31 30 100 J0110
0031 20 52 45 53 54 4F 50 4 RESTORE
0039 20 31 35 30 0D 0A 31 4 150 J012
0041 30 20 46 4F 50 20 46 0 FOR F%
0049 3D 26 48 43 30 30 30 =&HC000
0051 54 4F 20 26 40 30 30 TO &HC02
0059 44 0D 0A 31 30 30 30 DJ0130 R
0061 45 41 44 20 30 24 20 EAT X#
0069 20 50 4F 48 45 46 20 POKE F%
0071 2C 56 41 4C 40 26 48 VAL("&H
0079 2C 58 24 40 40 0A 31 "+X#) J01
-----

```

DIGITE RETURN

6.7 - PROGRAMADOR DE FUNÇÕES

O programa apresentado a seguir permite a programação das teclas de função de forma bastante simples. Após digitá-lo e conferi-lo, grave-o com o nome "FUNKEY.BAS".

O uso deste programa é mais indicado para o MSXDOS. Nesse caso, deve-se alterar o "END" da linha 640 para "_SYSTEM" e gerar um arquivo BAT com o comando "BASIC FUNKEY.BAS". Assim, estando em DOS, bastará executar o arquivo BAT para poder programar as teclas de função.

```
100 CLEAR 500 : DEFINT A-Z
110 SCREEN 0,,0 : WIDTH 38 : KEY OFF
120 FOR F=1 TO 4 : KEY(F) ON : NEXT F
130 ON KEY GOSUB 490,650,760,790
140 PRINT"REDEFINIDOR DE FUNCOES";
150 PRINT" - EDITORA ALEPH";
160 PRINTSTRING$(38,"");
170 PRINT : PRINT : PRINT : PRINT .
180 PRINT CHR$(27);"y4"
190 FOR F=1 TO 10
200 PRINT SPC(7);
210 PRINT "F";USING"##";F;
220 PRINT " --->[ ]"
230 NEXTF
240 PRINT : PRINT
250 PRINT SPC(11); "F1 - VOLTA AO DOS"
260 PRINT SPC(11); "F2 - REINICIALIZA"
270 PRINT SPC(11); "F3 - DEFAULT BASIC"
280 PRINT SPC(11); "F4 - LIMPA TEXTOS"
290 X=16 : Y=7
300 LOCATE X,Y,1
310 A$=INKEY$
320 IF A$<CHR$(28) THEN 350
330 IF A$>CHR$(31) THEN 350
340 A$=""
350 A=STICK(0)
360 IF A>0 AND A<3 OR A=8 THEN Y=Y-1
370 IF Y=6 THEN Y=16
380 IF A>3 AND A<7 THEN Y=Y+1
390 IF Y=17 THEN Y=7
400 IF A>1 AND A<5 THEN X=X+1
410 IF X>30 THEN X=16
420 IF A>5 THEN X=X-1
430 IF X=15 THEN X=30
440 IF A$="" THEN 300
450 LOCATE,,0
```

822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

460 VPOKE 40*Y+X+1,ASC(A%)
470 IF X<30 THEN X=X+1
480 GOTO 300
490 BEEP : BEEP : BEEP
500 FOR F=1 TO 10
510   LOCATE 0,0,0
520   A%="" : FL=0
530   X=(F+6)*40+17
540   FOR Y=X+14 TO X STEP -1
550     IF FL=1 THEN 580
560     IF VPEEK(Y)=32 THEN 590
570     FL=1
580     A%=CHR$(VPEEK(Y))+A%
590   NEXT Y
600   KEY F,A%
610 NEXT F
620 SCREEN 0 : WIDTH 40 : KEY ON
630 FOR F=1 TO 4 : KEY(F) OFF : NEXT F
640 PRINT CHR$(27);"x4" : END
650 BEEP : BEEP : BEEP
660 GOSUB 790 : LOCATE,,0
670 FOR F=0 TO 9
680   FOR G=0 TO 15
690     A=PEEK(&HF87F+16*F+G)
700     IF A=0 THEN 720
710     VPOKE40*(F+7)+17+G,A
720   NEXT G
730 NEXT F
740 LOCATE,,1
750 RETURN
760 BEEP : BEEP : BEEP
770 DEFUSR=&H3E : S=USR(0)
780 GOSUB 650 : RETURN
790 BEEP : BEEP : BEEP
800 LOCATE,,0 : PRINT " ";CHR$(8)
810 FOR F=0 TO 9
820   FOR G=0 TO 14
830     VPOKE40*(F+7)+17+G,32
840   NEXT G
850 NEXT F
860 LOCATE,,1
870 RETURN

```

```

0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099

```

TOTAL = 340

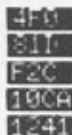
6.8 - CLS EM MSXDOS

O pequeno programa em BASIC listado a seguir gera um arquivo BATCH que limpa a tela ao ser executado a partir do MSXDOS.

Após digitá-lo, conferi-lo e gravá-lo, rode-o.

Dessa forma, deverá ter sido gerado no disco presente no drive A um arquivo BAT de nome "CLS.BAT". Esse arquivo deve ser usado a partir do MSXDOS, simplesmente digitando CLS e RETURN.

```
10 OPEN "A:CLS.BAT" AS #1
20 FIELD #1, 2 AS A%
30 LSET A%=CHR$(12)+CHR$(26)
40 PUT #1,1
50 END
```



Após rodar o programa, experimente carregar o MSXDOS e comande:

```
A>CLS
```

A tela deverá ser limpa.



6.9 - BEEP EM MSXDOS

O programa em BASIC listado a seguir gera um arquivo BATCH que produz um BEEP quando executado a partir do MSXDOS.

Seu uso é semelhante ao do programa CLS.BAT gerado na dica 6.C e, portanto, ele deve ser executado necessariamente a partir do MSXDOS.

```
10 OPEN "A:BEEP.BAT" AS #1
20 FIELD #1, 2 AS A%
30 LSET A%=CHR$(7)+CHR$(26)
40 PUT #1,1
50 END
```

```
54E
846
F11
10AF
1226
```

TOTAL = 1226

Após rodar o programa, carregue o MSXDOS e comande:

```
A>BEEP
```

Isso deverá produzir um beep.

6.A - "SWAP" DE ARQUIVOS EM MSXDOS

Estando em MSXDOS, digite a sequência de comandos listada a seguir para produzir um arquivo BATCH capaz de trocar os nomes de dois arquivos quaisquer.

```
A>COPY CON A:SWAP.BAT
REN %1 BABA.%%
REN %2 %1
REN BABA.%% %2
^Z
```

Com o arquivo SWAP.BAT já gravado no disco, você poderá usá-lo com a seguinte sintaxe:

```
A>SWAP arquivo1 arquivo2
```

Os parâmetros "arquivo1" e "arquivo2" são os nomes dos dois arquivos cujos conteúdos se deseja trocar.



DICAS DE PROCESSAMENTO

As dicas apresentadas neste capítulo são as mais diversas, abordando recursos múltiplos das máquinas MSX. Elas não são específicas para nenhum periférico, sendo mais relacionadas com o processamento de outros programas.

7.1 - Diferenciando o Expert do Hotbit	154
7.2 - Escondendo listagens de programas	155
7.3 - Desativando todos os comandos	157
7.4 - Evitando o "ON STOP GOSUB"	158
7.5 - Obtendo o valor de PI	159
7.6 - Tabelas de caracteres	160
7.7 - Sorteio de palavras da ROM	163
7.8 - Sorteio de palavras em linhas "DATA's"	164
7.9 - Mapeando linhas REM	165
7.A - Mapeando variáveis	166
7.B - Mapeando linhas de um programa	167
7.C - Gerando linhas DATA	168
7.D - Trocando tokens num programa	169
7.E - Aumentando a velocidade de execução ...	171
7.F - "SEARCH", pesquisador de strings	173
7.G - Recuperando programas apagados com NEW	175
7.H - Redefinindo mensagens de erros	176
7.I - Soma sintática	178
7.J - PSEUDO-RAMDISK	181
7.K - Rotacionando caracteres	183
7.L - Rotina de entrada com INKEY\$	184
7.M - Rotina para maiúsculas	186
7.N - Reduzindo a tela a um caractere	187
7.O - Grandes expoentes	188

7.1 - DIFERENCIANDO O EXPERT DO HOTBIT

Existem muitas diferenças facilmente identificáveis entre o HOTBIT e o EXPERT. A mais fácil de ser checada por um programa, quer esteja ele em BASIC ou em Linguagem de Máquina, é a mensagem de identificação do fabricante. No Expert, a partir do endereço 32513 encontramos na ROM a string "G r a d i e n t e". Basta então checar uma letra dessa mensagem que difira do HOTBIT, por exemplo, a letra "G" (código ASCII 71). O programa a seguir faz exatamente isso.

```
100 SCREEN,0 : WIDTH 39 : PRINT
110 PRINT " Conteúdo da ROM a partir"
120 PRINT " do endereço 32513: ";
130 FOR F=32513 TO 32531
140 PRINT CHR$(PEEK(F));
150 NEXT F
160 X$="é"
170 IF PEEK(32513)<>71 THEN X$="não é"
180 PRINT : PRINT : PRINT " ";
190 PRINT "Portanto, ";X$;" um EXPERT!"
200 PRINT : PRINT
```

TOTAL = 5703



BIBLIOGRAFIA RECOMENDADA.

Aprofundando-se no MSX - páginas 82 e 83.

7.2 - ESCONDENDO LISTAGENS DE PROGRAMAS

Um programa em BASIC pode estar presente na memória do micro sem que o comando LIST ou LLIST faça sua listagem aparecer na tela ou na impressora.

A seguir são apresentados dois pequenos programas que exemplificam duas das muitas formas disponíveis para se esconder a listagem.

Em ambos os programas os comandos responsáveis pela ocultação das listagens encontram-se na linha 10.

O primeiro programa deve ser rodado ao menos uma vez antes de ser gravado. Feito isso, o programa deve ser gravado no formato compactado, isto é, não pode ser gravado em ASCII! Para fita cassete deve-se usar o comando CSAVE e para disquete deve-se usar o comando SAVE sem a opção ",A"!

PROGRAMA 1

```
10 POKE&H8003,&HFF:POKE&H8004,&HFF          1523
20 FOR F=1 TO 200                             1584
30 PRINT "ESTOU AQUI ESCONDIDO !!!"          1584
40 NEXT F                                       1583
```

TOTAL = 1558

O segundo programa pode ser gravado mesmo sem ter sido rodado, pois de qualquer forma, ao ser carregado da fita ou do disquete, sua listagem estará visível. Para que ela se torne inacessível é necessário que ele seja rodado ao menos uma vez após ter sido carregado!

PROGRAMA 2

```
10 POKE&HFF89,&HD1                             1346
20 FOR F=1 TO 200                             1341
30 PRINT "ESTOU AQUI ESCONDIDO !!!"          1335
40 NEXT F                                       1300
```

TOTAL = 1201

Uma variante mais drástica do programa 2 é obtida alterando-se a linha 10 para:

```
10 POKE &HFF89,&HC3
```

Com isso, toda vez que o comando LIST for

executado, o micro será automaticamente ressetado.
Para desativar essa "trava" basta comandar:

```
POKE &HFF89,HC9
```

Note que apenas o comando LIST foi desabilitado. Os demais comandos do BASIC continuam funcionando normalmente.

Ao invés de ressetar o micro, podemos simplesmente evitar a listagem. Acrescente ao programa 2 as linhas a seguir:

```
11 POKE &HFF8A,&H72  
12 POKE &HFF8B,&H0
```



7.3 - DESATIVANDO TODOS OS COMANDOS

O hook do "Ok" é chamado pelo interpretador sempre que o "Ok" vai ser mostrado na tela. Isso pode servir para evitar que sejam usados quaisquer comandos do BASIC MSX. Para produzir um desvio no hook do "Ok" basta usar os seguintes comandos:

```
POKE &HFF07,&HC3  
POKE &HFF08,&H00  
POKE &HFF09,&H00
```

Com isso, sempre que o "Ok" for ser impresso na tela, o micro será ressetado. Para voltar a operar normalmente, basta comandar:

```
POKE &HFF07,&HC9
```

Um exemplo prático dessa "trava" pode ser imaginado ao se interromper a execução de um programa em BASIC com CONTROL + STOP. Logo a seguir, o "Ok" surge na tela, provocando um resset automático do micro.



7.4 - EVITANDO O "ON STOP GOSUB"

O MSX permite que se "trave" um programa em BASIC após o início de sua execução com as instruções:

```
STOP ON     e  
ON STOP GOSUB xxxxxx
```

Usando essas instruções nas primeiras linhas de um programa, após serem executadas, elas farão com que sempre que as teclas CONTROL e STOP forem pressionadas conjuntamente, a execução seja desviada para a linha de número xxxxx.

Isso pode ser muito útil depois que o programa está pronto, mas durante sua elaboração pode trazer transtornos ao programador.

O MSX, porém, possui o veneno e o antídoto. Com apenas um POKE numa variável do sistema é possível acionar o "warm start" do interpretador BASIC pelo teclado. Imagine que vamos executar um programa que usa as instruções STOP ON e ON STOP GOSUB. Se quisermos torná-lo facilmente interrompível, basta comandar:

```
POKE &HFBB0,1
```

Isso fará com que o pressionamento conjunto das teclas CONTROL, SHIFT, LGRA e RGRA no Expert (ou CTRL, SHIFT, GRAPH e CODE no HOTBIT) interrompa a execução do programa e devolva o comando ao usuário.

Para desligar o acionamento do "warm start" do interpretador basta comandar:

```
POKE &HFBB0,0
```

Esse recurso não é exclusivo do BASIC. Mesmo quando um programa em Linguagem de Máquina está sendo executado, se o conteúdo de &HFBB0 for diferente de 0, a execução pode ser interrompida, desde que o programa em L.M. não tenha desabilitado a interrupção!



7.5 - OBTENDO O VALOR DE PI

O MSX não possui uma variável reservada para o armazenamento da constante matemática π .

Entretanto, essa constante pode ser facilmente obtida e de várias formas diferentes.

A forma mais imediata é usar a expressão:

$$4 * \text{ATN}(1)$$

Experimente comandar:

```
PRINT 4*ATN(1)
```

O valor de π surgirá na tela, pois,

$$\text{Tan}(\pi/4) = 1$$

$$\text{Atn}(\text{tan}(\pi/4)) = \text{Atn}(1)$$

$$\pi/4 = \text{Atn}(1)$$

$$\pi = 4 * \text{Atn}(1)$$

Outra forma de obter o valor dessa constante é buscá-lo na própria ROM do micro. Os dois programas apresentados a seguir fazem exatamente isso.

```
10 REM PIROM I
20 A# = 0
30 FOR F=0 TO 7
40 POKE VARPTR(A#)+F,PEEK(&H2D43+F)
50 NEXT F
60 PI = 2*A#
70 PRINT PI
```

240
44F
898
F8F
1074
1488
1628

TOTAL = 1628

```
10 REM PIROM II
20 FOR F=0 TO 7
30 A$ = A$ + CHR$(PEEK(&H2D43+F))
40 NEXT F
50 PI = 2*CVD(A$)
60 PRINT PI
```

2F2
671
EF5
1077
1679
1892

TOTAL = 1892

Após a execução de um desses dois programas, o valor de π ficará armazenado na variável PI.

7.6 - TABELAS DE CARACTERES

O programa apresentado a seguir gera na tela a tabela de caracteres do seu micro. A apresentação é feita em duas partes: a primeira contém os caracteres de 0 a 127 e a segunda os caracteres de 128 a 255.

Além dos próprios caracteres, a tabela contém um quadro em branco logo abaixo de cada um deles. Esses quadros estão divididos em três campos e devem ser preenchidos por você, após a impressão da tabela em papel. Eles servirão para indicar as teclas a serem pressionadas para que o caractere correspondente seja apresentado na tela. Por exemplo, o quadro preenchido abaixo corresponde ao caractere "9".



O campo principal indica a tecla em que o caractere se encontra. O campo superior a direita, quando preenchido, indica que a tecla SHIFT deve ser pressionada. O campo inferior direito, quando preenchido, indica que a tecla LGRA (ou GRAPH) deve ser pressionada. Portanto, para produzir o caractere "9" deve-se pressionar as teclas:

SHIFT + LGRA + 9 (Expert)
ou

SHIFT + GRAPH + 9 (Hotbit)

O campo inferior direito serve para indicar se a tecla RGRA (ou CODE) deve ser usada. Nesse caso, ele deve ser preenchido.

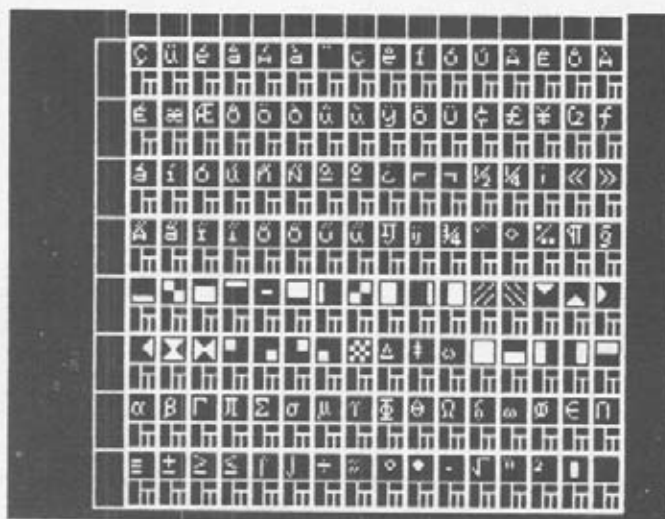
Caso o caractere não possa ser produzido através do teclado, sugerimos que não se preencha nada no campo reservado para a tecla. Cabe aqui uma crítica aos fabricantes: alguns caracteres não acessáveis pelo


```

430   IF F/22=F\22 THEN LINE(45,F+10)
                                     -(238,F+10)
440 NEXT F
450 RETURN
460 '
470 '   linhas verticais
480 '
490 FOR F=33 TO 238 STEP 12
500   IF F>33 THEN LINE (F,0)-(F,12)
510   LINE(F,12)-(F,187)
520   IF F>33 THEN LINE(F+1,12)-
                                     (F+1,187)
530 NEXT F
540 LINE (F-11,0)-(F-11,12)
550 RETURN
560 '
570 '   desenha T
580 '
590 FOR F=45 TO 226 STEP 12
600   FOR G=22 TO 176 STEP 22
610     LINE(F+6,G)-(F+6,G+10)
620     LINE(F+6,G+4)-(F+12,G+4)
630     LINE(F+9,G+4)-(F+9,G+10)
640   NEXT G
650 NEXT F
660 RETURN

```

TOTAL = 84F



7.7 - SORTEIO DE PALAVRAS DA ROM

O programa apresentado a seguir sorteia uma palavra reservada do BASIC MSX a partir da digitação de alguma tecla pelo usuário. Você pode usá-la em outros programas para obter palavras ao acaso.

```
100 ' rotina de sorteio
110 A$=INPUT$(1) : BEEP
120 A=65536!*RND(-TIME)
130 AZ=163*RND(A)+1
140 EN=14962 : A$="" : I=65 : C=0
150 A$=A$+CHR$(I)
160 P=PEEK(EN) : Q=PEEK(EN+1)
170 P$=CHR$(P)
180 IF P<128 THEN A$=A$+P$ : GOTO 240
190 A$=A$+CHR$(P-128)
200 EN=EN+1 : C=C+1
210 IF C=AZ THEN PRINT A$ : RUN
220 IF PEEK(EN+1)=0 THEN 240
230 A$="" : A$=A$+CHR$(I)
240 IF PEEK(EN)<>0 THEN 280
250 A$="" : I=I+1 : Q$=CHR$(I)
260 IF Q$="J" OR Q$="Q" THEN 280
270 A$=A$+Q$
280 EN=EN+1
290 IF EN<=15649 THEN 160
```

TOTAL = 4827

BIBLIOGRAFIA RECOMENDADA:

Coleção de Programas para MSX v.2 - páginas 50 a 55.

7.8 - SORTEIO DE PALAVRAS EM LINHAS "DATA'S"

O programa apresentado abaixo sorteia uma palavra entre uma coleção delas inseridas em linhas datas. Experimente acrescentar ao programa outras linhas DATA's com mais palavras. Note que é essencial que a última palavra seja "FIM" para que o programa pare de tentar ler mais palavras.

```
100 ' rotina de sorteio
110 A%=INPUT$(1) : BEEP :RESTORE
120 A=65536!*RND(-TIME) : FZ=0
130 READ A% : IF A%="FIM" THEN 150
140 FZ=FZ+1 : GOTO 130
150 A%=FZ*RND(A)+1 : RESTORE
160 FOR FZ=1 TO A%
170 READ A%
180 NEXT FZ
190 PRINT A%
200 GOTO 110
1000 REM
1010 DATA BABA,CACA,DADA,FAFA,GAGA,HAHA
1011 DATA JAJA,KAKA,LALA,MAMA,NANA,PAPA
1012 DATA PAPA,RARA,SASA,TATA,UUAU,VAVA
1013 DATA XAXA,ZAZA,NICK,GUTT,MINHOLETA
1020 DATA FIM
```

```
934
132
11610
21004
2006
18733
344E
4304
4509
480E
400F
4E04
630A
1908
8687
9332
966F
```

TOTAL = 966F



7.9 - MAPEANDO LINHAS REM

O programa apresentado a seguir vasculha a memória RAM do micro a partir de &H8000 a procura de pseudo-instruções REM. Ao encontrar no programa um REM (ou '), a linha em que ele se encontra terá seu número apresentado na tela.

Após digitá-lo, grave-o no formato ASCII e com a numeração bem alta (como usamos a seguir). Para usá-lo, faça um MERGE com o programa que estiver na memória e comande:

RUN 65100

65100	REM			115			
65110	REM	PROCURA	LINHAS	REM	974		
65120	REM			880			
65130	PRINT	CHR\$(12)		751			
65140	EI	=	32769!	1340			
65150	B1	=	PEEK (EI)	1944			
65160	B2	=	PEEK (EI + 1)	2176			
65170	B3	=	PEEK (EI + 2)	2368			
65180	B4	=	PEEK (EI + 3)	3110			
65190	PL	=	B1 + 256*B2	3928			
65200	NL	=	B3 + 256*B4	4480			
65210	IF	B1=0	AND	B2=0	THEN	350	5010
65220	PRINT	"=====		6840			
"							
65230	PRINT	"	NUL	:	"	NL	7768
65240	PRINT	"	CPL	:	"	PL - EI	8924
65250	PRINT	"	EPL	:	"	"&H";HEX\$(PL)	9200
65260	FOR	F	=	(EI + 4)	TO	(PL - 2)	9368
65270	IF	PEEK(F)	<	&H8F	THEN	310	10110
65280	PRINT	"&H";HEX\$(F);	"	=	REM";		1082
65290	PRINT	CHR\$(7)					1680
65300	F	=	PL - 2				1975
65310	NEXT	F					2125
65320	PRINT	"=====		10210			
"							
65330	EI	=	PL				2471
65340	GOTO	150					2521
65350	END						2598

TOTAL = 2038

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.A - MAPEANDO VARIÁVEIS

O programa apresentado a seguir vasculha a memória RAM do micro a partir de &H8000 a procura de uma dada variável. Se ela existir no programa, a linha em que ela se encontra terá seu número apresentado na tela.

Após digitá-lo, grave-o no formato ASCII e com a numeração bem alta (como usamos a seguir). Para usá-lo, faça um MERGE com o programa que estiver na memória e comande:

RUN 65000

```
65000 REM                                     0074
65001 REM MAPA DE VARIÁVEIS                 0075
65002 REM                                     0076
65003 PRINT CHR$(12)                        0077
65004 POKE &HFCAB,1                          0078
65005 INPUT "VARIÁVEL";V$                   0079
65006 EI = 32769!                            0080
65007 B1 = PEEK(EI) : B2 = PEEK(EI + 1)     0081
)
65008 B3 = PEEK(EI+2) : B4 = PEEK(EI + 3)   0082
)
65009 PL = B1 + 256*B2 : NL = B3 + 256*B4   0083
4
65010 IF B1=0 AND B2=0 THEN 65024           0084
65011 IF PEEK(EI+4)=&H8F THEN 65022        0085
65012 IF PEEK(EI+5)=&H8F THEN 65022        0086
65013 L$="" : IP=0                          0087
65014 FOR F = (EI+4) TO (PL-2)              0088
65015 IF PEEK(F)=34 THEN IP=(IP+1)         0089
65016 IF IP/2<>IP\2 THEN 65018             0090
65017 L$ = L$ + CHR$(PEEK(F))              0091
65018 NEXT F                                0092
65019 L=INSTR(L$,V$)                         0093
65020 IF L=0 THEN 65022                     0094
65021 PRINT ">>";STR$(NL)+" <<",          0095
65022 EI=PL                                  0096
65023 GOTO 65007                             0097
65024 POKE &HFCAB,0                          0098
65025 END                                    0099
```

TOTAL = 0099

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.8 - MAPEANDO LINHAS DE UM PROGRAMA

O pequeno programa listado a seguir mapeia as linhas de um programa em BASIC presente na memória do micro. Ele deve ser gravado em ASCII para poder ser emendado (com o MERGE) ao final dos programas que se deseja mapear.

Para usá-lo, após ter feito o MERGE, basta comandar:

```
RUN 60000
```

Os dados serão apresentados em 4 colunas. A 1ª conterà o nº da linha; a 2ª conterà seu endereço inicial; a 3ª seu endereço final e a 4ª o comprimento da linha em bytes.

```
60000 REM MAPEADOR DE LINHAS          15F
60010 EPL=32769!                       040
60020 PRINT"LINHA  E.I.    E.F      Comp. 152H
"
60030 PRINT"-----"                 0004
60040 NL=PEEK(EPL+2)+256*PEEK(EPL+3)   2700
60050 PRINT NL;"=>" ;HEX$(EPL);" a "; 0000
60060 C=EPL+1:EPL=PEEK(EPL)+256*PEEK(C) 0000
60070 PRINT HEX$(EPL-1);" =>" ;EPL-C+1 0001
60080 IF PEEK(EPL)=0 THEN END          0000
60090 GOTO 60040                       0000
```

```
TOTAL = 7020
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.C - GERANDO LINHAS DATA

Muitas vezes nos deparamos com situações em que necessitamos armazenar os bytes de uma região da memória em linhas DATA. O programa apresentado a seguir gera na tela, na impressora, em disquete ou em cassete, as linhas DATA correspondentes a uma certa região da memória.

Ao ser executado o programa pede a introdução do número da primeira linha DATA a ser gerada e os endereços inicial e final da área da memória a ser lida.

Na forma como está o programa a seguir, a listagem das linhas DATAS é apresentada na tela. Para fazer com que ela seja enviada à impressora, ao drive ou ao cassete, deve-se alterar a linha 130, substituindo o nome "CRT:" pelo do dispositivo desejado.

```
100 REM GERADATA2
110 REM
120 CLEAR 1000
130 OPEN"CRT:DATAS" FOR OUTPUT AS #1
140 DEFSNG A-Z
150 L$ = ""
160 INPUT"Entre a linha inicial";L
170 IF L<0 OR L>65000! THEN 160
180 INPUT"Entre o endereço inicial";I$
190 I = VAL("&H"+I$)
200 IF I<&H8000 OR I>&HFFFF THEN 190
210 INPUT"Entre o endereço final";F$
220 F = VAL("&H"+F$)
230 IF F<I OR F>&HFFFF THEN 220
240 L$ = L$ + STR$(L)+" DATA "
250 FOR N=I TO F
260   X$ = "00" + HEX$(PEEK(N))
270   X$ = RIGHT$(X$,2)
280   X = X + 1
290   L$ = L$ + X$
300   IF X/8<>X\8 THEN 350
310   PRINT #1,L$
320   L = L + 10
330   L$ = STR$(L)+" DATA "
340   GOTO 360
350   L$ = L$ + ", "
360 NEXT N
370 IF RIGHT$(L$,1)<>"," THEN END
380 L$ = LEFT$(L$,LEN(L$)-1)
390 PRINT #1,L$
```

7.D - TROCANDO TOKENS NUM PROGRAMA

Enfrentar o problema de transformar todos os comandos de um programa em outro comando não é uma tarefa das mais agradáveis, principalmente quando o programa é grande.

A rotina a seguir coloca um programa em Linguagem de Máquina a partir do endereço &HC000. Quando executada ela varre a área do programa em BASIC trocando os códigos iguais ao contido no endereço &HC09C pelo código contido em &HC09B.

Você pode usar o programa abaixo como uma rotina em BASIC, usando-o com um "MERGE", ou gravá-lo em binário e "pokear" os códigos da tokens manualmente.

Os códigos de todas as tokens pode ser encontrados no livro "APROFUNDANDO-SE NO MSX", na páginas 67 a 71 ou usando o programa da figura 1.4 do livro "PROGRAMAÇÃO AVANÇADA EM MSX", nas páginas 11 e 12.

Lembre-se que as tokens de funções tem sempre o bit 7 setado antes de serem inseridas num programa, portanto, se você desejar substituir tokens de funções deverá antes acrescentar &H80 ao seu código. Por exemplo, para substituir a token da função seno (&H09) pela da função cosseno (&H0C) você deverá "pokear" os valores &H89 e &H8C respectivamente nos endereços &HC09C e &HC09B.

50000	'*****	63F
50010	'* BY THE DOCTOR LUZ *	F84
50020	'*****	1920
50030	FOR L=&HC000 TO &HC09F	206F
50040	READ A\$:POKE L,VAL("&H"+A\$)	2217
50050	NEXT	2353
50060	DEFUSR=&HC000	2490
50070	PRINT"RODAR PROGRAMA ?"	262F
50080	A\$=INPUT\$(1)	2763
50090	IF A\$="S" THEN GOTO 50100 ELSE END	2892
50100	INPUT"CODIGO DA TOKEN";A	3022
50110	INPUT"NOVO CODIGO";B	3157
50120	POKE &HC09C,A	3283
50130	POKE &HC09B,B	3410
50140	A=USR(0)	3542
50150	LIST	3663
50160	DATA DD,21,FF,7F,DD,23,21,73	3780
50170	DATA C0,DD,7E,00,06,14,BE,28	3903
50180	DATA 21,23,23,10,F9,DD,7E,00	4023
50190	DATA FE,22,28,39,5F,3A,9C,C0	4143
50200	DATA DD,BE,00,20,DF,3A,9B,C0	4263

50210	DATA	DD,77,00,7B,FE,84,28,36	FFFF
50220	DATA	18,D2,DD,7E,00,FE,00,20	0055
50230	DATA	0E,DD,7E,01,FE,00,20,07	1088
50240	DATA	DD,7E,02,FE,00,28,09,23	E011
50250	DATA	7E,47,DD,23,10,FC,18,B4	F928
50260	DATA	DD,22,9D,C0,C9,DD,23,DD	5E
50270	DATA	7E,00,FE,16,28,A6,FE,00	627
50280	DATA	20,F3,DD,2B,18,9E,DD,23	F8E
50290	DATA	DD,7E,00,FE,00,20,F7,DD	0188
50300	DATA	2B,18,91,00,04,0B,03,0C	2008
50310	DATA	03,0D,03,0E,03,0F,02,11	8045
50320	DATA	01,12,01,13,01,14,01,15	407F
50330	DATA	01,16,01,17,01,18,01,19	5E0C
50340	DATA	01,1A,01,1C,03,1D,04,1F	7251
50350	DATA	0B,FF,02,00,00,00,00,20	7846

TOTAL = 7846



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 2.
 Programação Avançada em MSX - capítulo 1.

7.E - AUMENTANDO A VELOCIDADE DE EXECUÇÃO EM BASIC

Para ilustrar alguns métodos de se aumentar a velocidade de execução de programas em BASIC vamos usar uma pequena rotina que calcula os números primos compreendidos entre 2 e 1000. Rode o programa a seguir e anote o tempo que ele levou para rodar.

```
100 TIME=0
120 DIM A(2000)
130 FOR I=2 TO 1000
140 IF A(I)<>0 THEN GOTO 190
150 PRINT I : P=P+1
160 FOR N=2*I TO 2000 STEP I
170 A(N)=1
180 NEXT N
190 NEXT I
200 PRINT : PRINT "TEMPO:";TIME/60
210 END
```

```
2000
400
600
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
2800
3000
3200
3400
3600
3800
4000
```

TOTAL = 3377

O MSX trabalha normalmente com números em dupla precisão (14 casas) e faz os cálculos em BCD para reduzir ao máximo os erros de arredondamento. Com isso, ele perde mais tempo que outros micros menos inteligentes, como os APPLE's, os TRS-80, etc. Mas podemos fazer com que ele use precisão simples (6 casas), uma vez que as operações envolvidas são apenas a adição e a multiplicação de números inteiros. Isso faz o programa ficar ligeiramente mais rápido. Experimente acrescentar ao programa anterior a linha a seguir, rode-o novamente e anote o tempo de execução.

```
110 DEFSNG A-Z
```

Em nosso caso específico, sabemos que os dados manipulados pelo programa são todos inteiros. Todos os números primos são inteiros. Sabendo disso, podemos fazer com que o MSX diminua a precisão de seu cálculos um pouco mais. Altere a linha 110 como mostrada a seguir e rode mais uma vez o programa. Você deverá obter um tempo de execução bem menor.

```
110 DEFINT A-Z
```

Isso ainda não é tudo. Existem alguns procedimentos que o micro faz ao executar pela primeira vez uma linha do programa que não são refeitos se ela for executada outras vezes. Altere o programa, deixando-o como mostrado a seguir e rode-o mais uma vez.

```
0 TIME=0:KEYOFF:LOCATE0,0,0
1 DEFINTA-Z:POKE&HF3B1,3:DIMA(2000)
2 FORI=2T0100
3 IFA(I)<>0THEN7
4 PRINTI:P=P+1
5 FORN=2*IT02000STEP1
6 A(N)=1:NEXT
7 NEXT
8 PRINT:PRINT"TEMPO:";TIME/60
9 END
```

00:00
00:03
00:08
00:12
00:15
00:18
00:21
00:24
00:27
00:30
00:33
00:36
00:39
00:42

10:00 = 20:00

Além da renumeração, da supressão dos espaços em branco e da compactação de várias instruções por linha, o programa foi otimizado com mais algumas instruções.

A linha 0, além de apagar as teclas de funções, apaga definitivamente o cursor da tela.

Na linha 1, a instrução POKE na posição &HF3B1 faz com que a tela tenha apenas 3 linhas! Isso reduz bastante o tempo de 'SCROLL' da tela quando ela já está cheia.

Por fim, as instruções NEXT sem especificação do parâmetro nas linhas 6 e 7 também reduzem um pouco o tempo de execução.

Se você tiver um APPLE, um TRS-80 ou mesmo um PC, experimente rodar neles um programa equivalente e compare o tempo gasto com o do MSX. A não ser no caso de um PC com clock de 8 Mhz, o MSX será o mais rápido!

BIBLIOGRAFIA RECOMENDADA:

Curso de Basic MSX v.1 - páginas 37 a 39.
Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.F - "SEARCH", PESQUISADOR DE STRINGS

O programa apresentado a seguir gera, a partir do endereço &HE000, uma rotina em Linguagem de Máquina que pode ser chamada pelo BASIC para vasculhar a memória a procura de uma sequência de caracteres (string). Após gerar a rotina em L.M., o programa a grava em fita ou em disco com o nome SEARCH ou "SEARCH.BIN", respectivamente.

Para testar o programa, após ele ter sido executado, comande:

```
PRINT,HEX$(USR0("A L E P H"))
```

Ao fazer isso, a string "A L E P H" será procurada na memória e o endereço em que ela for encontrada será mostrado em hexadecimal na tela.

Após ter salvo o programa em BASIC, você pode apagá-lo da memória que mesmo assim a rotina em L.M. permanecerá funcionando. Para usá-la a sintaxe é sempre a mesma: basta chamar a função USR0 passando como parâmetro a string a ser procurada.

```
10 FOR FZ=&HE000 TO &HE14B
20 READ A$ : POKE FZ,VAL("&H"+A$)
30 NEXT FZ : DEFUSR0=&HE000
40 BSAVE"SEARCH.BIN",&HE000,&HE14B
50 END
100 DATA FE,03,C0,D5,11,4D,E0,21
110 DATA 4C,E0,36,00,01,FF,00,ED
120 DATA B0,D1,06,00,1A,4F,C5,13
130 DATA 1A,6F,13,1A,67,11,4C,E0
140 DATA ED,B0,21,00,00,01,FF,FF
150 DATA 11,4C,E0,1A,ED,B1,28,01
160 DATA C9,13,1A,B7,28,09,ED,A1
170 DATA E0,28,F6,1B,03,18,E9,AF
180 DATA C1,ED,42,22,F8,F7,3E,02
190 DATA 32,63,F6,C9,41,42,56,41
200 DATA 42,56,53,00,00,00,00,00
210 DATA 00,00,00,00,00,00,00,00
220 DATA 00,00,00,00,00,00,00,00
230 DATA 00,00,00,00,00,00,00,00
240 DATA 00,00,00,00,00,00,00,00
250 DATA 00,00,00,00,00,00,00,00
260 DATA 00,00,00,00,00,00,00,00
270 DATA 00,00,00,00,00,00,00,00
280 DATA 00,00,00,00,00,00,00,00
290 DATA 00,00,00,00,00,00,00,00
300 DATA 00,00,00,00,00,00,00,00
```

0000
0005
0010
0015
0020
0025
0030
0035
0040
0045
0050
0055
0060
0065
0070
0075
0080
0085
0090
0095
0100
0105
0110
0115
0120
0125
0130
0135
0140
0145
0150
0155
0160
0165
0170
0175
0180
0185
0190
0195
0200

310	DATA	00,00,00,00,00,00,00,00	8E08
320	DATA	00,00,00,00,00,00,00,00	4E00
330	DATA	00,00,00,00,00,00,00,00	8E07
340	DATA	00,00,00,00,00,00,00,00	0E00
350	DATA	00,00,00,00,00,00,00,00	8E06
360	DATA	00,00,00,00,00,00,00,00	8E05
370	DATA	00,00,00,00,00,00,00,00	8E04
380	DATA	00,00,00,00,00,00,00,00	8E03
390	DATA	00,00,00,00,00,00,00,00	8E02
400	DATA	00,00,00,00,00,00,00,00	8E01
410	DATA	00,00,00,00,00,00,00,00	0E00
420	DATA	00,00,00,00,00,00,00,00	0E00
430	DATA	00,00,00,00,00,00,00,00	0E00
440	DATA	00,00,00,00,00,00,00,00	0E00
450	DATA	00,00,00,00,00,00,00,00	0E00
460	DATA	00,00,00,00,00,00,00,00	0E00
470	DATA	00,00,00,00,00,00,00,00	0E00
480	DATA	00,00,00,00,00,00,00,00	0E00
490	DATA	00,00,00,00,00,00,00,00	0E00
500	DATA	00,00,00,00,00,00,00,00	0E00
510	DATA	00,00,00,00,00,00,00,00	0E00

Experimente, após ter rodado o programa ao menos uma vez, desligar o micro, ligá-lo novamente, carregar o programa SEARCH gravado em binário para a memória do micro e comandar:

```
DEFUSR0=&HE000 : ? HEX$(USR0("color"))
```

Você deverá obter como resultado o endereço da ROM onde existe essa palavra.

7.6 - RECUPERANDO PROGRAMAS APAGADOS COM NEW

O programa apresentado a seguir deve ser digitado e gravado. Ao ser executado ele gera e grava um programa em Linguagem de Máquina capaz de recuperar programas em BASIC apagados da memória do micro com o comando NEW.

Existem alguns raros casos em que a rotina não funcionará, porém certamente esse casos constituem menos de 1% das situações reais.

Para testar o programa, após tê-lo digitado, gravado e executado, comande NEW.

O programa foi "apagado" da memória. NÃO FAÇA NADA AINDA para evitar perder dados que ainda estão na memória do micro (apesar de ele não saber disso)! A primeira coisa a ser feita nessa situação é carregar o programa em Linguagem de Máquina gravado pelo programa em BASIC que você digitou. Comande:

```
BLOAD"WENNEW.BIN",R
```

Isso será o suficiente para recuperar seu programa em BASIC.

```
100 FOR F=&HE000 TO &HE032
110   READ A$:POKE F,VAL("&H"+A$)
120 NEXT F
130 BSAVE"WENNEW.BIN",&HE000,&HE032
140 END
150 DATA F3,21,04,80,23,7E,FE,00
160 DATA 20,FA,23,23,7E,FE,80,2B
170 DATA 20,F2,22,01,80,EB,1A,6F
180 DATA 13,1A,67,7E,FE,00,20,F5
190 DATA 23,22,C2,F6,22,C4,F6,22
200 DATA C6,F6,21,01,80,22,76,F6
210 DATA FB,C9,FF,00,00,00,00,00
```

```
588
098
227
1770
1350
241E
2020
4059
5100
642E
7981
8891
```

```
TOTAL = 8891
```

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulo 1.
Programação Avançada em MSX - capítulo 1.

7.H - REDEFININDO MENSAGENS DE ERROS

O programa apresentado a seguir permite a redefinição das mensagens de erro do BASIC. Na verdade, ele acrescenta novas mensagens às já existentes. Apenas a título de exemplo, usamos mensagens do tipo "MENSAGEM 1", "MENSAGEM 2", etc. Entretanto você pode criar suas próprias mensagens desde que sempre as termine pelo caractere "#".

100	REM		173
110	REM	ERROS ALTERNATIVOS	178D
120	REM		890
130	CLEAR	200,&HD000	044
140	FOR	F=0 TO 61	1099
150	READ	A\$: A=VAL("&H"+A\$)	1807
160	POKE	&HD000+F,A	1102
170	NEXT	F	207F
180	READ	A\$	2384
190	IF	A\$="FIM" THEN 250	2452
200	FOR	G=1 TO LEN(A\$)	324E
210	POKE	&HD000+F,ASC(MID\$(A\$,G,1))	3701
220	F=F+1		4307
230	NEXT	G	4635
240	GOTO	180	4956
250	POKE	&HD000+F,255	4788
260	DEFUSR	=&HD000	53F3
270	S=USR	(0)	5903
280	BSAVE	"ERROS",&HD000,&HD000+F	6592
290	END		6703
300	REM		6900
310	REM	DADOS	7048
320	REM		7240
330	DATA	11,0D,D0,21,FD,FE,36,C3	81F7
340	DATA	23,73,23,72,C9,E5,F5,C5	8794
350	DATA	21,3D,D0,7E,23,FE,FF,28	907C
360	DATA	16,FE,23,20,F6,0D,20,F3	96F3
370	DATA	7E,23,FE,FF,28,09,FE,23	9ED2
380	DATA	28,05,CD,A2,00,18,F1,3E	AE14
390	DATA	0D,CD,A2,00,3E,0A,CD,A2	0028
400	DATA	00,C1,F1,E1,C9,23	00E7
410	DATA	"MENSAGEM 1 #"	E104
420	DATA	"MENSAGEM 2 #"	F617
430	DATA	"MENSAGEM 3 #"	50
440	DATA	"MENSAGEM 4 #"	563
450	DATA	"MENSAGEM 5 #"	E58
460	DATA	"MENSAGEM 6 #"	1773
470	DATA	"MENSAGEM 7 #"	1FEC
480	DATA	"MENSAGEM 8 #"	333F

490	DATA	"MENSAGEM 9	#"	4490
500	DATA	"MENSAGEM A	#"	5AE4
510	DATA	"MENSAGEM B	#"	6E44
520	DATA	"MENSAGEM C	#"	8010
530	DATA	"MENSAGEM D	#"	8700
540	DATA	"MENSAGEM E	#"	8080
550	DATA	"MENSAGEM F	#"	9210
560	DATA	"MENSAGEM G	#"	9810
570	DATA	"MENSAGEM H	#"	A764
580	DATA	"MENSAGEM I	#"	B900
590	DATA	"MENSAGEM J	#"	0020
600	DATA	"MENSAGEM K	#"	E100
610	DATA	"MENSAGEM L	#"	F604
620	DATA	"MENSAGEM M	#"	25
630	DATA	"MENSAGEM N	#"	544
640	DATA	"MENSAGEM O	#"	E85
650	DATA	"MENSAGEM P	#"	1750
660	DATA	"FIM"	#"	1AE3

TOTAL = 1AE3

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - páginas 71 a 76 e 82.

7.1 - SOMA SINTÁTICA

Os erros mais frequentes cometidos durante a transcrição de programas listados para a memória do computador são devidos a digitação incorreta por parte do leitor. Raras as vezes em que o erro é devido a falhas de impressão e mais raras ainda as ocasiões em que o programa está realmente com algum erro lógico.

Mesmo considerando o fato de que todos os programas apresentados neste livro estão em BASIC são, em sua maioria, curtos e de fácil digitação, a probabilidade de falhas durante a digitação é algo considerável. Uma simples vírgula substituída involuntariamente por um ponto pode por todo um programa a perder.

Pensando em como diminuir a ocorrência de erros de digitação desenvolvemos uma pequena rotina em Linguagem de Máquina capaz de checar o programa na memória. Antes de continuarmos, digite o programa a seguir exatamente como o listamos abaixo, sem tirar nem por absolutamente nada e tomando o máximo cuidado para não cometer erro algum. Depois, salve o programa em disco ou em fita cassete.

```
100 REM-----
110 REM SOMA SINTÁTICA
120 REM                               Rubens Jr.
130 REM-----
140 FOR F=&HE000 TO &HE0B5
150   READ A% : POKE F,VAL("&H"+A%)
160 NEXT F : DEFUSR0=&HE000
170 BSAVE"SOMSIN.BIN",&HE000,&HE0B5
180 SS = USR0(0) : END
190 REM-----
200 DATA 3E,00,32,B4,E0,21,00,00
210 DATA 22,B0,E0,2A,76,F6,CD,F9
220 DATA 10,CD,63,E0,7A,B3,CA,4B
230 DATA E0,ED,53,B2,E0,CD,63,E0
240 DATA CD,68,E0,2B,2B,CD,7E,E0
250 DATA 23,E5,ED,5B,B2,E0,1B,B7
260 DATA ED,52,E1,20,F0,2A,B0,E0
270 DATA CD,94,E0,3E,0D,CD,A2,00
280 DATA 3E,0A,CD,A2,00,2A,B2,E0
290 DATA C3,0E,E0,21,A3,E0,CD,78
300 DATA 66,2A,B0,E0,CD,94,E0,3E
310 DATA 02,32,63,F6,2A,B0,E0,22
320 DATA F8,F7,C9,5E,23,56,23,C9
330 DATA E5,EB,CD,12,34,3E,09,CD
340 DATA A2,00,3E,3D,CD,A2,00,3E
```



```

350 DATA 09,CD,A2,00,E1,C9,5E,16
360 DATA 00,3A,B4,E0,3C,32,B4,E0
370 DATA AB,5F,E5,2A,B0,E0,19,22
380 DATA B0,E0,E1,C9,22,F8,F7,3E
390 DATA 02,32,63,F6,CD,22,37,CD
400 DATA 78,66,C9,0D,0A,0D,0A,54
410 DATA 4F,54,41,4C,09,3D,09,00
420 DATA 00,00,00,00,00,42,41,4F

```

```

3445
4870
6185
7661
8568
887E
8F45
9896

```

TOTAL = 9896

Com o programa já gravado, comande RUN. Você deverá obter na tela uma listagem terminada com a mensagem:

TOTAL = 9896

Se isso não ocorreu, carregue o programa SOMA SINTÁTICA (previamente gravado) para a memória do micro e confira-o novamente, pois há alguma coisa errada nele.

Após obter o valor correto para a soma TOTAL do programa, grave-o definitivamente.

Um programa em BASIC é armazenado na memória do micro como uma sequência de bytes. A rotina que gera a SOMA SINTÁTICA vai lendo a memória e para cada linha do programa em BASIC calcula uma soma "ponderada" dos bytes que a constituem. O valor com que cada byte participa na soma de cada linha depende de seu próprio valor e da posição que ele ocupa na linha. Para verificar isso, comande NEW e introduza a linha a seguir:

```
10 PRINT "AMOR"
```

A seguir, comande:

```
SS = USR(0)
```

Você deverá obter a soma TOTAL = 245

Agora, altere a linha 10 para:

```
10 PRINT "ROMA"
```

Comande mais uma vez:

```
SS = USR(0)
```

A soma TOTAL se alterou para **238** e isso aconteceu apesar de as duas linhas terem exatamente os mesmos caracteres.

Quando o programa tem várias linhas, o valor total de cada uma é adicionado na soma da linha subsequente. Desse modo, a soma total de um programa é sempre igual a soma indicada em sua última linha e quando um erro é cometido numa determinada linha, as somas de todas as linhas subsequentes são alteradas.

Note, porém, que a rotina não é infalível. Calcule, como contra-exemplo, as somas TOTAIS das duas linhas abaixo:

```
10 PRINT"ACD"
```

e

```
10 PRINT"ABC"
```

Apesar de as linhas serem diferentes, suas SOMAS são iguais!

Quase todos os demais programas deste livro estão acompanhados de sua soma TOTAL, obtida através do programa SOMA SINTÁTICA rodando num MSX EXPERT versão 1.1. Se o seu MSX for de outro tipo, a soma TOTAL dos programas poderá resultar diferente da que apresentamos, pois alguns caracteres (como o Ç, por exemplo) podem ter códigos diferentes. Cuidado, portanto, se esse for o seu caso!

Outra situação peculiar ocorre com programas que possuem GOTO ou GOSUB. Se calcularmos a soma total desses programas antes de executá-los e depois de executá-los obteremos valores diferentes, pois o interpretador BASIC da ROM do MSX usa uma técnica de otimização que altera as linhas com GOTO ou GOSUB após a primeira passada por elas (veja as páginas 16 e 17 do livro PROGRAMAÇÃO AVANÇADA EM MSX para maiores detalhes sobre essa técnica).

Neste livro, as somas TOTAIS apresentadas foram sempre calculadas ANTES de se executar o programa sequer uma única vez!

7.J - PSEUDO-RAMDISK

O programa apresentado a seguir permite o uso dos 32 Kbytes de RAM não disponíveis para o BASIC (entre os endereços 0 e &H8000). Após digitá-lo e gravá-lo, certificando-se de que ele esteja correto, rode-o. Com isso uma rotina em Linguagem de Máquina estará pronta para ser usada com outros programas, desde que ele não se sobreponham à área de memória entre &HD000 e &HD078.

A rotina permite que os programas presentes na RAM disponível do micro sejam passados para a RAM oculta e posteriormente recuperados de volta para a RAM disponível.

Para passar da RAM ativa para a RAM oculta deve-se comandar:

```
POKE 0,USR0(0%)
```

Fazendo isso você pode carregar outro programa na memória e usá-lo normalmente, pois o programa anterior estará "salvo" na RAM oculta.

Para recuperar o programa da RAM oculta para a RAM ativa deve-se comandar:

```
POKE 0,USR0(1%)
```

Com isso o programa "salvo" estará novamente presente na RAM ativa do micro.

Além desses recursos podemos também "trocar" o conteúdo da RAM ativa e da RAM oculta. Se temos um programa PROGR1 "salvo" na RAM oculta e um outro programa PROGR2 presente da RAM ativa, para trocá-los de posições devemos comandar:

```
POKE 0,USR0(2%)
```

Faça alguns testes com programas curtos para se habituar aos comandos.

```
100 SCREEN 0:WIDTH 39:CLEAR 200,&HD000
110 FOR F=&HD000 TO &HD076
120   READ A$:POKE F,VAL("&H"+A%)
130 NEXT F
140 DEFUSR0=&HD000
150 DATA F3,F5,C5,D5,E5,ED,73,FE
160 DATA F3,FE,02,20,5E,23,23,7E
170 DATA FE,00,20,0B,21,00,80,11
180 DATA 00,00,CD,53,D0,18,43,FE
```

```
595
188
128E
1418
1980
2291
2858
3054
4009
```

```

190 DATA 01,20,0B,21,00,00,11,00
200 DATA 80,CD,53,D0,18,34,FE,02
210 DATA 20,39,CD,53,D0,01,FF,5F
220 DATA 11,00,00,21,00,80,1A,32
230 DATA 75,D0,7E,12,3A,75,D0,77
240 DATA 23,13,0B,78,B1,20,EF,DB
250 DATA A8,18,14,DB,A8,47,CB,3F
260 DATA CB,3F,CB,3F,CB,3F,80,D3
270 DATA A8,C9,01,FF,5F,ED,B0,E6
280 DATA F0,D3,A8,ED,7B,FE,F3,E1
290 DATA D1,C1,F1,FB,C9,00,00,00

```

```

F800
1102
8800
8800
F800
5000
A500
8800
080E
080E
E-43

```

Você pode também programar as teclas de funções com os comandos de SALVAR, RECUPERAR e TROCAR programas na RAM oculta para facilitar a operação do programa em L.M.

BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - capítulos 0 e 1.
 Programação Avançada em MSX - capítulo 1.

7.K - ROTACIONANDO CARACTERES

A rotina apresentada a seguir permite a escrita com letras 'deitadas' na tela. Melhor do que tentar entender o que o programa faz, é rodá-lo e observar o efeito.

O programa em BASIC serve para gerar uma rotina em Linguagem de Máquina. Ao ser chamada por uma instrução `USR`, o caractere cujo código é passado como parâmetro será "girado" de 90°.

Analise as linhas de 200 a 240 para entender melhor como a rotina deve ser usada.

10 REM	00
20 REM GIRA CARACTERES NA SCREEN 1	010
30 REM	020
100 CLEAR 300,&HCFFF	040
110 SCREEN 1	070
120 DEFINT A	110
130 DEFUSR0=&HD000	150
140 FOR F=&HD000 TO &HD033	190
150 READ A\$:A=VAL("&H"+A\$)	230
160 POKE F,A	270
170 NEXT F	310
180 INPUT "MENSAGEM";A\$	350
190 PRINT	390
200 FOR F=1 TO LEN(A\$)	430
210 A=ASC(MID\$(A\$,F,1))*8	470
220 PRINT CHR\$(A/8)	510
230 X=USR0(A)	550
240 NEXT F	590
250 GOTO 250	630
260 REM	670
270 DATA 23,23,5E,23,56,62,68,E5	710
280 DATA 01,08,00,11,2C,D0,D5,CD	750
290 DATA 59,00,D1,E1,06,08,C5,06	790
300 DATA 08,D5,EB,CB,26,CB,1F,23	830
310 DATA 10,F9,EB,CD,4D,00,23,D1	870
320 DATA C1,10,EB,C9,00,00,00,00	910
330 DATA 00,00,00,00	950

TOTAL = 950

BIBLIOGRAFIA RECOMENDADA:

Linguagem de Máquina MSX - páginas 140 a 143.

7.1 - ROTINA DE ENTRADA COM INKEY\$

A rotina apresentada a seguir permite a introdução de dados em programas através da instrução INKEY\$.

As linhas de 100 a 170 simulam um programa qualquer. A única linha realmente necessária é a linha 110, onde uma função-string para posicionamento do cursor é definida. A rotina de entrada efetivamente começa na linha 1000.

Ao ser chamada, a sub-rotina da linha 1000 deve receber a LINHA (na variável LI) e a COLUNA (na variável CO) em que os dados introduzidos deverão aparecer na tela. Além disso deve-se também fornecer quantos caracteres poderão ser introduzidos, isto é, o TAMANHO DA LINHA (na variável TL) a ser introduzida.

Para entrar dados com essa rotina, dispõe-se das seguintes funções:

- RETURN - termina a inserção de dados;
- BS ou - Volta uma posição apagando;
- CLS - Apaga a linha já introduzida.

Ao retornar da sub-rotina (quando se digita RETURN), o programa traz na variável L\$ a linha introduzida.

Essa rotina pode ser particularmente útil nos programas de gerenciamento de dados, em que ao serem introduzidas, as informações de cada campo devem ser apresentadas de forma estética na tela.

```
100 SCREEN 0 : WIDTH 39 190E  
110 DEF FNPC$(LI,CO)=CHR$(27)+"Y"+ 15F0  
    CHR$(LI+32)+CHR$(CO+32)  
120 INPUT "LINHA:";LI 190E  
130 INPUT "COLUNA:";CO 1E90  
140 INPUT "Nº CARACTERES:";TL 2570  
150 GOSUB 1000 2238  
160 PRINT : PRINT,,,,,, "L$=";L$ 3788  
170 IF STRIG(0) THEN RUN ELSE 170 3FE8  
1000 ' 4208  
1010 ' SUB-ROTINA PARA ENTRAR LINHA 57EE  
1020 ' 557E  
1030 L$ = "" 6000  
1040 PRINT FNPC$(LI,CO);"-"; 6E8F  
1050 I$=INKEY$ 791E  
1060 IF I$="" THEN 1050 7E8F  
1070 IF I$>="" AND I$<="( THEN 1230 8200  
1080 C=CO+LEN(L$):PRINT FNPC$(LI,C); 8888
```

1090	IF LEN(L\$)<TL THEN PRINT " "	1000
1100	I=ASC(I\$)	1000
1110	IF I (> 8 THEN 1170 ' BS	1000
1120	IF LEN(L\$)=0 THEN 1140	1000
1130	L\$=LEFT\$(L\$,LEN(L\$)-1)	1000
1140	C = CO : PRINT FNPC\$(LI,C);	1000
1150	PRINT L\$;"_"	1000
1160	GOTO 1050	1000
1170	IF I (> 12 THEN 1220 'CLS	1000
1180	L\$=""	1000
1190	PRINT FNPC\$(LI,CO);"_";	1000
1200	PRINT STRING\$(TL-1," ")	1000
1210	GOTO 1050	1000
1220	IF I = 13 THEN RETURN 'RETURN	1000
1230	IF LEN(L\$)<TL THEN L\$=L\$+I\$	1000
1240	IF LEN(L\$)=TL THEN L\$=LEFT\$(L\$,TL-1	1000
) + I\$	
1250	C=C0-1+LEN(L\$)	1000
1260	PRINT FNPC\$(LI,C);	1000
1270	PRINT I\$;	1000
1280	IF LEN(L\$)<TL THEN PRINT"_"	1000
1290	GOTO 1050	1000

TOTAL = 5556

7.M - ROTINA PARA MAIÚSCULAS

O programa apresentado abaixo gera uma pequena rotina em Linguagem de Máquina que passa todas as letras comuns minúsculas de uma string para maiúsculas.

Experimente rodá-lo e introduza algumas seqüências de letras minúsculas. Note que a string é passada como parâmetro da função USR.

100 CLEAR 300,&HBFFF	837
110 FOR I = &HC000 TO &HC021	978
120 READ X\$	887
130 POKE I,VAL("&h"+X\$)	1128
140 NEXT I	1279
150 DATA 3A,63,F6,FE,03,C0,2A,F8	1048
160 DATA F7,7E,B7,C8,47,23,5E,23	8599
170 DATA 66,6B,7E,FE,61,38,07,FE	8608
180 DATA 7B,30,03,D6,20,77,23,10	4508
190 DATA F1,C9	4808
200 DEFUSR0=&HC000	5075
210 INPUT X\$	5248
220 PRINT USR0(X\$)	5256
230 GOTO 210	5261

TOTAL = 5261

7.N - REDUZINDO A TELA A UM CARACTERE

Você sabe que podemos usar o comando WIDTH para as telas de texto de modo a fazê-las ficar com apenas uma coluna. Experimente comandar:

```
WIDTH 1
```

Agora a tela tem apenas uma coluna. Para voltar ao normal, basta usar novamente o comando WIDTH.

Um fato menos conhecido é a possibilidade de fazer com que a tela tenha apenas 1 linha. Para isso, entretanto, não existe um comando dedicado do BASIC e é necessário usar o comando POKE para alterar o valor de uma variável do sistema, a CRTCNT, em &HF3B1.

Experimente comandar:

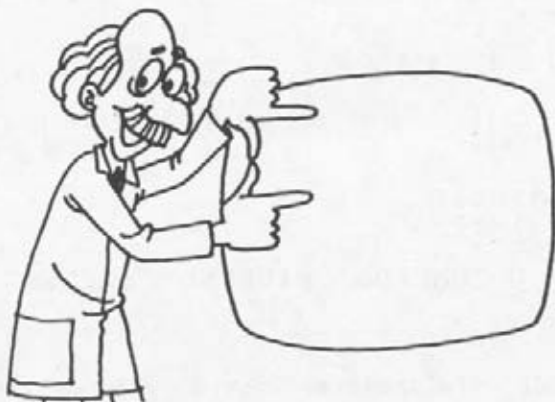
```
POKE &HF3B1,1
```

Com isso a tela deverá ficar com apenas uma linha.

Para reduzir a tela a apenas um caractere, basta usar o par de comandos:

```
WIDTH1:POKE&HF3B1,1
```

Isso, entretanto, tornará o teclado inoperante e você perderá o controle sobre o micro. Portanto, cuidado ao usar esta dica!



BIBLIOGRAFIA RECOMENDADA:

Aprofundando-se no MSX - pagina 47.

7.0 - GRANDES EXPOENTES

Recentemente, astrônomos americanos detectaram o que parecem ser galáxias em formação a 17 bilhões de anos-luz da Terra! Estes objetos estariam quase no limite do Universo observável, cujo raio, hoje, é estimado em 20 bilhões de anos-luz.

Aproveitando que você tem um computador na frente, você resolve calcular o raio do Universo em metros.

O cálculo é simples (pelo menos no computador!), você sabe que a luz se propaga com uma velocidade de 300.000 quilômetros por segundo. Basta portanto multiplicar esta velocidade de 300.000.000 m/s pelo número de segundos em 20 bilhões de anos (lembre-se que o ano tem 365 dias de 24 horas de 60 minutos de 60 segundos).

Digite, então, o seguinte programinha:

```
10 R=3000000000#*20000000000#*365*24*60*60
0
20 PRINT"RAIO DO UNIVERSO=";R;"METROS"
```

Rodando-o, você deve obter a considerável quantia de 1,89216 vezes 10 elevado à potência 26!

Entusiasmado com a rapidez de cálculo do seu MSX, você resolve calcular o volume do Universo (supondo ingenuamente que ele seja esférico).

Lembrando que o volume de uma esfera é dado pela expressão:

$$V = 4/3 \cdot \pi \cdot R^3$$

Você, então, completa seu programinha com as seguintes linhas:

```
30 PI=4*ATN(1)
40 X=(4/3)*PI
50 V=X*R^3
60 PRINT"VOLUME DO UNIVERSO=";V;"m3"
```

Ao rodar seu programa assim incrementado, você tem a decepção de obter um "overflow in 50", pois você "estourou" a capacidade de cálculo do MSX.

Nesta dica, vamos apresentar como contornar o problema de expoentes grandes demais. Basta lembrar

que:

$$\log (AxB) = \log A + \log B$$

$$\log A^n = n \times \log A$$

Substitua as linhas 50 e 60 por:

```
100 LV=LOG(X)+3*LOG(R)
110 LD=LV/LOG(10)
120 E=INT(LD)
130 M=10^(LD-E)
140 A$=STR$(M)+"E"+STR$(E)
150 PRINT"VOL. DO UNIVERSO="";A$;"M.C."
```

Na linha 100 você calcula o logaritmo neperiano do volume (o MSX só trabalha com logaritmos naturais). Na linha 110 você transforma o log natural em log decimal (é só dividir por LOG(10): esta é mais uma dica importante).

Para entender as linhas 120 e 130, lembre-se que se, por exemplo,

$$\log x = 79,4529$$

então

$$x = 10^{0,4529} \times 10^{79}$$

Portanto, toda vez que você se defrontar com números demasiadamente grandes para seu MSX, basta calcular seu logaritmo decimal, pegar a parte inteira com expoente de 10 e descobrir o multiplicando na frente dele, operando como aprendemos.





Como você deve ter percebido, as dicas deste livro foram apresentadas de forma bem prática e resumida, de modo a poderem ser usadas imediatamente. Se você quiser obter maiores detalhes sobre o funcionamento de cada uma delas é recomendável estudar detalhadamente os livros citados na "BIBLIOGRAFIA RECOMENDADA", ao final de cada dica. A seguir apresentamos um resumo do conteúdo de cada um desses livros para melhor orientá-lo.

LINGUAGEM BASIC MSX

Uma "enciclopédia" do BASIC MSX, com a sintaxe, função e exemplo de cada palavra do BASIC MSX.

CURSO DE BASIC MSX v.1

Uma introdução clara e didática ao BASIC residente do MSX, apresentada em 8 aulas com exercícios (e suas respostas!).

COLEÇÃO DE PROGRAMAS PARA MSX v.1 e v.2

Programas didáticos, aplicativos e utilitários explicados passo a passo para que o leitor aprenda a fazer seus próprios programas.

APROFUNDANDO-SE NO MSX

O "best seller" da literatura técnica sobre MSX, com a descrição detalhada da arquitetura da máquina e de cada uma de suas partes.

PROGRAMAÇÃO AVANÇADA EM MSX

Exemplos e rotinas utilitárias em ASSEMBLY ensinando ao leitor como se obtém o máximo das máquinas MSX.

LINGUAGEM DE MÁQUINA MSX

Uma introdução completa e didática aos poderosos recursos da Linguagem de Máquina Z80 aplicada aos micros MSX. Contém as instruções secretas do Z80.

USANDO O DISK DRIVE NO MSX

O MSXDOS, O CP/M para MSX e o DISK BASIC comentados exaustivamente.

SISTEMA DE DISCO PARA MSX

O SOLXDOS e o BASIC de DISCO comentados passo a passo de forma clara e didática.

DRIVES LEOPARD DE 3 1/2"

O primeiro livro sobre drives de 3 1/2" editado no Brasil. Contém todos os recursos do MSXDOS e do DISK BASIC MSX aplicados aos drives de 3 1/2".

Para receber gratuitamente o boletim informativo da ALEPH, contendo dicas de programação, artigos técnicos e informações sobre os últimos lançamentos para seu micro, envie seu nome e endereço completos (incluindo o CEP) para:

EDITORA ALEPH
Caixa Postal: 20.707
01498 São Paulo SP

Se você quiser adquirir todos os programas com mais de 512 bytes listados neste livro já gravados em DISCO (apenas em disco!!!), entre em contato conosco.

Para comprar nossos livros pelo correio, informe-se escrevendo ou telefonando para nós.

Nosso telefone é:

(011) 843-3202

COLEÇÃO MSX



100 DICAS PARA MSX

Ao longo de dois anos trabalhando com micros MSX pudemos atender a milhares (sem exagero!) de dúvidas sobre essas máquinas.

Esquentando a orelha ao telefone, atolando em montanhas de cartas, ou atendendo pessoalmente nossos leitores, surgiram as mais de cem dicas publicadas neste livro, como uma espécie de resposta coletiva.

Apesar de apelarmos frequentemente para o uso da Linguagem de Máquina, todos os programas listados neste livro estão em BASIC, prontos para serem usados e com checagem automática para detecção de erros de digitação.

Esperamos com isso abrir novos horizontes aos nossos leitores, tanto aos mais experientes quanto aos principiantes.

