

# **MSX**

## **Top Secret**

# **3**

**THE FINAL COMPILATION OF MSX INFORMATION**

**1<sup>a</sup> edição – 23 out 2022**

**Edison Moraes [2019-2022]**

## **NOTA DO AUTOR**

Depois do lançamento do MSX Top Secret 2, em abril de 2004, imaginei que não haveria mais necessidade de atualizá-lo, visto que o MSX não é mais fabricado comercialmente por grandes empresas. Além disso, a internet evoluiu e uma grande gama de informações passou a estar disponível para todos.

Entretanto, as informações estão esparsas, levando à necessidade de múltiplas e cansativas buscas nem sempre logrando êxito completo. Por isso, julguei conveniente escrever esta terceira – e última – edição do MSX Top Secret, coligindo todas as informações que pude encontrar em um único lugar.

Boas pesquisas!

Edison Antonio Pires de Moraes (autor)

P.S. Informações sobre erros são bem-vindas! Podem ser enviadas através do e-mail: “eapmoraes2012@gmail.com”.

# Índice

<b>Capítulo 1 - INTRODUÇÃO AO SISTEMA.....</b>	<b>25</b>
1.1 – ARQUITETURA INTERNA.....	26
1.1.1 – A CPU.....	27
1.1.1.1 – Wait States.....	27
1.1.2 – O VDP.....	27
1.1.3 – O PSG.....	28
1.1.4 – A PPI.....	28
<b>Capítulo 2 - SLOTS E CARTUCHOS.....</b>	<b>29</b>
2.1 – SLOTS.....	29
2.1.1 – Chamadas Inter-slots.....	32
2.1.2 – Área de Trabalho.....	33
2.2 – DESENVOLVENDO SOFTWARE EM CARTUCHO.....	35
2.2.1 – Alocando área de trabalho para cartuchos.....	40
<b>Capítulo 3 - A MEMÓRIA ROM.....</b>	<b>46</b>
3.1 – BIOS.....	47
3.2 – O MATH-PACK (PACOTE MATEMÁTICO).....	48
3.2.1 – Área de trabalho.....	49
3.2.2 – Entradas do Math-Pack.....	51
3.2.2.1 – Operações em ponto flutuante.....	51
3.2.2.2 – Operações com números inteiros.....	51
3.2.2.3 – Funções especiais.....	51
3.2.2.4 – Movimento.....	52
3.2.2.5 – Comparações.....	53
3.2.2.6 – Outras operações em ponto flutuante e I/O.....	53
3.3 – O INTERPRETADOR BASIC.....	55
3.3.1 – Os tokens.....	55
3.3.2 – Estrutura das linhas de programa.....	56
3.3.3 – Armazenamento de números.....	56
3.3.4 – Área de variáveis do interpretador.....	58
3.3.5 – Chamando programas de Assembly no BASIC.....	61
3.3.5.1 – Implementando novos comandos.....	62
3.3.6 – Rotinas do interpretador.....	63
3.3.7 – Chamando comandos do interpretador.....	66
3.3.8 – Mensagens de erro.....	73

<b>Capítulo 4 - A MEMÓRIA RAM.....</b>	<b>74</b>
4.1 – EXPANSÕES DE MEMÓRIA.....	74
4.1.1 – A Memória Mapeada.....	74
4.1.1.1 – Gerenciando a Memória Mapeada.....	76
4.1.2 – A Megaram.....	83
4.1.3 – Megaram x Memória Mapeada.....	84
4.2 – MAPEAMENTO DA RAM.....	85
4.2.1 – O FCB (File Control Block).....	86
4.2.2 – A área de trabalho.....	87
4.2.2.1 – Subrotinas inter-slot.....	88
4.2.2.2 – Os Hooks (ganchos).....	89
<b>Capítulo 5 - O VÍDEO E O VDP.....</b>	<b>91</b>
5.1 – O MSX-VIDEO.....	91
5.1.1 – Descrição dos registradores (V9918/38/58).....	93
5.1.2 – A VRAM (V9918/38/58).....	95
5.1.3 – A ADVRAM (V9918/38/58).....	96
5.1.4 – Portas de acesso ao VDP (V9918/38/58).....	97
5.2 – ACESSO À VRAM E AO VDP (V9918/38/58).....	98
5.2.1 – Acesso aos registradores de controle.....	98
5.2.1.1 – Acesso direto.....	98
5.2.1.2 – Acesso indireto.....	99
5.2.1.3 – Acesso indireto com autoincremento.....	99
5.2.2 – Acesso aos registradores de paleta.....	101
5.2.3 – Lendo os registradores de estado.....	102
5.2.4 – Acesso à VRAM pela CPU.....	103
5.3 – MODOS DE TELA (V9918/38/58).....	105
5.3.1 – Modo Texto 1.....	107
5.3.2 – Modo Texto 2.....	109
5.3.3 – Modo Multicor.....	112
5.3.4 – Modo Gráfico 1.....	115
5.3.5 – Modos Gráficos 2 e 3.....	117
5.3.6 – Modo Gráfico 4.....	120
5.3.7 – Modo Gráfico 5.....	122
5.3.8 – Modo Gráfico 6.....	124
5.3.9 – Modo Gráfico 7.....	126
5.3.10 – Modo Gráfico 8.....	128
5.3.11 – Modo Gráfico 9.....	132
5.3.12 – Variáveis de sistema dos modos de tela.....	135

5.4 – SPRITES (V9918/38/58).....	135
5.4.1 – Sprites modo 1.....	136
5.4.2 – Sprites modo 2.....	139
5.5 – COMANDOS DO VDP (V9938/58).....	142
5.5.1 – Descrição dos comandos do VDP.....	143
5.5.2 – Operações lógicas.....	144
5.5.3 – Especificação de áreas.....	145
5.5.4 – Usando os comandos do VDP.....	146
5.5.4.1 – HMMC (Transf. em bytes – CPU → VRAM).....	147
5.5.4.2 – YMMM (Transf. em bytes – VRAM direção Y).....	149
5.5.4.3 – HMMM (Transf. em bytes – VRAM → VRAM).....	151
5.5.4.4 – HMMV (Desenha retângulo em bytes).....	153
5.5.4.5 – LMMC (Transferência lógica – CPU → VRAM).....	155
5.5.4.6 – LMCM (Transferência lógica – VRAM → CPU).....	157
5.5.4.7 – LMMP (Transf. lógica – VRAM → VRAM).....	159
5.5.4.8 – LMMV (Pintura lógica da VRAM).....	161
5.5.4.9 – LINE (Desenha uma linha).....	163
5.5.4.10 – SRCH (Procura código de cor).....	165
5.5.4.11 – PSET (Desenha um ponto).....	167
5.5.4.12 – POINT (Lê código de cor de um ponto).....	168
5.5.5 – Tornando os comandos mais rápidos.....	169
5.6 – MISCELÂNEA DE FUNÇÕES DO VDP (V9938/58).....	170
5.6.1 – Ajuste da localização da tela.....	170
5.6.2 – Número de pontos na direção vertical.....	171
5.6.3 – Frequência de interrupção (PAL/NTSC).....	171
5.6.4 – Troca das páginas de vídeo.....	171
5.6.5 – Troca automática de tela.....	172
5.6.6 – Modo entrelaçado.....	172
5.6.7 – Scroll vertical.....	173
5.6.8 – Scroll horizontal (Só V9958).....	173
5.6.9 – Código de cor 0.....	174
5.6.10 – Interrupção por varredura de linha.....	174
5.6.11 – Liga/desliga a tela.....	175
5.6.12 – Comando de espera (Só V9958).....	175
5.7 – DESCRIÇÃO DOS REGISTRADORES (V9938/58).....	175
5.7.1 – Registradores de estado.....	175
5.7.2 – Registradores de modo.....	178
5.7.3 – Registradores de endereços das tabelas.....	183

5.7.4 – Registradores de cores e display.....	183
5.7.5 – Registradores de acesso.....	185
5.7.6 – Registradores de comando.....	185
5.8 – O V9990 (E-VDP-III).....	187
5.8.1 – Os registradores do V9990.....	188
5.8.2 – Acesso ao V9990.....	190
5.8.2.1 – Acesso aos registradores.....	190
5.8.2.2 – Acesso à VRAM.....	191
5.8.2.3 – Acesso à paleta de cores.....	192
5.8.2.4 – Acesso à Kanji ROM.....	193
5.8.3 – Modos de tela do V9990.....	194
5.8.3.1 – Modos por apresentação de padrões.....	194
5.8.3.2 – Modos bit-map.....	195
5.8.3.3 – Modos bit-map não documentados.....	197
5.8.3.4 – Sistemas de cores.....	198
5.8.3.5 – Valores dos registradores para cada modo.....	199
5.8.3.6 – Modo P1.....	199
5.8.3.7 – Modo P2.....	202
5.8.3.8 – Modo B0.....	204
5.8.3.9 – Modo B1.....	204
5.8.3.9 – Modo B2.....	205
5.8.3.10 – Modo B3.....	206
5.8.3.11 – Modo B4.....	207
5.8.3.12 – Modo B5.....	208
5.8.3.13 – Modo B6.....	208
5.8.3.14 – Modo B7.....	209
5.8.3.15 – Mapas de memória dos modos B0~B7.....	210
5.8.4 – Especificações de cores.....	211
5.8.4.1 – Modo BYUV.....	211
5.8.4.2 – Modo BYUVP.....	213
5.8.4.3 – Modo BYJK.....	216
5.8.4.4 – Modo BYJKP.....	216
5.8.4.5 – Modo BD16.....	216
5.8.4.6 – Modo BD8.....	217
5.8.4.7 – Modo BP6.....	218
5.8.4.8 – Modo BP4.....	218
5.8.4.9 – Modo BP2.....	219
5.8.4.10 – Cores para os modos P1 e P2.....	220

5.8.5 – SPRITES E CURSORES.....	221
5.8.5.1 – Sprites para os modos P1 e P2.....	221
5.8.5.2 – Cursos para os modos B0 a B7.....	223
5.8.6 – COMANDOS DO VDP V9990.....	224
5.8.6.1 – Formatos de dados para os comandos.....	226
5.8.6.2 – Parâmetros para os comandos.....	227
5.8.6.3 – Executando os comandos.....	232
5.8.6.4 – LMMC (Transferência Lógica CPU → VRAM).....	232
5.8.6.5 – LMMV (Desenha retângulo).....	233
5.8.6.6 – LMCM (Transferência lógica VRAM → CPU).....	235
5.8.6.7 – LMMM (Transferência lógica VRAM → VRAM).....	236
5.8.6.8 – CMMC (Transferência de caractere → VRAM).....	237
5.8.6.9 – CMMK (Transf. de caractere Kanji → VRAM).....	239
5.8.6.10 – CMMM (Transf. de caractere VRAM → VRAM).....	240
5.8.6.11 – BMXL (Transf. de bytes linear → coordenadas).....	242
5.8.6.12 – BMLX (Transf. de bytes coordenadas → linear).....	244
5.8.6.13 – BMLL (Transf. de bytes linear → linear).....	245
5.8.6.14 – LINE (Desenha uma linha).....	246
5.8.6.15 – SRCH (Procura código de cor de um ponto).....	248
5.8.6.16 – POINT (Lê código de cor de um ponto).....	249
5.8.6.17 – PSET (Desenha um ponto e avança).....	250
5.8.6.18 – ADVN (Avança coordenadas).....	251
5.8.7 – Scroll e área de imagem.....	252
5.8.8 – Descrição dos registradores do V9990.....	254
5.8.8.1 – Endereço de escrita na VRAM (Só escrita).....	254
5.8.8.2 – Endereço para leitura da VRAM (Só escrita).....	254
5.8.8.3 – Modo de tela (Leitura/escrita).....	254
5.8.8.4 – Controle do sistema (Leitura/escrita).....	255
5.8.8.5 – Controle de interrupções (Leitura/escrita).....	256
5.8.8.6 – Controle de paleta (Só escrita).....	257
5.8.8.7 – Cor de fundo (Leitura/escrita).....	257
5.8.8.8 – Ajuste de tela (Leitura/escrita).....	257
5.8.8.9 – Controle de scroll (Leitura/escrita).....	258
5.8.8.10 – Endereço da tabela de sprites (Leitura/escrita).....	259
5.8.8.11 – Controle do painel LCD (Leitura/escrita).....	259
5.8.8.12 – Controle de prioridade (Leitura/escrita).....	260
5.8.8.13 – Offset da paleta para sprite/cursor (Só escrita).....	260
5.8.8.14 – Comandos do VDP – Fonte.....	260

5.8.8.15 – Comandos do VDP – Destino.....	261
5.8.8.16 – Comandos do VDP – Tamanho.....	261
5.8.8.17 – Comandos do VDP – Argumentos.....	262
5.8.8.18 – Comandos do VDP – Operação lógica.....	263
5.8.8.19 – Comandos do VDP – Máscara de escrita.....	263
5.8.8.20 – Comandos do VDP – Cor da fonte.....	263
5.8.8.21 – Comandos do VDP – Controle de operação.....	264
5.8.8.22 – Comandos do VDP – Coord horizontal.....	264
<b>Capítulo 6 - GERADORES DE ÁUDIO.....</b>	<b>265</b>
6.1 – O PSG.....	265
6.1.1 – Descrição dos registradores.....	265
6.1.1.1 – Especificação da frequência.....	266
6.1.1.2 – Gerador de ruído branco.....	267
6.1.1.3 – Mixando os sons.....	268
6.1.1.4 – Ajuste de volume.....	268
6.1.1.5 – Frequência da envoltória.....	269
6.1.1.6 – Forma da envoltória.....	269
6.1.2 – Acesso ao PSG.....	270
6.2 – GERAÇÃO DE SONS PELA PORTA 1-bit.....	271
6.3 – O OPLL (MSX-MUSIC).....	271
6.3.1 – Descrição da síntese FM.....	272
6.3.2 – Mapa dos registradores do OPLL.....	273
6.3.3 – Descrição dos registradores.....	277
6.3.3.1 – Registrador de teste.....	277
6.3.3.2 – Registradores para definição de instrumentos.....	278
6.3.3.3 – Registradores de seleção.....	284
6.3.4 – A FM-BIOS.....	287
6.3.5 – O FM estéreo.....	290
6.3.6 – Acesso ao OPLL.....	291
6.3.6.1 – Acesso através da FM-BIOS.....	292
6.3.6.2 – Acesso direto.....	292
6.3.7 – Acesso ao DAC interno.....	294
6.4 – O PCM.....	295
6.4.1 – Acesso ao PCM.....	296
6.5 – O MSX-AUDIO.....	301
6.5.1 – Descrição da análise e síntese ADPCM.....	302
6.5.2 – Mapa dos registradores do MSX-Audio.....	304

6.5.3 – Descrição dos registradores.....	307
6.5.3.1 – Registrador de teste.....	307
6.5.3.2 – Registradores de tempo.....	308
6.5.3.3 – Controle de flags (sinalizadores).....	309
6.5.3.4 – Controle de teclado, memória e ADPCM.....	310
6.5.3.5 – Endereços de acesso.....	313
6.5.3.6 – Acesso ao ADPCM e I/O 4 bits.....	315
6.5.3.7 – Acesso ao gerador FM.....	317
6.5.3.8 – O registrador de estado.....	327
6.5.4 – Acesso à memória de áudio E ADPCM.....	328
6.5.4.1 – Análise do som (MSX-Audio → CPU).....	328
6.5.4.2 – Síntese de som (CPU → MSX-Audio).....	329
6.5.4.3 – Análise do som (MSX-Audio → Mem. Áudio).....	329
6.5.4.4 – Síntese de som (Mem. Áudio → MSX-Audio).....	330
6.5.4.6 – Leitura RAM/ROM áudio (Mem. Áudio → CPU)....	331
6.5.5 – Acesso ao MSX-AUDIO.....	332
6.5.5.1 – Acesso Direto.....	332
6.5.5.2 – Acesso através da Music BIOS.....	334
6.6 – O SCC.....	336
6.6.1 – O SCC “simples” (2212).....	337
6.6.1.1 – Forma de onda.....	338
6.6.1.2 – Ajuste da frequência.....	339
6.6.1.3 – Ajuste do volume.....	340
6.6.1.4 – Registrador de chaves.....	340
6.6.1.5 – Registrador de deformação.....	340
6.6.2 – O SCC+ (2312).....	341
6.6.3 – Acesso ao SCC.....	343
6.6.4 – Detectando o SCC.....	343
6.7 – O OPL4.....	346
6.7.1 – Descrição dos registradores para síntese wave.....	347
6.7.1.1 – Acesso à memória de áudio.....	349
6.7.1.2 – Acesso ao modo wave.....	351
6.7.1.3 – Formato da “wave table synthesis” .....	360
6.7.1.4 – Controle de mixagem wave/FM.....	362
6.7.2 – Descrição dos registradores para o gerador FM.....	362
6.7.2.1 – Timers.....	366
6.7.2.2 – Acesso ao modo FM.....	367
6.7.3 – Acesso ao OPL4.....	378

6.8 – COVOX.....	379
6.8.1 – Acesso ao Covox.....	379
<b>Capítulo 7 - SISTEMAS DE ARMAZENAMENTO DE MASSA.....</b>	<b>381</b>
7.1 – MSXDOS, MSXDOS2 e NEXTOR.....	382
7.1.1 – O COMMAND.COM.....	383
7.1.2 – O MSXDOS.SYS.....	384
7.1.3 – O DOS Kernel.....	384
7.1.4 – Estrutura dos arquivos no disco.....	384
7.1.4.1 – Setores.....	384
7.1.4.2 – Clusters (Aglomeradoss).....	385
7.1.4.3 – Divisão de dados no disco.....	385
7.1.4.4 – O setor de boot e o DPB.....	385
7.1.4.5 – O FIB (MSXDOS2).....	387
7.1.4.6 – A FAT (File allocation table).....	387
7.1.4.7 – O diretório.....	391
7.1.5 – Acesso aos arquivos em disco.....	393
7.1.5.1 – Abrindo um arquivo.....	396
7.1.5.2 – Fechando um arquivo.....	396
7.1.5.3 – Acesso sequencial e aleatório.....	396
7.1.5.4 – Headers (cabeçalhos).....	397
7.1.5.5 – Arquivos handle (MSXDOS2).....	398
7.1.6 – As funções do BDOS.....	399
7.1.6.1 – Leitura/escrita absoluta de setores.....	400
7.1.6.2 – Acesso aos arquivos usando o FCB.....	401
7.1.7 – Área de sistema para o MSXDOS.....	405
7.1.8 – Rotinas da interface de disco.....	406
7.1.9 – A página-zero.....	407
7.1.10 – O setor de boot.....	408
7.1.11 – A rotina de inicialização.....	410
7.1.12 – O NEXTOR.....	412
7.1.12.1 – O Kernel MSXDOS 1.....	413
7.1.12.2 – O Kernel do MSXDOS 2.....	413
7.1.12.3 – O Kernel do Nextor.....	415
7.1.12.4 – Criação de um Kernel Nextor.....	416
7.2 – O UZIX.....	427
7.2.1 – Sistemas de arquivos no Uzix.....	427
7.2.1.1 – Tipos de arquivos.....	428
7.2.1.2 – Estrutura hierárquica.....	429

7.2.1.3 – Permissões de acesso a arquivos.....	430
7.2.1.4 – Estrutura dos arquivos no disco.....	430
7.2.2 – Mapeamento de memória.....	434
7.2.3 – Desenvolvendo software para o Uzix.....	436
7.2.4 – Comandos de texto.....	437
7.2.5 – Chamadas de sistema.....	440
7.2.5.1 – Chamadas diretas.....	440
7.2.5.2 – Chamada indireta.....	441
7.2.5.3 – Chamadas via GETSET.....	442
7.2.6 – Módulo TCP/IP.....	442
7.2.7 – Códigos de erro.....	445
7.3 – SYMBOS.....	446
7.3.1 – Biblioteca do sistema SymStudio.....	447
7.3.2 – Configuração do sistema.....	447
7.3.3 – Kernel.....	447
7.3.3.1 – Acesso ao Kernel.....	448
7.3.4 – Rede (Network Daemon).....	451
7.3.5 – Gerenciador de Tela.....	452
7.3.6 – Protetor de Tela.....	452
7.3.7 – Terminal de Texto (SymShell).....	454
7.3.8 – Gerenciador de sistema.....	456
7.3.9 – Aplicativos.....	456
7.3.9.1 – Tipos de área de memória.....	456
7.3.9.2 – Estrutura dos aplicativos.....	459
7.3.10 – Menus pulldown.....	462
7.3.11 – Gráficos.....	462
7.3.11.1 – Gráficos padrão.....	462
7.3.11.2 – Gráficos com cabeçalho estendido.....	463
7.3.12 – Fontes.....	465
7.3.13 – Gerenciador de dispositivos.....	466
7.3.14 – Gerenciador de arquivos.....	466
7.3.15 – Gerenciador de diretório.....	466
7.4 – WiOS.....	467
7.4.1 – Aplicativos para o WiOS.....	468
7.4.1.1 – Estrutura do Programa.....	468
7.4.2 – Estrutura de Memória.....	469
7.4.2.1 – Área de Dados Globais (GDA).....	469
7.4.2.2 – Segmento de Dados.....	470

7.4.2.3 – Código e Segmento de Pilha.....	470
7.4.2.4 – Parte interna do WiOS.....	470
7.4.3 – Drivers.....	470
7.4.3.1 – Alças (handles).....	471
7.4.4 – Interface da Janela.....	471
7.4.4.1 – Parâmetros.....	472
7.4.4.2 – Gerenciamento de Janelas.....	473
7.4.4.3 – Tipos de Janelas.....	473
7.4.5 – Eventos (Comunicação entre WiOS e Tarefas).....	474
7.4.6 – Entrada do usuário.....	475
7.4.7 – Arrastar e Soltar.....	476
7.4.8 – Gerenciamento de Arquivos.....	476
7.4.9 – Linguagem de Programação.....	476
7.4.10 – Chamando funções WiOS.....	478
7.4.10.1 – Chamadas Diretas.....	480
7.4.11 – Referência do Evento.....	481
7.4.11.1 – Estrutura do Bloco de Eventos.....	482
7.4.12 – Especificação de troca de dados.....	482
7.4.12.1 – Execução da troca de dados.....	482
7.4.12.2 – Definição do tipo de dados.....	484
7.4.12.3 – Dados enviados com o evento ‘E_EDRAG’ .....	484
7.4.12.4 – Dados enviados com o evento ‘E_USERMSG’ .....	484
7.4.13 – Referência GDA.....	486
7.4.14 – Referência do Menu.....	488
7.4.14.1 – Tipos de menu.....	489
7.4.14.2 – Recursos do Menu.....	490
7.4.14.3 – Convenções gerais sobre menus.....	491
7.4.14.4 – Estrutura do Bloco de Menu.....	492
7.4.15 – Dicas de programação.....	495
7.4.15.1 – Janelas.....	495
7.4.15.2 – Entrada do usuário.....	497
7.4.15.3 – Usando o Segmento Temporário.....	497
7.4.15.4 – Ponteiro do Mouse.....	497
7.4.15.5 – Enviando dados para outras partes do programa...	498
7.4.15.6 – Pilhas.....	498
7.4.15.7 – Encerramento do Programa.....	499
7.4.16 – Compilação.....	499
7.4.16.1 – Aplicativos de parte única (código de até 16k).....	501

7.4.16.2 – Aplicativos de várias partes (mais de 16k).....	502
7.3.17 – Especificações da Tarefa.....	504
7.4.17.1 – Cabeçalhos.....	504
7.4.17.2 – Diretrizes de pré-processador.....	507
7.4.17.3 – Variáveis Externas.....	508
7.4.17.4 – Inicialização da Tarefa.....	508
7.4.17.5 – Rotina Principal.....	509
7.4.17.6 – Criação de Janelas.....	509
7.4.17.7 – “Polling” e Tratamento de Eventos.....	511
7.4.18 – Segurança.....	515
7.4.19 – Iniciando tarefas com a versão Alfa.....	515
7.4.20 – Descrição dos termos usados.....	516
<b>7.5 – ACESSO DIRETO AO CONTROLADOR DE DISCO.....</b>	<b>516</b>
7.5.1 – Comandos do FDC.....	517
7.5.1.1 – Comandos tipo I.....	518
7.5.1.2 – Comandos tipo II.....	519
7.5.1.3 – Comandos tipo III.....	520
7.5.1.4 – Comando tipo IV.....	522
7.5.2 – O registrador de estado.....	522
7.5.3 – Funções e cuidados adicionais.....	523
7.5.4 – Formatação.....	524
7.5.5 – Endereços de acesso ao FDC.....	525
<b>Capítulo 8 - DISPOSITIVOS ADICIONAIS.....</b>	<b>527</b>
8.1 – O RELÓGIO E A SRAM.....	527
8.1.1 – Funções do Clock-IC.....	527
8.1.2 – Estrutura e registradores do Clock-IC.....	528
8.1.2.1 – O registrador de modo (#13).....	528
8.1.2.2 – O registrador de teste (#14).....	529
8.1.2.3 – O registrador de reset (#15).....	529
8.1.2.4 – Acertando o relógio e o alarme.....	530
8.1.2.5 – Conteúdo da SRAM adicional.....	531
8.1.3 – Acesso ao Clock-IC.....	533
8.2 – INTERFACE DE IMPRESSORA.....	535
8.2.1 – Acesso à impressora.....	537
8.3 – INTERFACE DE TECLADO.....	537
8.3.1 – Acesso ao teclado.....	539
8.3.2 – Varredura de teclado.....	539
8.4 – INTERFACE UNIVERSAL DE I/O.....	540

<b>Capítulo 9 - O MSX TURBO R.....</b>	<b>543</b>
9.1 – ORGANIZAÇÃO DE SLOTS E PÁGINAS.....	543
9.2 – WAIT STATES.....	544
9.3 – MODOS DE OPERAÇÃO.....	545
9.3.1 – Comparação de velocidade.....	547
9.3.2 – Instruções específicas do R800.....	547
9.4 – A MSX-MIDI.....	549
9.4.1 – Acesso à MSX-MIDI.....	549
9.4.2 – Descrição das portas da MIDI externa.....	549
9.4.3 – Descrição das portas da MIDI interna.....	550
9.4.4 – MIDI interna e MIDI externa.....	552
9.5 – TEMPORIZAÇÃO PARA O V9958.....	553
9.6 – A SRAM INTERNA.....	553
9.7 – O MSX ENGINE S1990.....	554
<b>A PÊNDICE.....</b>	<b>556</b>
<b>1 – CARACTERES E TECLADO.....</b>	<b>557</b>
1.1 – TABELAS DE CARACTERES.....	557
1.1.1 – Tabela Japonesa.....	557
1.1.2 – Tabela Internacional.....	558
1.1.3 – Tabela Brasileira 1.0 (Expert 1.0).....	559
1.1.4 – Tabela Brasileira 1.1 (Expert 1.1 e Hotbit 1.2).....	560
1.1.5 – Tabela Cirílica (Russa).....	561
1.1.6 – Tabela Coreana.....	562
1.1.7 – Tabela Arábica (AX-170).....	563
1.1.8 – Tabela Arábica (AX-500).....	564
1.2 – MATRIZES DE TECLADO.....	565
1.2.1 – Matriz Japonesa.....	565
1.2.1.1 – Matriz Japonesa com a tecla かな/KANA travada.....	566
1.2.2 – Matriz do PX-7.....	567
1.2.3 – Matriz Internacional.....	568
1.2.5 – Matriz Argentina / Espanhola.....	569
1.2.6 – Matriz do Reino Unido (Inglaterra).....	569
1.2.7 – Matriz Cirílica (Russa).....	570
1.2.7.1 – Matriz Cirílica com tecla PYC/CODE travada.....	570
1.2.8 – Matriz Coreana.....	571
1.2.8.1 – Matriz coreana com a tecla 한글/CODE travada.....	571
1.2.9 – Matriz Arábica.....	572
1.2.9.1 – Matriz arábica com modo árabe ativado.....	572

1.3 – LAYOUTS DE TECLADO.....	573
1.3.1 – Layout Internacional.....	573
1.3.2 – Layout Japonês (JIS).....	573
1.3.3 – Layout Japonês (ANSI).....	573
1.3.4 – Layout Brasileiro 1.0 (Expert 1.0).....	574
1.3.5 – Layout Brasileiro 1.1 (Hotbit / Expert 1.1).....	574
1.3.6 – Layout do Reino Unido (Inglês).....	574
1.3.7 – Layout Argentino / Espanhol.....	575
1.3.8 – Layout Russo (Cirílico).....	575
1.3.9 – Layout Coreano (CPC-400).....	575
1.3.10 – Layout Arábico (AX-170).....	576
1.3.11 – Layout Francês (ML-F80).....	576
1.3.12 – Layout Germânico (HB-F700D).....	576
1.4 – CÓDIGOS DE CONTROLE.....	577
<b>2 – MAPA DAS PORTAS DE I/O.....</b>	<b>578</b>
<b>3 – MSX-BASIC.....</b>	<b>583</b>
3.1 – FORMATO.....	583
3.1.1 – Abreviações de Instruções.....	583
3.1.2 – Códigos de Operação Lógica.....	583
3.1.3 – Notações de Código.....	584
3.1.4 – Notações de Formato.....	584
3.2 – DESCRIÇÃO DOS COMANDOS.....	585
3.3 – DESCRIÇÃO DE COMANDOS ESTENDIDOS.....	615
3.3.1 – Lista dos comandos.....	621
A.....	621
B.....	624
C.....	627
D.....	638
E.....	641
F.....	643
G.....	646
H.....	646
I.....	648
J.....	650
K.....	651
L.....	655
M.....	660
N.....	667

O.....	688
P.....	689
Q.....	697
R.....	698
S.....	704
T.....	713
U.....	715
V.....	717
W.....	719
X.....	719
Y.....	720
<b>3.4 – CÓDIGOS DE ERRO DO MSX-BASIC.....</b>	<b>720</b>
<b>4 – MSXDOS.....</b>	<b>722</b>
<b>4.1 – NOTAÇÕES DE FORMATO.....</b>	<b>722</b>
<b>4.1.1 – Descrição das extensões de nomes de arquivos.....</b>	<b>723</b>
<b>4.2 – DESCRIÇÃO DOS COMANDOS.....</b>	<b>731</b>
<b>4.3 – CHAMADAS PARA A BDOS.....</b>	<b>745</b>
<b>4.3.1 – Manipulação de I/O.....</b>	<b>745</b>
<b>4.3.2 – Definição e leitura de parâmetros.....</b>	<b>747</b>
<b>4.3.3 – Leitura/escrita absoluta de setores.....</b>	<b>750</b>
<b>4.3.4 – Acesso aos arquivos usando o FCB.....</b>	<b>750</b>
<b>4.3.5 – Funções adicionadas pelo MSXDOS2.....</b>	<b>754</b>
<b>4.3.6 – Funções adicionadas pelo NEXTOR.....</b>	<b>768</b>
<b>4.4 – CÓDIGOS DE ERRO DO MSXDOS.....</b>	<b>775</b>
<b>4.5 – CÓDIGOS DE ERRO DO MSXDOS2.....</b>	<b>776</b>
<b>4.5.1 – Erros de Disco.....</b>	<b>776</b>
<b>4.5.2 – Erros das Funções do MSXDOS.....</b>	<b>776</b>
<b>4.5.3 – Erros Adicionados pelo Nextor.....</b>	<b>777</b>
<b>4.5.4 – Erros de Término de Programas.....</b>	<b>778</b>
<b>4.5.5 – Erros de Comando.....</b>	<b>778</b>
<b>5 – SYMBOS.....</b>	<b>779</b>
<b>5.1 – ROTINAS DO KERNEL.....</b>	<b>779</b>
<b>5.1.1 – Kernel Restarts.....</b>	<b>779</b>
<b>5.1.2 – Comandos do Kernel (Gerenciamento Multitarefa).....</b>	<b>781</b>
<b>5.1.3 – Respostas do Kernel (Gerenciamento Multitarefa).....</b>	<b>783</b>
<b>5.1.4 – Funções do Kernel (Gerenciamento de Memória).....</b>	<b>785</b>
<b>5.1.5 – Funções do Kernel (Bancos de Memória).....</b>	<b>787</b>
<b>5.1.6 – Funções do Kernel (Diversas).....</b>	<b>789</b>

5.2 – GERENCIADOR DA ÁREA DE TRABALHO.....	790
5.2.1 – Respostas do Gerenciador da Área de Trabalho.....	795
5.2.2 – Serviços do Gerenciador da Área de Trabalho.....	798
5.2.3 – Funções do Gerenciador da Área de Trabalho.....	800
5.2.4 – Registros de Gerenciamento da Área de Trabalho.....	802
5.2.4.1 – Janela de Registro de Dados.....	802
5.2.4.2 – Registro de Dados do Grupo de Controle.....	803
5.2.4.3 – Registros dos Dados de Controle.....	804
5.2.4.4 – Registro de Dados de Regra de Cálculo.....	804
5.3 – TIPOS DE CONTROLE.....	805
5.3.1 – Pintura.....	805
5.3.2 – Gráficos.....	808
5.3.3 – Botões.....	809
5.3.4 – Diversos.....	811
5.3.5 – Entrada de Texto.....	812
5.3.6 – Listas.....	815
5.3.7 – Menus PullDown.....	817
5.4 – GRÁFICOS.....	818
5.4.1 – Gráficos padrão.....	818
5.4.2 – Gráficos com cabeçalho estendido.....	819
5.4.3 – Fontes.....	820
5.5 – GERENCIADOR DE SISTEMA.....	820
5.5.1 – Gerenciamento de Aplicativos.....	820
5.5.2 – Comandos do Gerenciador de Sistema.....	823
5.5.3 – Serviços de Diálogo.....	824
5.5.4 – Funções do Gerenciador de Sistema.....	828
5.6 – GERENCIADOR DE ARQUIVOS E DISPOSITIVOS.....	830
5.6.1 – Mensagens do Gerenciador de Arquivos.....	830
5.6.2 – Códigos de Erro.....	830
5.6.3 – Funções do Dispositivo de Armazenamento de Massa.....	831
5.6.4 – Funções de Gerenciamento de Arquivos.....	834
5.6.5 – Funções de Gerenciamento de Diretório.....	838
5.6.6 – Funções do Gerenciador de Dispositivos.....	844
5.7 – LINHA DE COMANDO DO SYMSHELL.....	847
5.7.1 – Comandos do terminal de texto.....	847
5.7.1.1 – Aplicativos padrão.....	849
5.7.2 – Comandos e Respostas do Symshell.....	850
5.7.3 – Códigos de Controle do Terminal Symshell.....	853

5.7.4 – Códigos ASCII estendidos.....	855
5.7.5 – Códigos de Varredura do Teclado.....	856
5.8 – CONFIGURAÇÃO DO SISTEMA.....	856
5.8.1 – Cabeçalho.....	857
5.8.2 – Área Central.....	857
5.8.2.1 – Dispositivos de armazenamento de massa.....	857
5.8.2.2 – Exibição e diversos (1).....	858
5.8.2.3 – Teclado (1) e mouse.....	858
5.8.2.4 – Diversos (2) e links da área de trabalho.....	858
5.8.3 – Área de Dados.....	859
5.8.3.1 – Links da área de trabalho (2).....	859
5.8.3.2 – Protetor de tela.....	860
5.8.3.3 – Teclado (2).....	860
5.8.3.4 – Segurança.....	860
5.9 – APlicativos do PROTETOR DE TELA.....	861
5.10 – MAPA DE MEMÓRIA.....	862
5.10.1 – Mapa geral da memória.....	862
5.10.2 – Mapa da memória do aplicativo.....	863
5.10.3 – Configurações de memória.....	863
5.11 – GERENCIADOR DE TELA.....	864
5.12 – DAEMON DE REDE.....	865
5.12.1 – Configuração.....	865
5.12.2 – Serviços de Camada de Transporte.....	865
5.12.3 – Serviços de Camada de Aplicação.....	866
5.13 – CONSTANTES SYMBOS.....	866
5.13.1 – IDs de processos.....	866
5.13.2 – Mensagens.....	867
5.13.3 – Comandos do Kernel.....	867
5.13.4 – Respostas do Kernel.....	867
5.13.5 – Comandos do sistema.....	868
5.13.6 – Respostas do Sistema.....	869
5.13.7 – Comandos da área de trabalho.....	870
5.13.8 – Respostas da área de trabalho.....	871
5.13.9 – Comandos Shell.....	872
5.13.10 – Respostas do Shell.....	872
5.13.11 – Mensagens do protetor de tela.....	873
5.13.12 – Ações da Área de Trabalho.....	873
5.13.13 – Serviços da Área de Trabalho.....	874

5.13.14 – Jumps.....	874
5.13.15 – Funções de Gerenciador de Arquivos.....	875
<b>6 – UZIX.....</b>	<b>877</b>
6.1 – COMANDOS.....	877
6.1.1 – Convenções Usadas.....	877
6.1.1.1 – Notações de Formato.....	877
6.1.2 – Descrição dos Comandos.....	878
6.2 – ESTRUTURA HIERÁRQUICA.....	893
6.3 – MAPEAMENTO DE MEMÓRIA.....	894
6.4 – CHAMADAS DE SISTEMA.....	895
6.4.1 – Chamadas Diretas de Sistema.....	895
6.4.2 – Chamada Indireta de Sistema.....	910
6.4.3 – Chamadas via GETSET.....	911
6.4.4 – Módulo TCP/IP.....	914
6.4.5 – Códigos de erro.....	916
6.5 – CÓDIGOS DO TERMINAL VT-52.....	918
<b>7 – WiOS.....</b>	<b>919</b>
7.1 – DRIVER DE ARQUIVO DE SISTEMA.....	919
7.2 – DRIVER EXTERNO.....	922
7.3 – DRIVER GRÁFICO DE E/S.....	923
7.4 – DRIVER GRÁFICO.....	924
7.5 – DRIVER DE MEMÓRIA.....	929
7.6 – DRIVER PADRÃO.....	931
7.7 – DRIVER DE TAREFA.....	932
7.8 – DRIVER DE JANELA.....	933
7.8.1 – A Estrutura da Janela.....	934
7.8.2 – Funções do Window Driver.....	936
7.8.3 – Redesenhando Janelas.....	938
7.9 – DEFINIÇÃO DE EVENTOS.....	940
7.9.1 – Dados enviados com o evento ‘E_EDRAG’.....	943
7.9.2 – Dados enviados com o evento ‘E_USERMSG’ .....	943
7.10 – REFERÊNCIAS GDA.....	945
7.11 – ESTRUTURA DO BLOCO DE MENU.....	947
<b>8 – VARIÁVEIS DE SISTEMA.....</b>	<b>951</b>
8.1 – ÁREA DE SISTEMA PARA O MSXDOS1.....	951
8.1.1 – Hooks chamados pelas rotinas de disco.....	952
8.1.2 – Outros dados do DOS.....	954
8.1.3 – Hooks para a porta ‘COM’.....	955

8.1.4 – Teclado.....	955
8.1.5 – Variáveis do MSXDOS.....	955
8.1.6 – Endereços do DPB.....	957
8.1.7 – Rotinas usadas pelo MSXDOS.....	957
8.1.8 – Rotinas de movimento inter-slot.....	957
8.2 – ÁREA DE SISTEMA PARA O MSXDOS2.....	958
8.2.1 – Informações físicas dos discos.....	958
8.2.2 – Hooks chamados pelas rotinas de disco (1).....	958
8.2.3 – Informações lógicas dos discos.....	960
8.2.4 – Hooks chamados pelas rotinas de disco (2).....	960
8.2.5 – Variáveis do MSXDOS2.....	961
8.2.6 – Apontadores e buffers (FAT, DTA, FCB, DPB).....	964
8.2.7 – Jumps do sistema.....	965
8.3 – SUBROTINAS INTER-SLOT.....	965
8.4 – FUNÇÃO USR E MODOS TEXTO.....	966
8.5 – ÁREA USADA PELA TELA.....	966
8.5.1 – Screen 0.....	967
8.5.2 – Screen 1.....	968
8.5.3 – Screen 2.....	968
8.5.4 – Screen 3.....	969
8.5.5 – Outros valores para a tela.....	969
8.6 – ÁREA DOS REGISTRADORES DO VDP.....	970
8.6.1 – Área para o VDP V9938.....	970
8.6.2 – Área para o VDP V9958.....	972
7.7 – MISCELÂNEA.....	972
8.8 – ÁREA USADA PELO COMANDO PLAY.....	973
8.8.1 – Filas para o comando PLAY.....	975
8.8.2 – Offset para o buffer de parâmetros do comando PLAY.....	975
8.8.3 – Área de dados para o buffer de parâmetros.....	976
8.9 – ÁREA PARA O TECLADO.....	976
8.10 – ÁREA USADA PELO CASSETE.....	977
8.11 – ÁREA USADA PELO COMANDO CIRCLE.....	978
8.12 – ÁREA USADA INTERNAMENTE PELO BASIC.....	980
8.12.1 – Buffers de texto BASIC.....	981
8.12.2 – Dados Gerais.....	981
8.12.3 – Controle de linhas BASIC em tempo de execução.....	984
8.12.4 – Endereços de armazenamento do texto BASIC.....	985

8.12.5 – Área para as funções do usuário.....	986
8.12.6 – Área de dados do interpretador.....	987
8.13 – ÁREA PARA O MATH-PACK.....	988
8.14 – ÁREA DE DADOS DO SISTEMA DE DISCO.....	989
8.15 – ÁREA USADA PELO COMANDO PAINT.....	991
8.16 – ÁREA ADICIONADA PARA O MSX2.....	992
8.17 – ÁREA USADA PELA RS232C.....	994
8.18 – ÁREA DE DADOS GERAIS.....	996
8.19 – ROTINAS DE EXPANSÃO DA BIOS.....	1000
8.20 – ÁREA DE DADOS PARA OS SLOTS E PÁGINAS.....	1001
8.20.1 – Slot da Main-ROM.....	1003
8.20.2 – Registrador de slot secundário.....	1003
8.21 – DESCRIÇÃO DOS HOOKS.....	1003
<b>9 – ROTINAS DA BIOS.....</b>	<b>1017</b>
9.1 – ROTINAS DA MainROM.....	1017
9.1.1 – Rotinas RST.....	1017
9.1.2 – Rotinas para inicialização I/O.....	1020
9.1.3 – Rotinas para acesso ao VDP.....	1020
9.1.4 – Rotinas para acesso ao PSG.....	1026
9.1.5 – Rotinas para acesso ao teclado, tela e impressora.....	1027
9.1.6 – Rotinas de acesso I/O para jogos.....	1030
9.1.7 – Rotinas de acesso I/O para gravador cassete.....	1032
9.1.8 – Rotinas para a fila do PSG.....	1033
9.1.9 – Rotinas para as telas gráficas do MSX1.....	1034
9.1.10 – Miscelânea.....	1037
9.1.11 – Rotinas para acesso ao sistema de disco.....	1039
9.1.12 – Rotinas adicionadas para o MSX2.....	1040
9.1.13 – Rotinas adicionadas para o MSX2+.....	1042
9.1.14 – Rotinas adicionadas para o MSX turbo R.....	1042
9.1.15 – Rotinas inter-slot da área de trabalho.....	1044
9.2 – ROTINAS DA SubROM.....	1044
9.2.1 – Rotinas para funções gráficas do BASIC.....	1044
9.2.2 – Rotinas para funções gráficas.....	1047
9.2.3 – Rotinas duplicadas (iguais às da MainROM).....	1051
9.2.4 – Rotinas diversas para o MSX2 ou superior.....	1053
8.2.5 – Rotinas de manipulação da paleta de cores.....	1057
9.2.6 – Rotinas diversas usadas pelo BASIC.....	1058
9.2.7 – Rotinas de transferência de bloco (bit-blit).....	1061

9.3 – ROTINAS DO MATH-PACK.....	1064
9.3.1 – Funções matemáticas em ponto flutuante.....	1064
9.3.2 – Operações com números inteiros.....	1064
9.3.3 – Funções especiais.....	1064
9.3.4 – Movimento.....	1065
9.3.5 – Conversões.....	1066
9.4 – ROTINAS DO INTERPRETADOR BASIC.....	1067
9.4.1 – Rotinas de execução.....	1067
9.4.2 – Rotinas dos comandos e funções.....	1070
9.5 – ROTINAS DA BIOS ESTENDIDA.....	1075
9.5.1 – Entrada da BIOS estendida.....	1075
9.5.2 – Comandos internos (broadcast commands).....	1076
9.5.3 – Memória Mapeada.....	1077
9.5.3.1 – Rotinas de manipulação da Memória Mapeada.....	1079
9.5.4 – Porta serial RS232C e MSX Modem.....	1084
9.5.4.1 – Bytes de parâmetros.....	1085
9.5.4.2 – Rotinas de manipulação da porta serial RS232C.....	1085
9.5.4.3 – Rotinas de manipulação do MSX Modem.....	1089
9.5.5 – MSX-AUDIO.....	1094
9.5.5.1 – Rotinas de inicialização.....	1096
9.5.5.2 – Rotinas do PCM/ADPCM.....	1097
9.5.5.3 – Rotinas do teclado musical.....	1101
9.5.5.4 – Rotinas do sintetizador FM.....	1102
9.5.5.5 – Rotinas da MBIOS (Music BIOS).....	1104
9.5.6 – MSX-JE.....	1131
9.5.6.1 – Chamando as funções do MSX-JE.....	1132
9.5.6.2 – Interface do dicionário do MSX-JE.....	1134
9.5.7 – MSX UNAPI.....	1138
9.5.7.1 – RAM Helper.....	1139
9.5.7.2 – API para cartuchos Ethernet.....	1141
9.5.8 – MemMan.....	1144
9.5.8.1 – Fast Calls (Entradas alternativas preferenciais).....	1145
9.5.8.2 – Funções do MemMan.....	1146
9.5.9 – Comandos de sistema.....	1151
9.6 – ROTINAS DA INTERFACE DE DISCO.....	1152
9.6.1 – Inicialização da interface.....	1152
9.6.2 – Rotinas padrão da interface.....	1153
9.6.3 – Rotinas para acesso a Hard-Disks padrão IDE.....	1157

9.6.4 – Rotinas adicionadas pelo NEXTOR.....	1160
9.6.4.1 – Rotinas para drivers de dispositivos de disco.....	1163
9.6.4.2 – Rotinas para drivers de outros dispositivos.....	1170
9.6.5 – Rotinas para acesso a Hard-Disks padrão SCSI.....	1173
9.7 – ROTINAS DA MSX-MUSIC (FM/OPLL).....	1184
<b>10 – MSX-HID (Human Interface Device).....</b>	<b>1187</b>
10.1 – FINGERPRINTS DE DISPOSITIVOS MSX.....	1187
10.2 – FINGERPRINTS DE DISPOSITIVOS SEGA COMPATÍVEIS.	1187
10.3 – FINGERPRINTS DE DISP. QUE PODEM CONFLITAR.....	1187
10.4 – FINGERPRINTS DE DISPOSITIVOS CASEIROS.....	1188
10.5 – FINGERPRINTS RESERVADOS (NÃO USAR).....	1188
<b>11 – MEMÔNICOS Z80/R800.....</b>	<b>1189</b>
11.1 – GRUPO DE CARGA DE 8 BITS.....	1189
11.2 – GRUPO DE CARGA DE 16 BITS.....	1191
11.3 – GRUPO ARITMÉTICO DE 8 BITS.....	1193
11.4 – GRUPO ARITMÉTICO DE 16 BITS.....	1196
11.5 – GRUPO DE TROCA.....	1197
11.6 – GRUPO DE TRANSFERÊNCIA DE BLOCO.....	1198
11.7 – GRUPO DE PESQUISAS.....	1199
11.8 – GRUPO DE COMPARAÇÃO.....	1200
11.9 – GRUPO LÓGICO.....	1201
11.10 – GRUPO DE DESLOCAMENTO E ROTAÇÃO.....	1203
11.11 – GRUPO DE LIGAR, DESLIGAR E TESTAR BITS.....	1206
11.12 – GRUPO DE SALTO.....	1208
11.13 – GRUPO DE CHAMADA E RETORNO.....	1209
11.14 – GRUPO DE ENTRADA E SAÍDA.....	1210
11.15 – GRUPO DE CONTROLE E DE PROPÓSITO GERAL.....	1212
<b>12 – MAPAS DE REGISTRADORES DE CHIPS PADRÃO.....</b>	<b>1213</b>
12.1 – MAPA DOS REGISTRADORES DOS V9918/38/58.....	1213
12.1.1 – Portas de acesso aos VDPs 9918/9938/9958.....	1218
12.1.2 – Tabela de cores padrão.....	1219
12.2 – MAPA DOS REGISTRADORES DO V9990.....	1220
12.2.1 – Portas de acesso ao V9990.....	1225
12.3 – MAPA DOS REGISTRADORES DO PSG (AY-3-8910).....	1227
12.3.1 – Portas de acesso ao PSG.....	1228
12.4 – MAPA DOS REGISTRADORES DO FM-OPLL (YM2413).....	1229
12.4.1 – Portas de acesso ao OPLL.....	1231

12.5 – MAPA DOS REGISTRADORES DO MSX-AUDIO (Y8950)....	1232
12.5.1 – Portas de acesso ao MSX-Audio.....	1235
12.6 – MAPA DOS REGISTRADORES DO OPL4 (YMF278).....	1236
12.6.1 – Register Array #0.....	1236
12.6.2 – Register Array #1.....	1238
12.6.3 – Síntese Wave.....	1241
12.6.4 – Portas de acesso ao OPL4.....	1243
12.6.5 – Cabeçalho da “Wave Table Synthesis” .....	1244
12.6.6 – Tamanho dos dados “wave” .....	1245
12.7 – MAPA DOS REGISTRADORES DO SCC (2212/2312).....	1246
12.7.1 – Endereços de acesso ao SCC.....	1247
<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>1249</b>
<b>OUTROS LIVROS DO MESMO AUTOR.....</b>	<b>1252</b>

## ***Capítulo 1***

### **INTRODUÇÃO AO SISTEMA**

O sistema MSX foi criado em 1.983 e anunciado oficialmente no dia 27 de junho desse mesmo ano pela Microsoft, detentora do padrão na época. O MSX foi criado com arquitetura aberta, podendo qualquer empresa fabricá-lo sem ter que pagar “royalties”.

As especificações previam que todos os micros MSX seriam compatíveis em pontos estratégicos, e que todas as versões que viessem a ser criadas posteriormente manteriam a compatibilidade com o padrão original.

O MSX foi fabricado comercialmente até a quarta versão, o MSX turbo R, e na prática a compatibilidade tem se mantido. De fato, sempre há pequenas alterações em função do desenvolvimento tecnológico ou da não utilização de determinados recursos pelos programadores e pelos usuários em geral. Assim, do MSX1 para o MSX2, a expansão de memória em slots, de manuseio complicado, foi substituída por uma expansão chamada “Memory Mapper” ou “Memória Mapeada”. Seu VDP (processador de vídeo) também foi substituído por outro bem mais poderoso, além de ter vários periféricos opcionais padronizados, como o MSX Audio, por exemplo. Do MSX2 para o MSX2+ a memória principal passou a ser constituída pelos primeiros 64 Kbytes da Memória Mapeada, economizando com isso um slot, além de ter algumas funções do VDP alteradas. Já do MSX2+ para o MSX turbo R as mudanças foram mais radicais: eliminou-se a interface de cassete, que tornara-se totalmente obsoleta e introduziu-se uma nova CPU de 16 bits, a R800, totalmente compatível com o Z80, porém incrivelmente mais rápida que este.

Assim, apesar dessas pequenas alterações que teoricamente destruiriam a compatibilidade, na prática o MSX turbo R é compatível com todos os modelos anteriores. Na tabela da página seguinte estão ilustradas as principais características e diferenças entre as quatro versões de MSX lançadas oficialmente:

	MSX1 Junho/83	MSX2 Maio/86	MSX2+ Outubro/88	MSX turbo R Outubro/90
CPU	Z80 3,58MHz	Z80 3,58MHz	Z80 3,58MHz	Z80 3,58MHz R800 7,16MHz
RAM mínima	8 Kbytes	64 Kbytes	64 Kbytes	256 Kbytes
RAM máxima	1 Mbyte *	64 Mbytes *	64 Mbytes *	64 Mbytes *
VRAM	16 Kbytes	64/128 Kb	128 Kbytes	128 Kbytes
VDP	TMS9918	V9938	V9958	V9958
ROM Standard	32K Main 16K SubROM	32K Main 16K SubROM	32K Main 32K SubROM 16K DOS1	32K Main 48K SubROM 16K DOS1 48K DOS2
Interface CAS	Standard	Standard	Standard	Não
Interf. Impress.	Opcional	Standard	Standard	Standard
Slots Externos	1 ou 2	2	2	2
PSG	Standard	Standard	Standard	Standard
MSX Audio	Não	Opcional	Opcional	Opcional
FM Sound	Não	Opcional	Standard	Standard
PCM	Não	Não	Não	Standard
Disk Drive	Opcional	Opcional	Standard 3½ DD	Standard 3½ DD
MSX Basic	Versão 1.0	Versão 2.0	Versão 3.0	Versão 4.0
MSX-DOS	Versão 1.0	Versão 1.0 2.0 opcional	Versão 1.0 2.0 opcional	Versão 1.0 2.0 standard

A RAM máxima refere-se ao máximo teórico que poderia ser conectado à CPU. De fato, no caso do MSX1, podem ser conectados em cada slot apenas 64K de RAM e como há 16 slots ao todo, isso resulta no máximo teórico de 1 Mbyte. No caso do MSX2 em diante, até 4 Mbytes podem ser conectados em cada slot, através de uma expansão chamada Memória Mapeada, resultando num máximo teórico de 64 Mbytes. Entretanto, a maioria dos softwares que reconhecem a Mapper conseguirão usar apenas 4 Mbytes linearmente em um único slot (256 páginas lógicas), embora alguns deles reconheçam a Mapper em slots diferentes.

## 1.1 – ARQUITETURA INTERNA

No MSX1, existem 4 chips centrais que desempenham funções específicas no micro. Esses chips são designados pelas siglas CPU, VDP, PSG e PPI. Como cada um, à exceção da PPI, possui certa capacidade de

processamento, diz-se que o MSX possui multiprocessamento, teoria muitas vezes contestada.

### **1.1.1 – A CPU**

A CPU (Center Procedure Unit – Unidade Central de Processamento) é a responsável pela execução dos programas. O chip usado é o veterano Z80A, da Zilog, com um clock de 3,58 MHz. A capacidade de processamento desse chip, entretanto, se mostrou insuficiente. Por isso, apareceram MSX turbinados a 6 ou 7 MHz e a partir do MSX2+ não havia mais a exigência clock fixo de 3,58 MHz nas especificações básicas. Posteriormente foi desenhada uma CPU específica para o MSX, a R800, de 16 bits, que era bem mais rápida que o Z80A e que foi usada nos modelos MSX turbo R.

#### **1.1.1.1 – Wait States**

No MSX, é gerado um wait state durante a execução do ciclo M1. Assim, se uma instrução do Z80 tiver 2 ciclos M, serão gerados 2 wait states durante a execução da mesma.

### **1.1.2 – O VDP**

O VDP (Video Display Processor – Processador de Apresentação de Vídeo) é o chip responsável pelo processamento de vídeo. Ele está detalhadamente descrito no capítulo 5 (O vídeo e o VDP). O VDP usado no MSX1 é o TMS9918A, da Texas Instruments, que foi logo substituído pelo V9938, específico para o MSX2, bem mais poderoso. Depois veio o V9958, com algumas funções adicionais. Mas o V9958 mostrou-se muito lento e limitado. A Yamaha iniciou o desenvolvimento de um chip bem mais poderoso, denominado provisoriamente de V9978. Entretanto, este não chegou a ser produzido. Posteriormente, foi lançado um chip com a nomenclatura V9990, mas não era totalmente compatível com o V9958. O V9990 não foi usado em nenhum modelo oficial de MSX, sendo usado apenas em cartucho. Todos esses chips estão detalhadamente descritos no capítulo 5 (O vídeo e o VDP).

### 1.1.3 – O PSG

O PSG (Programmable Sound Generator – Gerador de Sons Programável) é o chip responsável pela geração de sons. Ele também controla as portas de joystick. O chip usado é o AY-3-8910A, da General Instruments. Por ser muito limitado, logo surgiram alternativas. A primeira foi o MSX-Audio (Y8950), que não se popularizou. Depois veio o OPLL (YM2413), este sim bastante popular, com muitos softwares. No mesmo caminho surgiu o SCC e depois o PCM nos modelos MSX turbo R. Um cartucho de som, o Moonsound, utilizava o OPL4 (YMF278B), um chip muito avançado com qualidade de som equivalente à dos CD's. Uma alternativa bastante barata foi o Covox, que é ligado à porta de impressora. Todos esses geradores estão descritos detalhadamente no capítulo 6 (Geradores de áudio).

### 1.1.4 – A PPI

A PPI (Peripheral Programmable Interface – Interface de Periféricos Programável) é a “pedra de toque” do MSX. É ela que faz o MSX ser tão flexível. Suas duas principais tarefas são o controle de teclado e a seleção de slots e páginas de memória. O chip responsável é o 8255A da Intel. Apesar de ter sido embutida no MSX Engine nos modelos mais recentes, seu funcionamento é idêntico ao MSX1. O esquema de páginas e slots, controlado pela PPI, está descrito detalhadamente no capítulo 2 (Slots e cartuchos) e a interface de teclado está descrita no capítulo 8, item 2 (Interface de teclado).

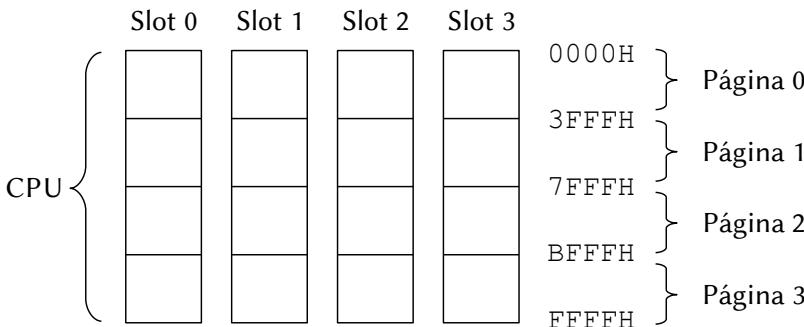
## **Capítulo 2**

### **SLOTS E CARTUCHOS**

A CPU Z80A, que é usada nos micros MSX, é capaz de endereçar diretamente apenas 64 Kbytes de memória. Nos micros MSX, entretanto, usando a técnica de slots e páginas, o Z80A pode acessar até 1 megabyte de memória, de forma não linear<sup>1</sup>.

#### **2.1 – SLOTS**

Há dois tipos de slots: os slots primários e os slots secundários. Os slots primários são em número de quatro e estão conectados diretamente à CPU. Cada slot é dividido em quatro partes de 16 Kbytes, perfazendo os 64 Kbytes do espaço de endereçamento do Z80. Essas partes são denominadas “páginas”. Uma página de mesmo número ocupa sempre o mesmo espaço de endereçamento da CPU, e por isso apenas quatro páginas de número diferente podem ficar ativas ao mesmo tempo, ainda que em slots diferentes. A ilustração abaixo mostra como os slots e páginas são organizados.

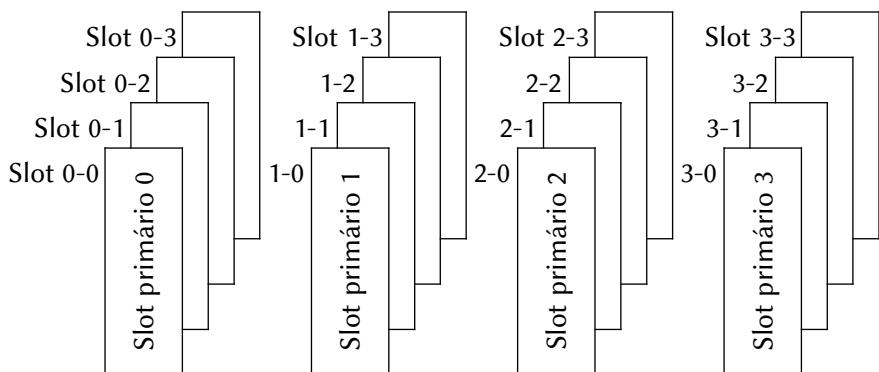


Cada slot primário pode suportar até quatro slots secundários. A escolha das páginas continua sendo possível da mesma forma que nos slots primários: apenas quatro páginas podem ficar ativas ao mesmo tempo, ainda que em slots primários e secundários diferentes.

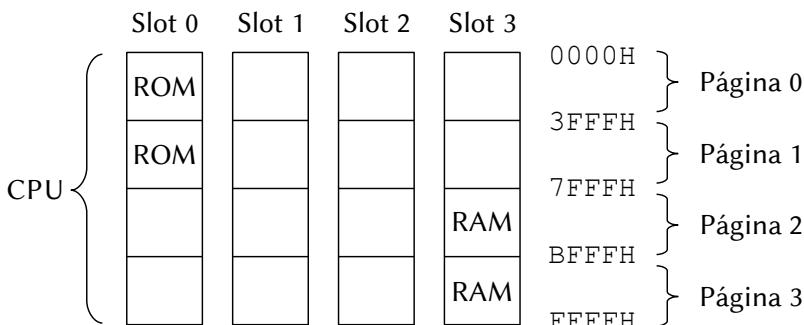
---

<sup>1</sup> É necessário um certo cuidado para não confundir a técnica de slots com a expansão de memória “Memory Mapper” que utiliza outro artifício para que cada slot possa acessar efetivamente até 4 megabytes não lineares.

Pode haver 16 slots no máximo, dos quais usualmente 8 são internos e reservados para a expansão do sistema e os outros 8 ficam disponíveis para o usuário, na forma de dois conectores de cartucho que podem ser expandidos para quatro conectores secundários cada. Abaixo está ilustrada a forma como os slots primários e secundários estão estruturados. O primeiro número é o slot primário e o segundo é o secundário, na forma “Slot P-S”.

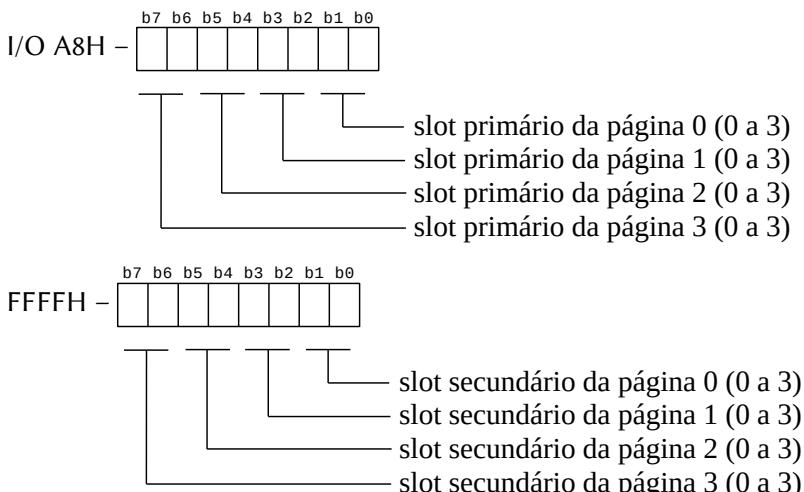


A seleção inicial mais comum de memória é a seguinte:



Quando o MSX é ligado ou resetado, o espaço de endereçamento é conectado automaticamente no slot 0. Por isso, a Main-ROM com a BIOS é colocada nas duas primeiras páginas do slot 0-0. Já a memória RAM pode estar em outros slots, porém o mais comum é que esteja nas duas últimas páginas do slot primário 3.

Normalmente, em uma máquina MSX1 padrão, há apenas slots primários. A cada slot primário podem ser atribuídos até quatro slots secundários. A seleção de slots e páginas é diferente para slots primários e secundários. Para os slots primários, ela é feita pela porta de I/O A8H e para os slots secundários é feita pelo registrador de slot secundário, que é que o endereço FFFFH da memória<sup>2</sup>. O formato dos registradores de slots está descrito abaixo.



Para obter o valor correto do slot secundário no endereço FFFFH, é necessário fazer uma inversão após a leitura (instrução NOT no BASIC ou CPL em Assembly). É importante observar que o valor só é invertido quando LIDO. Quando o valor for ESCRITO, não é invertido.

No MSX1, há apenas a Main-Rom de 32K. Para o MSX2 em dian-te, há também a Sub-ROM, que é colocada em outro slot. Os slots onde ficam instaladas a Main-ROM, a Sub-ROM e a RAM dependem de cada máquina. Em muitos casos, é necessário saber onde estão instaladas as memórias básicas do MSX, como no caso de estar rodando um programa sob o DOS e ser necessário acessar a Main-ROM, por exemplo. Os slots

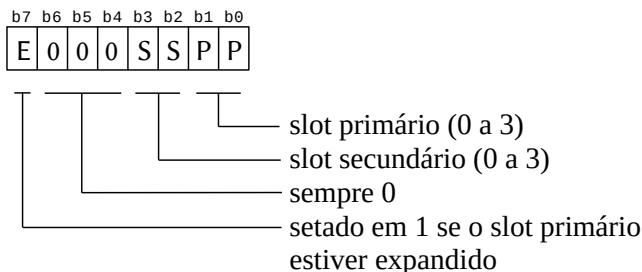
---

2 Não é recomendável que se troque slots e páginas diretamente e é necessário um cuidadoso planejamento para se chavear páginas e slots. Para utilizar rotinas em outras páginas é recomendável sempre usar a BIOS, que além de ser mais seguro e garantir a compatibilidade, simplifica muito a operação de slots e páginas.

onde estão instaladas a Main-ROM e a Sub-ROM são especificados nas seguintes variáveis de sistema:

- EXPTBL (FCC1H) – Slot da Main-ROM
- EXBRSA (FAF8H) – Slot da Sub-ROM (0 para MSX1)

Abaixo está ilustrada a estrutura desses registradores.



### 2.1.1 – Chamadas Inter-slots

Quando um programa está rodando em um determinado slot e deve chamar alguma rotina em outro slot, ele deverá fazer uma chamada inter-slot.

As chamadas inter-slot foram criadas para possibilitar que um determinado software tenha acesso às rotinas em outros slots em qualquer área de memória. Assim, um software rodando sob o MSXDOS, que ocupa toda a RAM linear disponível, pode acessar a BIOS ou o DOS Kernel residente na interface de disco. A partir do MSX2, há uma ROM de extensão da BIOS denominada Sub-ROM. As chamadas que a própria BIOS faz à Sub-ROM também são chamadas inter-slot.

Para facilitar as chamadas e assegurar a compatibilidade, existe um grupo de rotinas da BIOS denominado “grupo de chamadas inter-slot”. Algumas dessas rotinas também estão disponíveis para o MSXDOS, para que este possa acessar todas as rotinas da BIOS. As rotinas disponíveis para o MSXDOS estão descritas no capítulo sobre o sistema de disco.

As rotinas “inter-slot” da BIOS são as seguintes:

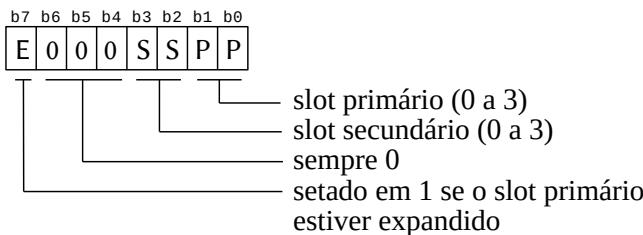
RDSLT	(000CH)	Lê um byte em qualquer slot
WRSLT	(0014H)	Escreve um byte em qualquer slot
CALSLT	(001CH)	Chama uma rotina em qualquer slot
ENASLT	(0024H)	Troca páginas e slots
CALLF	(0030H)	Chama uma rotina em qualquer slot
RSLREG	(0138H)	Lê o registrador de slot primário
WSLREG	(013BH)	Escreve no registrador de slot primário
SUBROM	(015CH)	Chama uma rotina na Sub-ROM
EXTROM	(015FH)	Chama uma rotina na Sub-ROM

A descrição completa de cada rotina pode ser vista no apêndice “CHAMADAS DA BIOS”.

## 2.1.2 – Área de Trabalho

A área de trabalho que contém as variáveis de sistema relativas aos slots são as seguintes (a descrição completa das variáveis de sistema pode ser vista no apêndice “DESCRIÇÃO DA ÁREA DE TRABALHO DO SISTEMA”):

EXRSA (FAF8H,1) – Slot da SUB-ROM

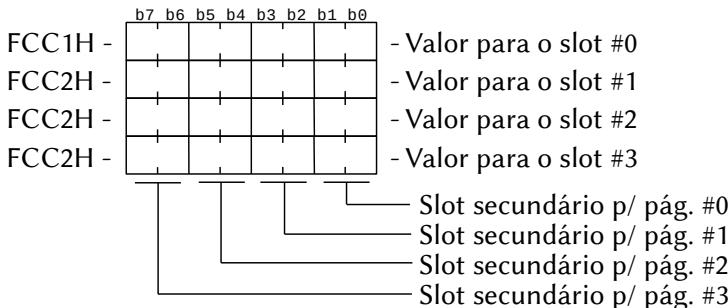


EXPTBL (FCC1H,4) – Indica se o slot primário está expandido.

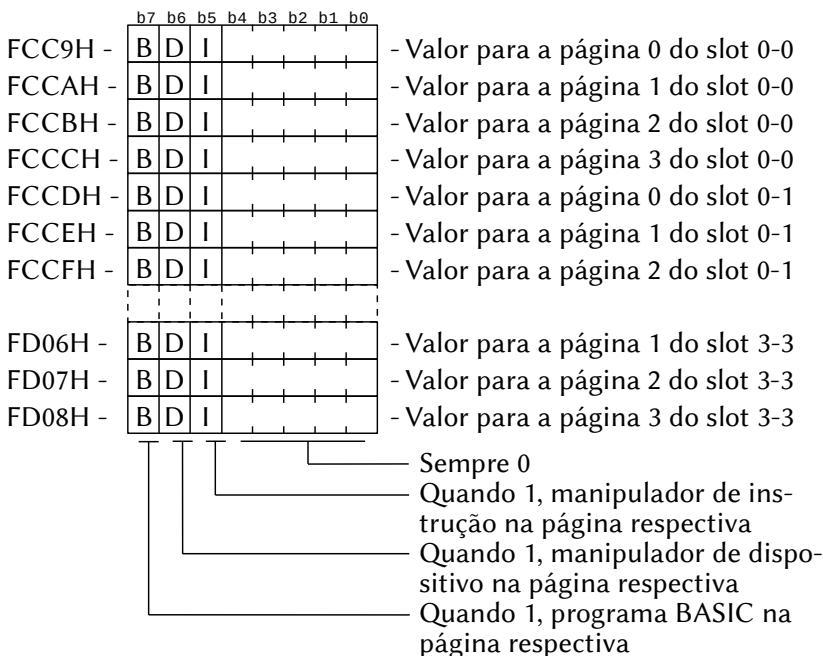
FCC1H -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr> <td>E</td><td>-</td><td>-</td><td>-</td><td>S</td><td>S</td><td>P</td><td>P</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	E	-	-	-	S	S	P	P	- Slot da Main-ROM
b7	b6	b5	b4	b3	b2	b1	b0											
E	-	-	-	S	S	P	P											
FCC2H -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>E</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </table>	E	-	-	-	-	-	-	-	- Slot primário #1								
E	-	-	-	-	-	-	-											
FCC2H -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>E</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </table>	E	-	-	-	-	-	-	-	- Slot primário #2								
E	-	-	-	-	-	-	-											
FCC2H -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>E</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </table>	E	-	-	-	-	-	-	-	- Slot primário #3								
E	-	-	-	-	-	-	-											
		0: slot primário NÃO está expandido 1: slot primário está expandido																

O primeiro byte (FCC1H) também indica o slot da Main-ROM em seus quatro últimos bits. Para os outros três bytes apenas o bit 7 é válido.

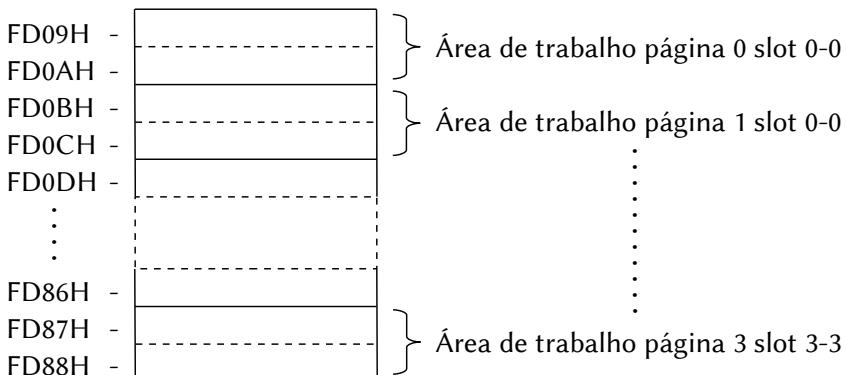
**SLTTBL (FCC5H,4)** – Área onde ficam armazenados os valores de expansão de cada slot primário.



**SLTATR (FCC9H,64)** – Indica a existência de rotinas em qualquer página de qualquer slot.



**SLTWRK (FD09H,128)** - Área de trabalho dos slots e páginas, reservando dois bytes para cada página.



## 2.2 – DESENVOLVENDO SOFTWARE EM CARTUCHO

Normalmente, os micros MSX possuem dois slots primários externos onde podem ser encaixados cartuchos contendo softwares, interfaces, etc. Programas BASIC ou Assembler podem ser facilmente armazenados em cartuchos contendo uma ROM ou EPROM.

Os cartuchos devem ter obrigatoriamente os primeiros 16 bytes reservados para o header. O header pode iniciar nos endereços 4000H ou 8000H, portanto somente nas páginas 1 ou 2. Os cartuchos não podem ocupar a área de endereçamento das páginas 0 e 3. Quando o micro é resetado, as informações contidas no header do cartucho são automaticamente reconhecidas para que o MSX possa executar corretamente as rotinas contidas no mesmo. A composição do header do cartucho é a seguinte:

+00H	ID	← 4000H ou 8000H
+02H	INIT	
+04H	STATEMENT	
+06H	DEVICE	
+08H	TEXT	
+0AH~+10H	RESERVADO	

**ID** – São dois bytes de identificação. No caso de cartuchos ROM, esses bytes devem ter o código “AB” (bytes 41H e 42H) e no caso de cartuchos de Sub-ROM, os bytes devem ser “CD” (43H e 44H).

**INIT** – Quando é necessário inicializar a área de trabalho ou I/O, esses dois bytes devem conter o endereço da rotina de inicialização; caso contrário devem conter o valor 0000H. Depois que a rotina de inicialização foi executada, a instrução RET retorna o controle ao micro. Todos os registradores podem ser modificados, exceto o registrador SP. Programas em Assembler também devem ser executados diretamente pelos bytes INIT.

**STATEMENT** – Quando o cartucho deve ser acessado pela instrução CALL do BASIC, esses dois bytes devem conter o endereço de início da rotina de expansão, caso contrário devem conter o valor 0000H.

A instrução CALL<sup>3</sup> tem o seguinte formato:

CALL <nome da instrução de expansão> (argumento)

O nome da instrução de expansão pode ter até 15 caracteres. Quando o interpretador BASIC encontra um comando CALL, o nome da instrução de expansão é colocado na variável de sistema PROCNM (FD89H) e o controle é transferido para a rotina cujo início é indicado pelos bytes STATEMENT. É essa rotina que deve reconhecer o nome da instrução em PROCNM. O registrador HL aponta exatamente para o primeiro caractere após a instrução expandida, conforme a ilustração abaixo:

CALL COMANDO (0,1,2):A=0



(HL)

PROCNM → 

C	O	M	A	N	D	O	00
---	---	---	---	---	---	---	----



Fim do nome da instrução  
expandida (byte 00H)

---

3 A abreviação da instrução CALL é o caractere sublinhado “\_”. Pode ser usado no lugar de CALL sem nenhum problema.

Quando a rotina de expansão não reconhece o comando, ela deve manter o valor de HL, setar a flag CY (CY=1) e devolver o controle ao interpretador (instrução RET). O interpretador vai então procurar outros cartuchos de expansão de comandos, se houver mais de um, e o procedimento será o mesmo. Se ao final a instrução não for reconhecida como válida, a flag CY ficará setada e será exibida a mensagem “Syntax Error” (erro de sintaxe). Esse procedimento está ilustrado abaixo.

```
CALL COMANDO (0,1,2):A=0
```



```
Flag CY=1      (HL)
```

Já se o comando for reconhecido como válido, a rotina correspondente será executada e no retorno ao interpretador, a flag CY deverá estar resetada (CY=0) e o registrador HL deve apontar para o primeiro sinalizador após o argumento da instrução expandida. O sinalizador pode ser o valor 00H (fim de linha) ou 3AH (dois pontos, separador de instruções). O processamento continuará normalmente. O exemplo abaixo ilustra o procedimento descrito.

```
CALL COMANDO (0,1,2):A=0
```



```
Flag CY=0      (HL)
```

**DEVICE** – Esses dois bytes podem apontar para uma rotina de expansão de dispositivos no caso do cartucho conter um dispositivo de I/O; caso contrário devem ser 0000H. A rotina para o dispositivo de expansão deve estar entre 4000H e 7FFFH. Um cartucho pode ter até quatro dispositivos, e o nome de cada um deles pode ter até 15 caracteres.

Quando o interpretador encontra um dispositivo indefinido, ele armazena o nome em PROCNM (FD89H), coloca o valor FFH no registrador A e passa o controle para o cartucho que tenha uma expansão de dispositivo.

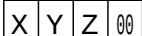
Para criar uma rotina de expansão de dispositivos, o descritor de arquivo deve ser identificado em PROCNM (FD89H) primeiro, e se não for o dispositivo correto, o controle deve ser devolvido ao interpretador com a flag CY setada (CY=1). O exemplo abaixo ilustra o que foi descrito.

OPEN "XYZ:"...

 Nome do dispositivo

Registrador A = FFH

Flag CY=1

PROCNM →  X Y Z 00

 Fim do descriptor de arquivo (byte 00H)

Já se o descriptor de dispositivo for reconhecido, a rotina respectiva deve ser processada e o número de identificação do dispositivo (device ID), que varia de 0 a 3, deve ser colocado no registrador A; a seguir, a flag CY deve ser resetada (CY=0) e o controle devolvido ao interpretador.

O interpretador procura cartucho após cartucho, e se ao final o nome de dispositivo não for reconhecido (ou seja, a flag CY sempre for 1), a mensagem de erro “Bad file name” (nome de arquivo errado) será mostrada.

Quando a operação de I/O atual é processada, ou seja, o dispositivo é válido, o interpretador coloca o nome do dispositivo (device ID – 0 a 3) na variável de sistema DEVICE (FD99H) e seta o dispositivo requerido em A com os valores descritos na tabela abaixo, para depois chamar a rotina de expansão do dispositivo.

Reg. A Dispositivo

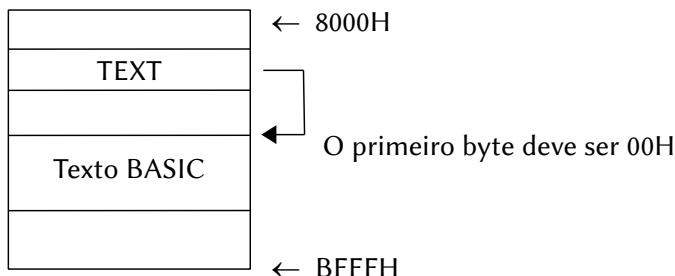
0	OPEN
2	CLOSE
4	Acesso aleatório
6	Saída sequencial
8	Entrada sequencial

Reg. A Dispositivo

10	Função LOC
12	Função LOF
14	Função EOF
16	Função FPOS
18	Caractere “Backup”

**TEXT** – Esses dois bytes apontam para um programa BASIC gravado em cartucho, autoexecutável quando o micro for resetado ou ligado. Se não houver programa BASIC, esses dois bytes devem conter 0000H. O tamanho do programa não pode ultrapassar 16 Kbytes (8000H a BFFFH).

O interpretador examina o conteúdo de TEXT, e se este contiver um endereço, inicia a execução do programa BASIC contido no endereço indicado. O primeiro byte apontado por TEXT deve ser 00H, que indica o início do texto BASIC. A figura abaixo mostra como o texto BASIC deve estar disposto no cartucho ROM para que a execução seja correta.



Para colocar programas BASIC em ROM alguns passos devem ser seguidos. Primeiro, troque o endereço inicial do texto BASIC para 8021H. Podem ser usados os seguintes comandos, em uma única linha:

```
POKE &HF676,&H21:POKE &HF677,&H80:POKE &H8020,0:NEW
```

Em seguida, carregue o programa BASIC desejado. Depois, crie a ID do cartucho. Podem ser usados os seguintes comandos, no modo direto<sup>4</sup>:

```
AD = &H8000
FOR I=0 TO 31:POKE AD+I,0:NEXT I
POKE &H8000,ASC("A")
POKE &H8001,ASC("B")
POKE &H8008,&H20
POKE &H8009,&H80
```

Em seguida, grave o conteúdo da página 2 (8000H a BFFFH) em um cartucho de ROM.

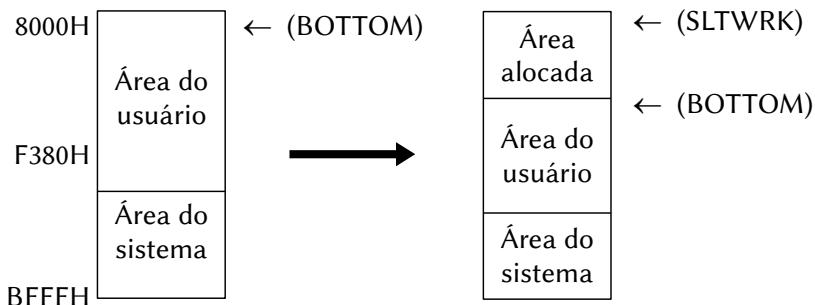
---

<sup>4</sup> A segunda linha (FOR I=0 TO 31:POKE AD+I,0:NEXT I) preenche a área de ID (8000H a 8020H) com zeros.

### 2.2.1 – Alocando área de trabalho para cartuchos

Para programas que não requeiram softwares de outros cartuchos, como jogos, por exemplo, a área de memória abaixo de F380H pode ser usada livremente. Mas em programas que são executados usando as funções do interpretador BASIC ou da BIOS, há três opções:

1. Colocar RAM no próprio cartucho (o melhor método).
2. Quando apenas um ou dois bytes forem requeridos para a área de trabalho, podem ser usados os dois bytes correspondentes na variável de sistema SLTWRK (FD09H~FD88H).
3. Quando mais de dois bytes são requeridos para a área de trabalho, é necessário alocar RAM usada pelo BASIC. Para fazer isso, é necessário colocar o conteúdo de BOTTOM (FC48H) na área correspondente em SLTWRK (FD09H~FD88H) e incrementar o valor de BOTTOM até o valor requerido para a área de trabalho. Essa área pode então ser usada como área de trabalho pelo cartucho. A figura abaixo ilustra esse método.



Também pode ser usada a área logo abaixo da área de trabalho, embora esse método seja desaconselhado. Ao conectar uma ROM com o DOS Kernel (Sistema de Disco), uma parte da RAM é alocada logo abaixo de F380H. A área de trabalho do cartucho deve ser alocada logo abaixo desta, para uma segurança mínima. Com dois drives lógicos conectados para o DOS1, é ocupada uma área até próximo de DA00H; no caso do DOS2, com oito drives lógicos conectados, até próximo de E100H. A área alocada deve estar logo abaixo. A melhor forma de alocá-la é chamando o comando CLEAR do BASIC, pois muitas variáveis de sistema tem que

ser alteradas. Para mais detalhes, pode ser vista a seção “CHAMANDO COMANDOS EM BASIC” no capítulo 3 (MEMÓRIA ROM).

A seguinte rotina foi publicada no MSX2 Technical Handbook como exemplo de alocação de área de trabalho para cartuchos.

```
;*****
;Subroutines to support slot for ROM in 1 page
;*****
;

RSLREG      EQU    00138H
EXPTBL      EQU    0FCC1H
BOTTOM       EQU    0FC48H
HIMEM        EQU    0FC4AH
SLTWRK      EQU    0FD09H
;

;-----
;

;GTSL1      Get slot number of designated page
;Entry       None
;Return      A - Slot address as follows
;Modify      Flags
;

;      FxxxSSPP
;      |  || |
;      |  ||+-- primary slot # (0-3)
;      |  +--- secondary slot # (0-3)
;      |          00 if not expanded
;      +----- 1 if secondary slot # specified
;
;This value can later be used as an input parameter
;for the RDSL, WRSLT, CALSLT, ENASLT and 'RST 10H'
;
PUBLIC        GTSL10
GETSL10:
    PUSH   HL           ;Save registers
    PUSH   DE

    CALL   RSLREG      ;read primary slot #
    RRCA
    RRCA
    AND    11B          ;[A]=000000PP
    LD     E,A
```

```

LD      D, 0           ; [DE]=000000PP
LD      HL, EXPTBL
ADD    HL, DE          ; [HL]=EXPTBL+000000PP
LD      E, A           ; [E]=000000PP
LD      A, (HL)        ; A=(EXPTBL+000000PP)
AND    80H             ; Use only MSB
JR     Z, GTSL1NOEXP
OR     E               ; [A]=F00000PP
LD      E, A           ; save primary slot number
INC    HL               ; point to SLTTBL entry
INC    HL
INC    HL
INC    HL
LD      A, (HL)        ; get current expansion slot
                           ; register
RRCA
RRCA
AND    11B             ; [A] = 000000SS
RLCA
RLCA             ; [A] = 0000SS00
OR     E               ; [A] = F000SSPP
;
GTSL1END:
POP    DE
POP    HL
RET
GTSL1NOEXP:
LD      A, E           ; [A] = 000000PP
JR     GTSL1END

;-----
;
;ASLW1      Get address of slot work
;Entry      None
;Return     HL      address of slot work
;Modify     None
;
PUBLIC      ASLW10
ASLW10:
PUSH    DE
PUSH    AF
CALL   GTSL10          ; [A] = F000SSPP, SS = 00 if

```

```

        ;not expanded
AND    00001111B      ;[A] = 0000SSPP
LD     L,A            ;[A] = 0000SSPP
RLCA
RLCA
RLCA
RLCA      ;[A] = SSPP0000
AND    00110000B      ;[A] = 00PP0000
OR     L               ;[A] = 00PPSSPP
AND    00111100B      ;[A] = 00PPSS00
OR     01B             ;[A] = 00PPSSBB

;
;Now, we have the sequence number for this cartridge
;as follows.

;
;      00PPSSBB
;      | | | | |
;      | | | +--- higher 2 bits of memory address (1)
;      | | +---- secondary slot # (0..3)
;      +---- primary slot # (0..3)

;
RLCA      ;*=2
LD     E,A
LD     D,0      ;[DE] = 0PPSSBB0
LD     HL,SLTWRK
ADD   HL,DE
POP   AF
POP   DE
RET

-----
;

;RSLW1      Read slot work
;Entry      None
;Return     HL      Content of slot work
;Modify     None
;
PUBLIC      RSLW10
RSLW10:
    PUSH  DE
    CALL  ASLW10      ;[HL] = address of slot work

```

```

LD      E, (HL)
INC    HL
LD      D, (HL)      ; [DE] = (slot work)
EX      DE, HL        ; [HL] = (slot work)
POP    DE
RET

;-----
;

;WSLW1      Write slot work
;Entry       HL      Data to write
;Return      None
;Modify      None
;
PUBLIC      WSLW10
WSLW10:
PUSH    DE
EX      DE, HL        ; [DE] = data to write
CALL    ASLW10        ; [HL] = address of slot work
LD      (HL), E
INC    HL
LD      (HL), D
EX      DE, HL        ; [HL] = data tow write
POP    DE
RET

;-----
;

; How to allocate work area for cartridges.
; If the work area is greater than 2 bytes, make the
; SLTWRK point to the system variable BOTTOM
; (0FC48H), then update it by the amount of memory
; required. BOTTOM is set up by the initialization
; code to point to the bottom of equipped RAM.
;
;     Ex, if the program is at 4000H..7FFFH.
;
;WORKB      allocate work area from BOTTOM
;           (my slot work) <- (old BOTTOM)
;Entry       HL      required memory size
;Return      HL      start address of my work area
;           = old BOTTOM
;
```

```
;           0 if cannot allocate
;Modify      None
;
PUBLIC      WORKB0
WORKB0:
    PUSH   DE
    PUSH   BC
    PUSH   AF

    EX     DE,HL      ;[DE] = Size
    LD     HL,(BOTTOM) ;Get current RAM bottom
    CALL   WSLW10     ;Save BOTTOM to slot work
    PUSH   HL          ;Save old BOTTOM
    ADD    HL,DE      ;[HL] = (BOTTOM) + SIZE
    LD     A,H         ;Beyond 0FFFFH?
    CP     0E0H
    JR     NC,NOROOM  ;Yes, cannot allocate this
                       ;much
    LD     (BOTTOM),HL ;Updtae (BOTTOM)
    POP   HL          ;[HL] = old BOTTOM

WORKBEND:
    POP   AF
    POP   BC
    POP   DE
    RET

;
;      BOTTOM became greater than 0FFFFH, there is
;      no RAM to be allocated.
;

NOROOM:
    LD     HL,0
    CALL  WSLW10     ;Clear slot work
    JR     WORKBEND  ;Return 0 in [HL]

END
```

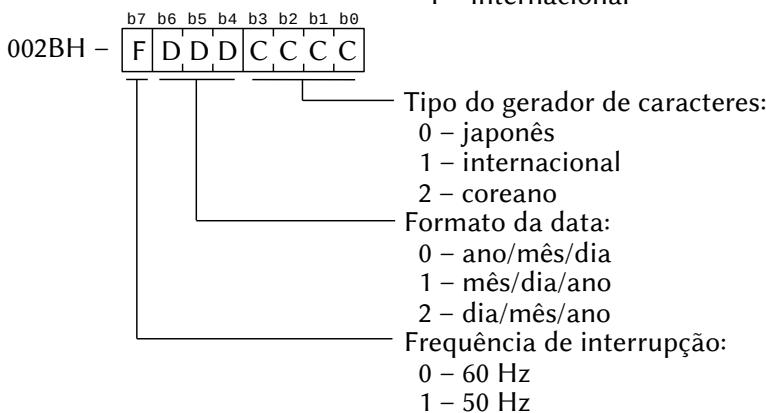
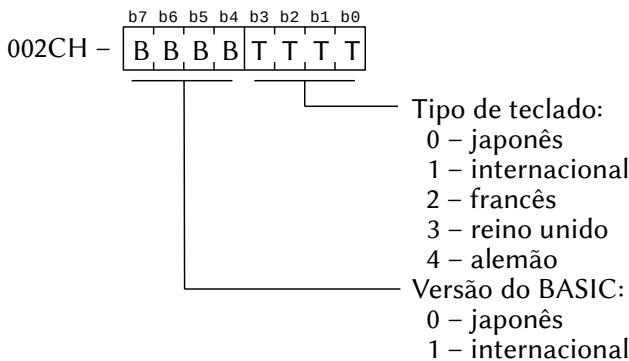
## **Capítulo 3**

### A MEMÓRIA ROM

A memória ROM é vital para o funcionamento de qualquer microcomputador. No caso do MSX, ela incorpora a rotina de inicialização, a BIOS, a tabela inicial de caracteres, o MSX-DOS (DOS Kernel), etc.

Além disso, existem alguns bytes no início da ROM que contêm algumas informações importantes que podem ser úteis ao programador. Esses bytes são:

- |             |  |
|-------------|--|
| 0004H/0005H | - Endereço da tabela de caracteres na ROM. |
| 0006H       | - Porta de leitura de dados do VDP.        |
| 0007H       | - Porta de escrita de dados no VDP.        |



## 002DH – Versão do Hardware

- 00H = MSX1
- 01H = MSX2
- 02H = MSX2+
- 03H = MSX turbo R

### 3.1 – BIOS

Praticamente todos os programas, seja em assembler ou linguagem de alto nível, incluindo o próprio interpretador BASIC residente no MSX, requer um conjunto de funções primárias para poder operar. Essas funções incluem acionadores de tela, impressoras, drives e outras funções relacionadas ao hardware. No MSX, essas funções primárias são realizadas pelas rotinas da BIOS, que significa “Basic Input/Output System”, ou “Sistema Básico de Entrada e Saída”.

Existem 137 rotinas da BIOS disponíveis ao usuário, se o micro for um MSX turbo R. Para versões anteriores, o número de rotinas disponíveis é menor.

Existem dois tipos de rotinas da BIOS: as que estão na Main-ROM e as que estão na Sub-ROM. Para o MSX1 não existe Sub-ROM; para o MSX2, MSX2+ e MSX turbo R há 16K, 32K e 48K de Sub-ROM, respectivamente. As rotinas da Main-ROM e da Sub-ROM usam diferentes sequências de chamada. Para a Main-ROM, pode ser usada uma instrução CALL ou RST. As chamadas para a Sub-ROM são feitas com o auxílio da rotina EXTROM (015FH) ou SUBROM (015CH) da Main-ROM, carregando IX com o endereço da rotina da Sub-ROM a ser chamada, e procedendo conforme o exemplo abaixo:

```
LD    IX, ROTINA      ;carrega IX com o endereço
CALL EXTROM          ;executa a rotina
...                  ;retorno da rotina aqui
```

Quando o conteúdo de IX não deve ser destruído, a seguinte sequência de chamada deve ser usada:

```
INIROT: PUSH IX        ;salva IX
       LD    IX, ROTINA  ;carrega IX com o endereço
       JP    SUBROM      ;executa a rotina
       ...              ;retorno da chamada INIROT
```

Todas as rotinas da Main-ROM e da Sub-ROM estão descritas detalhadamente no apêndice, capítulo 8, seções 8.1 e 8.2 – Rotinas da MainROM e da SubROM.

### 3.2 – O MATH-PACK (PACOTE MATEMÁTICO)

O Math-Pack (Pacote Matemático) é um conjunto de rotinas matemáticas que não pertencem à BIOS e que constituem o centro das operações matemáticas do MSX-BASIC. Essas rotinas podem ser utilizadas por programas Assembly, tornando disponíveis operações com ponto flutuante, aritméticas, logarítmicas e trigonométricas, além de várias operações especiais.

As operações envolvendo números reais com o Math-Pack são realizadas em BCD (Binary Coded Decimal). Os números podem ser inteiros de 2 bytes (-32 768 a +32 767), de precisão simples (6 dígitos, com expoente de -63 a +63) ocupando 4 bytes ou de precisão dupla (14 dígitos com expoente de -63 a +63), ocupando 8 bytes.

Para armazenar um número inteiro, o bit b15 é reservado para o sinal, restando 15 bits para armazenar o módulo. Os números inteiros têm o seguinte formato:

b15	B14~b0	decimal
0	0000000000000000	+0
0	0000000000000001	+1
:	:	:
0	1111111111111111	+32767
1	0000000000000000	-32768
1	0000000000000001	-32767
:	:	:
1	1111111111111111	-1

Um número real é composto por uma mantissa, um sinal e um expoente. O sinal da mantissa é representado por 0 (positivo) ou 1 (negativo). O expoente é uma expressão binária de 7 bits que representa uma potência de 10 e pode variar de -63 a +63. A forma como os números de ponto flutuante são armazenados na memória está ilustrada abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0	
+/-								expoente	0
1º dígito								1	
3º dígito								2	
5º dígito								3	
7º dígito								4	
9º dígito								5	
11º dígito								6	
13º dígito								7	

precisão simples

precisão dupla

mantissa

Abaixo está ilustrada a expressão binária formadora do expoente e do sinal da mantissa.

b7	b6	b5	b4	b3	b2	b1	b0	
+/-	expoente							
0	0	0	0	0	0	0	0	expoente positivo
1	0	0	0	0	0	0	0	indefinido (-0 ?)
x	0	0	0	0	0	0	0	-63ª potência de 10
x	0	0	0	0	0	0	0	0ª potência de 10
x	1	1	1	1	1	1	1	+63ª potência de 10

Exemplo de número de precisão simples:

123.456 → 0,123456E+6

DAC → 

0	1	2	3
46	12	34	56

Exemplo de número de precisão dupla:

123.456,78901234 → 0,12345 678901234E+6

DAC → 

0	1	2	3	4	5	6	7
46	12	34	56	78	90	12	34

Os dígitos que constituem a mantissa são sempre considerados como colocados logo após a vírgula.

### 3.2.1 – Área de trabalho

Para que o Math-Pack possa realizar suas funções, existem duas áreas de memória reservadas, que são o “DAC” (Decimal ACumulator, F7F6H) e o “ARG” (F847H). Por exemplo, numa multiplicação o produto

dos números contidos em DAC e ARG é calculado e o resultado é colocado em DAC.

No DAC, podem ser armazenados números de precisão dupla, precisão simples ou inteiros de dois bytes, sendo que nesse último caso os dois bytes que representam o número inteiro são armazenados em DAC+2 e DAC+3. Para que as rotinas do Math-Pack possam distinguir que tipo de número está armazenado em DAC, a variável de sistema VALTYP (F663H) é usada, devendo conter o valor 2 para números inteiros, 4 para números de precisão simples e 8 para números de precisão dupla.

Ao usar as rotinas do Math-Pack em assembly, deve-se tomar um cuidado especial. Como são rotinas usadas pelo interpretador BASIC, caso ocorra algum erro (como divisão por zero ou overflow, por exemplo), o controle é automaticamente transferido para o manipulador de erro que depois devolve o controle ao interpretador. Para evitar que isso ocorra, pode ser usado o hook HERRO (FFB1H) para interceptar o erro antes que o controle seja devolvido ao interpretador. O código de erro fica no registrador E da CPU, podendo também ser usado pelo programa assembly.

Para usar as rotinas do Math-Pack em programas Assembly deve-se proceder exatamente da mesma forma como se chama as rotinas da BIOS. Colocam-se os devidos valores em ARG, DAC e VALTYP e eventualmente em algum registrador da CPU e chama-se a rotina desejada através da instrução CALL ou através das rotinas CALSLT ou CALLF. A única observação a fazer é que pouquíssimas rotinas preservam algum registrador; portanto é necessário sempre salvar na pilha os registradores que não devem ser destruídos. Abaixo e na página seguinte estão descritas variáveis de sistema usadas pelo Math-Pack.

VALTYP (F663H, 1 byte)

Formato do número contido em DAC (2, 4 ou 8)

DAC (F7F6H, 16 bytes)

Acumulador de ponto flutuante no formato BCD.

ARG (F847H, 16 bytes)

Argumento para uso com DAC.

### 3.2.2 – Entradas do Math-Pack

#### 3.2.2.1 – Operações em ponto flutuante

Label	Endereço	Função
DECSUB	268CH	DAC $\leftarrow$ DAC – ARG (Precisão dupla)
DECADD	2691H	DAC $\leftarrow$ DAC + ARG (Precisão dupla)
DECMUL	27E6H	DAC $\leftarrow$ DAC * ARG (Precisão dupla)
DECDIV	289FH	DAC $\leftarrow$ DAC / ARG (Precisão dupla)
SGNEXP	37C8H	DAC $\leftarrow$ DAC $^$ ARG (Precisão simples)
DBLEXP	37D7H	DAC $\leftarrow$ DAC $^$ ARG (Precisão dupla)
COS	2993H	DAC $\leftarrow$ COS (DAC)
SIN	29ACh	DAC $\leftarrow$ SIN (DAC)
TAN	29FBH	DAC $\leftarrow$ TAN (DAC)
ATN	2A14H	DAC $\leftarrow$ ATN (DAC)
LOG	2A72H	DAC $\leftarrow$ LOG (DAC)
SQR	2AFFH	DAC $\leftarrow$ SQR (DAC)
EXP	2B4AH	DAC $\leftarrow$ EXP (DAC)

#### 3.2.2.2 – Operações com números inteiros

Label	Endereço	Função
UMULT	314AH	DE $\leftarrow$ BC * DE (unsigned mult)
ISUB	3167H	HL $\leftarrow$ DE – HL
IADD	3172H	HL $\leftarrow$ DE + HL
IMULT	3193H	HL $\leftarrow$ DE * HL
IDIV	31E6H	HL $\leftarrow$ DE / HL
INTEXP	383FH	DAC $\leftarrow$ DE $^$ HL
IMOD	323AH	{ HL $\leftarrow$ DE mod HL DE $\leftarrow$ DE / HL

#### 3.2.2.3 – Funções especiais

Label	Endereço	Função
DECNRM	26FAH	Normaliza DAC <sup>5</sup>
DECROU	273CH	Arredonda DAC

5 Zeros excessivos na mantissa são removidos (Ex. 0,00123  $\rightarrow$  0,123E-2)

RND	2BDFH	DAC $\leftarrow$ RND (DAC)
SIGN	2E71H	A $\leftarrow$ sinal da mantissa em DAC
ABSFN	2E82H	DAC $\leftarrow$ ABS (DAC)
NEG	2E8DH	DAC $\leftarrow$ NEG (DAC)
SGN	2E97H	DAC $\leftarrow$ SGN (DAC) <sup>6</sup>
FRCINT	2F8AH	Converte DAC em inteiro de 2 bytes <sup>7</sup>
FRCSNG	2FB2H	Converte DAC para precisão simples
FRCDBL	303AH	Converte DAC para precisão dupla
FIXER	30BEH	DAC $\leftarrow$ SGN(DAC) * INT(ABS(DAC))

### 3.2.2.4 – Movimento

Label	Endereço	Função	
MAF	2C4DH	ARG $\leftarrow$ DAC	precisão dupla
MAM	2C50H	ARG $\leftarrow$ (HL)	precisão dupla
MOV8DH	2C45H	(DE) $\leftarrow$ (HL)	precisão dupla
MFA	2C59H	DAC $\leftarrow$ ARG	precisão dupla
MFM	2C5CH	DAC $\leftarrow$ (HL)	precisão dupla
MMF	2C67H	(HL) $\leftarrow$ DAC	precisão dupla
MOV8HD	2D6AH	(HL) $\leftarrow$ (DE)	precisão dupla
XTF	2C6FH	(SP) $\leftrightarrow$ DAC	precisão dupla
PHA	2CC7H	ARG $\leftarrow$ (SP)	precisão dupla
PHF	2CCCH	DAC $\leftarrow$ (SP)	precisão dupla
PPA	2CDCH	(SP) $\leftarrow$ ARG	precisão dupla
PPF	2CE1H	(SP) $\leftarrow$ DAC	precisão dupla
PUSHF	2EB1H	DAC $\leftarrow$ (SP)	precisão simples
MOVFM	2EBEH	DAC $\leftarrow$ (HL)	precisão simples
MOVFR	2EC1H	DAC $\leftarrow$ CBED	precisão simples
MOVRF	2ECCH	CBED $\leftarrow$ DAC	precisão simples
MOVRMI	2ED6H	CBED $\leftarrow$ (HL)	precisão simples
MOVRM	2EDFH	BCDE $\leftarrow$ (HL)	precisão simples
MOVMF	2EE8H	(HL) $\leftarrow$ DAC	precisão simples
MOVE	2EEBH	(HL) $\leftarrow$ (DE)	precisão simples

6 Para a função SGN, o resultado é representado por um inteiro de 2 bytes.

7 O resultado da conversão é armazenado em DAC +2 e +3.

VMOVAM	2EEFH	ARG $\leftarrow$ (HL)	VALTYP
MOVVFM	2EF2H	(DE) $\leftarrow$ (HL)	VALTYP
VMOVE	2EF3H	(HL) $\leftarrow$ (DE)	VALTYP
VMOVFA	2F05H	DAC $\leftarrow$ ARG	VALTYP
VMOVFM	2F08H	DAC $\leftarrow$ (HL)	VALTYP
VMOVAF	2F0DH	ARG $\leftarrow$ DAC	VALTYP
VMOVMF	2F10H	(HL) $\leftarrow$ DAC	VALTYP

**Obs.:** (HL) e (DE) significam os endereços de memória apontados por HL e DE. Quatro nomes de registradores juntos (CBED ou BCDE) contém um número de precisão simples (sinal + expoente, 1º e 2º dígitos, 3º e 4º dígitos, 5º e 6º dígitos). Quando o objeto for VALTYP, o movimento será de acordo com o tipo indicado por VALTYP (F663H), ou seja, 2, 4 ou 8 bytes.

### 3.2.2.5 – Comparações

Label	Endereço		Esquerdo	Direito
ICOMP	2F4DH	Inteiro de 2 bytes	DE	HL
FCOMP	2F21H	Precisão simples	CBED	DAC
XDCOMP	2F5CH	Precisão dupla	ARG	DAC

**Obs.:** Na comparação de números reais de precisão simples, CBED deverá conter um dos operandos de precisão simples (sinal + expoente, 1º e 2º dígitos, 3º e 4º dígitos, 5º e 6º dígitos). O resultado da comparação será colocado no registrador A, conforme mostrado abaixo:

A = 1  $\rightarrow$  esquerdo < direito

A = 0  $\rightarrow$  esquerdo = direito

A = -1  $\rightarrow$  esquerdo > direito

### 3.2.2.6 – Outras operações em ponto flutuante e I/O

#### FIN (3299H)

**Função:** Converte uma string representando um número real para o formato BCD e o armazena em DAC.

**Entrada:** HL – Endereço do primeiro caractere da string  
A – Primeiro caractere da string

**Saída:** DAC – Número real em BCD  
C – FFH – sem ponto decimal; 0 – com ponto decimal  
B – Número de dígitos após o ponto decimal  
D – Número total de dígitos

**FOUT** (3225H)

**Função:** Converte um número real contido em DAC para um string sem formatar.

**Entrada:** A – Sempre 0

B – Número de dígitos antes do ponto decimal

C – Número de dígitos depois do ponto decimal, incluindo este.

**Saída:** HL – Endereço do primeiro caractere da string

**PUFOUT** (3426H)

**Função:** Converte um número real contido em DAC para uma string, formatando.

**Entrada:** A – bit 7: 0- não formatado 1- formatado

bit 6: 0- sem vírgulas 1- c/ vírgulas cada 3 díg.

bit 5: 0- sem significado 1- preenche espaços c/ “\*”

bit 4: 0- sem significado 1- adiciona “\$” antes do núm.

bit 3: 0- sem significado 1- coloca “+” p/ núm. posit.

bit 2: 0- sem significado 1- sinal depois do número

bit 1: 0- não utilizado

bit 0: 0- ponto fixo 1- ponto flutuante

B – Número de dígitos antes do ponto decimal

C – Número de dígitos depois do ponto decimal, incluindo este.

**Saída:** HL – Endereço do primeiro caractere da string.

**FOUTB** (371AH)

**Função:** Converte um número inteiro para uma expressão binária

**Entrada:** DAC+2 – número inteiro

VALTYP – 2

**Saída:** HL – Endereço do primeiro caractere da string.

**FOUTO** (371EH)

**Função:** Converte um número inteiro para uma expressão octal

**Entrada:** DAC+2 – número inteiro

VALTYP – 2

**Saída:** HL – Endereço do primeiro caractere da string.

**FOUTH** (3722H)

**Função:** Converte um número inteiro para uma expressão hexadecimal.

Entrada: DAC+2 – número inteiro  
 VALTYP – 2

Saída: HL – Endereço do primeiro caractere da string.

### 3.3 – O INTERPRETADOR BASIC

A maior parte do interpretador BASIC reside na página 1 da ROM. A área de texto de um programa BASIC inicia normalmente no endereço 8000H (que corresponde ao início da página 2) mas pode ser alterada mudando-se a variável de sistema TXTTAB (F676H) que contém inicialmente o valor 8000H e indica o início da área de texto BASIC.

#### 3.3.1 – Os tokens

Para cada palavra reservada do BASIC existe um código correspondente chamado “token” ou “átomo”. Um token nada mais é que um único byte representando uma palavra reservada do BASIC.

Como se pode concluir, o texto BASIC não é armazenado na forma ASCII, mas em uma forma mais compacta. A finalidade dos tokens não é apenas tornar o texto BASIC mais compacto, mas também mais rápido, visto que, durante o processamento, em vez de decodificar toda a sequência ASCII do comando, o interpretador precisa apenas decodificar um byte.

Um comando BASIC, por exemplo, “PRINT A”, estará armazenado na área de texto BASIC da seguinte forma:

byte 91H – Token do comando PRINT  
 byte 20H – Espaço  
 byte 41H – Código ASCII da variável ‘A’

Já as funções do BASIC são armazenadas de uma forma um pouco diferente. Os tokens das funções são precedidos por um byte FFH e têm seu bit 7 setado. Por exemplo, uma função do BASIC tipo “X=SIN(A)” é armazenada da seguinte forma:

byte 58H – Código ASCII da variável ‘X’  
 byte EFH – Token do sinal ‘=’  
 byte FFH – Identificador de função  
 byte 89H – Token setado da função SIN  
 byte 28H – Código ASCII de ‘(’

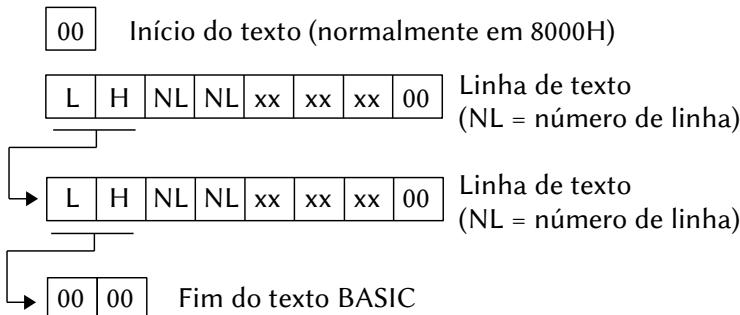
- byte 41H – Código ASCII da variável ‘A’  
 byte 29H – Código ASCII de ‘’

Todos os comandos e funções BASIC com seus respectivos tokens podem ser vistos na seção “CHAMANDO COMANDOS EM BASIC”.

### 3.3.2 – Estrutura das linhas de programa

A maneira pela qual as linhas de programa são armazenadas na área de texto BASIC é bastante simples.

Os dois primeiros bytes (normalmente 8001H e 8002H) contêm o endereço de início da próxima linha; os dois seguintes contêm o número da linha (que pode variar de 0 a 65 529) e em seguida vêm os bytes que armazenam a linha propriamente dita, podendo ter até 254 bytes, sendo que o último deve ser 00H, indicando o fim de linha. Quando for o fim do programa, são acrescidos mais dois bytes 00H, indicando esse fato. A forma de armazenamento das linhas está ilustrada abaixo.



### 3.3.3 – Armazenamento de números

Os números são armazenados de uma forma especial, visando economizar o máximo possível de memória na área de texto. Os números inteiros são tratados de uma forma bastante peculiar. São divididos em três grupos: 0 a 9, 10 a 255 e 256 a 32 767. Para os números inteiros de 0 a 9, há uma espécie de token que o identifica como tal, conforme a tabela abaixo:

0 – 11H	1 – 12H	2 – 13H	3 – 14H	4 – 15H
5 – 16H	6 – 17H	7 – 18H	8 – 19H	9 – 20H

Para os números inteiros de 10 a 255, é colocado um byte de identificação antes do número, que neste caso é o 0FH. Logo após o byte de identificação, está o byte que representa numericamente o valor, de 10 a 255. Para os números inteiros de 256 a 32 767, também existe um byte de identificação (1CH) seguido de dois bytes que armazenam o número na forma LSB-MSB. Se um número inteiro for negativo, ele será precedido pelo token do sinal de “-” (F2H). Veja os exemplos:

4	<table border="1"><tr><td>15H</td></tr></table>	15H			
15H					
-4	<table border="1"><tr><td>F2H</td><td>15H</td></tr></table>	F2H	15H		
F2H	15H				
35	<table border="1"><tr><td>0FH</td><td>23H</td></tr></table> (23H = 35 em decimal)	0FH	23H		
0FH	23H				
-35	<table border="1"><tr><td>F2H</td><td>0FH</td><td>23H</td></tr></table>	F2H	0FH	23H	
F2H	0FH	23H			
1000	<table border="1"><tr><td>1CH</td><td>E8H</td><td>03H</td></tr></table> (03E8H = 1000 em decimal)	1CH	E8H	03H	
1CH	E8H	03H			
-1000	<table border="1"><tr><td>F2H</td><td>1CH</td><td>E8H</td><td>03H</td></tr></table>	F2H	1CH	E8H	03H
F2H	1CH	E8H	03H		

Os números de precisão simples são armazenados em quatro bytes, na forma BCD, precedidos pelo byte de identificação 1DH. Os números de precisão dupla são armazenados em oito bytes, também na forma BCD, precedidos pelo byte de identificação 1FH.

Os números armazenados em outras bases (binário, octal e hexadecimal) também têm seus bytes de identificação. Para um número binário, são dois bytes ID (26H e 42H, ou “&B”), sendo que ele é armazenado na forma ASCII. Já os números octais têm como ID o byte 0BH e são armazenados na forma LSB-MSB. Para os números hexadecimais, o byte ID é 0CH e o número também é armazenado na forma LSB-MSB.

Os números que referem linhas de programas (nas instruções GOTO e GOSUB, por exemplo) têm um tratamento bem peculiar. Durante a digitação do programa, o número de linha é armazenado em dois bytes na forma LSB-MSB, precedidos pelo byte de identificação 0EH. Quando a linha for executada pela primeira vez, o interpretador mudará o byte ID para 0DH e os dois bytes seguintes conterão o endereço de início da linha respectiva, e não mais o número de linha. Isso é feito para acelerar a execução do programa na próxima vez que for executado.

As diversas formas de armazenamento estão ilustradas na figura abaixo.

Inteiro de 0 a 9	<table border="1"><tr><td>xx</td><td>(xx pode variar entre 11H a 1AH)</td></tr></table>	xx	(xx pode variar entre 11H a 1AH)						
xx	(xx pode variar entre 11H a 1AH)								
Inteiro de 10 a 255	<table border="1"><tr><td>0FH</td><td>xx</td></tr></table>	0FH	xx						
0FH	xx								
Inteiro de 256 a 32 767	<table border="1"><tr><td>1CH</td><td>xx</td><td>xx</td></tr></table>	1CH	xx	xx					
1CH	xx	xx							
Precisão simples	<table border="1"><tr><td>1DH</td><td>xx</td><td>xx</td><td>xx</td></tr></table>	1DH	xx	xx	xx				
1DH	xx	xx	xx						
Precisão dupla	<table border="1"><tr><td>1FH</td><td>xx</td><td>xx</td><td>xx</td><td>xx</td><td>xx</td><td>xx</td><td>xx</td></tr></table>	1FH	xx	xx	xx	xx	xx	xx	xx
1FH	xx	xx	xx	xx	xx	xx	xx		
Número octal	<table border="1"><tr><td>0BH</td><td>xx</td><td>xx</td></tr></table>	0BH	xx	xx					
0BH	xx	xx							
Número hexadecimal	<table border="1"><tr><td>0CH</td><td>xx</td><td>xx</td></tr></table>	0CH	xx	xx					
0CH	xx	xx							
Número binário	<table border="1"><tr><td>26H</td><td>42H</td><td>número na forma ASCII (0~1)</td></tr></table>	26H	42H	número na forma ASCII (0~1)					
26H	42H	número na forma ASCII (0~1)							
Linha (antes de RUN)	<table border="1"><tr><td>0EH</td><td>xx</td><td>xx</td></tr></table>	0EH	xx	xx					
0EH	xx	xx							
Linha (depois de RUN)	<table border="1"><tr><td>0DH</td><td>xx</td><td>xx</td></tr></table>	0DH	xx	xx					
0DH	xx	xx							

### 3.3.4 – Área de variáveis do interpretador

A área de memória logo acima do final do texto BASIC é alocada para armazenar as variáveis do programa. Essa área inicia no endereço apontado por VARTAB (F6C2H) e termina no endereço apontado por STREND (F6C6H)<sup>8</sup>. Cada vez que uma variável for consultada, o interpretador procura a mesma na área delimitada por VARTAB e STREND e, caso não a encontre, assume o valor 0 para variáveis numéricas ou nulo para variáveis string.

Sempre que uma nova linha BASIC é introduzida, deletada ou o comando CLEAR é executado, o valor de STREND é igualado ao valor de VARTAB e consequentemente todas as variáveis do programa são limpas e ficam nulas.

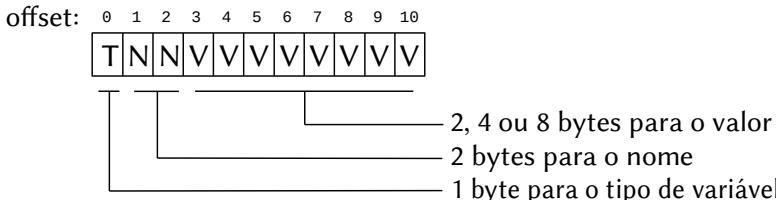
Existem 4 tipos de variáveis do BASIC:

---

8 As variáveis string e as matrizes são tratadas de forma diferente. Essa forma está descrita no final do item 3.4.

- numéricas inteiras: ocupam 2 bytes
- numéricas de precisão simples: ocupam 4 bytes
- numéricas de precisão dupla: ocupam 8 bytes
- alfanuméricicas (strings): ocupam 3 bytes

As variáveis numéricas possuem a sintaxe de armazenamento ilustrada abaixo:



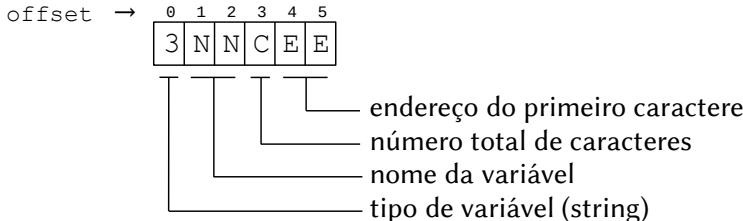
O primeiro byte indica o tipo de variável numérica que está armazenada: 2, 4 ou 8. Esse valor também já indica o número de bytes ocupados pela variável.

O interpretador assume como default as variáveis de dupla precisão, mas o tipo de variável pode ser alterado pelos comandos DEFINT, DEFSNG, DEFDBL e DEFSTR. Esses comandos possuem uma tabela que inicia em F6CAH e tem 26 bytes, um para cada letra do alfabeto, que indica que a variável cujo nome inicia com aquela letra deve assumir o tipo indicado:

- |              |                       |
|--------------|-----------------------|
| 02 – inteira | 04 – precisão simples |
| 03 – string  | 08 – precisão dupla   |

Os sinais de identificação imediata do tipo de variável (% , ! , # e \$) têm precedência sobre os valores indicados por essa tabela.

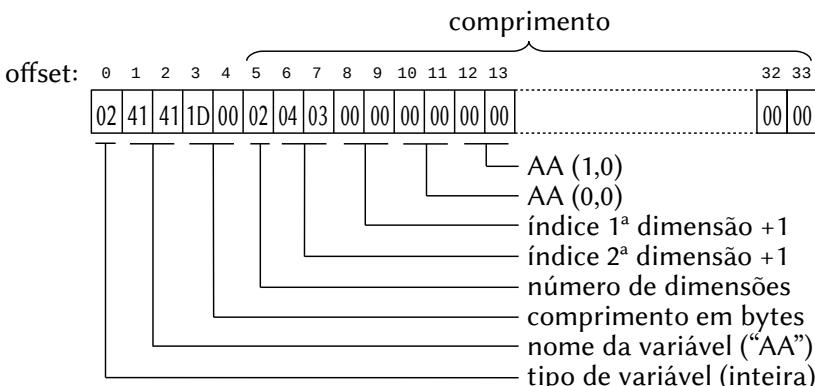
As variáveis alfanuméricicas (strings) têm uma forma de armazenamento ligeiramente diferente, cuja sintaxe está descrita abaixo:



A variável de sistema FRETOP (F69BH) armazena o endereço que receberá o último caractere da string que está sendo armazenada.

Se houver uma atribuição direta a uma variável alfanumérica (tipo A\$="XYZ"), o endereço que o apontador indicará estará na própria área de texto do programa BASIC e não na área reservada para variáveis string, evitando duplicação de dados e economizando memória. Isso também acontece quando forem lidos dados armazenados em instruções "DATA": o apontador indicará o texto logo após a instrução "DATA", não o transferindo para a área string. Entretanto, qualquer operação feita com a variável que a modifique fará com que os dados por ela representados sejam transferidos para a área reservada e o apontador conterá o endereço respectivo nesta.

As matrizes têm uma forma de armazenamento diferente, mas o formato de armazenamento é igual ao das variáveis simples. Primeiro, vem o byte ID seguido do nome da variável; depois dois bytes indicam o comprimento total da matriz (considerando 3 bytes do apontador para as variáveis alfanuméricas ou 2, 4 ou 8 bytes para variáveis numéricas). O comprimento indicado inclui todos os valores que vêm em seguida. Logo depois do comprimento, vem um byte que indica o número de dimensões da matriz, seguido de tantas sequências de dois bytes quanto sejam as dimensões da matriz. Esses dois bytes são apontadores para cada uma das dimensões da matriz, acrescidos de 1. Após, vem o armazenamento das variáveis propriamente ditas. Abaixo está ilustrado como fica armazenada a matriz DIM AA%(2,3).



### 3.3.5 – Chamando programas de Assembly no BASIC

Para usar programas Assembly juntamente com o BASIC, existem 3 comandos reservados para tal fim: USR, CMD e IPL. O uso mais comum é com a função USR; podem ser definidas até 10 rotinas com ela. Para usá-la, basta seguir os três passos seguintes:

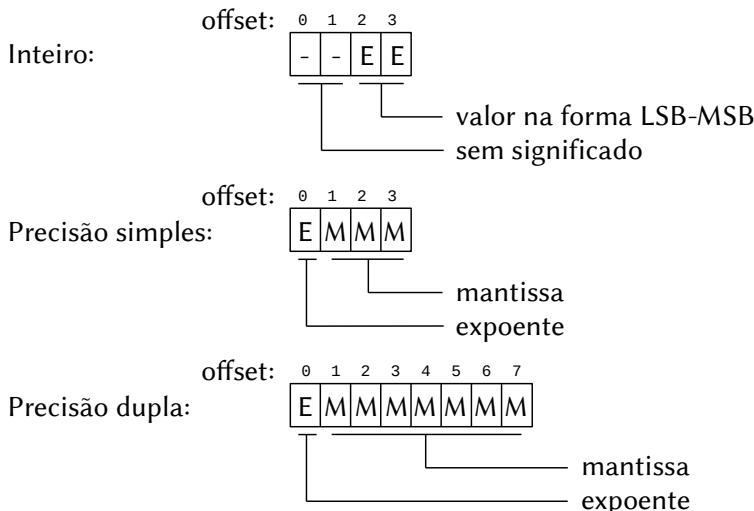
- 1 – Especificar o endereço de execução da rotina através do comando DEFUSR;
- 2 – Chamar a rotina através do comando USR;
- 3 – Para retornar o controle ao interpretador, usar uma instrução RET.

Um argumento qualquer pode ser passado ao programa Assembly pela função USR. Nesse caso, o registrador A conterá o tipo de variável passada e o registrador DE conterá o endereço de um apontador no caso de variáveis string ou HL conterá o endereço da própria variável, caso seja numérica, conforme as ilustrações abaixo.

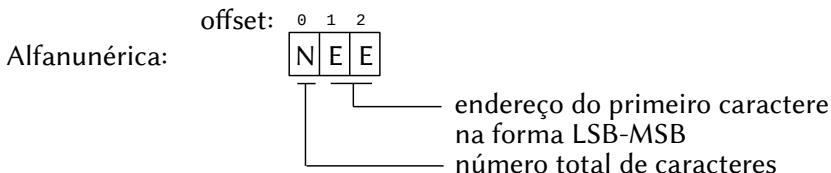
Valores de A:

02 – numérico inteiro	04 – precisão simples
03 – string	08 – precisão dupla

Endereços apontados por HL:



Endereço apontado por DE:



A função USR também permite passar variáveis alteradas pela rotina Assembly para o BASIC. Nesse caso, os valores de HL e DE devem conter o endereço inicial da variável ou apontador. Variáveis numéricas podem ser trocadas de tipo livremente, bastando alterar a variável de sistema VALTYP (F663H) e respeitar a forma de armazenamento da variável. Porém, variáveis string não podem ser trocadas de tipo e nem podem ter alterado o número de caracteres.

### 3.3.5.1 – Implementando novos comandos

Novos comandos também podem ser implementados, usando as palavras reservadas CMD ou IPL. Para isso, basta alterar o hook respectivo (FE0DH para CMD e FE03H para IPL) fazendo-o apontar para a rotina Assembly.

Se for inserida uma instrução POP AF logo no início do hook, caso ocorra algum erro na execução do comando, não haverá geração de erro quando do retorno ao BASIC. Nesse caso, a própria rotina Assembly poderá imprimir mensagem de erro interna.

No caso de algum argumento ser passado para a rotina assembly por esses comandos, o par HL apontará o primeiro caractere após o comando, conforme ilustrado abaixo:



No retorno, o par HL deverá apontar para o primeiro sinalizador após o comando implementado, que pode ser 00H (fim de linha) ou 3AH (dois pontos, separador de instruções).

Para implementações mais sofisticadas, argumentos podem ser passados para o comando CMD. Como o registrador HL aponta para o primeiro caractere após o comando quando a rotina ASM é chamada,

esta pode interpretar uma string complexa. Neste caso, há diversas rotinas do interpretador que podem ser muito úteis. Há uma descrição detalhada destas rotinas no item seguinte.

### 3.3.6 – Rotinas do interpretador

Existem algumas rotinas padrão do interpretador que estão disponíveis para os programas assembly. Elas estão listadas abaixo da mesma forma que as rotinas da BIOS. Como são rotinas do interpretador, caso ocorra algum erro, o controle será transferido ao manipulador de erro e depois devolvido ao interpretador. Para evitar que isso ocorra, pode ser usado o hook HERRO (FFB1H) para interceptar o erro. O código de erro fica no registrador E, podendo ser usado pela rotina assembly. Todos os registradores são alterados pelas rotinas.

**READYR** (409BH/Main)

Função: Retorna ao nível de comandos (partida a quente do BASIC).

Entrada: Nenhuma

Saída: Nenhuma

**CRUNCH** (42B2H/Main)

Função: Converte um texto BASIC da forma ASCII para a forma tokenizada.

Entrada: HL - endereço do texto em ASCII a ser convertido, finalizado por um byte 00H.

Saída: KBUF (F41FH) – texto BASIC convertido

**NEWSTT** (4601H/Main)

Função: Executa um texto BASIC. O texto deverá estar na forma tokenizada.

Entrada: HL – apontador para o início do texto a ser executado. O texto deverá estar na forma ilustrada abaixo:

3AH 94H 00H ...

:	NEW		...
---	-----	--	-----

↑

(HL)

Saída: Nenhuma

**CHRGTR** (4666H/Main) – de 0010H

**Função:** Extrai um caractere do texto BASIC, iniciando por (HL)+1.  
Espaços são ignorados.

**Entrada:** HL – endereço inicial do texto

**Saída:** HL – endereço do caractere extraído

A – código ASCII do caractere extraído

Flag Z – ligada se for fim de linha (00H ou 3AH “:”)

Flag CY – ligada se for um caractere de 0 a 9

**FRMEVL** (4C64H/Main)

**Função:** Avalia uma expressão e devolver o resultado.

**Entrada:** HL - endereço inicial da expressão no texto BASIC.

**Saída:** HL - endereço final da expressão +1.

VALTYP (F663H) – 2, 3, 4 ou 8 (tipo de variável<sup>9</sup>)

DAC (F7F6H) – resultado da expressão avaliada

**GETBYT** (521CH/Main)

**Função:** Avalia uma expressão e retornar um resultado de 1 byte.  
Quando o resultado extrapolar o valor de 1 byte será gerado erro de “Função Ilegal” e a execução retornará ao nível de comandos.

**Entrada:** HL – endereço inicial da expressão a ser avaliada

**Saída:** HL - endereço final da expressão +1.

A, E – resultado da avaliação (A e E contêm o mesmo valor)

**FRMQNT** (542FH/Main)

**Função:** Avalia uma expressão e retornar um resultado de 2 bytes (número inteiro). Quando o resultado extrapolar o valor de 2 bytes, será gerado um erro de “Overflow” e a execução retornará ao nível de comandos.

**Entrada:** HL – endereço inicial da expressão a ser avaliada

**Saída:** HL - endereço final da expressão +1.

DE - resultado da avaliação

<sup>9</sup> O significado dos valores está descrito detalhadamente na seção “ÁREA DE VARIÁVEIS DO INTERPRETADOR”.

### SYNCHR (558CH/Main) – de 0008H

**Função:** Testa se o caractere apontado por (HL) é o especificado.  
 Se não for, gera “Syntax error”; caso contrário chama CHRGTR (4666H/Main).

**Entrada:** HL - aponta para o caractere a ser testado  
 O caractere para comparação deve ser colocado após uma intrução “RST 0008H” na forma de parâmetro em linha, conforme exemplo abaixo:

```
LD   HL,CARACT
RST  008H
DEFB 'A'
|
```

CARACT: DEFB 'B'

**Saída:** HL é incrementado em um e A recebe (HL). Quando o caractere testado for numérico, a flag CY é setada. O fim de declaração (00H ou 3AH “:”) seta a flag Z.

### GETYPR (5597H/Main) – de 0028H

**Função:** Obtém o tipo de operando contido em DAC.

**Entrada:** Nenhuma

**Saída:** Flags CY, S, Z e P/V, conforme tabela abaixo:

Inteiro:	C=1	S=1*	Z=0	P/V=1
----------	-----	------	-----	-------

Precisão simples:	C=1	S=0	Z=0	P/V=0*
-------------------	-----	-----	-----	--------

Precisão dupla:	C=0*	S=0	Z=0	P/V=1
-----------------	------	-----	-----	-------

String:	C=1	S=0	Z=1*	P/V=1
---------	-----	-----	------	-------

**Obs:** Os tipos podem ser reconhecidos unicamente pelas flags marcadas com “\*”.

### PTRGET (5EA4H/Main)

**Função:** Obtém o endereço para o armazenamento de uma variável ou matriz. O endereço também é obtido quando a variável não foi atribuída. Quando o valor de SUBFLG (F5A5H) for diferente de 0, o endereço inicial de uma matriz será obtido; caso contrário, será obtido o endereço do elemento da matriz.

**Entrada:** HL - endereço inicial do nome da variável no texto BASIC  
 SUBFLG (F6A5H) – 0: variável simples, outro valor: matriz

**Saída:** HL - endereço após o nome da variável  
 DE – endereço de onde o conteúdo da variável está armazenado.

## FRESTR (67D0H/Main)

Função: Registra o resultado de uma string obtida por FRMEVL (4C64H) e obtém o respectivo descritor. Quando avaliando uma string, esta rotina é, geralmente, combinada com FRMEVL da forma descrita abaixo:

```
CALL FRMEVL
PUSH HL
CALL FRESTR
EX DE, HL
POP HL
LD A, (DE)
...
```

Entrada: VALTYP (F663H) – tipo de variável (deve ser 3)  
 DAC (F7F7H) – apontador para o descritor da string  
 Saída: HL – apontador para o descritor da string

### 3.3.7 – Chamando comandos do interpretador

É possível usar as rotinas do interpretador em programas Assembly. Entretanto, ao chamar um comando BASIC passa-se literalmente a trabalhar em BASIC, devendo-se levar em consideração duas coisas. Primeira: algum erro ou bug acidental que ocorra durante a execução da rotina fará como que o controle seja devolvido automaticamente ao nível de comandos BASIC. Para evitar que isso ocorra, pode ser usado o hook HERRO (FFB1H) para interceptar o erro. O código de erro fica no registrador E, podendo ser usado pela rotina assembly. Segunda: um comando em BASIC só deve ser chamado caso o algoritmo a ser usado seja muito complexo, como as instruções CIRCLE, LINE, DRAW, PLAY e outras de execução complexa. Sempre deve ser dada preferência às rotinas da BIOS quando estas puderem realizar o mesmo trabalho, pois são muito mais rápidas e fáceis de usar que as rotinas do BASIC.

Para chamar um comando BASIC, normalmente basta setar no par HL o endereço de uma falsa linha BASIC terminada por um byte 00H, preferencialmente na forma tokenizada. Porém alguns comandos exigem que mais registradores e até variáveis de sistema sejam carregadas, mas são comandos sem utilidade alguma para programas assembly. Para obter a forma tokenizada do comando, há uma técnica simples:

basta digitar a linha de programa desejada e depois usar um programa monitor (dump) para observar a linha tokenizada.

Após setar o par HL, deve ser usada a rotina CALBAS da BIOS para executar o comando. Também podem ser usadas as rotinas CALSLT (001CH) ou CALLF (0030H), setando em IY o slot da Main-ROM.

Para usar o comando, é necessário saber em qual endereço está a rotina que executa o mesmo. Isso pode ser feito consultando uma tabela de endereços que inicia em 392EH, onde os endereços por ela apontados seguem em ordem crescente de token do comando. As funções também têm sua tabela, com início em 39DEH. Os endereços apontados por essas tabelas, apesar de não serem padronizados, permaneceram os mesmos em todos os modelos MSX. Conclui-se que não há necessidade de consultar a tabela para garantir a compatibilidade, bastando setar o endereço da rotina diretamente. Entretanto, nada impede que a tabela seja consultada. Abaixo segue uma relação de todos os comandos e funções BASIC com seus respectivos tokens (incluindo o token setado precedido de FFH no caso de funções e os tokens especiais dos comandos ELSE e REM), endereço na tabela e pontos de entrada.

Comando	Token	Token de função	Endereço na tabela	Endereço da rotina
>	EEH	-	Afat	-
=	EFH	-	Afat	-
<	F0H	-	Afat	-
+	F1H	-	Afat	-
-	F2H	-	Afat	-
*	F3H	-	Afat	-
/	F4H	-	Afat	-
^	F5H	-	Afat	-
\$	FCH	-	Afat	-
ABS	06H	FF86H	39E8H	2E82H
AND	F6H	-	Afat	-
ASC	15H	FF95H	3A06H	680BH
ATN	0EH	FF8EH	39F8H	2A14H
ATTR\$	E9H	-	Afat	7C43H
AUTO	A9H	-	3973H	49B5H
BASE	C9H	-	39BEH	7B5AH
BEEP	C0H	-	39ACH	00C0H
BIN\$	1DH	FF9DH	3A16H	6FFFH
BLOAD	CFH	-	39CAH	6EC6H

BSAVE	D0H	-	39CCH	6E92H
CALL	CAH	-	39C0H	55A8H
CDBL	20H	FFAOH	3A1CH	303AH
CHR\$	16H	FF96H	3A08H	681BH
CINT	1EH	FF9EH	3A18H	2F8AH
CIRCLE	BCH	-	39A4H	5B11H
CLEAR	92H	-	3950H	64AFH
CLOAD	9BH	-	3962H	703FH
CLOSE	B4H	-	3994H	6C14H
CLS	9FH	-	396AH	00C3H
CMD	D7H	-	39DAH	7C34H
COLOR	BDH	-	39A6H	7980H
CONT	99H	-	395EH	6424H
COPY	D6H	-	39D8H	7C2FH
COS	0CH	FF8CH	39F4H	2993H
CSAVE	9AH	-	3960H	6FB7H
CSNG	1FH	FF9FH	3A1AH	2FB2H
CSRLIN	E8H	-	Afat	790AH
CVD	2AH	FFAAH	3A30H	7C70H
CVI	28H	FFA8H	3A2CH	7C66H
CVS	29H	FFA9H	3A2EH	7C6BH
DATA	84H	-	3934H	485BH
DEF	97H	-	395AH	501DH
DEFDBL	AEH	-	3988H	4721H
DEFINT	ACH	-	3984H	471BH
DEFSNG	ADH	-	3986H	471EH
DEFSTR	ABH	-	3982H	4718H
DELETE	A8H	-	397CH	53E2H
DIM	86H	-	3938H	5E9FH
DRAW	BEH	-	39A8H	5D6EH
DSKF	26H	FFA6H	3A28H	7C39H
DSKI\$	EAH	-	Afat	7C3EH
DSKO\$	D1H		39CEH	7C16H
ELSE	A1H	3AA1H	396EH	485DH
END	81H		396EH	63EAH
EOF	2BH	FFFABH	3A32H	6D25H
EQV	F9H	-	Afat	-
ERASE	A5H		3976H	6477H
ERL	E1H	-	Afat	4E0BH
ERR	E2H	-	Afat	4DFDH
ERROR	A6H	-	3978H	49AAH
EXP	0BH	FF8BH	39F2H	2B4AH

FIELD	B1H	-	398EH	7C52H
FILES	B7H	-	39AAH	6C2FH
FIX	21H	FFA1H	3A1EH	30BEH
FN	DEH	-	Afat	5040H
FOR	82H	-	3920H	4524H
FPOS	27H	FFA7H	3A2AH	6D39H
FRE	0FH	FF8FH	39FAH	69F2H
GET	B2H	-	3990H	775BH
GOSUB	8DH	-	3948H	47B2H
GOTO	89H	-	393EH	47E8H
GO TO	89H	-	393EH	47E8H
HEX\$	1BH	FF9BH	3A12H	65FAH
IF	8BH		3942H	49E5H
IMP	FAH	-	3A20H	7940H
INKEY\$	ECH	-	Afat	7347H
INP	10H	FF90H	39FCH	4001H
INPUT	85H	-	3936H	4B6CH
INSTR	E5H	-	39F6H	29FBH
INT	05H	FF85H	39E6H	30CFH
IPL	D5H	-	39D6H	7C2AH
KEY	CCH	-	3964H	786CH
KILL	D4H	-	39D4H	7C25H
LEFT\$	01H	FF81H	39DEH	6861H
LEN	12H	FF92H	3A00H	67FFH
LET	88H	-	393CH	4880H
LFILES	BBH	-	39A2H	6C2AH
LINE	AFH	-	398AH	4B0EH
LIST	93H	-	3952H	522EH
LLIST	9EH	-	3968H	5229H
LOAD	B5H	-	3996H	6B5DH
LOC	2CH	FFACH	3A34H	6D03H
LOCATE	D8H	-	39DCH	7766H
LOF	2DH	FFADH	3A36H	6D14H
LOG	0AH	FF8AH	39F0H	2A72H
LPOS	1CH	FF9CH	3A14H	4FC7H
LPRINT	9DH	-	394CH	4A1DH
LSET	B8H	-	399CH	7C48H
MAX	CDH	-	39C6H	7E4BH
MERGE	B6H	-	3998H	6B5EH
MIDS	03H	FF83H	39E2H	689AH
MKDS	30H	FFB0H	3A3CH	7C61H
MKI\$	2EH	FFAEH	3A38H	7C57H

MKS\$	2FH	FFAFH	3A3AH	7C5CH
MOD	FBH	-	Afat	-
MOTOR	CEH	-	39C8H	73B7H
NAME	D3H	-	39D2H	7C20H
NEW	94H	-	3954H	6286H
NEXT	83H	-	3932H	6527H
NOT	E0H	-	Afat	-
OCT\$	1AH	FF9AH	3A10H	7C70H
OFF	EBH	-	3A02H	3A02H
ON	95H	-	3956H	48E4H
OPEN	B0H	-	398CH	6AB7H
OR	F7H	-	Afat	-
OUT	9CH	-	3964H	4016H
PAD	25H	FFA5H	3A26H	7969H
PAINT	BFH	-	39AAH	59C5H
PDL	24H	FFA4H	3A24H	795AH
PEEK	17H	FF97H	3A0AH	541CH
PLAY	C1H	-	39AEH	73E5H
POINT	EDH	-	Afat	5803H
POKE	98H	-	395CH	5423H
POS	11H	FF91H	39FEH	4FCCCH
PRESET	C3H	-	39B2H	57E5H
PRINT	91H	-	394EH	4A24H
PSET	C2H	-	39B0H	57EAH
PUT	B3H	-	3992H	7758H
READ	87H	-	393AH	4B9FH
REM	8FH	3A8FH	394AH	485DH
RENUM	AAH	-	3980H	5468H
RESTORE	8CH	-	3944H	63C9H
RESUME	A7H	-	397AH	495DH
RETURN	8EH	-	3948H	4821H
RIGHT\$	02H	FF82H	39E0H	6891H
RND	08H	FF88H	39ECH	2BDFH
RSET	B9H	-	399EH	7C4DH
RUN	8AH	-	3940H	479EH
SAVE	BAH	-	39A0H	6BA3H
SCREEN	C5H	-	39B6H	79CCH
SET	D2H	-	39D0H	7C1BH
SGN	04H	FF84H	39E4H	2E97H
SIN	09H	FF89H	39EEH	29ACH
SOUND	C4H	-	39B4H	73CAH
SPACE\$	19H	FF99H	3A0EH	6848H

SPC (	DFH	-	Afat	-
SPRITE	C7H	-	39BAH	7A48H
SQR	07H	FF87H	39EAH	2AFFH
STEP	DCH	-	Afat	-
STICK	22H	FFA2H	3A20H	7940H
STOP	90H	-	394CH	63E3H
STR\$	13H	FF93H	3A02H	6604H
STRIG	23H	FFA3H	3A22H	794CH
STRING\$	E3H	-	Afat	6829H
SWAP	A4H	-	3974H	643EH
TAB (	DBH	-	Afat	-
TAN	0DH	FF8DH	39F6H	29FBH
THEN	DAH	-	Afat	-
TIME	CBH	-	39C2H	7911H
TO	D9H	-	Afat	-
TROFF	A3H	-	3972H	6439H
TRON	A2H	-	3970H	6438H
USING	E4H	-	Afat	-
USR	DDH	-	Afat	4FD5H
VAL	14H	FF94H	3A04H	68BBH
VARPTR	E7H	-	39FAH	4E41H
VDP	C8H	-	39BCH	7B37H
VPEEK	18H	FF98H	3A0CH	7BF5H
VPOKE	C6H	-	39B8H	7BE2H
WAIT	96H	-	3958H	401CH
WIDTH	A0H	-	396CH	51C9H
XOR	F8H	-	Afat	-

Nem todos os comandos estão nas tabelas da ROM e alguns sequer tem rotinas próprias para execução. Esses comandos estão marcados com a expressão “Afat”, pois são executados diretamente pela rotina padrão em 4DC7H (Avaliador de Fatores). Em particular, os tokens dos comandos ELSE e REM são precedidos pelo byte 3AH (“：“) e os tokens de todas as funções (tokens menores que 80H) têm seu bit 7 setado e são precedidos pelo byte FFH no texto BASIC.

No início dessa seção foi dito que a linha BASIC deveria estar preferencialmente na forma tokenizada. Entretanto, é possível usá-la na forma ASCII. O único cuidado nesse caso é substituir dez caracteres-chave pelos tokens respectivos, podendo o restante do texto estar na forma ASCII. Esses caracteres com seus respectivos tokens são:

```
' E6H      = EFH      + F1H      * F3H      ^ F5H
> EEH      < F0H      - F2H      / F4H      $ FCH
```

Assim, por exemplo, uma linha de texto BASIC tipo:

```
LINE (10,10)-(50,50),1
```

deve ser colocada na linha em código de máquina da seguinte forma:

```
DEFB ` (10,10) `,0F2H, ` (50,50) ,1` ,000H
```

Entretanto, se algum dos caracteres-chave vier entre aspas no texto BASIC, como nos comandos DRAW ou PLAY, deverá ser mantido em sua forma original. Por exemplo:

PLAY "A-BC+" em assembly ficará:

```
DEFB '"A-BC+"',000H
```

Um ótimo lugar para colocar o texto a ser executado é na variável de sistema KBUF (F41FH), por dois motivos: é usada pelo interpretador exatamente para isso e fica na página 3, podendo ser executada a partir do DOS sem problemas. Abaixo está ilustrado um exemplo prático com a instrução CIRCLE. Essa rotina funciona tanto sob o DOS quanto sob o BASIC, em qualquer endereço.

```
CIRCLE: EQU 05B11H
INIGRP: EQU 00072H
CHGET: EQU 0009FH
CALS LT: EQU 0001CH
SLTROM: EQU 0FCC1H
KBUF: EQU 0F41FH
LD HL,LINBAS
LD DE,KBUF
LD BC,12 LDIR
LD IX,INIGRP
LD IY,(SLTROM+1)
CALL CALSLT
LD HL,KBUF
LD IX,CIRCLE
LD IY,(SLTROM+1)
CALL CALSLT
LD IX,CHGET
LD IY,(SLTROM+1)
```

```

CALL CALSLT
RET
LINBAS: DEFB ` (128,96),70` ,000H

```

### 3.3.8 – Mensagens de erro

Na ROM, há uma tabela que contém uma lista das mensagens de erro do BASIC, que inicia em 3D75H. Cada uma delas é armazenada como texto, terminado com um byte 00H. Os códigos de erro associados são mostrados abaixo apenas para referência, pois não fazem parte da tabela.

01 NEXT without FOR	19 Device I/O error
02 Syntax error	20 Verify error
03 RETURN without GOSUB	21 No RESUME
04 Out of DATA	22 RESUME without error
05 Illegal function call	23 Unprintable error
06 Overflow	24 Missing operand
07 Out of memory	25 Line buffer overflow
08 Undefined line number	50 FIELD overflow
09 Subscript out of range	51 Internal error
10 Redimensioned array	52 Bad file number
11 Division by zero	53 File not found
12 Illegal direct	54 File already open
13 Type mismatch	55 Input past end
14 Out of string space	56 Bad file name
15 String too long	57 Direct statement in file
16 String formula too complex	58 Sequential I/O only
17 Can't CONTINUE	59 File not OPEN
18 Undefined user function	

## **Capítulo 4**

### **A MEMÓRIA RAM**

A CPU Z80 pode acessar diretamente o máximo de 64 Kbytes de memória. Essa quantidade de memória já era insuficiente para muitas aplicações mesmo em 1983 quando foi criado o padrão MSX. Tendo em vista esse fato, foram desenvolvidos alguns sistemas para ampliar a quantidade de memória que o Z80 pode acessar.

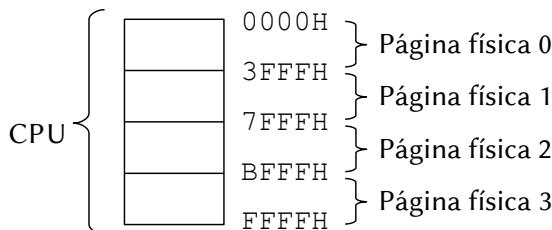
#### **4.1 – EXPANSÕES DE MEMÓRIA**

O primeiro sistema de expansão de memória que foi desenvolvido para o MSX foi o esquema de slots e páginas, que permitia ao Z80 acessar um máximo teórico de 1 Mbyte. O sistema de slots e páginas era excelente para atualizar o hardware, mas muito complexo para funcionar como expansão de memória. Pouquíssimos softwares chegaram a utilizá-lo.

Em 1985, com o lançamento do MSX2, foi criado um novo conceito de expansão de memória, a Memória Mapeada, de fácil manipulação, que permitia a conexão de até 4 Mbytes em cada slot, valor mais que suficiente para a época.

##### **4.1.1 – A Memória Mapeada**

A Memória Mapeada<sup>10</sup> usa as portas de I/O do Z80 como complemento ao barramento de endereços. Quatro portas são usadas, de FCH a FFH, uma para cada página física. Páginas físicas são as quatro páginas de 16 Kbytes que podem estar ativas ao mesmo tempo, cada uma em endereços diferentes, conforme a ilustração abaixo:



Para cada página física há uma porta de I/O correspondente, como ilustrado abaixo:

<sup>10</sup> A Memória Mapeada é comumente chamada apenas de Mapper, do termo em inglês “Memory Mapper”.

Página física 0 → porta FCH  
 Página física 1 → porta FDH  
 Página física 2 → porta FEH  
 Página física 3 → porta FFH

O valor que pode ser escrito em uma porta do Z80 varia de 0 a 255 e cada valor define uma *página lógica*. Dessa forma pode-se ter até 256 páginas lógicas. Na memória mapeada as portas de I/O são usadas para associar uma página lógica a uma página física; por isso cada página, tanto física como lógica, tem 16 Kbytes. Então, para saber o total que este método pode manipular, basta fazer 16 Kbytes vezes 256, o que dá 4 Mbytes em cada slot. Como cada slot é tratado de modo diferente e pode haver até 4 Mbytes de memória mapeada em cada um, fazemod 4 Mbytes vezes 16, o que resulta em um máximo teórico de 64 Mbytes.

No MSX2 é usado um slot com 64 Kbytes de RAM e a memória mapeada deve estar em outro slot. Já do MSX2+ em diante os 64 Kbytes de RAM principal correspondem aos primeiros 64 Kbytes da memória mapeada. A seleção padrão inicial de páginas é a seguinte:

Página física 0 = página lógica 3  
 Página física 1 = página lógica 2  
 Página física 2 = página lógica 1  
 Página física 3 = página lógica 0

A troca entre as páginas físicas e lógicas é muito simples. Basta usar uma instrução OUT do Z80 para posicionar a página lógica desejada na página física correspondente. Assim, para a seleção inicial dos 64 Kbytes, a seguinte sequência de instruções é usada:

```
OUT 0FCH,3 ;posiciona a pág. lóg. 3 na pág. fís. 0
OUT 0FDH,3 ;posiciona a pág. lóg. 2 na pág. fís. 1
OUT 0FEH,3 ;posiciona a pág. lóg. 1 na pág. fís. 2
OUT 0FFH,3 ;posiciona a pág. lóg. 0 na pág. fís. 3
```

Com essa disposição, a página lógica 0 (primeira página da Memória Mapeada) conterá as rotinas e variáveis de sistema e é a única que nunca poderá ser movida.

Como as páginas lógicas têm sempre o mesmo número, eventualmente uma página lógica pode estar em duas ou mais páginas físicas ao mesmo tempo. Por exemplo, as instruções

```
OUT 0FDH,5
OUT 0FEH,5
```

posicionam a página lógica 5 nas páginas físicas 1 e 2.

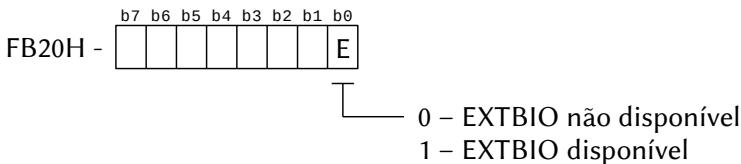
Uma observação importante é que a seleção de slots e páginas físicas têm precedência sobre a seleção de páginas lógicas. Por isso, ao selecionar uma página lógica, é necessário certificar-se que a página física correspondente esteja habilitada.

Normalmente apenas as páginas físicas 1 e 2 são usadas para a seleção de páginas lógicas, uma vez que a página física 0 contém a BIOS e a página física 3 contém a área de trabalho do sistema e não pode ser desligada, sob pena de paralisar todo o sistema.

#### 4.1.1.1 – Gerenciando a Memória Mapeada

Se o MSXDOS2 estiver instalado, são disponibilizadas várias funções para manipulação da Memória Mapeada, ou simplesmente Mapper. Estas funções são acessadas através de uma entrada adicional chamada de EXTBIO, localizada no endereço FFCAH.

Se o programa não for carregado diretamente pelo MSXDOS2, é necessário verificar se a rotina EXTBIO está disponível, o que pode ser feito testando a flag HOKVLD no endereço FB20H, da seguinte forma:



Se o programa for carregado a partir do MSXDOS2, o passo acima é desnecessário, pois o Kernel já disponibiliza a entrada. Para chamar a função, é necessário setar número do dispositivo no registrador D, a função no registrador E e os parâmetros requeridos nos outros registradores. O formato de chamada da EXTBIO pode ser o seguinte:

```

EXTBIO: EQU 0FFCAH
        XOR A           ; A deve ser sempre 0
        LD  D,4          ; ID do dispositivo (mapper)
        LD  E,1          ; função GETMAP
        CALL EXTBIO      ; executa EXTBIO

```

O formato da rotina é o seguinte:

#### EXTBIO (FFCAH/Sys)

Função: Acessa funções estendidas da BIOS

Entrada: A – sempre 00H

D – ID do dispositivo

E – número da função

Saída: A – ID do slot da mapper primária

DE - reservado

BC, HL, BC', DE', HL', IX, IY – depende do dispositivo e função chamadas

O dispositivo de manipulação de mapper é 4 e as funções são as seguintes:

#### GETMAP (Disp.: 04H, função: 01H)

Função: Retorna o endereço da tabela de variáveis da mapper

Entrada: Através de EXTBIO (A=00H, D=04H, E=01H)

Saída: A – ID do slot da mapper primária.

DE – Reservado.

HL – Endereço inicial da tabela de variáveis, cuja estrutura é a seguinte:

- +00H ID do slot da mapper primária
- +01H Número total de segmentos de 16K
- +02H Número de segmentos de 16K livres
- +03H Número de segmentos de 16k  
alocados pelo sistema (mínimo de  
6 para a mapper primária)
- +04H Número de segmentos de 16K  
alocados para o usuário
- +05H~+07H Reservados (Sempre 00H)
- +08H... Entradas para outras mappers em  
outros slots. Se não houver  
nenhuma, conterá 00H.

## GETSUP (Disp.: 04H, função: 02H)

Função: Retorna diversos parâmetros relativos à mapper

Entrada: Através de EXTBIO (A=00H, D=04H, E=02H)

Saída: A – Número total de segmentos (páginas lógicas) da mapper primária

B – ID do slot da mapper primária

C – Número de segmentos (páginas lógicas) livres na mapper primária

DE – Reservado

HL – Endereço inicial de uma tabela de chamada de subrotinas de suporte à mapper. O formato desta tabela é o seguinte:

+00H ALL_SEG	Aloca um segmento de 16K
+03H FRE_SEG	Libera um segmento de 16K
+06H RD_SEG	Lê um byte do endereço A:HL para A
+09H WR_SEG	Escreve o conteúdo de E no endereço A:HL
+0CH CAL-SEG	Chamada inter-segmento pelo endereço IYh:IX
+0FH CALLS	Chamada inter-segmento. Parâmetros em linha após a instrução CALL
+12H PUT_PH	Coloca um segmento na página física (HL)
+15H GET_PH	Retorna o segmento atual para a página física (HL)
+18H PUT_P0	Coloca um segmento na página física 0
+1BH GET_P0	Retorna o segmento atual da página 0.
+1EH PUT_P1	Coloca um segmento na página física 1
+21H GET_P1	Retorna o segmento atual da página 1.
+24H PUT_P2	Coloca um segmento na página física 2
+27H GET_P2	Retorna o segmento atual da página 2.

+2AH PUT_P3	Não suportada visto que a página 3 não pode ser trocada. Se chamada, apenas retorna.
+2DH GET_P3	Retorna o segmento atual da página 3.

Uma vez obtido o endereço inicial da tabela, podem ser chamadas as subrotinas da BIOS estendida. Elas estão descritas a seguir.

#### ALL\_SEG (00H/Ext)

Função: Alocar um segmento de 16K da mapper.

- Entrada: A – 00H – aloca um segmento de usuário  
           01H – aloca um segmento de sistema<sup>11</sup>
- B – 00H – aloca somente na mapper primária
- |                |                |                |                |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| b <sub>7</sub> | b <sub>6</sub> | b <sub>5</sub> | b <sub>4</sub> | b <sub>3</sub> | b <sub>2</sub> | b <sub>1</sub> | b <sub>0</sub> |
| E              | A              | A              | A              | S              | S              | P              | P              |
- 
- Slot primário
- Slot secundário
- 000 – aloca apenas no slot especificado
- 001 – aloca em qualquer slot
- 010 – tenta alocar no slot especificado; se falhar tenta outro slot
- 011 – tenta alocar em outros slots que não o especificado; se falhar tenta no slot especif.
- 0 – slot primário NÃO expandido
- 1 – slot primário expandido

---

11 Um segmento de sistema só será liberado com o uso da rotina FRE\_SEG. No caso de um segmento de usuário, sempre que o programa que o utiliza for fechado, os segmentos são liberados, o que não acontece com os segmentos de sistema.

Saída CY = 1 – Não há segmentos livres  
 CY = 0 – Segmento alocado  
     A = número do segmento  
     B = ID do slot do segmento

#### FRE\_SEG (03H/Ext)

Função: Liberar um segmento de 16K da mapper.  
 Entrada: A – número do segmento a ser liberado  
     B – Se for 00H, libera somente na mapper primária; se  
         for diferente de 00H libera em qualquer outra  
         mapper que não a primária.  
 Saída: CY = 0 – segmento liberado  
     CY = 1 – erro na liberação do segmento

#### RD\_SEG (06H/Ext)

Função: Ler um byte da mapper  
 Entrada: A – número do segmento de onde o byte será lido  
     HL – endereço a ser lido (0000H to 3FFFH)  
 Saída: A - byte lido  
     Todos os outros registradores são preservados

#### WR\_SEG (09H/Ext)

Função: Escrever um byte na mapper  
 Entrada: A – número do segmento onde o byte será escrito  
     HL – endereço a ser escrito (0000H to 3FFFH)  
     E - valor a escrever  
 Saída: A - corrompido durante a escrita  
     Todos os outros registradores são preservados

#### CAL\_SEG (0CH/Ext)

Função: Chama uma rotina em qualquer área da mapper.  
 Entrada: IYh – número do segmento a ser chamado  
     IX – endereço a ser chamado (0000H to FFFFH)  
     AF, BC, DE e HL podem conter parâmetros para a rotina.  
     Não usar AF', BC', DE' e HL' pois são corrompidos durante  
         a chamada  
 Saída: AF, BC, DE, HL, IX e IY podem conter valores de retorno  
         válidos. AF', BC', DE' e HL' retornam corrompidos.

### CALLS (0FH/Ext)

Função: Chama uma rotina em qualquer área da mapper através de parâmetros em linha.

Entrada: AF, BC, DE e HL podem conter parâmetros para a rotina. Não usar AF', BC', DE' e HL' pois são corrompidos durante a chamada. A sequência de chamada deve estar no seguinte formato:

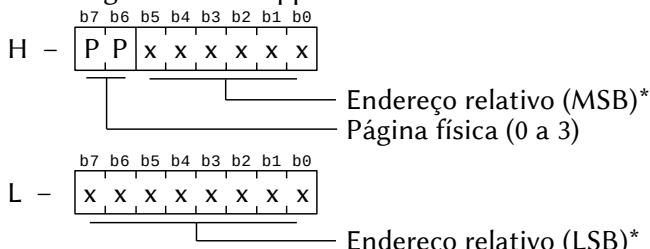
```
CALL CALLS
DEFB SEGMENTO
DEFW ENDEREÇO
```

Saída: AF, BC, DE, HL, IX e IY podem conter valores de retorno válidos. AF', BC', DE' e HL' retornam corrompidos.

### PUT\_PH (12H/Ext)

Função: Habilita um segmento da mapper em uma página física.

Entrada: A – Segmento da mapper:

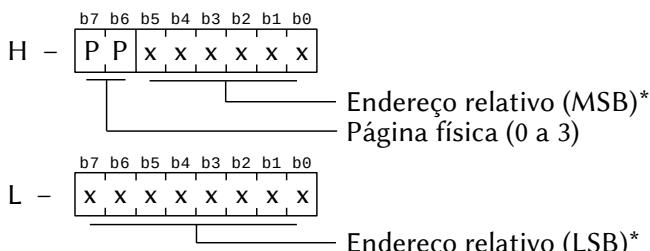


\* O endereço relativo é opcional.

Saída: Nenhuma. Todos os registradores são preservados.

### GET\_PH (15H/Ext)

Função: Retorna o segmento atual ativo em uma página física.



\* O endereço relativo é opcional.

Saída: A – número do segmento.  
Todos os outros registradores são preservados.

#### PUT\_P0 (18H/Ext)

Função: Habilita um segmento da mapper na página física 0.  
Entrada: A – Número do segmento a ser habilitado.  
Saída: Nenhuma. Todos os registradores são preservados.

#### GET\_P0 (1BH/Ext)

Função: Retorna o segmento ativo na página física 0.  
Entrada: Nenhuma.  
Saída: A – Número do segmento ativo.  
Todos os outros registradores são preservados.

#### PUT\_P1 (1EH/Ext)

Função: Habilita um segmento da mapper na página física 1.  
Entrada: A – Número do segmento a ser habilitado.  
Saída: Nenhuma. Todos os registradores são preservados.

#### GET\_P1 (21H/Ext)

Função: Retorna o segmento ativo na página física 1.  
Entrada: Nenhuma.  
Saída: A – Número do segmento ativo.  
Todos os outros registradores são preservados.

#### PUT\_P2 (24H/Ext)

Função: Habilita um segmento da mapper na página física 2.  
Entrada: A – número do segmento a ser habilitado.  
Saída: Nenhuma. Todos os registradores são preservados.

#### GET\_P2 (27H/Ext)

Função: Retorna o segmento ativo na página física 2.  
Entrada: Nenhuma.  
Saída: A – Número do segmento ativo.  
Todos os outros registradores são preservados.

#### PUT\_P3 (2AH/Ext)

Não suportada uma vez que a página física 3 não pode ser trocada.  
Uma chamada a esta função tem efeito nulo.

### GET\_P3 (2DH/Ext)

Função: Retorna o segmento ativo na página física 0.

Entrada: Nenhuma.

Saída: A – Número do segmento ativo.

Todos os outros registradores são preservados.

## 4.1.2 – A Megaram

Apesar de não ser reconhecida oficialmente como expansão de memória para o MSX, a Megaram é bastante popular no Brasil. Ela foi idealizada para que se pudesse rodar jogos megaram sem necessidade de convertê-los para a Memória Mapeada.

A Megaram também envolve conceito de páginas lógicas e físicas, mas sua operação é mais complicada que a da Memória Mapeada. Cada página lógica da Megaram tem 8 Kbytes e como podem ser definidas até 256 páginas lógicas, o máximo possível de memória que pode ser conectado em cada slot é 2 Mbytes.

O gerenciamento das páginas da Megaram é feito através da porta 08EH do Z80. Para habilitar a Megaram, primeiro deve ser executada a seguinte instrução:

```
OUT (08EH),A
```

O valor de A não tem importância. Essa instrução apenas indica à Megaram que ela vai ser usada. Como cada página lógica da Megaram tem apenas 8 Kbytes, são necessárias duas páginas lógicas para cada página física. Cada página lógica pode começar em um dos seguintes endereços:

```
4000H – 6000H – 8000H – A000H
```

Depois de executada a instrução “OUT (08EH),A”, deve-se carregar em A o número desejado da página lógica da Megaram e executar a instrução “LD (xxxxH),A”, onde “xxxxH” é o endereço inicial da página lógica na página física. Para colocar as páginas lógicas 0 e 1 da Megaram na página física 1 da memória, deve-se executar as seguintes instruções:

```

OUT  (08EH),A    ;habilita a megaram
LD   A,0         ;seleciona página lógica 0
LD   (04000H),A ;posiciona pág. lóg. 0 em 4000H
LD   A,1         ;seleciona página lógica 1
LD   (06000H),A ;posiciona pág. lóg. 1 em 6000H

```

Executando essas instruções, as páginas lógicas 0 e 1 da Megaram estarão ocupando a página física 1 do micro, e estarão prontas para serem lidas, mas não para serem escritas. Para poder escrever dados na Megaram, deve-se executar a instrução “IN A,(08EH)”. Logo abaixo está listada a sequência de instruções que colocam as páginas 0 e 1 da Megaram na página física 1 e as habilita para leitura e escrita.

```

OUT  (08EH),A    ;habilita a megaram
LD   A,0         ;seleciona página lógica 0
LD   (04000H),A ;posiciona pág. lóg. 0 em 4000H
LD   A,1         ;seleciona página lógica 1
LD   (06000H),A ;posiciona pág. lóg. 1 em 6000H
IN   A,(08EH)   ;habilita leitura e escrita

```

Ao ser executada, essa rotina posiciona as páginas lógicas 0 e 1 da Megaram na página física 1 e as habilita para serem lidas e escritas. Como na Memória Mapeada, a seleção de páginas físicas tem precedência sobre as páginas lógicas; por isso, para habilitar as páginas lógicas, é necessário que a página física correspondente esteja habilitada.

Para a Megaram não há nenhuma rotina padrão para gerenciamento da mesma. Todo o acesso é direto e o gerenciamento fica a cargo do programador.

#### **4.1.3 – Megaram x Memória Mapeada**

Tanto a Megaram quanto a Memória Mapeada devem ser reconhecidas pelo software que a utiliza. Não existe nenhuma rotina da BIOS para manipulação direta dessas expansões. No BDOS existe um meio para a procura de páginas lógicas livres para a memória mapeada.

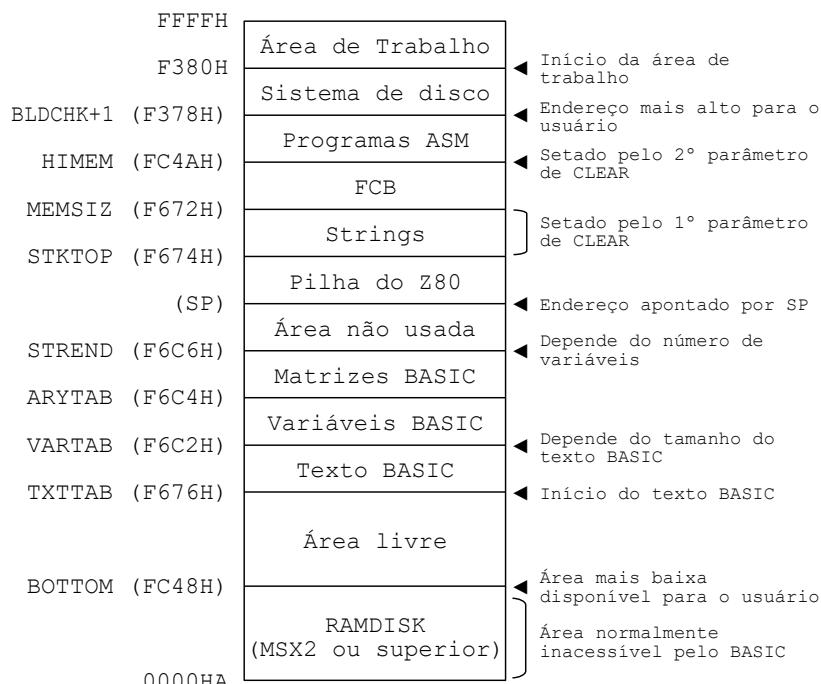
Uma dúvida que pode surgir aos programadores é sobre qual expansão de memória usar: Megaram ou Memória Mapeada. Como já descrito, a Memória Mapeada é a expansão padrão do MSX; entretanto a Megaram é muito popular no Brasil.

Uma solução razoável a essa questão é que os programas desenvolvidos reconheçam as duas expansões. Primeiro, o programa deve procurar a Memória Mapeada, já que é a expansão padrão. Caso esta não seja encontrada, faz-se a procura pela Megaram. Deve ser levado em conta que a Megaram é quase inexistente em outros países.

## 4.2 – MAPEAMENTO DA RAM

Independente de slots, páginas e expansões de memória, existe um mapeamento específico para a RAM, residente no topo da página física 3. Embora os endereços inferiores também sejam mapeados, não há problemas de troca entre as páginas físicas 0, 1 e 2, desde que tomados os devidos cuidados, como, por exemplo, não desligar a página onde o programa está sendo executado. A página física 3 jamais deve ser desligada, pois contém a área de trabalho do sistema.

Ao entrar no BASIC, logo após um reset, a RAM é mapeada como ilustrado na página seguinte.



O valor hexadecimal ao lado das variáveis de sistema acima são os endereços das mesmas; são as variáveis, ou os valores contidos nos respectivos endereços, que apontam para o endereço real no mapeamento mostrado acima.

Esse é o mapeamento padrão do MSX2/2+/TR com unidade de disco. Sem unidade de disco, basta desconsiderar a área respectiva (a área de disco será descrita com detalhes no capítulo “O SISTEMA DE DISCO”). Para o MSX1, a RAMDISK deve ser desconsiderada. Todas as áreas relativas ao BASIC (de TXTTAB até MEMSIZ) e a área para strings estão descritas no capítulo 1, seção “INTERPRETADOR BASIC”. A área para programas assembly é setada pelo comando CLEAR e fica reservada para rotinas do usuário; o interpretador não interferirá nela a não ser que seja instruído para isso (função USR ou expansão de comandos).

#### **4.2.1 – O FCB (File Control Block)**

O FCB (File Control Block, ou Bloco de Controle de Arquivos) é um buffer de 265 bytes usado para comunicação com periféricos.

Os periféricos padrão têm os seguintes nomes:

- A: até H: – Drives (A: até F: para MSXDOS1)
- AUX: – Auxiliar (Ex. RS232C, só DOS)
- CAS: – Interface de cassette
- CON: – Teclado ou tela de texto (Só DOS)
- CRT: – Tela de texto
- GRP: – Tela gráfica
- LPT: – Impressora (Só BASIC)
- LST: – Impressora (Só DOS)
- MEM: – Ramdisk dos 32K inferiores da RAM  
(Só BASIC, MSX2 ou superior)
- NUL: – Nulo (sem efeito, só DOS)
- PRN: – Impressora (Só DOS)

Toda vez que o BASIC ou o DOS referem a um desses dispositivos, um FCB é aberto. No BASIC podem ser abertos até 15 FCB's simultaneamente, especificados pelo comando MAXFILES. No DOS, o número máximo depende da memória disponível, e os FCB's podem ser abertos pelo comando BUFFERS. No reset é alocado apenas um FCB para o BASIC e 5 para o DOS. O formato do FCB está ilustrado abaixo.

Offset	Label	Descrição
+ 0	FL.MOD	Modo do arquivo aberto
+ 1	FL.FCA	Apontador para o FCB do BDOS (low)
+ 2	FL.LCA	Apontador para o FCB do BDOS (high)
+ 3	FL.LSA	Caractere de backup
+ 4	FL.DSK	Número do dispositivo
+ 5	FL.SLB	Uso interno do interpretador
+ 6	FL.BPS	Localização de FL.BUF
+ 7	FL.FLG	Flag de informações
+ 8	FL.OPS	Informação da cabeça virtual
+ 9...	FL.BUF	Início do buffer de 256 bytes

#### 4.2.2 – A área de trabalho

A área de trabalho do sistema vai do endereço F380H até FFFFH. O uso dessa área pelo programador deve ser bem controlado, sob pena de alterações indesejáveis nas funções básicas do micro ou até mesmo uma paralisação total do sistema. Essa área é mapeada como mostrado na ilustração abaixo.

FFFFH	Seleção de slot
FFFDH ~ FFFEH	Reservado
FFFAH ~ FFFCH	Registradores V9958
FFF8H ~ FFF9H	Reservado
FFF7H	Slot da MainROM
FFE7H ~ FFE6H	Registradores V9938
FFCFH ~ FFE6H	Programas expansão BIOS
FFCAH ~ FFCEH	Hooks expansão BIOS
FD9AH ~ FFC9H	Hooks
F39AH ~ FD99H	Variáveis do sistema
F380H ~ F399H	Rotinas inter-slot

#### 4.2.2.1 – Subrotinas inter-slot

A área compreendida entre F380H e F399H contém algumas rotinas para acesso em outros slots e páginas. Estas rotinas podem acessar qualquer área desde que a página física 3 esteja ativa. Elas permanecem as mesmas desde do MSX1 até o MSX turbo R e podem ser consideradas padrão por causa disso. São as seguintes:

##### RDPRIM (F380H)

Função: Lê um byte de qualquer endereço de qualquer slot

Entrada: A - slot primário a ser lido

D - slot atual para retorno

Saída: E - byte lido

##### WRPRIM (F385H)

Função: Escreve um byte em qualquer endereço de qualquer slot

Entrada: A - slot primário a ser lido

D - slot atual para retorno

E - byte a ser escrito

Saída: Nenhuma

##### CLPRIM (F38CH)

Função: Chama um endereço em qualquer slot

Entrada: A - slot primário que contém a rotina

IX - endereço a ser chamados

PUSH AF – slot atual para retorno (em A)

Saída: Depende da rotina chamada

Observe que alguns valores de entrada são pouco convencionais. Isso acontece porque estas rotinas foram especificamente escritas para serem usadas pelas rotinas padrão RDSL (000CH), WRSLT (0014H) e CALSLT (001CH) da BIOS. Os códigos em ASM dessas rotinas é o seguinte:

```
F380H RDPRIM: OUT  (0A8H),A ; seleciona slot prim.
F382H          LD   E, (HL)  ; lê byte
F383H          JR   WRPRM1 ; restaura slot prim.
F385H WRPRIM: OUT  (0A8H),A ; seleciona slot prim.
F387H          LD   (HL),E  ; escreve byte
F388H WRPRM1: LD   A,D    ; A = slot anterior
F389H          OUT  (0A8H),A ; restaura slot prim.
F38CH CLPRIM: OUT  (0A8H),A ; seleciona slot prim.
```

F38EH	EX AF,AF'	;troca AF com AF'
F38FH	CALL CLPRM1	;chama rotina
F392H	EX AF,AF'	;troca AF com AF'
F393H	POP AF	;A = slot anterior
F394H	OUT (0A8H),A	;restaura slot prim.
F396H	EX AF,AF'	;troca AF com AF'
F397H	RET	;retorna
F398H CLPRM1:	JP (IX)	;chama endereço (IX)

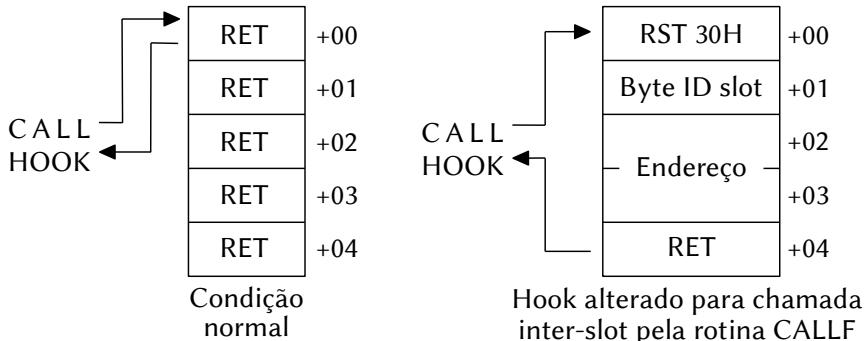
#### 4.2.2.2 – Os Hooks (ganchos)

A área de trabalho compreendida entre FD9AH e FFC9H é a área que contém os Hooks ou Ganchos. Cada hook é composto por cinco bytes que normalmente são preenchidos com o valor C9H (instrução RET).

Os hooks são chamados de posições estratégicas da BIOS de modo que as operações da BIOS/Interpretador possam ser modificadas ou ampliadas. Cada hook tem espaço suficiente para uma chamada distante para qualquer slot. Como são preenchidos inicialmente com instruções RET, isso causa apenas um retorno para a BIOS. Entretanto, ele pode ser modificado para chamar uma rotina em qualquer slot.

Não é necessário que o hook seja preenchido de modo a acessar rotinas em outros slots. Se a operação a ser realizada couber em cinco bytes, o hook, por si só, pode constituir o código a ser executado.

O modo como o hook pode ser modificado para acessar rotinas em outros slots está ilustrado abaixo.



Uma sugestão de rotina ASM para a alteração dos hooks está ilustrada abaixo.

```
HOOK: EQU XXXXH
      LD A, 0F7H           ; F7H = instrução RST 030H
      LD (HOOK), A
      LD A, (ID_SLOT)     ; A <- ID do slot da rotina
      LD (HOOK+1), A
      LD HL, (ENDEREÇO)   ; HL <- endereço da rotina
      LD (HOOK+2), HL
      LD A, 0C9H           ; C9H = instrução RET
      LD (HOOK+4), A
```

## ***Capítulo 5***

### **O VÍDEO E O VDP**

As máquinas MSX, com sua constante evolução, necessitaram de cada vez mais capacidade gráfica. Assim, no MSX1 é usado o VDP (Video Display Processor) TMS9918A ou TMS9928A, que apresentava apenas 16 cores e tinha poucos recursos, representando um fator muito limitante ao processamento gráfico dessas máquinas. Apenas dois anos depois de lançado, surgiu o MSX2 em 1985, com o novo VDP V9938, totalmente compatível com o TMS9918A. Mais tarde, em 1988, surgiu o MSX2+, com o V9958. Posteriormente, foi lançado o V9990, mas não era totalmente compatível com o V9958 e não chegou a ser usado em nenhum modelo MSX, mas foi utilizado em alguns car-tuchos de vídeo. O V9958 também foi usado nos modelos MSX turbo R, lançados em 1990 e 1991.

#### **5.1 – O MSX-VÍDEO**

O MSX1 usa o TMS9918A para geração do sinal de vídeo, cujas características principais são as seguintes:

- Resolução máxima de 256 x 192 pixels;
- Máximo de 16 cores apresentadas simultaneamente;
- Modo texto de 40 colunas por 24 linhas;
- Apresenta até 32 sprites monocromáticos na tela, com o máximo de 4 em uma linha horizontal;
- 16 Kbytes de VRAM

O V9938, usado no MSX2, tem as seguintes características principais:

- Paleta de 512 cores;
- Resolução máxima de 512 x 424 pontos com 16 cores;
- Máximo de 256 cores apresentadas simultaneamente;
- Modos gráficos bit-mapped de fácil manipulação;
- Modo texto de 80 caracteres por linha com recurso de “blink”;
- Linha, procura e movimentação de áreas implementadas em hardware;
- Apresenta até 32 sprites na tela, com o máximo de 8 sprites na mesma linha horizontal;
- Cada linha de cada sprite pode ter uma cor diferente;

- Endereços de memória podem ser representados por coordenadas gráficas;
- Funções de operação lógica;
- Scroll vertical fino por hardware;
- Capacidade interna de digitalização;
- Capacidade interna de “superimpose”;
- Até 128 Kbytes de VRAM.

O V9958, usado no MSX2+ e no MSX turbo R, acrescentou novas características ao vídeo, dentre as quais:

- Máximo de 19 268 cores apresentadas simultaneamente;
- Capacidade de sincronização externa;
- Possibilidade de múltiplas configurações MSX-VIDEO;
- Paletas de cores externas podem ser adicionadas usando a saída color-bus;
- Scroll fino vertical e horizontal por hardware;
- DAC de 5 bits por cor primária.

O V9990 não possui as Screens 0 a 4, e as Screens 5 a 12 foram modificadas, de forma que não são totalmente compatíveis com o V9958. Entretanto, possui muitos modos a mais, além de ser muito mais rápido (tem de 9 a 17 vezes a velocidade do V9958, dependendo do modo) e acessar até 512K de VRAM. Suas características principais são:

- Resolução máxima de 768 x 480 pontos (modo overscan);
- Máximo de 32 768 cores apresentadas simultaneamente;
- Paleta de até 64 cores selecionadas de 32 768;
- Scroll suave em todas as direções;
- Apresenta até 2 planos simultaneamente;
- Capacidade interna de digitalização;
- Capacidade interna de “superimpose”;
- Comandos de hardware disponíveis;
- Apresenta até 125 sprites na tela com até 16 cada linha horizontal;
- Cada ponto de cada sprite pode ter uma cor diferente;
- Resolução VGA adicional de 640 x 400 e 640 x 480 com até 16 cores selecionadas de 32 768.

### 5.1.1 – Descrição dos registradores (V9918/38/58)

O TMS9918A tem 9, o V9938 tem 49 e o V9958 tem 52 registradores internos para controlar as operações de vídeo. Esses registradores são divididos em três grupos. O grupo de controle e o grupo de estado podem ser acessados diretamente pelo BASIC e pela BIOS. O terceiro grupo, inexistente no TMS9918A, é o de paletas, e não pode ser acessado diretamente.

O grupo de registradores de controle é numerado de R#0 a R#7 para o TMS9918A, de R#0 a R#23 e de R#32 a R#46 para o V9938 e existem mais três, R#25 a R#27 para o V9958. São registradores de 8 bits apenas de escrita (para obter seus valores, existe uma cópia dos mesmos na área de trabalho do sistema). O subgrupo que vai de R#0 a R#2710 são registradores que controlam todos os modos de tela. O outro subgrupo, de R#32 a R#46, executa comandos de hardware do VDP. Esses comandos serão descritos com detalhes mais adiante. As tabelas abaixo resumidamente as funções de cada registrador dos grupos de controle e de estado.

R#0	VDP(0) Registrador de modo #0
R#1	VDP(1) Registrador de modo #1
R#2	VDP(2) Endereço da tabela de nomes dos padrões
R#3	VDP(3) Endereço da tabela de cores (low)
R#4	VDP(4) Endereço da tabela geradora de padrões
R#5	VDP(5) Endereço da tabela de atributos dos sprites (low)
R#6	VDP(6) Endereço da tabela de padrões dos sprites
R#7	VDP(7) Cor da borda e dos caracteres no modo texto

Os registradores seguintes só estão disponíveis para V9938 ou superior:

R#8	VDP(9) Registrador de modo #2
R#9	VDP(10) Registrador de modo #3
R#10	VDP(11) Endereço da tabela de cores (high)
R#11	VDP(12) End. da tabela de atributos dos sprites (high)
R#12	VDP(13) Cor dos caracteres para a função “blink”
R#13	VDP(14) Período de “blinking”
R#14	VDP(15) Endereço de acesso à VRAM (high)
R#15	VDP(16) Especificação indireta para S#n (preset: 0)
R#16	VDP(17) Especificação indireta para P#n (preset: 0)

R#17	VDP(18)	Especificação indireta para R#n (preset: 0)
R#18	VDP(19)	Ajuste de tela (SET ADJUST)
R#19	VDP(20)	Examina linha ao ocorrer interrupção
R#20	VDP(21)	Burst de cor para a fase 0 (preset 00 000 000B)
R#21	VDP(22)	Burst de cor para a fase 1/3 (preset 00 111 011B)
R#22	VDP(23)	Burst de cor para a fase 2/3 (preset 00 000 101B)
R#23	VDP(24)	Scroll vertical
R#24	O registrador R#24 não existe	
R#25	VDP(26)	Registrador de modo #4 (só V9958)
R#26	VDP(27)	Scroll horizontal (só V9958)
R#27	VDP(28)	Scroll horizontal fino (só V9958)

Os registradores R#28 a R#31 não existem.

R#32	VDP(33)	SX: coordenada horizontal inicial (low)
R#33	VDP(34)	SX: coordenada horizontal inicial (high)
R#34	VDP(35)	SY: coordenada vertical inicial (low)
R#35	VDP(36)	SY: coordenada vertical inicial (high)
R#36	VDP(37)	DX: coordenada horizontal de destino (low)
R#37	VDP(38)	DX: coordenada horizontal de destino (high)
R#38	VDP(39)	DY: coordenada vertical de destino (low)
R#39	VDP(40)	DY: coordenada vertical de destino (high)
R#40	VDP(41)	NX: número de pontos a transferir na direção horizontal (low)
R#41	VDP(42)	NX: número de pontos a transferir na direção horizontal (high)
R#42	VDP(43)	NY: número de pontos a transferir na direção vertical (low)
R#43	VDP(44)	NY: número de pontos a transferir na direção vertical (high)
R#44	VDP(45)	CLR: transferência de dados para a CPU
R#45	VDP(46)	ARGT: registrador de argumento
R#46	VDP(47)	CMR: envia comando de hardware ao VDP

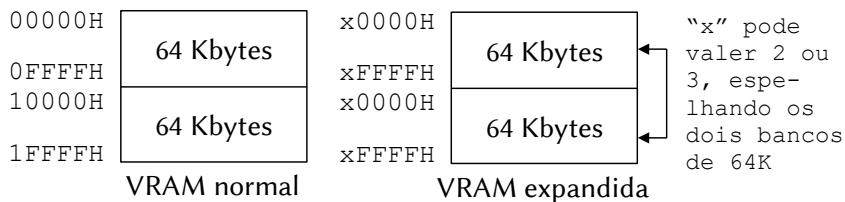
O grupo seguinte é o grupo de registradores de estado. São registradores de 8 bits somente de leitura designados por S#0 a S#9. O único presente no TMS9918 é o S#0. A listagem abaixo descreve resumidamente suas funções:

S#0	VDP(8)	Informação de interrupção
S#1	VDP(-1)	Informação de interrupção
S#2	VDP(-2)	Registrador de informação e controle
S#3	VDP(-3)	Coordenada horizontal detectada (low)
S#4	VDP(-4)	Coordenada horizontal detectada (high)
S#5	VDP(-5)	Coordenada vertical detectada (low)
S#6	VDP(-6)	Coordenada vertical detectada (high)
S#7	VDP(-7)	Dado obtido por algum comando do VDP
S#8	VDP(-8)	Coordenada horizontal obtida por algum comando de procura (low)
S#9	VDP(-9)	Coordenada vertical obtida por algum comando de procura (high)

## 5.1.2 – A VRAM (V9918/38/58)

O TMS9918A pode ser conectado a apenas 16 Kbytes de memória. Já o V9938 pode ser conectado a 64 ou 128 Kbytes e o V9958 deve obrigatoriamente ser conectado a 128 Kbytes, sem o que os modos de tela acrescentados não poderão ser utilizados. Essa memória é controlada pelo VDP e não pode ser acessada diretamente pela CPU; por isso é chamada de VRAM (Video RAM).

Para o V9938 e V9958 pode ser conectado mais um banco de 64 Kbytes de expansão. Entretanto, esta expansão serve apenas para armazenar dados, pois seu conteúdo não é exibido na tela. Além disso, por limitações de hardware, ela só pode ser usada adequadamente nas screens até 6. Screens de 7 para cima exigem uma velocidade de acesso que o banco adicional não comporta. Para essas screens, no caso dos 128K padrão e por questão da baixa velocidade das memórias, o VDP guarda os bytes pares da imagem em um banco de 64K e os bytes ímpares em outro banco, tornando inviável o uso dos 64K adicionais nesta situação. Os 64K de expansão são espelhados, conforme a ilustração abaixo:



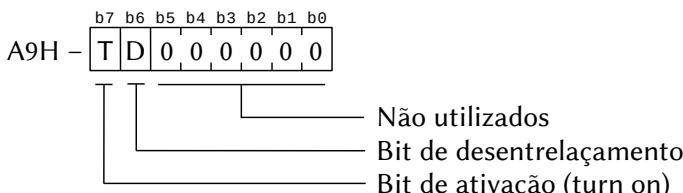
### 5.1.3 – A ADVRAM (V9918/38/58)

A ADVRAM foi um periférico desenvolvido no Brasil que permite à CPU acessar diretamente a VRAM. Para isso, ela é conectada diretamente em um slot, que no caso é onde está a Main-ROM. Ela funciona sempre na página 2 (8000H a BFFFH) desse slot.

Como uma página física tem apenas 16 Kbytes, é necessário fazer um chaveamento para que se possa acessar os 128 Kbytes da VRAM. Esse chaveamento é feito exatamente da mesma forma que para a Memória Mapeada, inclusive usando a mesma porta de I/O (0FEH). A ADVRAM é uma Memória Mapeada especial, espelhando a VRAM. Abaixo está descrita a porção de VRAM acessada pelo comando OUT:

OUT (0FEH), 0	; 00000H a 03FFFH
OUT (0FEH), 1	; 00000H a 03FFFH
OUT (0FEH), 7	; 1C000H a 1FFFFH

Para evitar que programas aplicativos confundam a ADVRAM com uma Memória Mapeada normal, ela é desativada no reset. Para ativá-la, existe um registrador de controle, que pode ser acessado pela porta de I/O 9AH. A estrutura desse registrador está ilustrada abaixo.



O bit de ativação deve ser setado para habilitar a ADVRAM. Seu valor na inicialização é 0; portanto no reset a ADVRAM está desabilitada. Já o bit de desentrelaçamento serve para fazer com que a ADVRAM se comporte de forma linear quando for usado o modo entrelaçado do VDP, facilitando a programação.

Para escrever um dado no registrador de controle, deve ser usada a seguinte sequência de instruções:

```
LD   A, valor
IN   A, (09AH)
```

Não houve erro; é mesmo usada uma instrução IN para escrever no registrador de controle. O Z80, ao ler um dado através de uma porta de I/O, coloca o conteúdo do registrador A nas linhas de endereço de A8~A15 e o endereço do dispositivo nas linhas A0~A7. Essa característica permite escrever um dado enquanto se lê outro. No caso, a ADVRAM lê o conteúdo do registrador A presente nas linhas de endereço A8~15 e depois o escreve no registrador de controle.

#### 5.1.4 – Portas de acesso ao VDP (V9918/38/58)

O TMS9918 tem duas portas de acesso enquanto o V9938 e o V9958 têm quatro portas para comunicação com a CPU. As funções dessas portas estão listadas na tabela abaixo. Elas são expressadas por “r” e “w” e seus valores estão armazenados, respectivamente, nos endereços 0006H e 0007H da Main-ROM.

r = (0006H) = porta de leitura (RDVDP)  
 w = (0007H) = porta de escrita (WRVDP)

Quando a comunicação com o VDP requerer maior velocidade, essas portas podem ser usadas para acessar o VDP diretamente. Entretanto, o VDP é lento, requerendo até 8 µS de intervalo entre acessos consecutivos. Por isso, é bom evitar instruções tipo OTIR/OTDR, que podem acarretar falhas na leitura de dados pelo VDP. Dê preferência a loops com as instruções OUTI/OTDR. As portas são descritas abaixo:

Porta #0	(leitura)	r	Lê dados da VRAM (MSX1)
Porta #0	(escrita)	w	Escreve dados na VRAM (MSX1)
Porta #1	(leitura)	r+1	Lê registrador de estado (MSX1)
Porta #1	(escrita)	w+1	Escreve no registrador de controle (MSX1)
Porta #2	(escrita)	w+2	Escreve nos registradores de paleta (MSX2 ou superior)
Porta #3	(escrita)	w+3	Escreve no registrador especificado indiretamente (MSX2 ou superior)

Muito embora as portas de acesso ao VDP não tenham sido padronizadas, elas são as mesmas em todos os modelos MSX lançados. São as seguintes:

Porta #0 – 98H  
 Porta #1 – 99H  
 Porta #2 – 9AH  
 Porta #3 – 9BH

## 5.2 – ACESSO À VRAM E AO VDP (V9918/38/58)

O VDP e a VRAM podem ser acessados diretamente pelas portas de I/O já descritas. Esta seção descreve como fazê-lo.

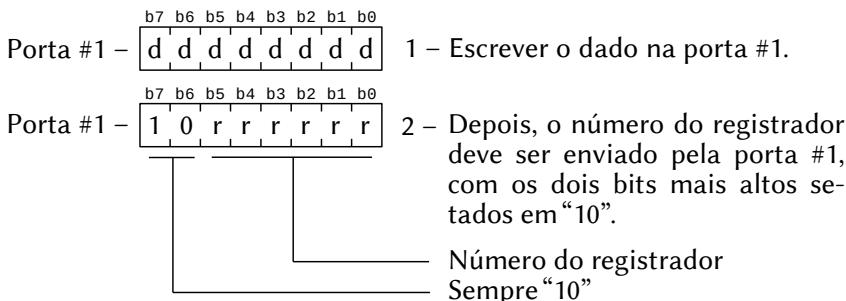
### 5.2.1 – Acesso aos registradores de controle

Os registradores de controle são apenas de escrita. Entre-tanto, o conteúdo do primeiro subgrupo (R#0 a R#27) pode ser obtido pelo comando VDP(n) do BASIC porque há uma cópia deles na área de trabalho do sistema (endereços F3DFH a F3E6H, FFE7H a FFF6H e FFFAH a FFFCH). Essa cópia é feita pela BIOS.

Existem três meios para escrever dados nos registradores de controle, descritos abaixo.

#### 5.2.1.1 – Acesso direto

O primeiro meio é especificar o dado e escrevê-lo diretamente. O dado é escrito primeiro, na porta #1, seguido do número do registrador respectivo, conforme ilustrado abaixo:

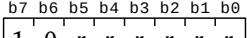
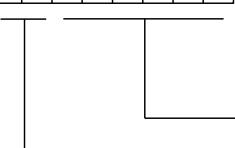
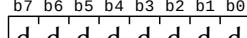
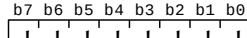


Exemplo de código ASM:

```
LD  A,x          ; x = dado a ser escrito
OUT (099H),A      ; envia o dado
LD  A,10000000B   ; seta b7-b6 em “10”
OR  Y             ; y = número registr. (0~46)
OUT (099H),A      ; envia o registrador
```

### 5.2.1.2 – Acesso indireto

O segundo meio é escrever o dado no registrador especificado por R#17. Para isso, é necessário usar o método direto para colocar o número do registrador desejado em R#17, com os dois bits mais altos setados em “10”. Depois pode-se enviar dados continuamente pela porta #3 para o mesmo registrador. Esse meio é útil para executar comandos de hardware do VDP.

R#17 –  	<p>1 – O número do registrador deve ser colocado em R#17 com os dois bits mais altos setados em “10”</p> <p>Número do registrador Sempre “10”</p>
Porta #3 – 	<p>2 – O dado enviado pela porta #3 será colocado na registrador especificado em R#17.</p>
Porta #3 – 	<p>3 – Outros dados podem ser enviados pela porta #3 para o mesmo registrador, indefinidamente.</p>

Exemplo de código ASM:

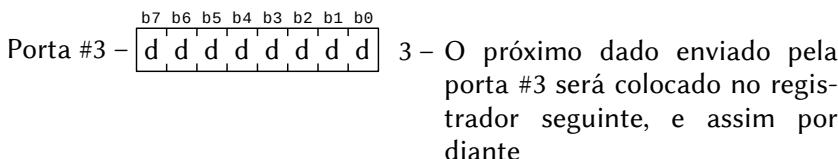
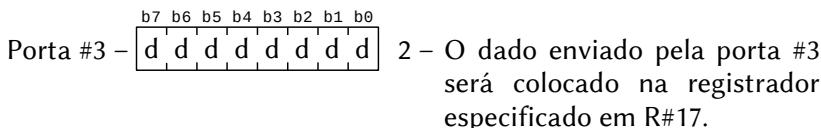
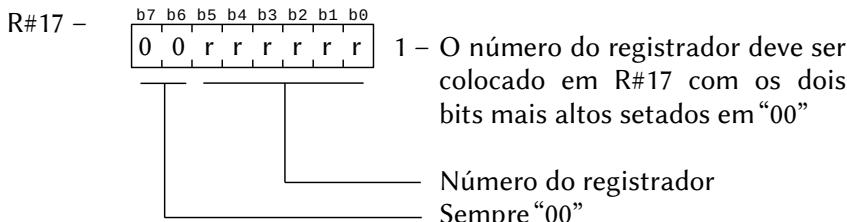
```

LD   A,10000000B ; seta b7-b6 em “10”
OR   Y             ; y = número registr. (0~46)
OUT  (099H),A      ; envia o número do registr.
LD   A,10000000B ; b7,b6 = 10 → sem autoinc
OR   17            ; A = R#17 com b7,b6 = 10
OUT  (099H),A      ; envia o reg. para R#17
LD   A,x           ; x = dado a ser escrito
OUT  (09BH),A      ; envia um dado
LD   A,x           ; x = dado a ser escrito
OUT  (09BH),A      ; envia outro dado para o
                   ; mesmo registrador, indefinidamente

```

### 5.2.1.3 – Acesso indireto com autoincremento

Esse meio é similar ao anterior, com a diferença que, cada vez que o dado for escrito pela porta #3, R#17 é incrementado em 1 e o próximo dado enviado será escrito no registrador seguinte. Para usá-lo, é necessário colocar, pelo método direto, o número do primeiro registrador em R#17 com os dois bits mais altos setados em “00”. Depois é só enviar os dados pela porta #3.



### Exemplo de código ASM:

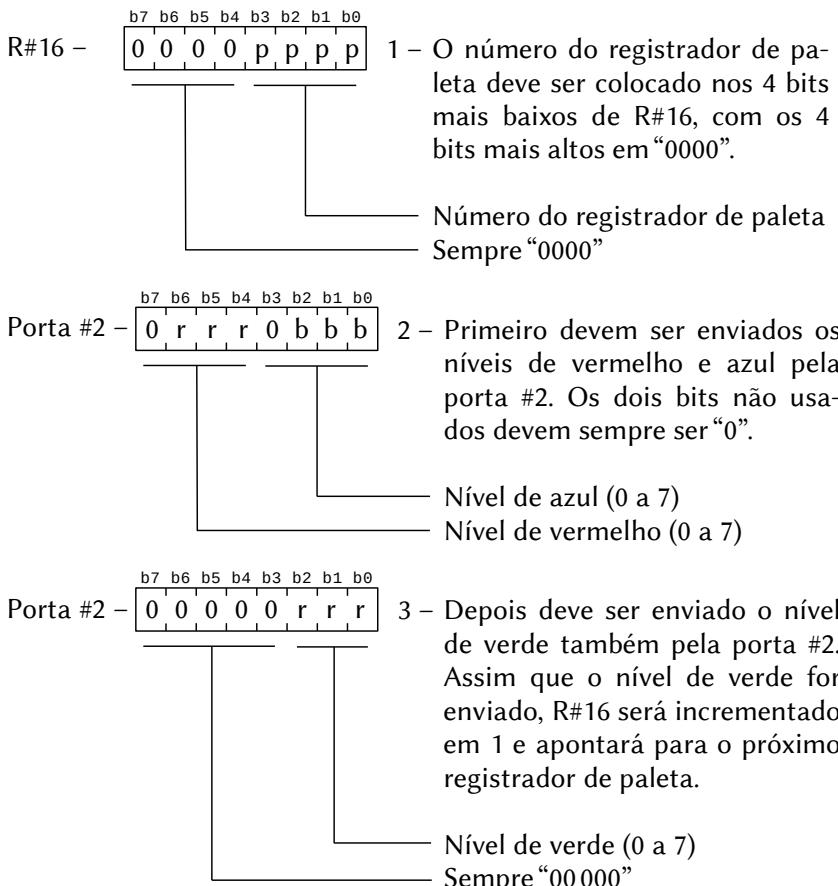
```

LD   A, 10000000B ; seta b7-b6 em “10”
OR   Y              ; y = número registr. (0~46)
OUT  (099H),A       ; envia o número do registr.
LD   A,17            ; A = R#17 com b7,b6 = 00
                  ; (função autoinc ativa)
OUT  (099H),A       ; envia o reg. para R#17
LD   A,x             ; x = dado a ser escrito
OUT  (09BH),A        ; envia um dado para R#n
LD   A,x             ; x = dado a ser escrito
OUT  (09BH),A        ; envia um dado para R#n+1
LD   A,x             ; x = dado a ser escrito
OUT  (09BH),A        ; envia um dado para R#n+2
                  ; (e assim sucessivamente)

```

### 5.2.2 – Acesso aos registradores de paleta

Para escrever nos registradores de paleta (P#0 a P#15), é necessário especificar o número da paleta nos quatro bits mais baixos de R#16 e enviar os dados pela porta #2. Como cada registrador tem 9 bits, os dados devem ser enviados por dois bytes consecutivos. Depois que os dois bytes forem enviados, R#16 é automaticamente incrementado em 1, apontando para o próximo registrador de paleta. Essa característica facilita a inicialização da paleta.



Nível de verde (0 a 7)  
Sempre “00 000”

Exemplo de código ASM:

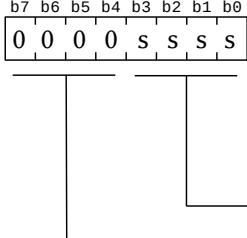
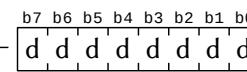
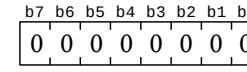
```

LD   A,x          ; x = nûm.reg.paleta (0~15)
OUT (099H),A      ; envia o nûmero do registr.
LD   A,16          ; R#16
OUT (099H),A      ; envia reg. pal. para R#16
LD   A,x          ; x = nûvel vermelho/azul
OUT (09AH),A      ; envia verm/azul p/ P#n
LD   A,x          ; x = nûvel de verde
OUT (09AH),A      ; envia verde para P#n
LD   A,x          ; x = nûvel vermelho/azul
OUT (09AH),A      ; envia verm/azul p/ P#n+1
LD   A,x          ; x = nûvel de verde
OUT (09AH),A      ; envia verde para P#n+1
; (e assim sucessivamente)

```

### 5.2.3 – Lendo os registradores de estado

Os registradores de estado são apenas de leitura. O conteúdo deles pode ser lido pela porta #1, colocando em R#15 o número do registrador de estado a ser lido. As interrupções devem ser desativadas (DI) durante a leitura dos registradores de estado. Depois de lido, o registrador R#15 deve ser zerado antes das interrupções serem reabilitadas.

R#15 – 	1 – O nûmero do registrador de estado deve ser colocado nos 4 bits mais baixos de R#15. As interrupções devem ser desabilitadas.  Número do registrador de estado Sempre “0000”
Porta #1 – 	2 – O dado fica disponível para ser lido pela porta #1
R#15 – 	3 – R#15 deve ser zerado antes de reabilitar as interrupções.

Exemplo de código ASM:

```

STA:  DEFB 00H
      DI           ; desabilita interrupções

```

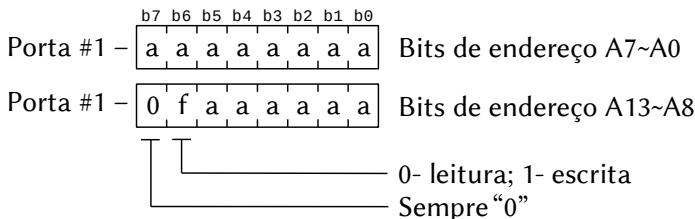
```

LD   A,x           ; x = núm.reg.estado (0~9)
OUT  (099H),A      ; envia o número do registr.
LD   A,15          ; R#15
OUT  (099H),A      ; envia reg. estado p/ R#15
IN   A,(099H)       ; A recebe reg. estado
LD   (STA),A        ; salva valor em STA
XOR  A              ; A = 0
OUT  (099H),A      ; envia o valor 0
LD   A,15          ; R#15
OUT  (099H),A      ; zera R#15
EI                   ; habilita interrupções

```

### 5.2.4 – Acesso à VRAM pela CPU

Para acessar os 16 Kbytes do TMS9918A do MSX1 é usado um bus de endereços de 14 bits. Para que a CPU possa acessar a VRAM, esses 14 bits devem ser enviados pela porta #1 em dois bytes consecutivos. O bit 6 do segundo byte é uma flag para indicar leitura ou escrita, conforme ilustrado abaixo:



Depois de setar o endereço da VRAM, o dado pode ser lido ou escrito através da porta #0. A flag de leitura/escrita deve estar ajustada conforme já descrito. O contador de endereços é automaticamente incrementado em 1 cada vez que um byte for lido ou escrito através da porta #0, de forma a facilitar o acesso contínuo à VRAM.

Exemplo de código ASM:

```

LD   HL,x           ; X = 0000H ~ 3FFFH
DI                   ; desabilita interrupções
LD   A,L             ; A = LSB 8 bits
OUT  (099H),A        ; envia LSB 8 bits
LD   A,H             ; A = MSB 6 bits
OR   64               ; OR 01000000 seta gravação
OUT  (099H),A        ; envia MSB 6 bits + flag

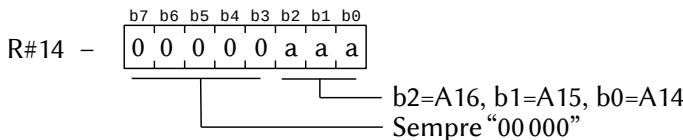
```

```

DI           ; habilita interrupções
LD  A,x      ; X = dado a ser enviados
OUT (098H),A ; envia o dado p/ VRAM

```

Para acessar os 128 Kbytes de VRAM do V9938/V9958, é usado um bus de endereços de 17 bits. Desses, os três bits mais altos são armazenados em R#14. Dessa forma é possível selecionar até 8 páginas de 16 Kbytes cada.



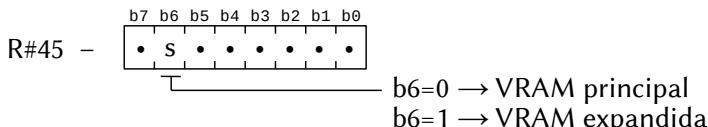
Exemplo de código ASM:

```

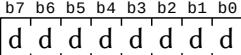
LD  A,x      ; x = MSB 3 bits (páginas)
OUT (099H),A ; envia a página
LD  A,14      ; R#14
OUT (099H),A ; envia página p/ R#14
LD  HL,x     ; X = 0000H ~ 3FFFH
DI           ; desabilita interrupções
LD  A,L       ; A = LSB 8 bits
OUT (099H),A ; envia LSB 8 bits
LD  A,H       ; A = MSB 6 bits
OR  64         ; OR 01000000 seta gravação
OUT (099H),A ; envia MSB 6 bits + flag
DI           ; habilita interrupções
LD  A,x      ; X = dado a ser enviados
OUT (098H),A ; envia o dado p/ VRAM

```

Os primeiros 64 Kbytes da VRAM e a VRAM expandida ocupam o mesmo espaço de endereçamento do VDP. Se o micro não possuir VRAM expandida, sempre será selecionada a VRAM principal. O bit 6 de R#45 pode ser usado para alternar entre os dois bancos de 64 Kbytes, conforme a ilustração abaixo:



Depois de setar o endereço da VRAM e se é expandida ou não, o dado pode ser lido ou escrito através da porta #0. A flag de leitura/ escrita deve estar ajustada conforme já descrito. O contador de endereços é incrementado em 1 cada vez que um byte for lido ou escrito através da porta #0. Essa característica facilita o acesso contínuo à VRAM.

Porta #0 –  A escrita ou leitura de dados é feita através da porta #0.

Os tempos de acesso entre leituras/escritas na VRAM dependem da screen e do VDP. Os tempos mínimos estão listados abaixo. O valor seguido de T significa o número de ciclos T de um Z80 trabalhando no clock padrão do MSX (3,58 MHz).

		TMS9918/9928	V9938/V9958
Screen 0, width 40	Texto 1	3,4 µS (12T)	5,6 µS (20T)
Screen 0, width 80	Texto 2		5,6 µS (20T)
Screen 1	Gráfico 1	8,2 µS (29T)	4,2 µS (15T)
Screen 2	Gráfico 2	8,2 µS (29T)	4,2 µS (15T)
Screen 3	Multicor	3,7 µS (13T)	4,2 µS (15T)
Screen 4	Gráfico 3		4,2 µS (15T)
Screen 5	Gráfico 4		4,2 µS (15T)
Screen 6	Gráfico 5		4,2 µS (15T)
Screen 7	Gráfico 6		4,2 µS (15T)
Screen 8	Gráfico 7		4,2 µS (15T)
Screen 10	Gráfico 8		4,2 µS (15T)
Screen 11	Gráfico 8		4,2 µS (15T)
Screen 12	Gráfico 9		4,2 µS (15T)

Observe que, no caso do MSX turbo R, o acesso ao VDP é temporizado em 8 µS entre acessos consecutivos aos registradores.

### 5.3 – MODOS DE TELA (V9918/38/58)

O MSX1 tem 4 modos de tela. Já o MSX2 tem seis modos a mais e no MSX2+ e MSX turbo R, além desses, existem mais dois. Na tabela abaixo os modos marcados com “\*\*” foram acrescentados para o MSX2 e

os marcados com “\*\*” foram acrescentados para o MSX2+ e MSX turbo R. Juntamente com os modos, há uma curta descrição dos mesmos.

MODO	SCREEN	DESCRIÇÃO RESUMIDA
TEXTO 1	SCREEN 0 WIDTH 40	Até 40 caracteres por linha de texto; uma cor para todos os caracteres
TEXTO 2 *	SCREEN 0 WIDTH 80	Até 80 caracteres por linha de texto; função de “blink” inclusa
MULTICOR	SCREEN 3	Pseudográfico; 64 x 48 pontos; 16 cores independentes para cada ponto
GRÁFICO 1	SCREEN 1	32 caracteres por linha de texto; caracteres de várias cores disponíveis
GRÁFICO 2	SCREEN 2	256 x 192 pontos; 16 cores limitadas a 2 cada 8 pontos horizontais; sprites modo 1
GRÁFICO 3 *	SCREEN 4	Igual a Gráfico 2, mas usa sprites modo 2
GRÁFICO 4 *	SCREEN 5	256 x 212 pontos; 16 cores de 512 para cada ponto; sprites modo 2
GRÁFICO 5 *	SCREEN 6	512 x 212 pontos; 4 cores de 512 para cada ponto; sprites modo 2
GRÁFICO 6 *	SCREEN 7	512 x 212 pontos; 16 cores de 512 para cada ponto; sprites modo 2
GRÁFICO 7 *	SCREEN 8	256 x 212 pontos; 256 cores para cada ponto; sprites modo 2
GRÁFICO 8 **	SCREEN 10 SCREEN 11	256 x 212 pontos; 65 536 cores para cada 4 pontos horizontais ou 16 cores de 512 para cada ponto; máximo de 12 499 cores simultâneas; sprites modo 2
GRÁFICO 9 **	SCREEN 12	256 x 212 pontos; 131 072 cores para cada 4 pontos horizontais; máximo de 19 268 cores simultâneas; sprites modo 2

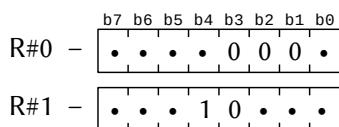
Existem também algumas configurações especiais que u-sam alguns dos modos gráficos para texto. Elas são as seguintes:

Screen 9	Width 1~40	Graphic 4	Korean MSX2
	Width 41~80	Graphic 5	
Kanji0	Width 1~32	Graphic 4	
	Width 33~64	Graphic 6	
Kanji1	Width 1~40	Graphic 4	MSX2+ MSX turbo R
	Width 41~80	Graphic 6	
Kanji2	Width 1~32	Graphic 4 interlaced	
	Width 33~64	Graphic 6 interlaced	
Kanji3	Width 1~40	Graphic 4 interlaced	
	Width 41~80	Graphic 6 interlaced	

### 5.3.1 – Modo Texto 1

- 24 linhas de até 40 caracteres cada;
- uma cor de fundo e uma para os caracteres, selecionadas de 16 (MSX1) ou 512 (MSX2 ou superior);
- 256 caracteres disponíveis com resolução de 6 pontos horizontais por 8 verticais;
- requer 2048 bytes para a fonte e 960 bytes para a tela;
- compatível com Screen 0, Width 1~40.

O modo texto 1 é selecionado pelos registradores R#0 e R#1, conforme ilustrado abaixo:



A área onde a fonte de caracteres é armazenada chama-se *tabela geradora de padrões*. Nela, cada caractere é definido por 8 bytes, mas os dois bits mais baixos de cada byte não são exibidos. Por isso, a célula onde cada caractere é mostrado tem 6 x 8 pontos. A fonte contém 256 caracteres distintos, numerados de 0 a 255.

A localização da tabela geradora de padrões está especificada em R#4. Os 11 bits mais baixos (b10 ~ b0) não são especificados; são sempre 0. Apenas os 6 bits mais altos podem ser especificados. Por isso, a tabela sempre começa em um múltiplo de 2 Kbytes a partir de 00 000H. Esse endereço pode ser obtido pela variável de sistema BASE(2) do BASIC.

R#4 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	a	a	a	a	a	a

 - A16 ~ A11

A *tabela de nomes dos padrões* armazena a posição de cada caractere que deve ser apresentado na tela. Um byte é usado para cada caractere; ele contém o código ASCII do caractere a ser apresentado na posição respectiva.

O endereço inicial da tabela é especificado em R#2. Os 10 bits mais baixos (b9 ~ b0) não são especificados; são sempre 0. Apenas os 7 bits mais altos são especificados. Por isso, a tabela de nomes sempre começa em um múltiplo de 1 Kbyte a partir de 00 000H. Esse endereço pode ser obtido pela variável de sistema BASE(0) do BASIC.

R#2 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	a	a	a	a	a	a	a

 - A16 ~ A10

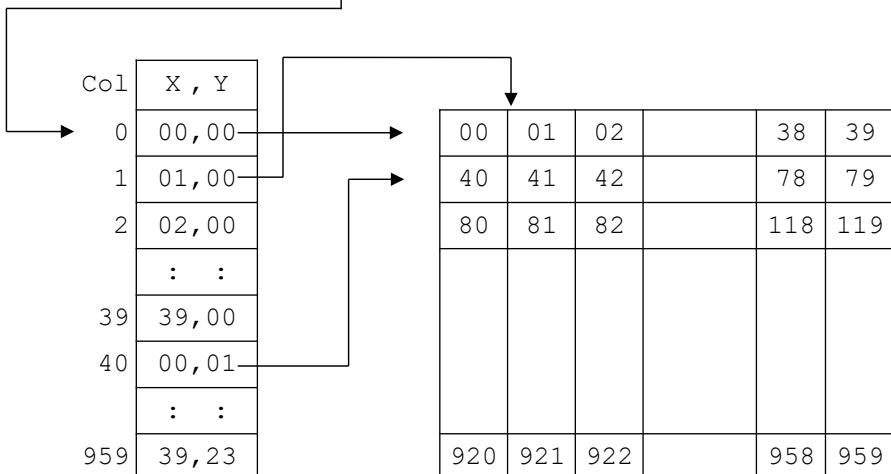
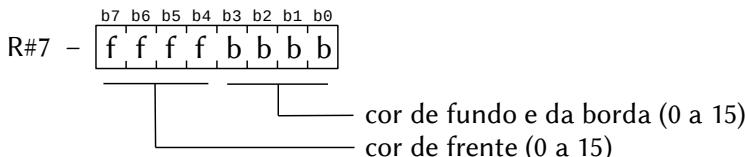


Tabela de nomes

Posições correspondentes na tela

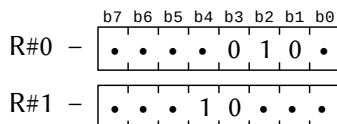
Para especificar a cor dos caracteres e a cor de fundo, é usado o registrador R#7. Os quatro bits mais altos de R#7 especificam a cor dos caracteres (cor de frente) e os quatro bits mais baixos especificam a cor de fundo e da borda.



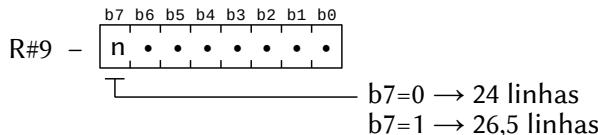
### 5.3.2 – Modo Texto 2

- 24 ou 26,5 linhas de até 80 caracteres cada;
- uma cor de fundo e uma para os caracteres, selecionadas de 512;
- 256 caracteres disponíveis com resolução de 6 pontos horizontais por 8 verticais;
- função de “blink” (piscar) independente para cada posição na tela;
- requer 2048 bytes para a fonte;
- para 24 linhas requer 1920 bytes para a tela (80 caracteres x 24 linhas) e 240 bytes (1920 bits) para os atributos de “blinking”;
- para 26,5 linhas requer 2160 bytes para a tela (80 caracteres x 27 linhas) e 270 bytes (2160 bits) para os atributos de “blinking”;
- compatível com Screen 0, Width 41~80.

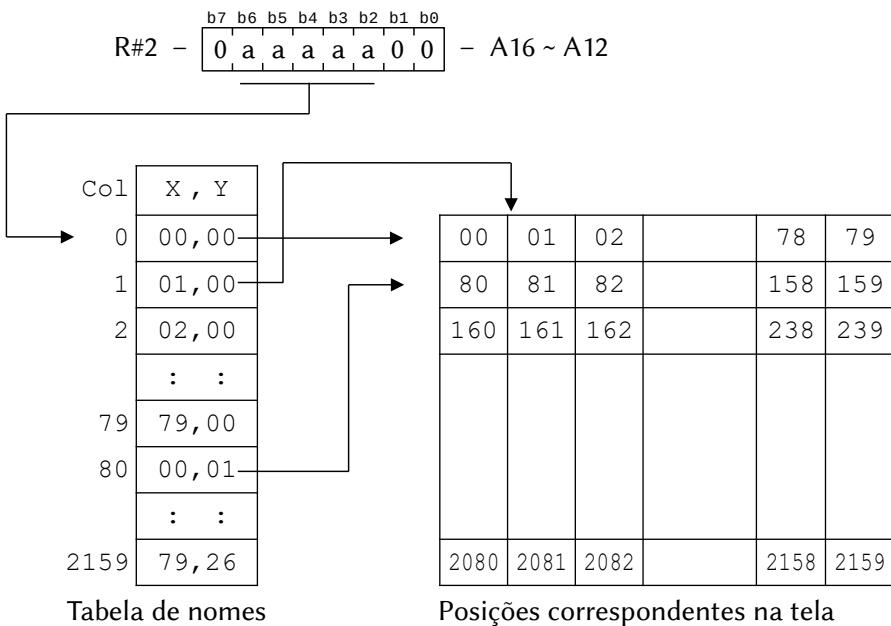
O modo texto 2 é selecionado pelos registradores R#0 e R#1, conforme ilustrado abaixo:



O modo texto 2 pode apresentar 24 ou 26,5 linhas, dependendo do valor do bit 7 de R#9. Na última linha do modo 26,5 linhas é apresentada apenas a metade superior dos caracteres. Esse modo não é suportado pelo BASIC.



A tabela geradora de padrões do modo texto 2 tem a mesma estrutura, função e tamanho que a do modo texto 1. Como o número de caracteres que podem ser mostrados nesse modo foi aumentado para um máximo de 2160 (80 x 27), a memória máxima ocupada pela tabela de nomes é de 2160 bytes. O endereço inicial da tabela de nomes deve ser especificado em R#2. Apenas os 5 bits mais altos são especificados; os 12 bits mais baixos são sempre 0. Por isso, o endereço inicial da tabela de nomes é sempre um múltiplo de 4 Kbytes a partir de 00 000H.



No modo texto 2 é possível fazer os caracteres piscarem. Esse recurso é chamado de “blink”. A tabela de “blinking” armazena a posição de cada caractere na tela; um bit na tabela corresponde a um caractere. Quando esse bit for 1, a função de “blink” é ativada para o caractere respectivo. O endereço da tabela é armazenado em R#3 e R#10. Os 8 bits mais altos especificam o endereço e os 9 bits mais baixos são sempre 0. Por isso, o endereço da tabela de “blinking” é sempre um múltiplo de 512 bytes a partir de 00 000H.

R#10 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	0	0	0	a	a	a

 R#3 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
a	a	a	a	a	0	0	0

 – A16 ~ A9

byte 0	00,00	01,00	02,00	03,00	04,00	05,00	06,00	07,00
byte 1	08,00	09,00	10,00	11,00	12,00	13,00	14,00	15,00
:	:	:	:	:	:	:	:	:
byte 269	72,26	73,26	74,26	75,26	76,26	77,26	78,26	79,26

Tabela de “blinking” com as posições correspondentes na tela

As cores dos caracteres no modo texto 2 são especificadas em R#7 de R#12. Os quatro bits mais altos de R#7 especificam a cor dos caracteres (cor de frente) e os quatro bits mais baixos especificam a cor de fundo e da borda. Quando a função de “blink” estiver ativa, a cor do caractere será especificada pelos quatro bits mais altos de R#12 e a cor de fundo do caractere pelos quatro bits mais baixos de R#12.

R#7 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
f	f	f	f	b	b	b	b

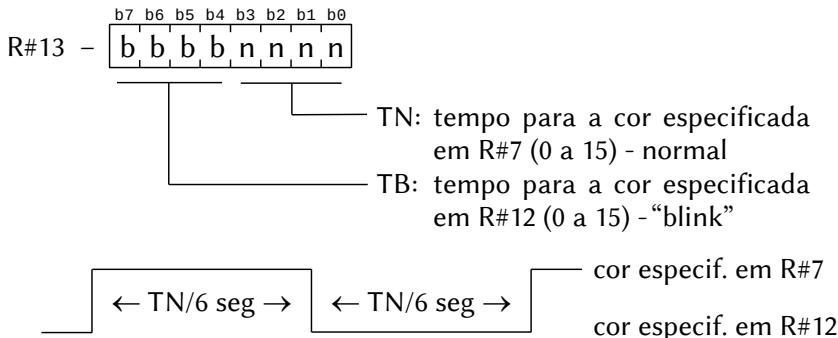
 ← cor normal dos caracteres

R#12 - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
f	f	f	f	b	b	b	b

 ← cor com a função “blink” ativa

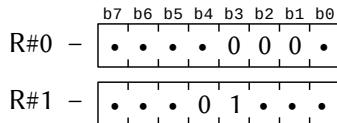
O tempo de “blinking” (tempo que o caractere assume as cores de “blink” e depois volta às cores normais) é especificado em R#13. Os quatro bits mais altos de R#13 definem o tempo em que o caractere fica com a cor original e os quatro bits mais baixos definem o tempo em que os caracteres ficam com a cor de “blink”. O período de tempo é especificado em unidades de 1/6 de segundo. Quando o registrador estiver zerado, as cores são assumidas permanentemente. A ilustração da página seguinte mostra como o tempo é dividido para a função de “blink”.



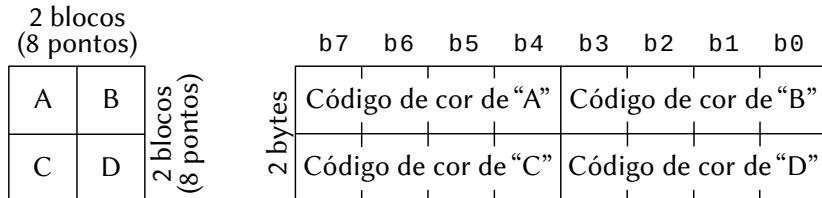
### 5.3.3 – Modo Multicor

- 64 (horizontal) x 48 (vertical) blocos;
- até 16 cores simultâneas;
- cada bloco tem 4x4 pontos e uma cor;
- requer 2048 bytes para a tabela de cores e 768 para especificar a localização dos blocos na tela;
- sprites modo 1;
- compatível com Screen 3.

O modo multicor é selecionado por R#0 e R#1, conforme ilustrado abaixo:



A organização desse modo é um pouco complexa. Cada padrão (caractere) corresponde a 4 blocos, numa construção 2x2. Dois bytes na tabela de padrões representam a cor de cada bloco. A organização de cada padrão está ilustrada abaixo.



O endereço da tabela de cores é especificado em R#4. Apenas os 6 bits mais altos são especificados; por isso o endereço inicial da tabela sempre será um múltiplo de 2 Kbytes a partir de 00 000H.

Cada 8 bytes da tabela de padrões correspondem a 1 caractere de largura (2 blocos horizontais) e varre a tela no sentido vertical de forma intercalada, conforme ilustração abaixo.

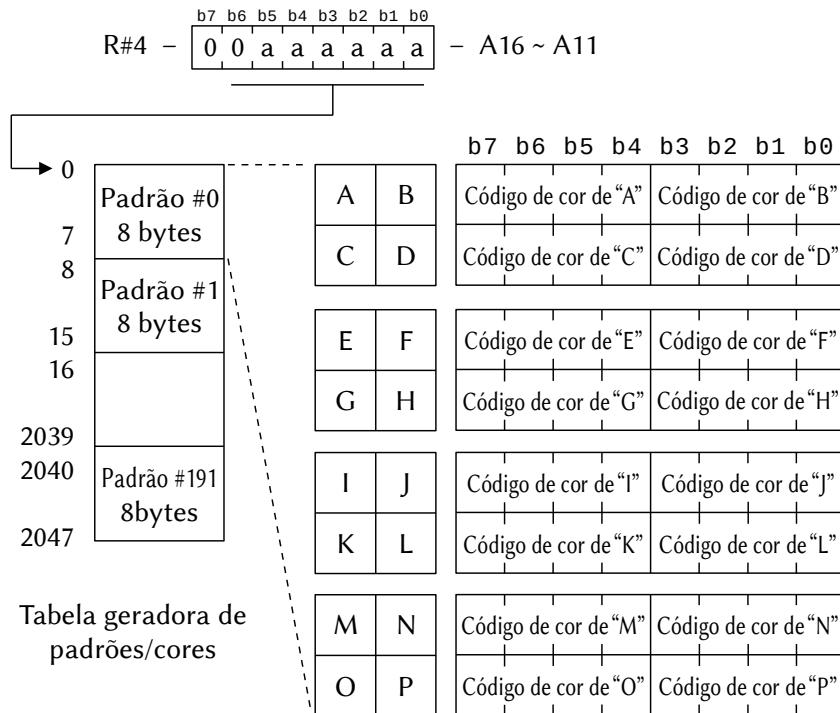


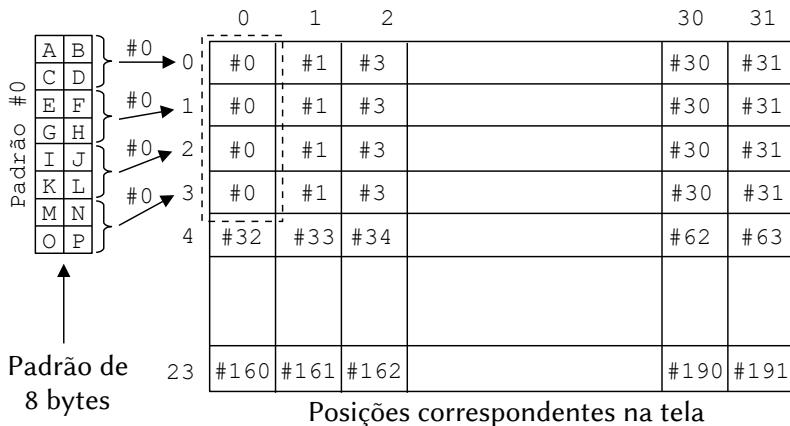
Tabela ABCD: válida quando Y for 0, 4, 8, 12, 16 ou 20

Tabela EFGH: válida quando Y for 1, 5, 9, 13, 17 ou 21

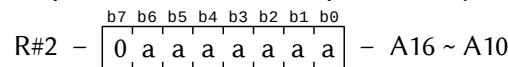
Tabela I J K L: válida quando Y for 2, 6, 10, 14, 18 ou 22

Tabela MNOP: válida quando Y for 3, 7, 11, 15, 19 ou 23

A tabela de nomes especifica as coordenadas de tela onde os caracteres (2 x 2 blocos) serão exibidos. Ela também é organizada de 4 em 4 de forma intercalada, conforme ilustrado abaixo.



O endereço inicial da tabela de nomes dos padrões é especificado em R#2. Só os 7 bits mais altos são especificados; por isso, a tabela de nomes sempre inicia em um múltiplo de 1 Kbyte a partir de 00 000H.



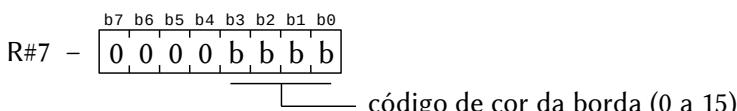
Col	X , Y
0	00,00
1	01,00
2	02,00
	:
31	39,00
32	00,01
	:
767	31,23

## Tabela de nomes

00	01	02		30	31
32	33	34		62	63
64	65	66		94	95
736	737	738		766	767

#### Posições correspondentes na tela

A cor da borda no modo multicor deve ser especificada nos 4 bits mais baixos de R#7.



### 5.3.4 – Modo Gráfico 1

- 32 (horizontal) x 24 (vertical) padrões;
- até 16 cores podem ser apresentadas simultaneamente (escolhidas de 512 para MSX2 ou superior);
- cada padrão tem 8 x 8 pontos e pode ser definido livremente;
- cores diferentes para cada 8 padrões pode ser definidas;
- requer 2048 bytes para a fonte de padrões, 768 bytes para a tabela de nomes e 32 bytes para a tabela de cores;
- sprites modo 1;
- compatível com Screen 1.

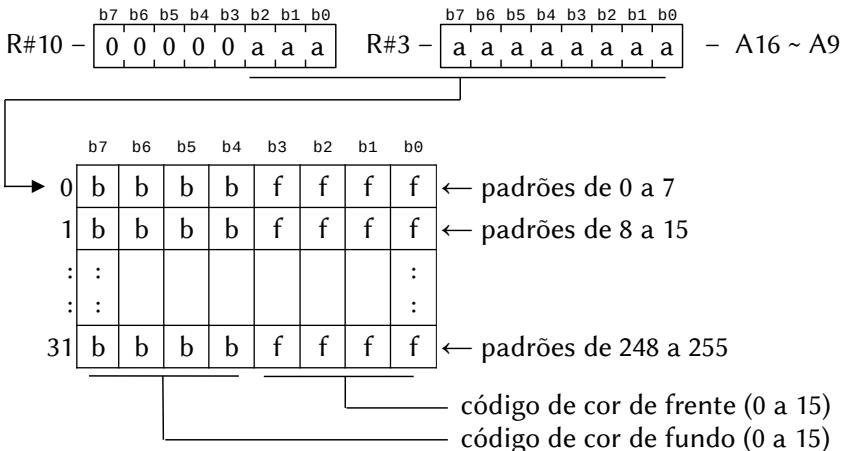
O modo gráfico 1 é selecionado por R#0 e R#1 conforme ilustrado abaixo.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#0 -	•	•	•	•	0	0	0	•
R#1 -	•	•	•	0	0	•	•	•

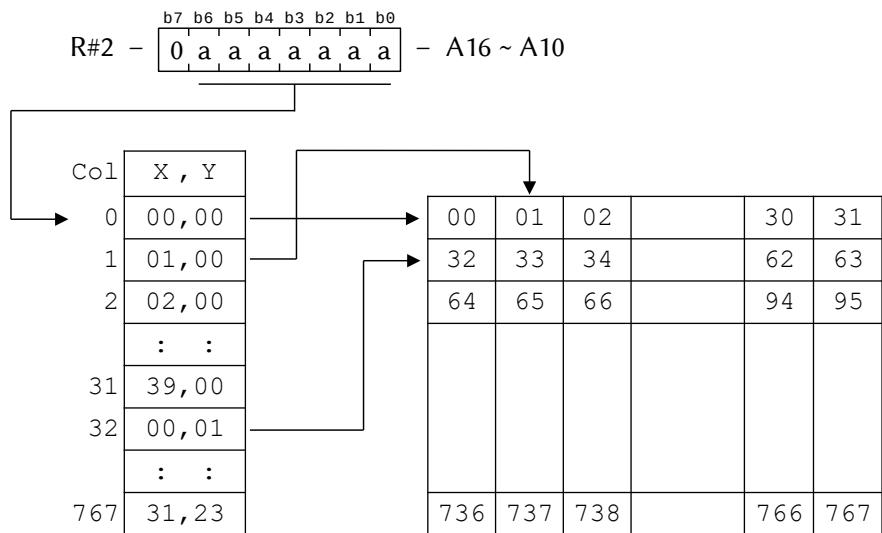
Nesse modo, 256 tipos de padrões, ou caracteres, podem ser apresentados na tela. A fonte de cada padrão é definida pela tabela geradora de padrões. O endereço inicial da tabela de padrões é especificado em R#4. Somente os 6 bits mais altos são especificados; por isso essa tabela sempre inicia em um múltiplo de 2 Kbytes a partir de 00 000H.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#4 -	0	0	a	a	a	a	a	a
								endereço inicial da tabela geradora de padrões
								sem significado (sempre 00)

A tabela de cores especifica uma cor para cada 8 padrões consecutivos na tabela de padrões. O endereço inicial da tabela de cores é especificado em R#3 e R#10. Somente os 11 bits mais altos são especificados (A16 a A6); por isso essa tabela sempre inicia num múltiplo de 64 bytes a partir de 0000H.



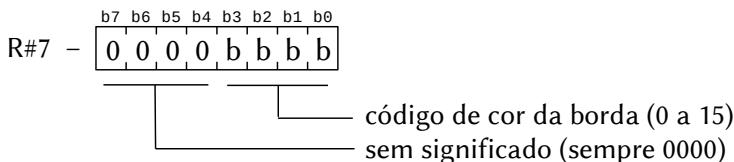
A tabela de nomes dos padrões tem 768 bytes e é a responsável pela localização dos mesmos na tela. O endereço inicial dessa tabela é especificado em R#2. Apenas os 7 bits mais altos (A16 a A10) são especificados; os bits A9 a A0 são sempre 0. Por isso, a tabela de nomes dos padrões sempre inicia num múltiplo de 1 Kbyte a partir de 0000H. A organização dessa tabela está ilustrada abaixo.



## Tabela de nomes

## Posições correspondentes na tela

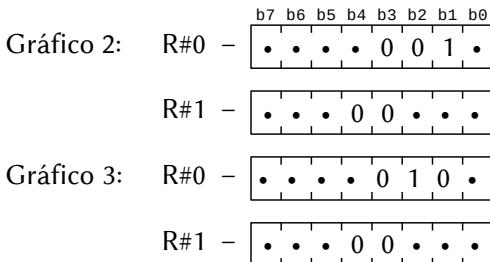
A cor da borda no modo gráfico 1 deve ser especificada nos 4 bits mais baixos de R#7.



### 5.3.5 – Modos Gráficos 2 e 3

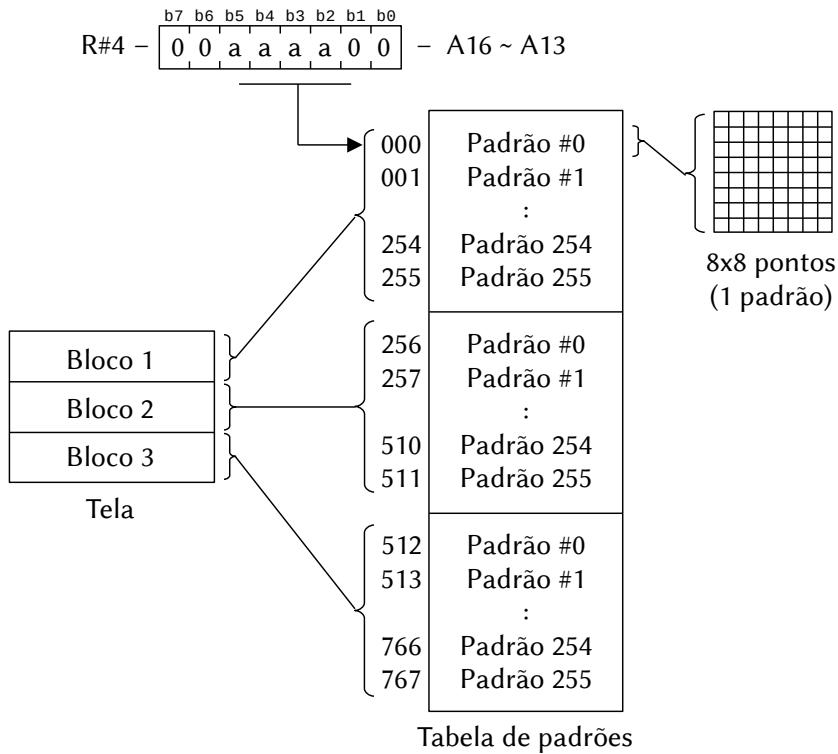
- 32 (horizontal) por 24 (vertical) padrões;
- até 16 cores podem ser apresentadas simultaneamente;
- 768 padrões diferentes são disponíveis;
- cada padrão tem 8 x 8 pontos;
- qualquer figura pode ser definida para cada padrão;
- apenas 2 cores podem ser definidas para cada 8 pontos horizontais;
- requer 6144 bytes para a fonte de padrões e mais 6144 bytes para a tabela de cores;
- sprites modo 1 para gráfico 2 e modo 2 para gráfico 3;
- gráfico 2 compatível com Screen 2 e gráfico 3 com Screen 4.

Os modos gráficos 2 e 3 são selecionados por R#0 e R#1 conforme ilustração abaixo. Eles são idênticos, exceto que o modo gráfico 2 usa sprites modo 1 e o gráfico 3 usa sprites modo 2.

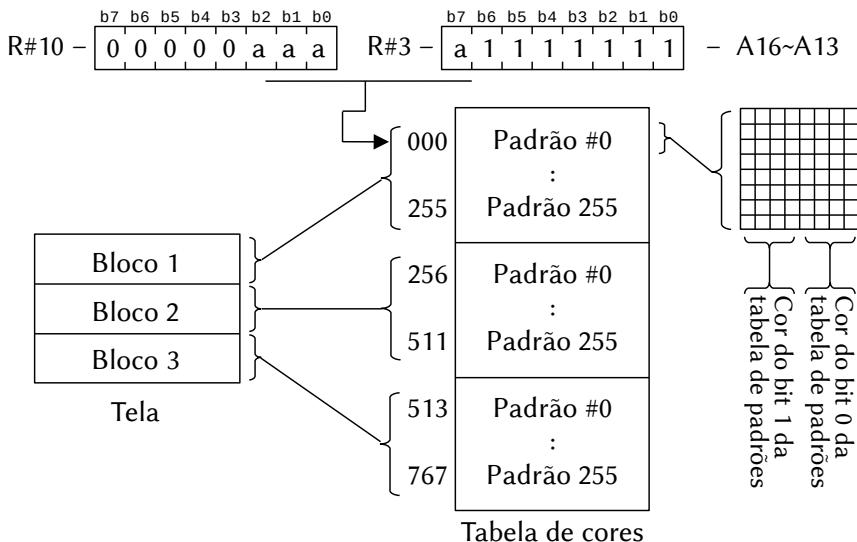


Nesses dois modos, a tabela geradora de padrões é compatível com o modo gráfico 1, onde 768 padrões diferentes podem ser mostrados. Como cada padrão tem 8 x 8 pontos e pode ter um desenho diferente, há uma simulação de apresentação de 256 x 192 pontos na tela. O en-

Endereço inicial da tabela geradora de padrões é especificado em R#4. Apesar dos 4 bits mais altos de endereço são válidos (A16 a A13); por isso o endereço inicial será sempre um múltiplo de 8 Kbytes a partir de 00000H. Nesse modo, a tela é dividida em três blocos de 256 padrões cada um, perfazendo um total de 768 padrões.



O tamanho da tabela de cores é o mesmo da tabela geradora de padrões, e as cores podem ser especificadas para cada bit 0 ou 1 de cada linha horizontal de cada padrão. O endereço inicial da tabela de cores é especificado em R#3 e R#10, mas apenas os quatro bits mais altos são especificados; por isso, a tabela de cores sempre inicia num múltiplo de 8 Kbytes a partir de 00000H.



A tabela de nomes dos padrões é dividida em três partes, uma para cada bloco de tela. Cada parte tem 256 bytes e é responsável pela apresentação de 256 padrões na tela. O endereço inicial da tabela de nomes é especificado em R#2.

R#2 - 

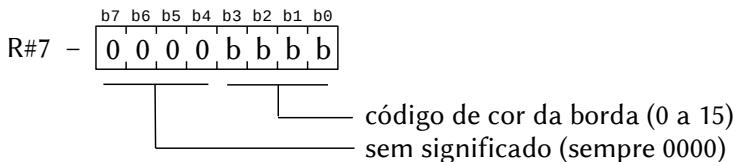
b7	b6	b5	b4	b3	b2	b1	b0
0	a	a	a	a	a	a	a

 - A16 ~ A10

0,0	0	0,0	31,0
1,0	1		
:			
31,7	255	0,7	31,7
1ª parte			
0,8	256	0,8	31,8
:			
31,15	511	0,15	31,15
2ª parte			
0,16	512	0,16	31,16
:			
30,23	766	0,23	31,23
31,23	767		

Tabela de nomes Tela

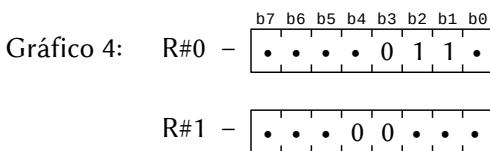
A cor da borda nos modos gráficos 2 e 3 é especificada nos 4 bits mais baixos de R#7.



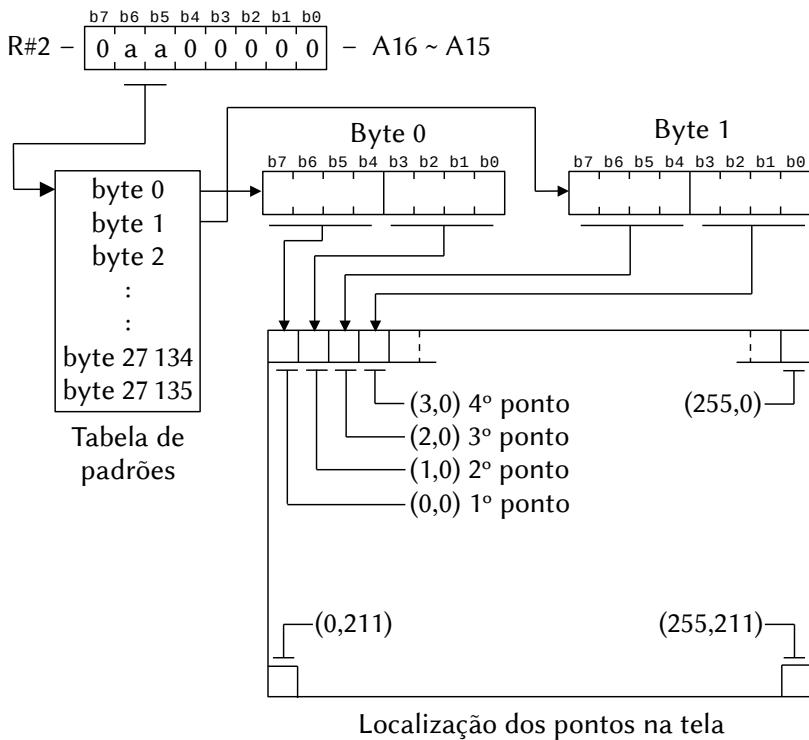
### 5.3.6 – Modo Gráfico 4

- 256 (horizontal) por 212 (vertical) pontos;
- apresenta até 16 cores escolhidas de 512 para cada ponto;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 26,5 Kbytes (4 bits x 256 pontos x 212 pontos) de memória;
- gráficos bit-mapped de fácil manipulação;
- compatível com Screen 5.

O modo gráfico 4 é selecionado por R#0 e R#1, conforme ilustrado abaixo:



No modo gráfico 4, um byte na tabela geradora de padrões corresponde a dois pontos na tela. Cada ponto é representado por 4 bits; portanto até 16 cores podem ser especificadas para cada ponto. O endereço inicial da tabela geradora de padrões é especificado em R#2. Apenas os dois bits mais altos são especificados; por isso a tabela geradora de padrões sempre inicia num múltiplo de 32 Kbytes a partir de 00000H.

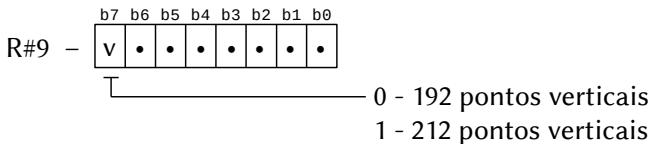


Nesse modo, cada byte corresponde a dois pontos horizontais em sequência na tela e como a resolução horizontal é de 256 pontos, 128 bytes são necessários para cada linha de tela. Nesse modo não há necessidade da tabela de nomes. O endereço na VRAM de cada ponto pode ser calculado pela seguinte expressão:

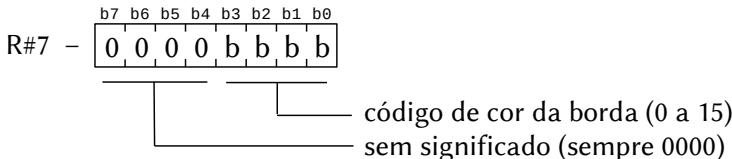
$$\text{Endereço} = X/2 + Y \cdot 128 + \text{Endereço inicial}$$

Onde X é a coordenada horizontal e Y é a coordenada vertical do ponto. O ponto é especificado pelos 4 bits mais altos do endereço se X for par e pelos 4 bits mais baixos de X for ímpar.

Nesse modo, o número de pontos verticais pode ser aumentado para 212. Para isso, basta setar o bit 7 de R#9, conforme ilustração abaixo.



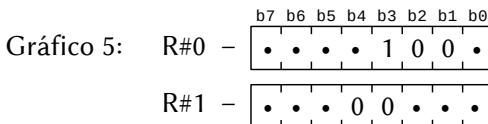
A cor da borda no modo gráfico 4 deve ser especificada nos 4 bits mais baixos de R#7, conforme ilustração abaixo.



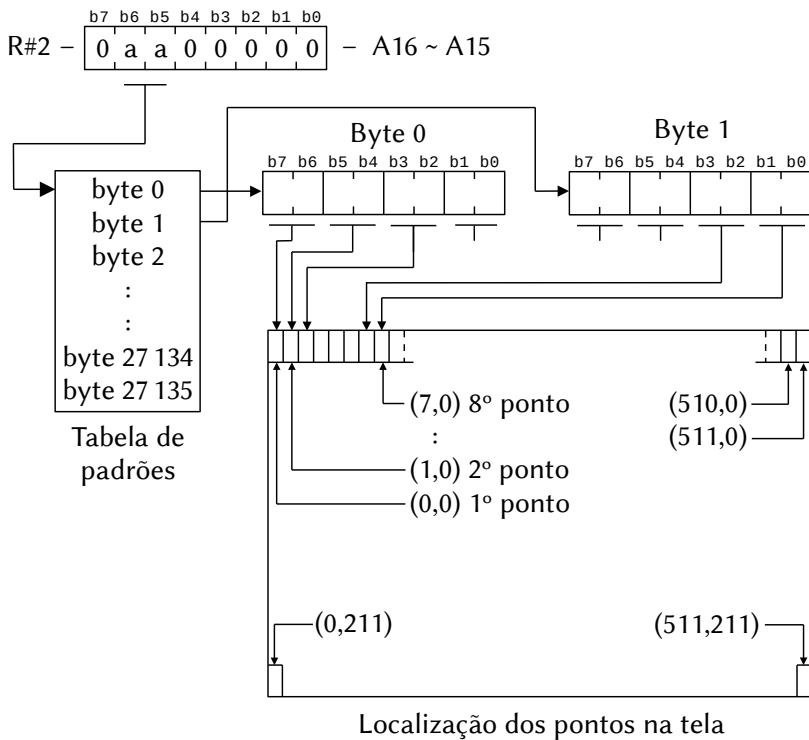
### 5.3.7 – Modo Gráfico 5

- 512 (horizontal) por 212 (vertical) pontos;
- apresenta até 4 cores escolhidas de 512 para cada ponto;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 26,5 Kbytes (2 bits x 512 pontos x 212 pontos) de memória;
- gráficos bit-mapped de fácil manipulação;
- compatível com Screen 6.

O modo gráfico 5 é selecionado por R#0 e R#1, conforme ilustração abaixo.



No modo gráfico 5 um byte na tabela de padrões corresponde a quatro pontos na tela. Cada ponto é representado por 2 bits e pode ter até 4 cores. O endereço inicial da tabela geradora de padrões é especificado em R#2, sendo que apenas os dois bits mais altos são válidos. Por isso a tabela sempre inicia num múltiplo de 32 Kbytes a partir de 0000H.



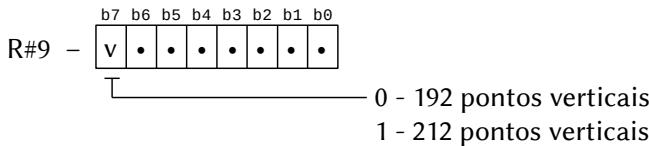
Cada byte representa quatro pontos em sequência horizontal na tela e como há 512 pontos de resolução horizontal, são necessários 128 bytes para representar cada linha da tela. O endereço na VRAM de cada ponto pode ser calculado pela seguinte expressão:

$$\text{Endereço} = X/4 + Y*128 + \text{Endereço inicial}$$

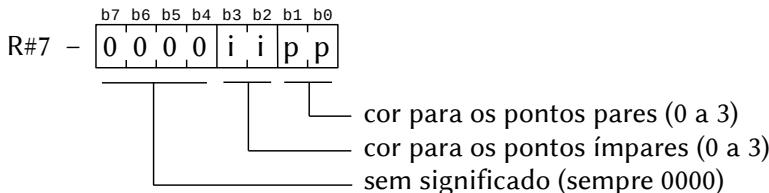
Onde X é a coordenada horizontal e Y a coordenada vertical. Como cada byte representa quatro pontos é necessária mais uma operação para determinar qual par de bits representa cada ponto:

- Se  $X \bmod 4 = 0$ , o ponto é representado pelos bits 7 e 6;
- Se  $X \bmod 4 = 1$ , o ponto é representado pelos bits 5 e 4;
- Se  $X \bmod 4 = 2$ , o ponto é representado pelos bits 3 e 2;
- Se  $X \bmod 4 = 3$ , o ponto é representado pelos bits 1 e 0.

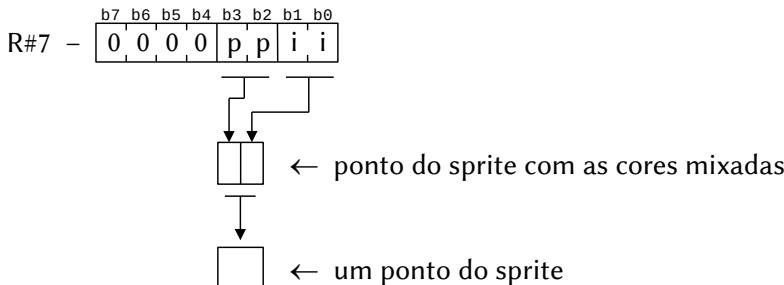
Nesse modo, o número de pontos verticais pode ser aumentado para 212. Para isso, basta setar o bit 7 de R#9, conforme ilustração abaixo.



No modo gráfico 5, há um tratamento especial para a cor da borda e dos sprites. A cor é especificada por 4 bits em R#7, dois para os pontos pares e dois para os ímpares.



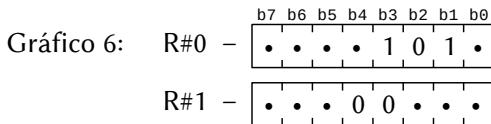
Para os sprites, as cores são especificadas como ilustrado abaixo, onde os códigos para os pontos pares e ímpares estão invertidos em relação à cor da borda:



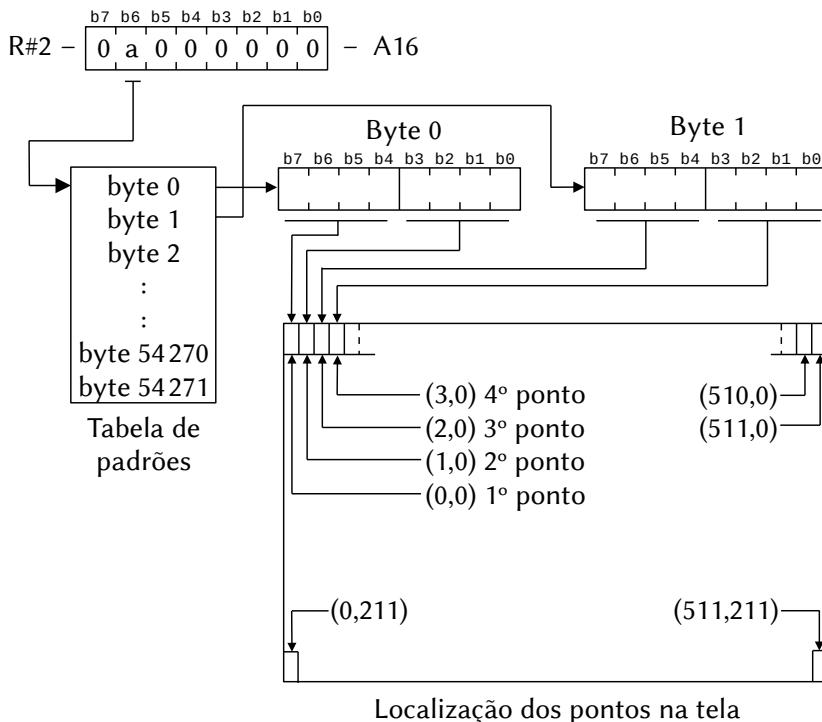
### 5.3.8 – Modo Gráfico 6

- 512 (horizontal) por 212 (vertical) pontos;
- apresenta até 16 cores escolhidas de 512 para cada ponto;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 53 Kbytes (4 bits x 512 pontos x 212 pontos) de memória;
- gráficos bit-mapped de fácil manipulação;
- compatível com Screen 7.

O modo gráfico 6 é selecionado por R#0 e R#1 conforme a ilustração abaixo.



No modo gráfico 6, um byte na tabela de padrões corresponde a dois pontos na tela. Cada ponto é representado por 4 bits; portanto 16 cores podem ser especificadas. O endereço de início da tabela de padrões é especificado por um único bit em R#2; por isso, a tabela geradora de padrões sempre inicia em 00 000H ou 10 000H.

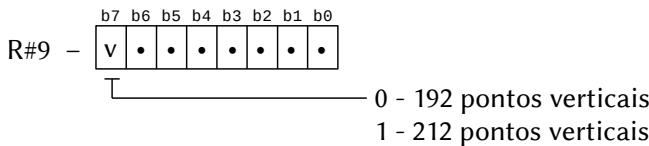


Como há 512 pontos em cada linha horizontal e cada byte representa dois pontos, são necessários 256 bytes para cada linha de tela. O endereço de cada ponto na tabela pode ser calculado pela seguinte expressão:

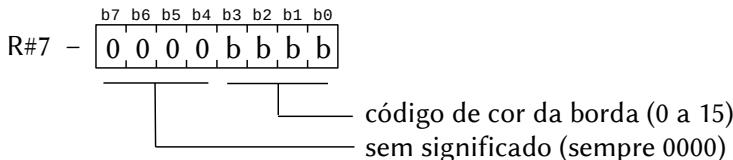
Endereço = X/2 + Y\*256 + Endereço inicial

Onde X é a coordenada horizontal do ponto e Y é a coorde-nada vertical. Se o ponto for par, ele será representado pelos 4 bits mais altos do endereço e se for ímpar, será representado pelos 4 bits mais baixos.

Nesse modo, o número de pontos verticais pode ser aumenta-do para 212. Para isso, basta setar o bit 7 de R#9, conforme ilustração abaixo.



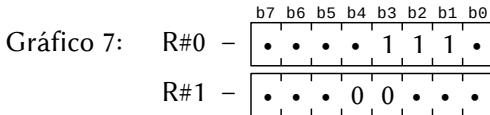
A cor da borda no modo gráfico 4 deve ser especificada nos 4 bits mais baixos de R#7, conforme ilustração abaixo.



### 5.3.9 – Modo Gráfico 7

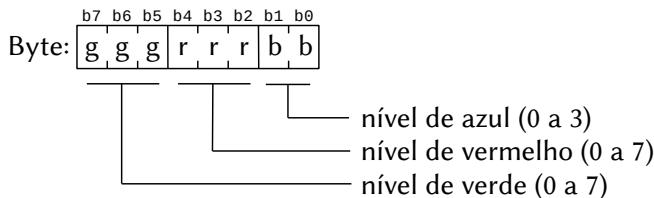
- 256 (norizontal) por 212 (vertical) pontos;
- apresenta até 256 cores simultâneas para cada ponto;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 53 Kbytes (8 bits x 256 pontos x 212 pontos) de memória;
- gráficos bit-mapped de fácil manipulação;
- compatível com Screen 8.

O modo gráfico 7 é selecionado por R#0 e R#1, conforme ilustração abaixo:

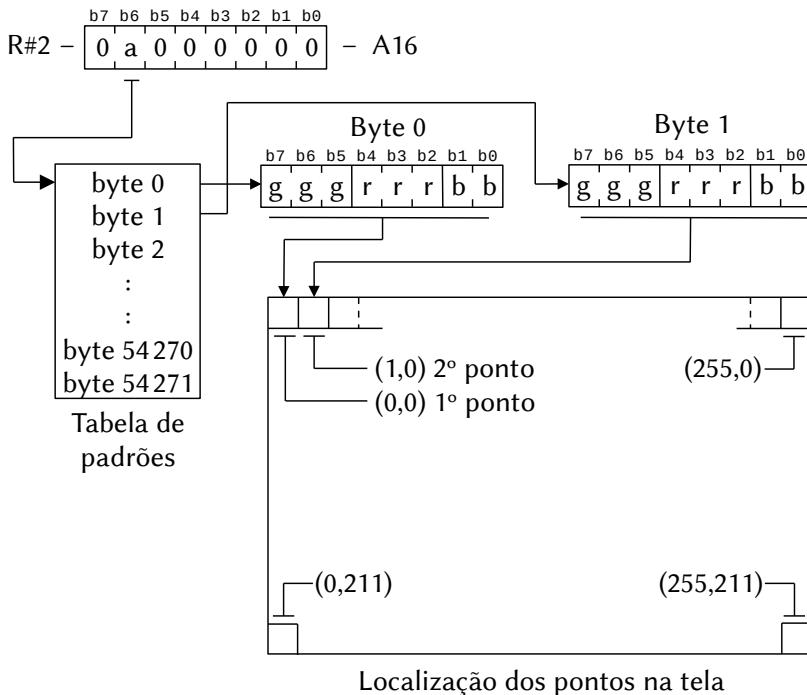


A configuração do modo gráfico 7 é a mais simples de todas; um ponto na tela corresponde a um byte na tabela de padrões, poden-do apresentar até 256 cores simultâneas. Nesse modo não é usada a paleta

de cores, sendo que cada byte de dados reserva 3 bits de intensidade para o verde, 3 bits para o vermelho e 2 bits para o azul.



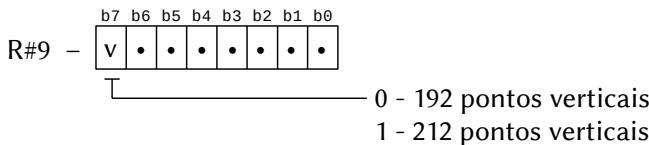
O endereço inicial da tabela de padrões é especificado em R#2 por um único bit; por isso, a tabela de padrões sempre inicia em 00 000H ou 10 000H.



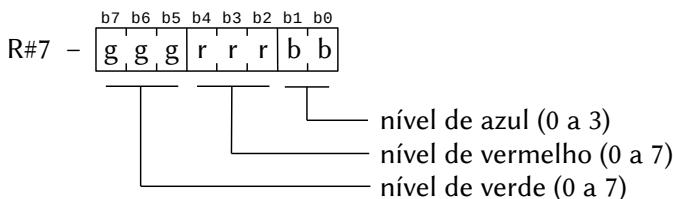
O endereço de cada ponto na tela pode ser calculado pela seguinte expressão:

$$\text{Endereço} = X + Y \cdot 256 + \text{Endereço inicial}$$

Onde X é a coordenada horizontal e Y a vertical. Nesse modo, o número de pontos verticais pode ser aumentado para 212. Para isso, basta setar o bit 7 de R#9, conforme ilustração abaixo.



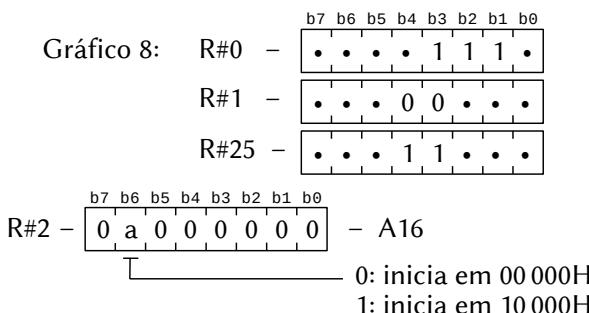
A cor da borda deve ser especificada em R#7 no mesmo formato dos bytes de dados da tela. Todos os bits de R#7 são válidos.



### 5.3.10 – Modo Gráfico 8

- 256 (horizontal) por 212 (vertical) pontos;
- apresenta até 12 499 cores simultâneas na tela;
- cores são especificadas para cada quatro pontos horizontais;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 53 Kbytes (256 pontos x 212 pontos) de memória;
- mapeamento gráfico YJK e bit-mapped mixados;
- compatível com Screens 10 e 11.

O modo gráfico 8 é selecionado por R#0, R#1 e R#25, conforme ilustração abaixo.

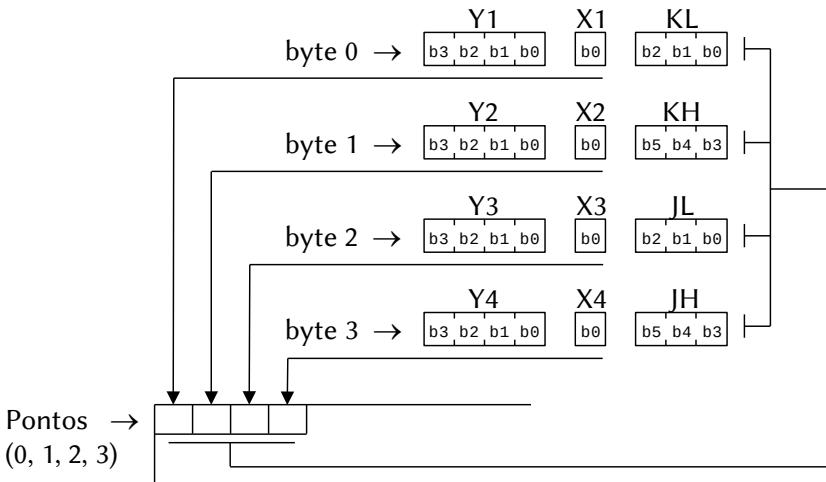


O endereço de início da tabela geradora de padrões é especificado por um único bit de R#2; por isso ela pode iniciar em 00 000H ou 10 000H.

A configuração de pontos e cores do modo gráfico 8 é um pouco complexa. Nos modos já vistos, as cores são dosadas pelo sistema RGB. Nesse novo modo gráfico, o sistema de cores usado é o YJK mixado com o RGB.

Nesse modo, os pontos estão organizados de quatro em quatro na horizontal. Cada grupo de 4 pontos pode ter uma única cor, escolhidas de 4096, com até 16 níveis de brilho para cada ponto individual. Ou então cada ponto pode ter até 16 cores escolhidas de uma paleta de 512, tal qual o modo gráfico 4.

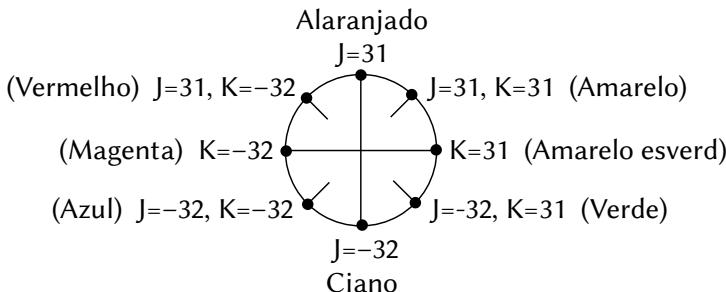
É de se notar que as cores de cada grupo de 4 pontos horizontais não são totalmente independentes quando for usado o sistema YJK. Isso pode causar pequenos borrões na tela; entretanto o método é excelente para representar imagens fotografadas. A estrutura do modo gráfico 8 está ilustrada abaixo.



Quando os bits Xn forem 1, a cor para cada ponto será escolhida da paleta de 512, com os 4 bits Yn respectivos variando de 0 a 15, tal qual as cores são escolhidas para o modo gráfico 4. Nesse caso, os bits J e K são ignorados. Não é obrigatório que todos os bits X de um grupo de 4

pontos sejam iguais, podendo haver mistura nos 4 bytes que compõem o grupo.

Quando os bits  $X_n$  forem 0, será usado o sistema YJK. Nesse sistema, as cores são escolhidas pelos vetores J e K, sendo que J é representado por 6 bits e K por outros 6, conforme o esquema acima. No gráfico abaixo está ilustrado como as cores são representadas por esses vetores.



Os binários correspondentes aos valores mostrados na ilustração são os seguintes:

	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0 →	0	0	0	0	0	0
31 →	0	1	1	1	1	1
-32 →	1	0	0	0	0	0
-1 →	1	1	1	1	1	1

Como há 12 bits para representar a cor (6 bits para J e 6 bits para K), fazemos  $2^{12} = 4096$  cores, que é o número máximo de cores que podem ser definidas. Cada grupo de 4 pontos horizontais só pode ter uma cor escolhida dessas 4096. Entretanto, cada ponto individual desse grupo pode ter uma variação de brilho de 16 níveis, representada pelos bits  $Y_n$ , onde 0000B representa brilho mínimo e 1111B brilho máximo.

Os vetores J e K podem variar de -32 a 31, conforme ilustração da página anterior. Com a combinação dos valores extremos, pode-se

formar as quatro cores principais do sistema YJK: verde, vermelho, azul e amarelo. Utilizando os valores intermediários, podem ser geradas as 4096 cores. A conversão do sistema YJK para o RGB e vice-versa pode ser feita através das seguintes fórmulas:

$$Y = R/4 + G/8 + B/2$$

$$J = R - Y$$

$$K = G - Y$$

$$R = Y + J$$

$$G = Y + K$$

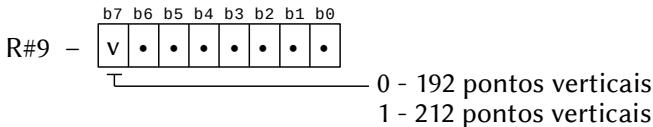
$$B = 5/4 Y - 1/2 J - 1/4 K$$

Um detalhe importante é quanto ao número de cores. Como há 4096 cores e 16 níveis de brilho para cada uma, na verdade são  $16^4 \cdot 4096 = 65\,536$  cores possíveis. Acontece que nesse modo as cores não são totalmente independentes, o que causa uma redução no número de cores apresentadas simultaneamente para 12499.

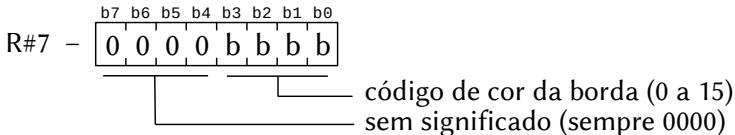
O endereço de cada ponto na tela pode ser calculado pela seguinte expressão:

$$\text{Endereço} = X + Y \cdot 256 + \text{Endereço inicial}$$

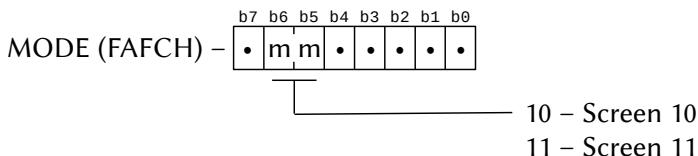
Onde X é a coordenada horizontal e Y a vertical. Nesse modo, o número de pontos verticais pode ser aumentado para 212. Para isso, basta setar o bit 7 de R#9, conforme ilustração abaixo.



A cor da borda no modo gráfico 4 deve ser especificada nos 4 bits mais baixos de R#7, conforme ilustração abaixo, obedecendo a paleta de cores, como no modo gráfico 4.



No BASIC, este modo pode ser selecionado por Screen 10 ou Screen 11. A diferença entre uma e outra está no tratamento dado a elas pelo interpretador BASIC. A Screen 10 é tratada como a Screen 5 (ou pelo sistema RGB) e a Screen 11 é tratada como Screen 8 (ou sistema YJK). Para diferenciar uma da outra, é usada uma flag na variável de sistema MODE (FAFCH), conforme ilustrado abaixo:

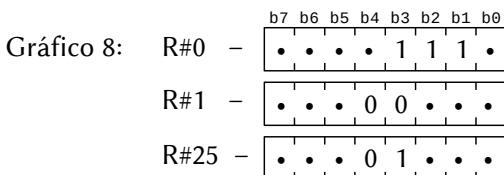


Observe que o BASIC não tem nenhum suporte especial para desenhar no sistema YJK e que a troca entre Screen 10 e Screen 11 não limpa a tela.

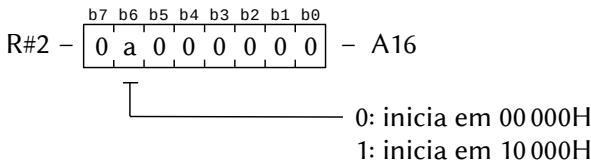
### 5.3.11 – Modo Gráfico 9

- 256 (horizontal) por 212 (vertical) pontos;
- apresenta até 19 268 cores simultâneas na tela;
- cores são especificadas para cada quatro pontos horizontais;
- comandos de hardware de alta velocidade são disponíveis;
- sprites modo 2;
- requer 53 Kbytes (256 pontos x 212 pontos) de memória;
- mapeamento gráfico YJK;
- compatível com Screen 12.

O modo gráfico 9 é selecionado por R#0, R#1 e R#25, conforme ilustrado abaixo:

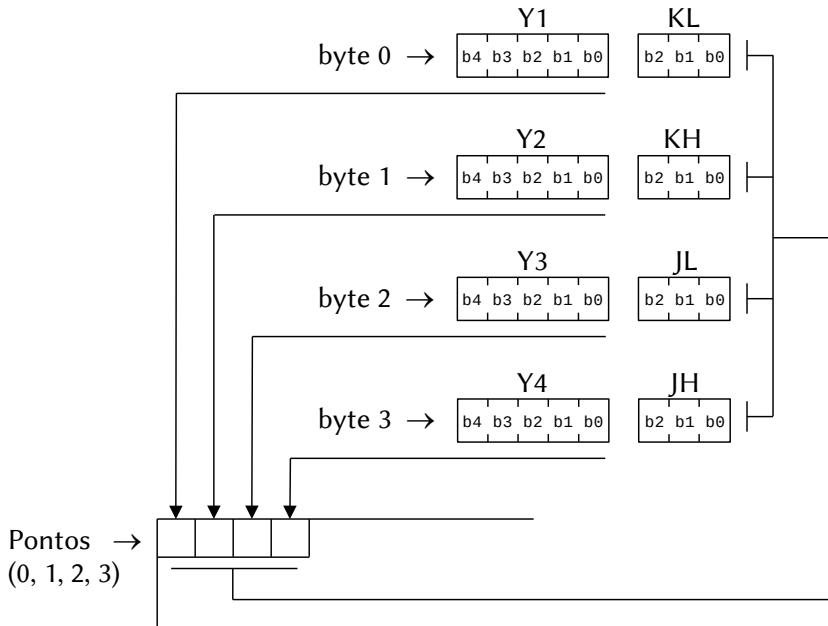


A organização do modo gráfico 9 é semelhante porém mais simples que a do modo gráfico 8. O endereço da tabela geradora de padrões é especificado em um único bit de R#2; por isso, a tabela inicia sempre em 00 000H ou 10 000H.

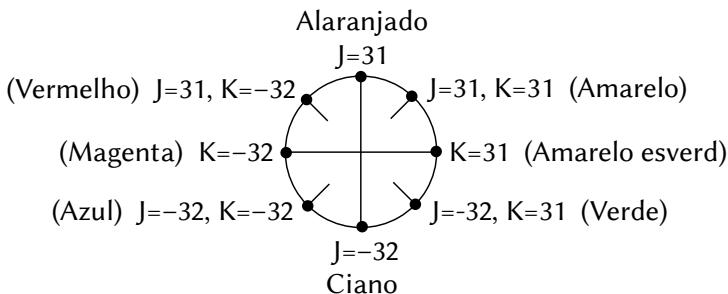


No modo gráfico 9 é usado o sistema YJK puro. Os pontos estão organizados de quatro em quatro na horizontal, sendo que cada grupo de 4 pontos pode ter uma única cor, escolhida de 4096, com até 32 níveis de brilho para cada ponto individual

A cor é escolhida pelos vetores J e K exatamente da mesma forma que no modo gráfico 8. Já o valor de Y, que é o valor do brilho, pode variar de 11111B (maior brilho) até 00000B (menor brilho). Como há 4096 cores e 32 níveis de brilho para cada ponto, na verdade existem 131072 cores possíveis ( $32 * 4096$ ). Entretanto, pelo fato dos pontos não serem totalmente independentes, há uma redução do número de cores que podem ser apresentadas simultaneamente para 19 268. A organização do modo gráfico 9 está ilustrada abaixo.



As cores são escolhidas pelos vetores J e K com os mesmos valores do modo gráfico 8, cuja ilustração está reproduzida abaixo.



Os binários correspondentes aos valores mostrados na ilustração são os seguintes:

	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0 →	0	0	0	0	0	0
31 →	0	1	1	1	1	1
-32 →	1	0	0	0	0	0
-1 →	1	1	1	1	1	1

Como há 12 bits para representar a cor (6 bits para J e 6 bits para K), fazemos  $2^{12} = 4096$  cores, que é o número máximo de cores que podem ser definidas. Cada grupo de 4 pontos horizontais só pode ter uma cor escolhida dessas 4096. Entretanto, no modo 9, cada ponto individual desse grupo pode ter uma variação de brilho de 32 níveis, representada pelos bits Y<sub>n</sub>, onde 00000B representa brilho mínimo e 11111B brilho máximo.

O endereço de cada ponto na tela para o modo gráfico 9 pode ser calculado pela seguinte expressão:

$$\text{Endereço} = X + Y \cdot 256 + \text{Endereço inicial}$$

Onde X é a coordenada horizontal e Y a vertical. O número de pontos verticais deve ser especificado em R#9, conforme já descrito. A cor da borda deve ser especificada em R#7, obedecendo à paleta de cores, tal qual no modo gráfico 4.

### 5.3.12 – Variáveis de sistema dos modos de tela

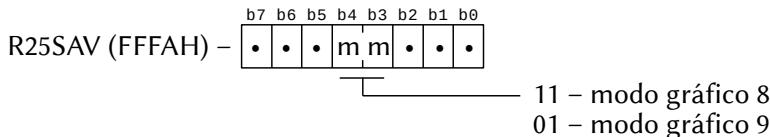
As seguintes variáveis de sistema são usadas pela BIOS e pelo BASIC para diferenciar os modos de tela:

LINL40	(F3AEH, 1)
MODE	(FAFCH, 1)
SCRMOD	(FCAFH, 1)
R25SAV	(FFFAH, 1)

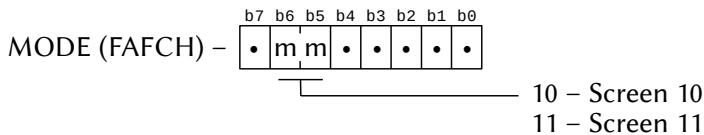
Nos modos multicor e gráficos 1 a 8, não há segredo; o valor da Screen em uso é salvo na variável SCRMOD com o mesmo valor usado pelo interpretador na instrução SCREEN.

Já nos modos texto, é salvo o valor 0 na variável SCRMOD e a largura de tela na variável LINL40 (1 a 40 para modo texto 1 e 41 a 80 para modo texto 2).

Para os modos gráficos 8 e 9 a variável SCRMOD sempre conterá o valor 8 (como na Screen 8). Para diferenciar um do outro, é usada a variável R25SAV, conforme abaixo.



Para diferenciar a Screen 10 da Screen 11, é preciso ler a variável de sistema MODE, como ilustrado abaixo.



### 5.4 – SPRITES (V9918/38/58)

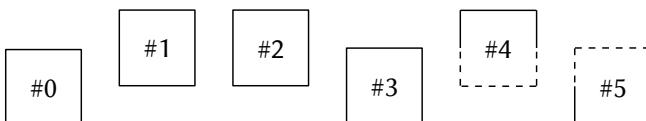
Sprites são padrões ou desenhos móveis e 8x8 ou 16x16 pontos. São usados principalmente em jogos. Existem dois modos de sprites para o MSX2 em diante. O modo 1 é compatível com o TMS9918A do MSX1. O modo 2 inclui algumas funções novas que foram implementadas nos VDPs V9938 e V9958. Os sprites do V9990 não são compatíveis com os sprites dos V9918/38/58 e serão descritos mais adiante.

Até 32 sprites podem ser apresentados simultaneamente na tela. Eles podem ter 2 tamanhos: 8x8 ou 16x16 pontos. Apenas um tamanho pode ser apresentado na tela de cada vez. O tamanho de um ponto do sprite é normalmente do tamanho de um ponto da tela, mas nos modos gráficos 5 e 6 (que tem resolução de 512x212), o tamanho horizontal é de dois pontos da tela, de forma que o tamanho absoluto do sprite é sempre o mesmo em qualquer modo de tela.

O modo do sprite é automaticamente selecionado de acordo com a screen em uso. Para gráfico 1 e 2 e multicor, o modo 1 é selecionado e para os modos gráficos 3 a 9 é selecionado o modo 2. Os modos texto não comportam sprites.

#### 5.4.1 – Sprites modo 1

Os sprites modo 1 são exatamente iguais aos sprites do MSX1. Podem haver na tela até 32 sprites numerados de 0 a 31. Os sprites de número mais baixo têm prioridade de apresentação mais alta. Quando os sprites são colocados na mesma linha horizontal, até 4 sprites de prioridade mais alta são mostrados integralmente. A parte dos sprites com prioridade maior que 3 (5º sprite em diante) coexistente na mesma linha horizontal não é mostrada, conforme a ilustração abaixo.

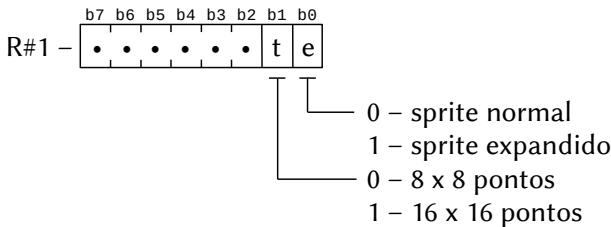


Prioridade dos sprites modo 1

Na ilustração, os sprites #0, #1, #2 e #3 são mostrados inteiros, mas apenas a metade superior do sprite #4 aparece, bem como a metade inferior do sprite #5. A parte tracejada não é exibida.

O tamanho dos sprites, de 8x8 ou 16x16, é selecionado pelo bit 1 de R#1. O tamanho default é de 8x8 pontos.

Os sprites também podem ser expandidos para o dobro do tamanho, na vertical e na horizontal. Nesse caso, cada ponto do sprite corresponde a quatro pontos na tela (exceto nos modos gráficos 5 e 6, onde corresponderá a 8 pontos, sendo 4 na horizontal e 2 na vertical). Essa função é controlada pelo bit 0 de R#1.

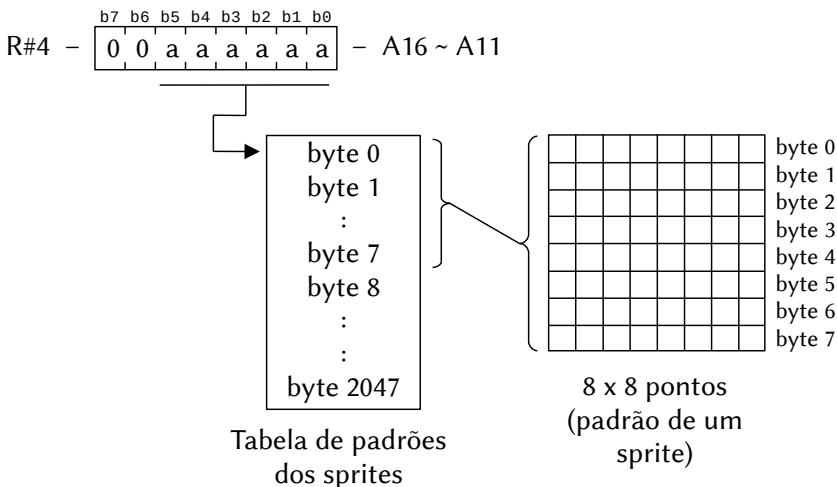


Os padrões dos sprites são definidos na VRAM. Até 256 sprites podem ser definidos se o tamanho for 8 x 8, e até 64 se o tamanho for 16 x 16. Os padrões são numerados de 0 a 255. Para formar um sprite 16 x 16 são usados 4 sprites 8 x 8, na sequência ilustrada abaixo:

01	03
02	04

Sequência de sprites 8 x 8  
para formar um sprite 16 x 16

O endereço da tabela de padrões dos sprites é especificado em R#16. Apenas os 6 bits mais altos são especificados; por isso, a tabela sempre inicia num múltiplo de 2 Kbytes a partir de 00 000H. A tabela de padrões dos sprites tem 2048 bytes de tamanho e reserva 8 bytes para cada sprite 8 x 8, conforme ilustrado abaixo.



Quando o bit correspondente ao ponto do sprite for 0, o ponto respectivo será transparente; quando o bit for 1, o ponto terá a cor especificada na Tabela de Atributos dos Sprites. Essa tabela tem 128 bytes e reserva 4 bytes para cada sprite a ser apresentado na tela, num máximo de 32 sprites simultâneos. Ela inicia no endereço apontado por R#11 e R#5. Como apenas os 7 bits mais altos são especificados, a tabela sempre inicia num múltiplo de 1 Kbyte a partir de 00 000H. Os quatro bytes reservados pela tabela para cada sprite contêm os seguintes dados:

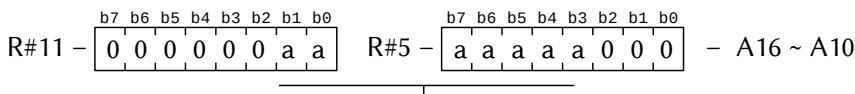
**Coordenada Y:** Especifica a coordenada vertical do canto superior esquerdo do sprite. A linha do topo da tela não é 0, mas 255. Colocando esse valor em 208 (D0H), todos os sprites de prioridade menor não são mostrados.

**Coordenada X:** Especifica a coordenada horizontal vertical do canto superior esquerdo do sprite.

**Nº do padrão:** Especifica qual padrão da tabela geradora de padrões dos sprites será apresentado.

**Código de cor:** Especifica a cor, de acordo com a paleta, dos bits setados em 1 da tabela de padrões.

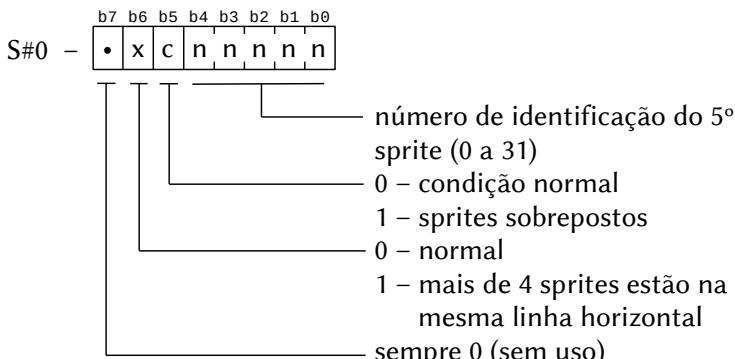
**EC:** Setando este bit em 1, o sprite respectivo será deslocado 32 pontos à esquerda da coordenada especificada.



	b7 b6 b5 b4 b3 b2 b1 b0	Atributos do sprite #0
0	Coordenada Y (0 a 255)	
1	Coordenada X (0 a 255)	
2	Número do padrão (0 a 255)	
3	EC   0   0   0   Código de cor	
4	:	
:	:	
127		← Atributos do sprite #31

Tabela de atributos dos sprites

Quando dois sprites se sobrepõem na tela, o bit 5 de S#0 é setado, informando a situação. A informação de sobreposição ou conflito acontece somente quando os bits 1 se encontram, ou seja, quando a parte “desenhada” dos sprites se sobrepõem. Quando mais de quatro sprites são colocados na mesma linha horizontal, o bit 6 de S#0 é setado e o número do 5º sprite é colocado nos 5 bits mais baixos de S#0.

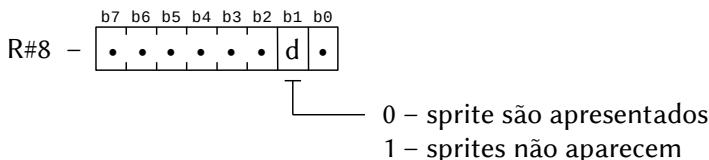


#### 5.4.2 – Sprites modo 2

Os sprites modo 2 foram adicionados aos VDPs V9938/9958 trazendo novas características e maior flexibilidade que os sprites modo 1.

O número de sprites que podem ser apresentados simultaneamente é de 32, e até 8 sprites podem ser colocados na mesma linha horizontal. Os sprites de número menor têm prioridade de apresentação maior, como no modo 1. O tamanho dos sprites (8x8 ou 16x16) e a expansão para o dobro do tamanho são selecionados da mesma forma que para os sprites modo 1.

Os sprites modo 2 dispõem de uma função de liga-desliga a apresentação na tela, controlada pelo bit 1 de R#8. Quando esse bit for 0, os sprites aparecerão normalmente na tela, mas quando for 1 nenhum sprite aparecerá.



A tabela geradora de padrões é setada da mesma forma que para o modo 1, mas a tabela de atributos sofreu mudanças. Nos sprites modo 2, uma cor diferente pode ser especificada para cada linha do sprite. Essa informação é armazenada na Tabela de Cores dos Sprites, que é independente da tabela de atributos. A tabela de atributos armazena as seguintes informações:

**Coordenada Y:** Coordenada vertical do sprite. A linha do topo da tela não é 0, mas 255. Colocando este valor em 208 (D0H), todos os sprites com prioridade maior não são mostrados. Colocando em 216, são os sprites de prioridade menor que não são mostrados.

**Coordenada X:** Especifica a coordenada horizontal do sprite.

**Nº do padrão:** Especifica qual padrão da tabela geradora de padrões será apresentado.

A tabela de cores dos sprites é setada automaticamente 512 bytes antes do endereço inicial da tabela de atributos. Ela aloca 16 bytes para cada padrão, e cada linha de cada sprite contém os seguintes dados:

- Cor:** Cor da linha respectiva, podendo variar de 0 a 15, obedecendo à paleta de cores.
- EC:** Quando esse bit for 1, a linha respectiva será deslocada 32 pontos à esquerda da coordenada especificada.
- CC:** Quando esse bit for 1, o sprite respectivo terá a mesma prioridade que os sprites de prioridade maior. Quando sprites de mesma prioridade se sobrepõem, é feita uma operação lógica OR entre as cores dos sprites para determinar a nova cor. Nesse caso, a sobreposição não causa conflito e não é detectada como colisão de sprites.
- IC:** Quando esse bit for 1, a linha respectiva não causará conflito quando ocorrer sobreposição com outros sprites.

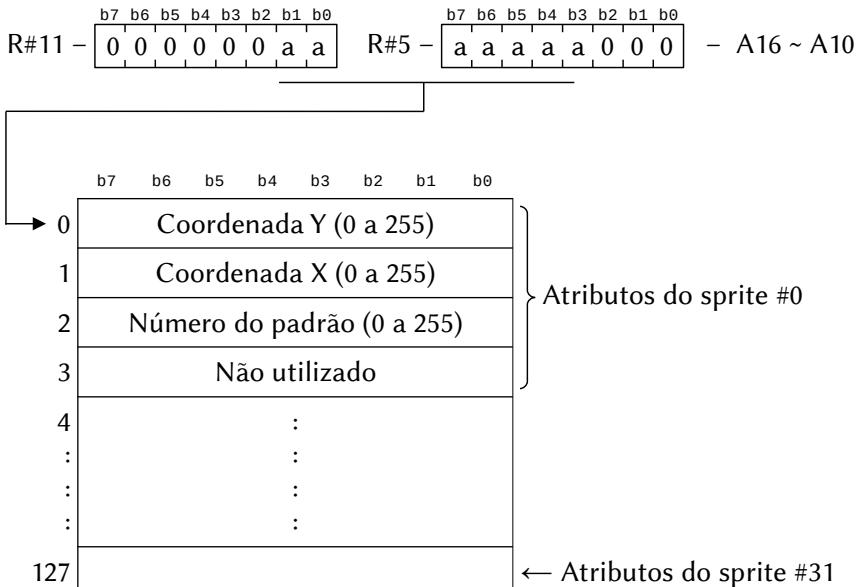


Tabela de atributos dos sprites

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
0	EC	CC	IC	0	Código de cor				
1	EC	CC	IC	0	Código de cor				
:	:	:	:	:	:				
15	EC	CC	IC	0	Código de cor				
:	:	:	:	:	:				
:	:	:	:	:	:				
496	EC	CC	IC	0	Código de cor				
497	EC	CC	IC	0	Código de cor				
:	:	:	:	:	:				
511	EC	CC	IC	0	Código de cor				

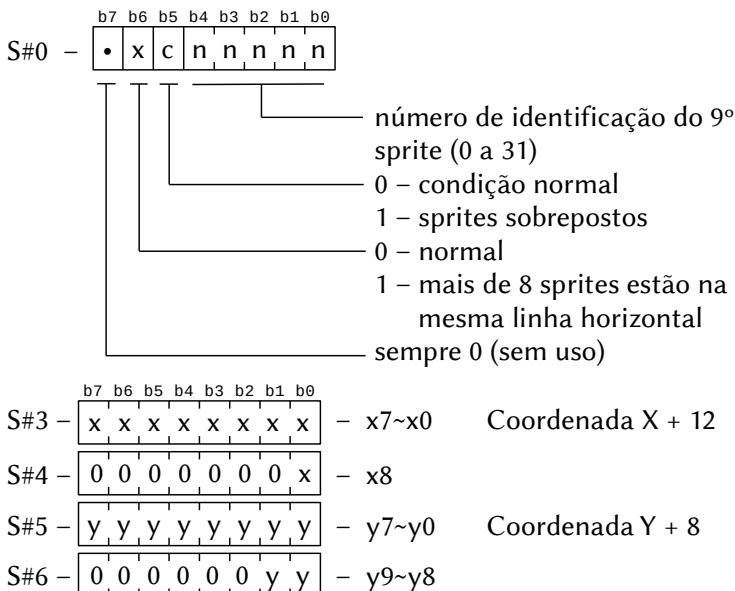
1<sup>a</sup> linha      2<sup>a</sup> linha      16<sup>a</sup> linha      1<sup>a</sup> linha      2<sup>a</sup> linha      16<sup>a</sup> linha      Sprite #0      Sprite #31

Tabela de cores dos sprites

A sobreposição ou conflito de sprites modo 2 é detectada quando a cor do ponto do sprite não for transparente e os bits CC da tabela de cores do sprite for 0. Quando a sobreposição é detectada, o bit 5 de S#0 é setado em 1 e a coordenada da sobreposição é colocada em S#3 a S#6. A coordenada devolvida por esses registros não é aquela onde o conflito ocorreu. Para obter as coordenadas exatas, a seguinte expressão deve ser usada:

$$\text{Coordenada } X = (S\#3 \text{ e } S\#4) - 12$$

$$\text{Coordenada } Y = (S\#5 \text{ e } S\#6) - 8$$



Quando mais de 8 sprites são colocados na mesma linha horizontal, o bit 6 de S#0 é setado em 1 e o número do plano do sprite de menor prioridade (9º sprite) é colocado nos 5 bits mais baixos de S#0, conforme ilustração acima.

## 5.5 – COMANDOS DO VDP (V9938/58)

O MSX-VIDEO pode executar algumas operações gráficas básicas, denominadas Comandos do VDP. São executadas por hardware e estão disponíveis para os modos gráficos 4 a 9. Quando os comandos do VDP são executados, a localização dos pontos de início e destino são representadas por coordenadas (X, Y) e não há divisão de páginas de vídeo;

os 128 Kbytes de VRAM são tratados como um único bloco, conforme ilustração abaixo.

Gráfico 4 (Screen 5)		Endereço	Gráfico 5 (Screen 6)	
(0,0)	(255,0)	00000H	(0,0)	(511,0)
Página 0			Página 0	
(0,255)	(255,255)	07FFFH	(0,255)	(511,255)
(0,256)	(255,256)		(0,256)	(511,256)
Página 1			Página 1	
(0,511)	(255,511)	0FFFH	(0,511)	(511,511)
(0,512)	(255,512)		(0,512)	(511,512)
Página 2			Página 2	
(0,767)	(255,767)	17FFFH	(0,767)	(511,767)
(0,768)	(255,768)		(0,768)	(511,768)
Página 3			Página 3	
(0,1023)	(255,1023)	1FFFH	(0,1023)	(511,1023)

Gráficos 7 a 9 (Screens 8 a 12)		Endereço	Gráfico 6 (Screen 7)	
(0,0)	(255,0)	00000H	(0,0)	(511,0)
Página 0			Página 0	
(0,255)	(255,255)	0FFFH	(0,255)	(511,255)
(0,256)	(255,256)		(0,256)	(511,256)
Página 1			Página 1	
(0,511)	(255,511)	1FFFH	(0,511)	(511,511)

### 5.5.1 – Descrição dos comandos do VDP

Existem 12 comandos disponíveis no MSX-VIDEO. Estes comandos estão resumidos na tabela abaixo.

Tipo de Comando	Memônico	Origem	Destino	Unidade	Código R#46-4msb
Movimentos Rápidos	HMMC	CPU	VRAM	bytes	1111
	YMMM	VRAM	VRAM	bytes	1110
	HMMR	VRAM	VRAM	bytes	1101
	HMMV	VDP	VRAM	bytes	1100

Movimentos Lógicos	LMMC LMCM LMMM LMMV	CPU VRAM VRAM VDP	VRAM CPU VRAM VRAM	pontos pontos pontos pontos	1011 1010 1001 1000
Linha Procura	LINE SRCH	VDP	VRAM	pontos	0111 0110
Pset Point	PSET POINT	VDP	VRAM	pontos	0101 0100
Reservado	-	-	-	-	0011
Reservado	-	-	-	-	0010
Reservado	-	-	-	-	0001
Parada	STOP	-	VDP	-	0000

Quando um dado é escrito em R#46 (registraror de comando), o VDP começa a executar o comando e seta o bit 0 (CE / command execute) do registraror de estado S#2. Os parâmetros necessários de-vem ser colocados em R#32 a R#45 antes do comando ser executado. Quando a execução do comando termina, o bit 0 se S#2 é resetado (0). Para interromper a execução de um comando, pode ser usado o co-mando de parada (0000B). Os comandos do VDP só funcionam nos modos gráficos 4 a 9, mas nos modos 8 e 9 devem ser usados com cautela, pois a tela pode borrar, já que nesses modos os pontos estão organizados em blocos de quatro na horizontal.

### 5.5.2 – Operações lógicas

Quando os comandos são executados, várias operações lógi-cas podem ser feitas entre a VRAM e um dado especificado. Essas operações estão descritas na tabela abaixo.

NOME	OPERAÇÃO	R#46-4lsb
IMP	DC = SC	0000
AND	DC = SC and DC	0001
OR	DC = SC or DC	0010
XOR	DC = SC xor DC	0011
NOT	DC = not (SC)	0100

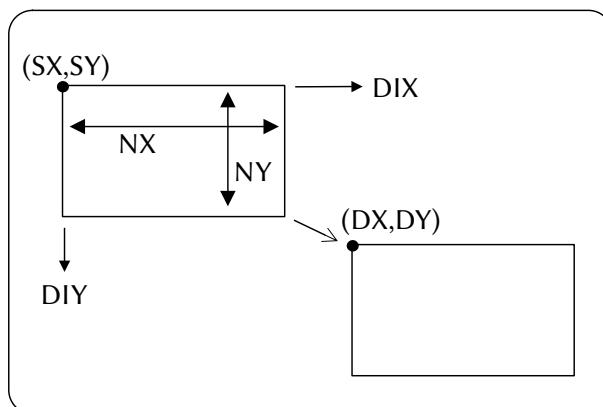
TIMP	Se $SC=0$ , $DC = DC$ senão $DC = SC$	1000
TAND	Se $SC=0$ , $DC = DC$ senão $DC = SC \text{ and } DC$	1001
TOR	Se $SC=0$ , $DC = DC$ senão $DC = SC \text{ or } DC$	1010
TXOR	Se $SC=0$ , $DC = DC$ senão $DC = SC \text{ xor }$	1011
TNOT	Se $SC=0$ , $DC = DC$ senão $DC = \text{not}(SC)$	1100

Na tabela, SC representa o código de cor da origem e DC o código de cor do destino. IMP, AND, OR, XOR e NOT são as operações lógicas possíveis. Nas operações com nomes precedidos por “T”, os pontos de origem que tiverem a cor 0 não serão objeto de operação lógica no destino. Usando esse recurso, somente as porções coloridas são sobrepostas. Esse recurso é especialmente efetivo para animações.

### 5.5.3 – Especificação de áreas

Os comandos de movimentação de áreas transferem os dados dentro de uma área especificada por um retângulo. A área a ser transferida é especificada por um vértice, a partir do qual são informados os tamanhos dos lados do retângulo, juntamente com a direção em que os dados serão transferidos e as coordenadas de destino.

SX e SY são as coordenadas de origem; NX e NY são o comprimento de cada lado do retângulo em pontos e DIX e DIY especificam a direção em que os dados serão transferidos e dependem do tipo de comando. DX e DY especificam as coordenadas de destino.



Especificação de áreas para os comandos do VDP

### 5.5.4 – Usando os comandos do VDP

Os comandos do VDP são classificados em três tipos: comandos de transferência rápida (high speed transfer), comandos de transferência lógica (logical transfer) e comandos de desenho. Eles devem ser acessados por via direta; por isso, deve ser tomado um certo cuidado com a sincronização, aguardando que o VDP esteja pronto. Deve haver uma pausa de 8 µs entre acessos consecutivos (para tanto, pode ser usado um loop com a instrução OUTI, e não a instrução OTIR, para uma máquina MSX padrão a 3,58 MHz). No MSX turbo R, o MSX Engine S1990 gera pausas de 8 µs a cada byte escrito em uma das portas do VDP, do modo que não é necessário se preocupar com a sincronização nessas máquinas. É importante observar, entretanto, que, uma vez escrito o dado, ele fica armazenado no S1990, liberando a CPU. Mas se outro dado for escrito logo em seguida, será gerado um HALT até atingir os 8 µs necessários. Essa característica pode ser usada para que o R800 processe dados entre acessos consecutivos ao VDP<sup>12</sup>.

Também deve ser aguardado que o VDP termine de executar o comando para enviar outro comando ou dado, o que deve ser feito lendo o bit CE do registrador de estado S#2. Para tanto, a seguinte rotina pode ser usada:

```

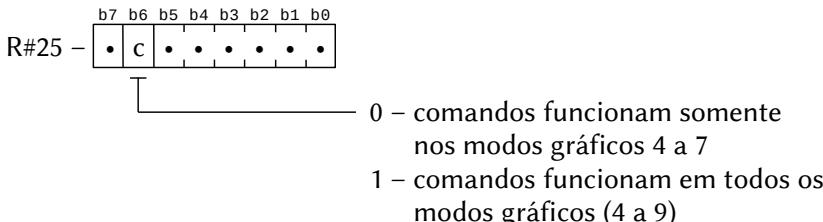
WAIT:   LD    A,2
        CALL  STATUS
        AND   1
        JR    NZ,WAIT
        XOR   A
        CALL  STATUS
        RET
        ;
STATUS: OUT   099H,A
        LD    A,08FH
        OUT   099H,A
        IN    A,099H
        RET

```

---

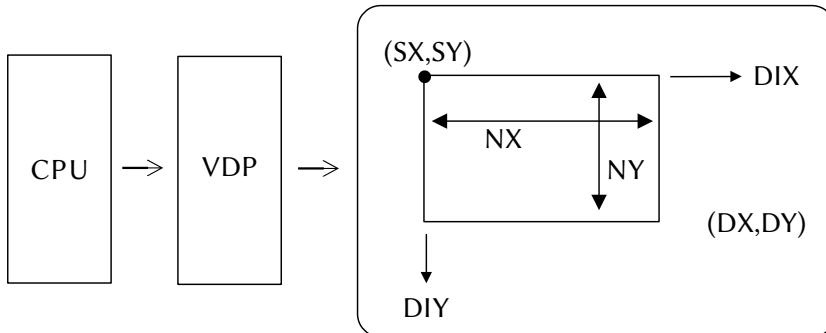
12 Uma pausa de 8 µs corresponde a 57 ciclos T do R800, o que permite bastante processamento entre acessos. Observe que não é obrigatório que o R800 termine o processamento em 57 ciclos; pode usar quantos forem necessários, apenas com a consequência de tornar mais lento o acesso ao VDP.

No caso de usar comandos no V9958, há uma peculiaridade. Deve ser observado o bit 6 de R#25 (CMD). Se esse bit for 0, os comandos só funcionarão nos modos gráficos 4 a 7. É o valor default. Para que os comandos funcionem nos modos gráficos 8 e 9 é necessário setar esse bit em 1. Nesse caso, os comandos funcionarão da mesma forma que para o modo gráfico 7.



#### 5.5.4.1 – HMMC (Transf. em bytes – CPU → VRAM)

Nesse comando, os dados são transferidos da CPU para uma área especificada na VRAM. Operações lógicas não são possíveis; os dados são transferidos em bytes.



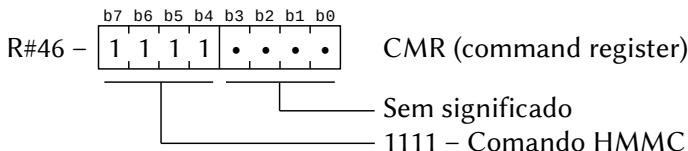
Os parâmetros mostrados na tabela abaixo devem ser colocados nos registradores apropriados. Nesse ponto, deve ser escrito, em R#44, o primeiro byte de dados a ser transferido da CPU para a VRAM. Para executar o comando, é necessário escrever o código de comando F0H em R#46 e o byte contido em R#44 será escrito na VRAM. Então, o VDP esperará o segundo byte de dados, que também deverá ser escrito em R#44, e assim sucessivamente. O byte só será transferido depois que o VDP recebê-lo (se o bit TR de S#2 for 1). Quando o bit CE de S#2 for 0, significa que todos os bytes de dados foram transferidos.

R#36 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal da tela para onde os bytes de dados serão transferidos (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													

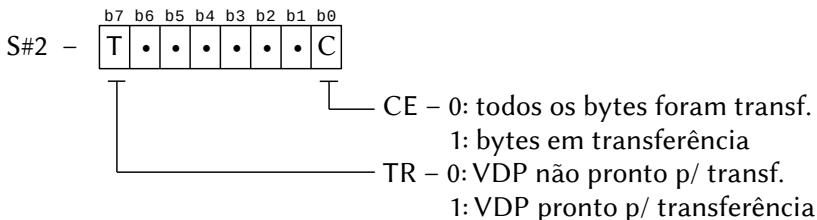
Para os modos gráficos 4 e 6 o bit de endereço mais baixo (b0) não é válido visto que cada byte transfere dois pontos. Para o modo gráfico 5, os dois bits mais baixos (b1 e b0) não são válidos porque cada byte transfere 4 pontos. Para os modos gráficos 7, 8 e 9 todos os bits são válidos.

R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical da tela para onde os bytes de dados serão transferidos (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#40 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	SX	Número de pontos a transferir na direção horizontal (0 a 511) Obs. em R#36/R#37
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	SY	Número de pontos a transferir na direção vertical (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x = 2n</td><td></td><td>x = 2n+1</td><td></td><td></td><td></td><td></td><td></td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x = 2n		x = 2n+1						CLR Gráficos 4 e 6 4: n = 0 a 127; 6: n = 0 a 255	Formato dos dados a serem transferidos	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x = 2n		x = 2n+1																		
	<table border="1"><tr><td>x=4n</td><td>x=4n+1</td><td>x=4n+2</td><td>x=4n+3</td></tr></table>	x=4n	x=4n+1	x=4n+2	x=4n+3	CLR Gráfico 5 n = 0 a 127														
x=4n	x=4n+1	x=4n+2	x=4n+3																	
	<table border="1"><tr><td colspan="8">x = n</td></tr></table>	x = n								CLR Gráficos 7, 8 e 9 n = 0 a 511										
x = n																				
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>•</td><td>•</td><td>m</td><td>•</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	m	•	y	x	•	•	ARGT (argument register)		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
•	•	m	•	y	x	•	•													
		DIX (direção X) 0 – para a direita 1 – para a esquerda																		
		DIY (direção Y) 0 – para baixo 1 – para cima																		
		MXD (memória de destino) 0 – VRAM 1 – VRAM expandida																		

Para executar o comando HMMC, basta escrever o valor F0H em R#46:

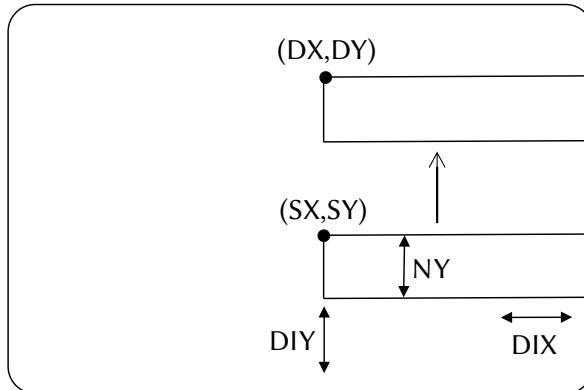


O estado da transferência pode ser lido em S#2:



#### 5.5.4.2 – YMMM (Transf. em bytes – VRAM direção Y)

Nesse comando, os dados de uma área especificada da VRAM são transferidos para outra área da VRAM. Os dados são transferidos apenas na direção Y (vertical), conforme ilustração abaixo:



Os registradores devem ser carregados de acordo com a ilustração abaixo.

R#34 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7~y0</sub>	SY	Coordenada vertical de origem (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y <sub>9~y8</sub>										
0	0	0	0	0	0	y	y													
R#36 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x <sub>7~x0</sub>	DX	Coordenada horizontal de origem e de destino (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x <sub>8</sub>										
0	0	0	0	0	0	0	x													

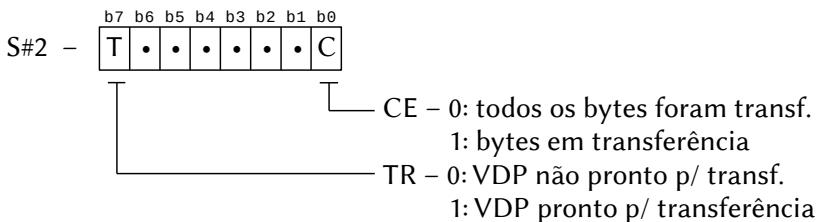
Para os modos gráficos 4 e 6 o bit de endereço mais baixo (b<sub>0</sub>) não é válido visto que cada byte transfere dois pontos. Para o modo gráfico 5, os dois bits mais baixos (b<sub>1</sub> e b<sub>0</sub>) não são válidos porque cada byte transfere 4 pontos. Para os modos gráficos 7, 8 e 9 todos os bits são válidos.

R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7~y0</sub>	DY	Coordenada vertical de destino (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y <sub>9~y8</sub>										
0	0	0	0	0	0	y	y													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7~y0</sub>	SY	Número de pontos a transferir na direção vertical (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y <sub>9~y8</sub>										
0	0	0	0	0	0	y	y													
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>•</td><td>•</td><td>m</td><td>•</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	m	•	y	x	•	•	ARGT (argument register)		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
•	•	m	•	y	x	•	•													
		DIX (direção X) 0 – para a direita 1 – para a esquerda																		
		DIY (direção Y) 0 – para baixo 1 – para cima																		
		MXD (memória de destino) 0 – VRAM 1 – VRAM expandida																		

Para executar o comando YM**MM**, basta escrever o valor E0H em R#46:

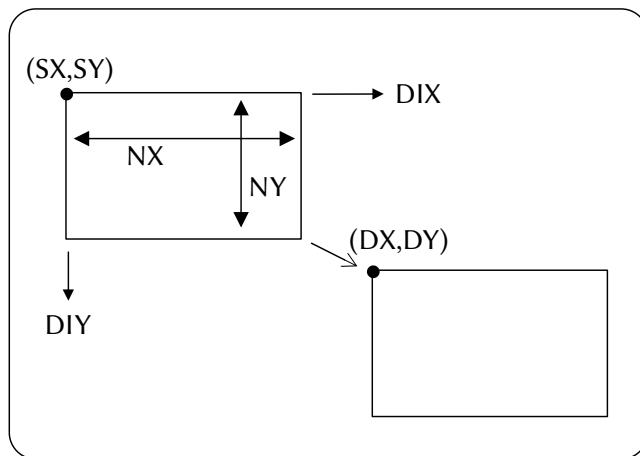
R#46 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	1	0	•	•	•	•	CMR (command register)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
1	1	1	0	•	•	•	•											
		Sem significado 1110 – Comando YM <b>MM</b>																

O estado da transferência pode ser lido em S#2:



#### 5.5.4.3 – HMMM (Transf. em bytes – VRAM → VRAM)

Nesse comando, os dados são transferidos de uma área da VRAM para outra. Os dados são transferidos em áreas retangulares, conforme ilustração abaixo:



Os seguintes registradores devem ser carregados:

R#32 – 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
x	x	x	x	x	x	x	x

 – x<sub>7</sub>~x<sub>0</sub>      SX      Coordenada horizontal de origem (0 a 511)

R#33 – 

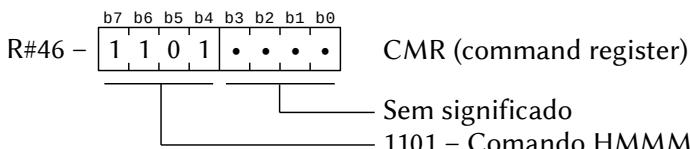
0	0	0	0	0	0	0	x
---	---	---	---	---	---	---	---

 – x<sub>8</sub>

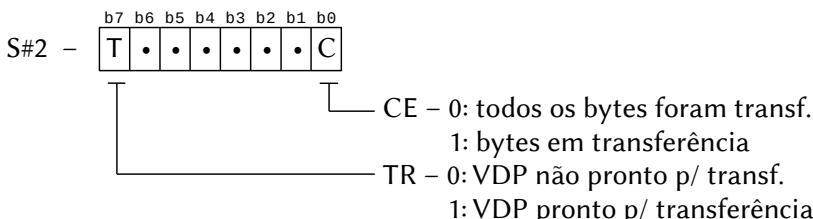
Para os modos gráficos 4 e 6 o bit de endereço mais baixo (b0) não é válido visto que cada byte transfere dois pontos. Para o modo gráfico 5, os dois bits mais baixos (b1 e b0) não são válidos porque cada byte transfere 4 pontos. Para os modos gráficos 7, 8 e 9 todos os bits são válidos.

R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8	SY Coordenada vertical de origem (0 a 1023)								
0	0	0	0	0	0	y	y												
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX Coordenada horizontal de destino (0 a 511) * veja observação em R#32/R#33.
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8									
0	0	0	0	0	0	0	x												
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY Coordenada vertical de destino (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8									
0	0	0	0	0	0	y	y												
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	SX Número de pontos a transferir na direção horizontal (0 a 511) * Obs em R#32/R#33
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8									
0	0	0	0	0	0	0	x												
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY Número de pontos a transferir na direção vertical (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8									
0	0	0	0	0	0	y	y												
R#45 -	<table border="1"><tr><td>•</td><td>•</td><td>d</td><td>s</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	•	•	d	s	y	x	•	•	ARGT (argument register)									
•	•	d	s	y	x	•	•												
		DIX (direção X) 0 – para a direita 1 – para a esquerda																	
		DIY (direção Y) 0 – para baixo 1 – para cima																	
		MXS (memória de origem) 0 – VRAM 1 – VRAM expandida																	
		MXD (memória de destino) 0 – VRAM 1 – VRAM expandida																	

Para executar o comando HMMM, basta escrever o valor D0H em R#46:

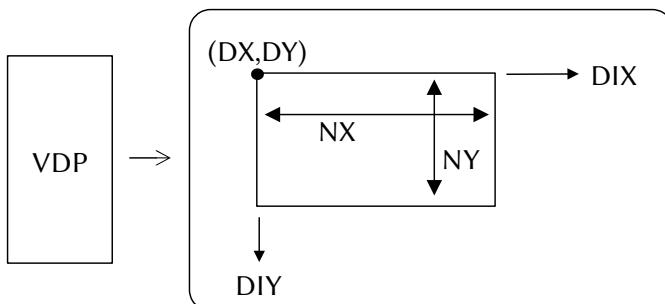


O estado da transferência pode ser lido em S#2:



#### 5.5.4.4 – HMMV (Desenha retângulo em bytes)

Nesse comando, cada byte de dados especificado é desenhado na VRAM com o código de cor respectivo, conforme a ilustração.



Os seguintes registradores devem ser carregados:

R#36 -

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
x	x	x	x	x	x	x	x

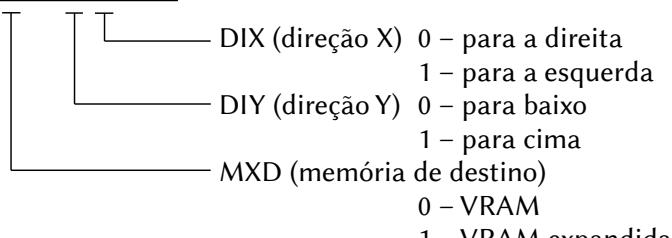
- x7~x0      DX      Coordenada horizontal da tela onde a pintura irá começar (0 a 511)

R#37 -

0	0	0	0	0	0	0	x
---	---	---	---	---	---	---	---

- x8

Para os modos gráficos 4 e 6 o bit de endereço mais baixo (b0) não é válido visto que cada byte transfere dois pontos. Para o modo gráfico 5, os dois bits mais baixos (b1 e b0) não são válidos porque cada byte transfere 4 pontos. Para os modos gráficos 7, 8 e 9 todos os bits são válidos.

R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7</sub> ~y <sub>0</sub>	DY	Coordenada vertical da tela onde a pintura irá começar (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y <sub>9</sub> ~y <sub>8</sub>										
0	0	0	0	0	0	y	y													
R#40 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x <sub>7</sub> ~x <sub>0</sub>	SX	Número de pontos a pintar na direção horizontal (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x <sub>8</sub>		Obs. em R#36/R#37								
0	0	0	0	0	0	0	x													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7</sub> ~y <sub>0</sub>	SY	Número de pontos a pintar na direção vertical (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y <sub>9</sub> ~y <sub>8</sub>										
0	0	0	0	0	0	y	y													
R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x = 2n</td><td></td><td>x = 2n+1</td><td></td><td></td><td></td><td></td><td></td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x = 2n		x = 2n+1						CLR	Gráficos 4 e 6 4: n = 0 a 127; 6: n = 0 a 255	Formato dos dados de cores para a pintura
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x = 2n		x = 2n+1																		
	<table border="1"><tr><td>x=4n</td><td>x=4n+1</td><td>x=4n+2</td><td>x=4n+3</td></tr></table>	x=4n	x=4n+1	x=4n+2	x=4n+3	CLR	Gráfico 5 n = 0 a 127													
x=4n	x=4n+1	x=4n+2	x=4n+3																	
	<table border="1"><tr><td>x = n</td></tr></table>	x = n	CLR	Gráficos 7, 8 e 9 n = 0 a 511																
x = n																				
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>•</td><td>•</td><td>m</td><td>•</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	m	•	y	x	•	•		ARGT (argument register)	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
•	•	m	•	y	x	•	•													
																				
			DIX (direção X) 0 – para a direita 1 – para a esquerda																	
			DIY (direção Y) 0 – para baixo 1 – para cima																	
			MXD (memória de destino) 0 – VRAM 1 – VRAM expandida																	

Para executar o comando HMMV, basta escrever o valor C0H em																		
R#46:																		
R#46 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	0	0	•	•	•	•	CMR (command register)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
1	1	0	0	•	•	•	•											
		Sem significado 1100 – Comando HMMV																

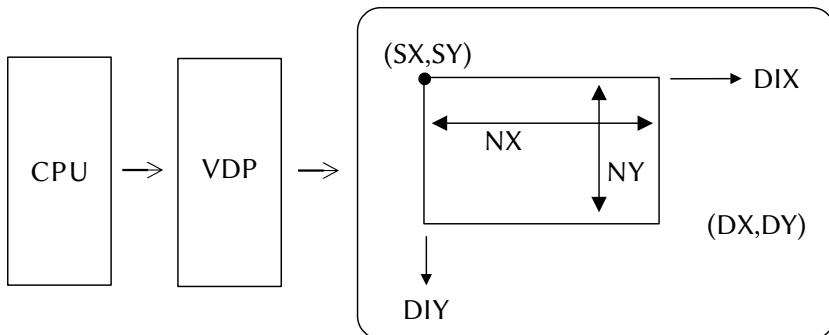
O estado da pintura pode ser lido em S#2:

S#2 -	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	C
	•	•	•	•	•	•	•	•	C

CE – 0: retângulo já desenhado  
1: desenho em progresso

#### 5.5.4.5 – LMMC (Transferência lógica – CPU → VRAM)

Nesse comando, os bytes de dados são transferidos da CPU para uma área específica da VRAM em pontos. Operações lógicas durante a transferência são possíveis. Nos comandos de transferência lógica os dados são transferidos em pontos e um byte é requerido para cada ponto em todos os modos de tela. Por isso, todos os bits das coordenadas horizontais são válidos. O código da operação lógica deve ser especificado nos 4 bits mais baixos do registrador R#46.



Os seguintes registradores devem ser carregados:

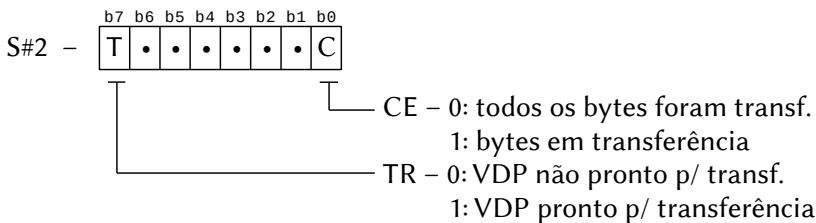
R#36 -	<table border="1"> <tr> <th>b<sub>7</sub></th><th>b<sub>6</sub></th><th>b<sub>5</sub></th><th>b<sub>4</sub></th><th>b<sub>3</sub></th><th>b<sub>2</sub></th><th>b<sub>1</sub></th><th>b<sub>0</sub></th></tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal da tela para onde os bytes de dados serão transferidos (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td> </tr> </table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#38 -	<table border="1"> <tr> <th>b<sub>7</sub></th> <th>b<sub>6</sub></th> <th>b<sub>5</sub></th> <th>b<sub>4</sub></th> <th>b<sub>3</sub></th> <th>b<sub>2</sub></th> <th>b<sub>1</sub></th> <th>b<sub>0</sub></th> </tr> <tr> <td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td> </tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical da tela para onde os bytes de dados serão transferidos (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td> </tr> </table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													

R#40 -		- x7~x0	NX	Número de pontos a transferir na direção horizontal (0 a 511)
R#41 -		- x8		
R#42 -		- y7~y0	NY	Número de pontos a transferir na direção vertical (0 a 1023)
R#43 -		- y9~y8		
R#44 -		CLR Gráficos 4 e 6		Formato dos dados de cores a serem transferidos
		CLR Gráfico 5		
		CLR Gráficos 7, 8 e 9		
R#45 -		ARGT (argument register)		
		DIX (direção X) DIY (direção Y) MXD (memória de destino)	0 – para a direita 1 – para a esquerda 0 – para baixo 1 – para cima 0 – VRAM 1 – VRAM expandida	

Para executar o comando LMMC, deve ser escrito o valor 1011B nos quatro bits mais altos de R#46 e o código de operação lógica nos quatro bits mais baixos.

R#46 -		CMR (command register)
		Código de operação lógica 1011 – Comando LMMC

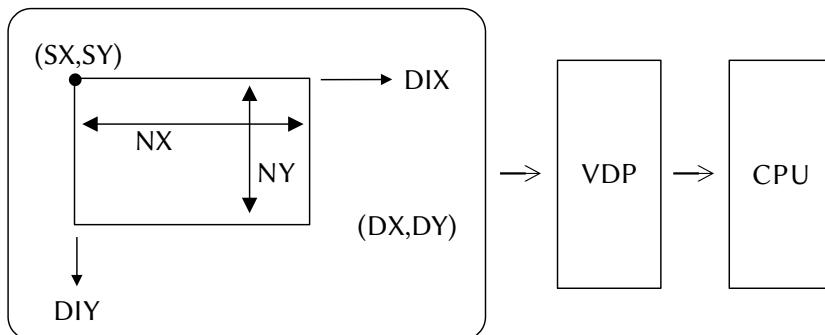
O estado da transferência pode ser lido em S#2:



#### 5.5.4.6 – LMCM (Transferência lógica – VRAM → CPU)

Nesse comando, os dados são transferidos de uma área especificada na VRAM para a CPU em pontos. Um byte é requerido para cada ponto.

Ao iniciar a execução desse comando, a CPU deve verificar o bit TR de S#2. Se esse bit for 1, o byte de dados estará disponível para ser lido em S#7. Quando o bit CE de S#2 for 0, os bytes de dados a serem transferidos terminaram.



Os seguintes registradores devem ser carregados:

R#32 –	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	– x7~x0	SX	Coordenada horizontal de origem (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#33 –	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr> </table>	0	0	0	0	0	0	0	x	– x8										
0	0	0	0	0	0	0	x													
R#34 –	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	– y7~y0	SY	Coordenada vertical de origem (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#35 –	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr> </table>	0	0	0	0	0	0	y	y	– y9~y8										
0	0	0	0	0	0	y	y													

R#40 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX	Número de pontos a transferir na direção horizontal (0 a 511)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr> </table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													

R#42 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY	Número de pontos a transferir na direção vertical (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr> </table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													

R#44 – Não utilizado por este comando.

R#45 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>•</td><td>•</td><td>•</td><td>m</td><td>y</td><td>x</td><td>•</td><td>•</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	m	y	x	•	•	ARGT (argument register)
b7	b6	b5	b4	b3	b2	b1	b0											
•	•	•	m	y	x	•	•											
		DIX (direção X) 0 – para a direita 1 – para a esquerda																
		DIY (direção Y) 0 – para baixo 1 – para cima																
		MXS (memória de origem) 0 – VRAM 1 – VRAM expandida																

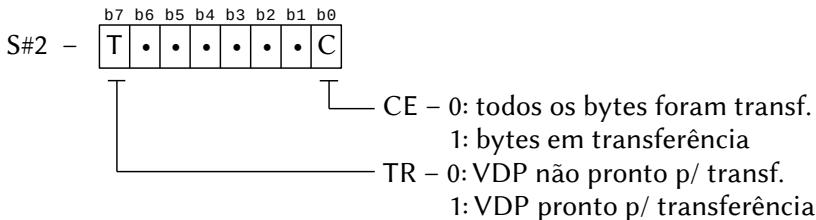
Para executar o comando LMCM, deve ser escrito o valor 1010B nos quatro bits mais altos de R#46 e o código de operação lógica nos quatro bits mais baixos.

R#46 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	1	0	1	0	0	0	0	0	CMR (command register)
b7	b6	b5	b4	b3	b2	b1	b0											
1	0	1	0	0	0	0	0											
		Código de operação lógica 1010 – Comando LMCM																

O valor dos bytes lidos fica disponível no registrador S#7, no formato ilustrado abaixo. Quando o último dado for escrito em S#7 e o bit TR de S#2 for 1, o comando será terminado pelo VDP e o bit CE de S#2 será 0.

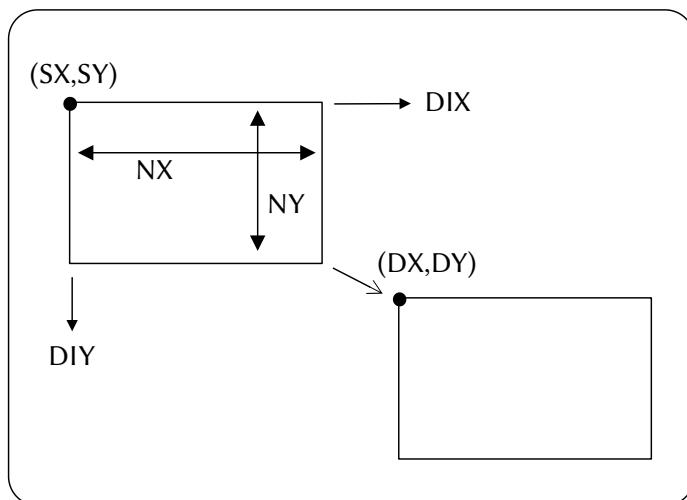
S#7 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	•	c	c	c	c	CLR Gráficos 4 e 6	Formato dos dados de cores lidos da VRAM
b7	b6	b5	b4	b3	b2	b1	b0												
•	•	•	•	c	c	c	c												
	<table border="1"> <tr><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td></tr> </table>	•	•	•	•	•	•	c	c	CLR Gráfico 5									
•	•	•	•	•	•	c	c												
	<table border="1"> <tr><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td></tr> </table>	c	c	c	c	c	c	c	c	CLR Gráficos 7, 8 e 9									
c	c	c	c	c	c	c	c												

O estado da transferência pode ser lido em S#2:

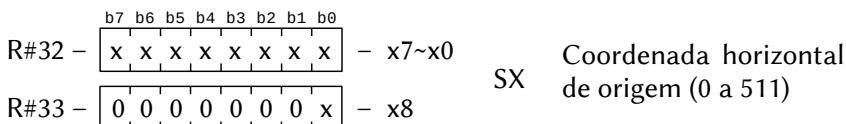


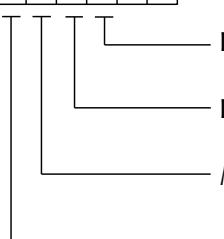
#### 5.5.4.7 – LMMM (Transf. lógica – VRAM → VRAM)

Nesse comando, os dados de uma área especificada na VRAM são transferidos para outra área da VRAM em pontos. Operações lógicas no destino são possíveis. Enquanto o bit CE de S#2 for 1, o comando estará sendo executado.

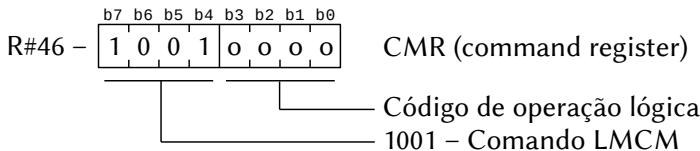


Os seguintes registradores devem ser carregados:

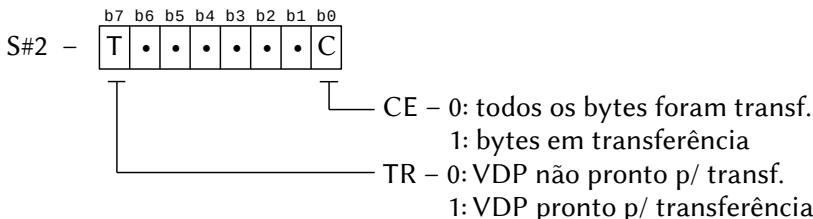


R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY	Coordenada vertical de origem (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal de destino (0 a 511)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical de destino (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	SX	Número de pontos a transferir na direção horizontal (0 a 511)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY	Número de pontos a transferir na direção vertical (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#45 -	<table border="1"><tr><td>•</td><td>•</td><td>d</td><td>s</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	•	•	d	s	y	x	•	•	ARGT (argument register)										
•	•	d	s	y	x	•	•													
			DIX (direção X)	0 – para a direita 1 – para a esquerda																
			DIY (direção Y)	0 – para baixo 1 – para cima																
			MXS (memória de origem)	0 – VRAM 1 – VRAM expandida																
			MXD (memória de destino)	0 – VRAM 1 – VRAM expandida																

Para executar o comando LMMM, deve-se escrever o valor 1001B nos quatro bits mais altos de R#46, sendo que os quatro bits mais baixos devem conter o código de operação lógica.

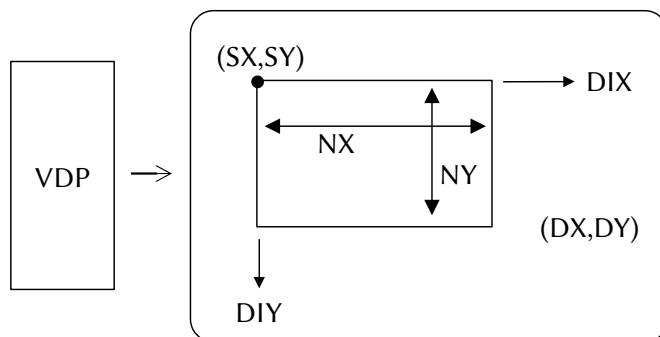


O estado da transferência pode ser lido em S#2:

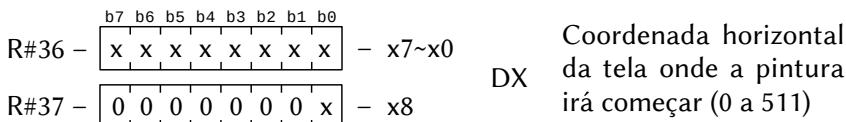


#### 5.5.4.8 – LMMV (Pintura lógica da VRAM)

Uma área retangular qualquer da VRAM pode ser pintada com pontos de uma determinada cor e operações lógicas no destino com a cor especificada são possíveis.



Os seguintes registradores devem ser carregados:

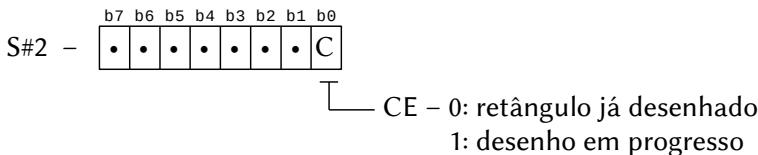


R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical da tela onde a pintura irá começar (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#40 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	SX	Número de pontos a pintar na direção horizontal (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	SY	Número de pontos a pintar na direção vertical (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	•	•	c	c	c	c	CLR	Gráficos 4 e 6	Formato dos dados de cores a para a pintura
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
•	•	•	•	c	c	c	c													
	<table border="1"><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td></tr></table>	•	•	•	•	•	•	c	c	CLR	Gráfico 5									
•	•	•	•	•	•	c	c													
	<table border="1"><tr><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td></tr></table>	c	c	c	c	c	c	c	c	CLR	Gráficos 7, 8 e 9									
c	c	c	c	c	c	c	c													
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>•</td><td>•</td><td>m</td><td>•</td><td>y</td><td>x</td><td>•</td><td>•</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	m	•	y	x	•	•	ARGT (argument register)		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
•	•	m	•	y	x	•	•													
			DIX (direção X)	0 – para a direita 1 – para a esquerda																
			DIY (direção Y)	0 – para baixo 1 – para cima																
			MXD (memória de destino)	0 – VRAM 1 – VRAM expandida																

Para executar o comando LMMV deve-se escrever o valor 1000B nos quatro bits mais altos de R#46 sendo que os quatro bits mais baixos devem conter o código de operação lógica.

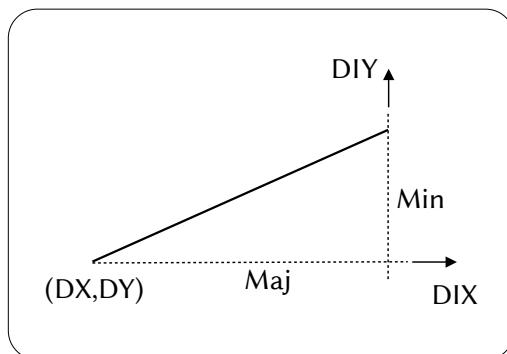
R#46 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>o</td><td>o</td><td>o</td><td>o</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	0	0	0	o	o	o	o	CMR (command register)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
1	0	0	0	o	o	o	o											
		Código de operação lógica 1000 – Comando LMMV																

O estado da pintura pode ser lido em S#2:



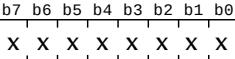
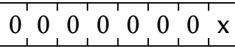
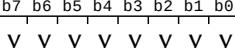
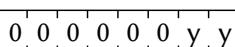
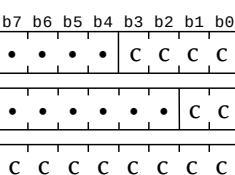
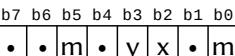
#### 5.5.4.9 – LINE (Desenha uma linha)

Esse comando desenha uma linha entre coordenadas da tela. Os parâmetros são especificados incluindo a coordenada (X,Y) de início da linha e o comprimento horizontal e vertical até o ponto final, conforme a ilustração abaixo:

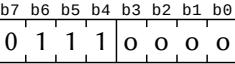


Antes da execução do comando LINE, os seguintes registradores devem ser carregados:

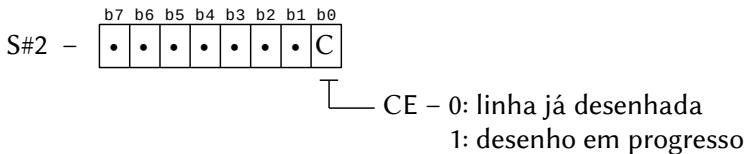
R#36 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td> </tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td> </tr> </table> – x7~x0	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	DX	Coordenada horizontal da tela a partir da qual a linha será desenhada (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
x	x	x	x	x	x	x	x											
R#37 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td> </tr> </table> – x8	0	0	0	0	0	0	0	x	DY	Coordenada vertical da tela a partir da qual a linha será desenhada (0 a 1023)								
0	0	0	0	0	0	0	x											
R#38 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td> </tr> <tr> <td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td> </tr> </table> – y7~y0	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
y	y	y	y	y	y	y	y											
R#39 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td> </tr> </table> – y9~y8	0	0	0	0	0	0	y	y										
0	0	0	0	0	0	y	y											

R#40 -		- x7~x0		Número de pontos do cateto maior do triângulo retângulo de referência para desenho
R#41 -		- x8	Maj	
R#42 -		- y7~y0		Número de pontos do cateto menor do triângulo retângulo de referência para desenho
R#43 -		- y9~y8	Min	
R#44 -		CLR	Gráficos 4 e 6	Formato dos dados de cores a para o desenho da linha
		CLR	Gráfico 5	
		CLR	Gráficos 7, 8 e 9	
R#45 -		ARGT (argument register)		
			MAJ – 0, se o lado maior for horizontal 1, se o lado maior for vertical ou igual ao lado menor	
			DIX (direção X) 0 – para a direita 1 – para a esquerda	
			DIY (direção Y) 0 – para baixo 1 – para cima	
			MXD (memória de destino) 0 – VRAM 1 – VRAM expandida	

Para executar o comando LINE deve-se escrever o valor 0111B nos quatro bits mais altos de R#46 sendo que os quatro bits mais baixos devem conter o código de operação lógica.

R#46 -		CMR (command register)		
			Código de operação lógica 0111 – Comando LINE	

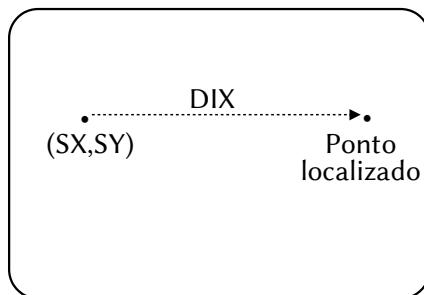
O estado do desenho pode ser lido em S#2:



#### 5.5.4.10 – SRCH (Procura código de cor)

Esse comando procura a existência de um ponto com uma cor específica a partir de uma coordenada na VRAM, sempre na horizontal, para a esquerda ou para a direita. É um comando útil para rotinas de pintura ou preenchimento.

O comando termina quando o ponto com a cor especificada é encontrado ou quando a borda da tela é atingida. Enquanto o bit CE de S#2 for 1, o comando estará sendo executado. Terminado o comando, se o ponto com a cor foi encontrado, o bit BD se S#2 será 1 e a coordenada horizontal do ponto ficará disponível em S#8 e S#9.



Antes da execução do comando SRCH, os seguintes registradores devem ser carregados:

R#32 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td> </tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td> </tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	– x <sub>7~0</sub>	SX	Coordenada horizontal da tela para início da procura (0 a 511)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
x	x	x	x	x	x	x	x												
R#33 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td> </tr> </table>	0	0	0	0	0	0	0	x	– x <sub>8</sub>										
0	0	0	0	0	0	0	x												
R#34 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td> </tr> <tr> <td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td> </tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	– y <sub>7~0</sub>	SY	Coordenada vertical da tela para início da procura (0 a 1023)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
y	y	y	y	y	y	y	y												
R#35 – <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td> </tr> </table>	0	0	0	0	0	0	y	y	– y <sub>9~y8</sub>										
0	0	0	0	0	0	y	y												

R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	•	c	c	c	c	CLR Gráficos 4 e 6	Código de cor para a procura
b7	b6	b5	b4	b3	b2	b1	b0												
•	•	•	•	c	c	c	c												
	<table border="1"><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td></tr></table>	•	•	•	•	•	•	c	c	CLR Gráfico 5									
•	•	•	•	•	•	c	c												
	<table border="1"><tr><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td></tr></table>	c	c	c	c	c	c	c	c	CLR Gráficos 7, 8 e 9									
c	c	c	c	c	c	c	c												
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>•</td><td>•</td><td>m</td><td>•</td><td>•</td><td>x</td><td>e</td><td>•</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	m	•	•	x	e	•	ARGT (argument register)	
b7	b6	b5	b4	b3	b2	b1	b0												
•	•	m	•	•	x	e	•												
		EQ – condição p/ término de execução 0 – termina ao encontrar a cor especificada 1 – termina ao encontrar outra cor que não a especificada																	
		DIX (direção X) 0 – para a direita 1 – para a esquerda																	
		MXD (memória de destino) 0 – VRAM 1 – VRAM expandida																	

Para executar o comando SRCH, basta escrever o valor 60H (0110B) no registrador R#46. Ao terminar a execução, se o bit BD se S#2 for 1, o ponto com a cor especificada foi encontrado e sua coordenada horizontal está armazenada em S#8 e S#9. Se o ponto não foi encontrado, o bit BD de S#2 será 0.

R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	1	1	0	•	•	•	•	CMR (command register)
b7	b6	b5	b4	b3	b2	b1	b0											
0	1	1	0	•	•	•	•											
		Sem significado 0110 – Comando SRCH																

O estado da procura pode ser lido em S#2:

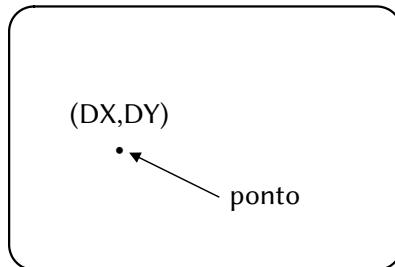
S#2 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>•</td><td>•</td><td>•</td><td>B</td><td>•</td><td>•</td><td>•</td><td>C</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	B	•	•	•	C	CE – 0: busca terminada 1: busca em progresso
b7	b6	b5	b4	b3	b2	b1	b0											
•	•	•	B	•	•	•	C											
		BD – 0: cor desejada não localizada 1: cor desejada foi encontrada																

Se o ponto com as características selecionadas for encontrado, sua coordenada horizontal poderá se lida em S#8 e S#9:

S#8 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	Coordenada horizontal do ponto com a cor especificada, quando encontrado
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
S#9 -	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr> </table>	0	0	0	0	0	0	0	x	- x8									
0	0	0	0	0	0	0	x												

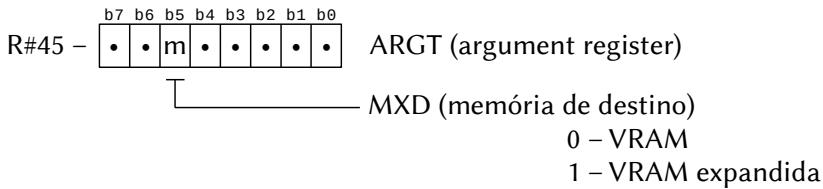
#### 5.5.4.11 – PSET (Desenha um ponto)

Usando esse comando, pode-se desenhar um ponto em qualquer coordenada da VRAM. Operações lógicas no destino são possíveis.

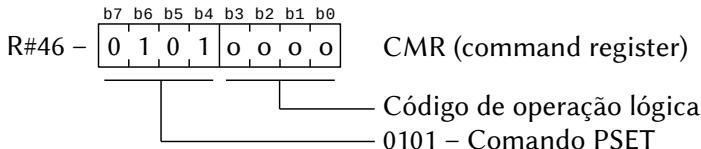


Os seguintes registradores devem ser carregados:

R#36 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal do ponto a desenhar (0 a 511)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr> </table>	0	0	0	0	0	0	0	x	- x8										
0	0	0	0	0	0	0	x													
R#38 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical do ponto a desenhar (0 a 1023)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr> </table>	0	0	0	0	0	0	y	y	- y9~y8										
0	0	0	0	0	0	y	y													
R#44 -	<table border="1"> <tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	•	c	c	c	c	CLR	Gráficos 4 e 6	Código de cor do ponto a desenhar
b7	b6	b5	b4	b3	b2	b1	b0													
•	•	•	•	c	c	c	c													
	<table border="1"> <tr><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td></tr> </table>	•	•	•	•	•	•	c	c	CLR	Gráfico 5									
•	•	•	•	•	•	c	c													
	<table border="1"> <tr><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td></tr> </table>	c	c	c	c	c	c	c	c	CLR	Gráficos 7, 8 e 9									
c	c	c	c	c	c	c	c													

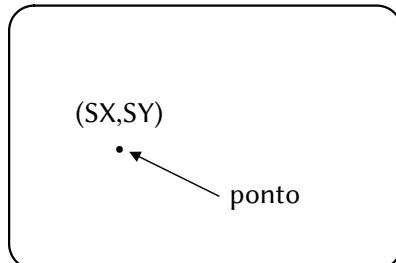


Para executar o comando PSET, basta escrever o valor 0101B nos quatro bits mais altos de R#46 e o código de operação lógica nos quatro bits mais baixos.

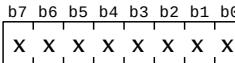
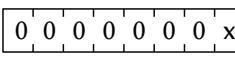
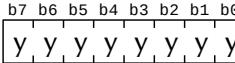
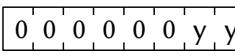


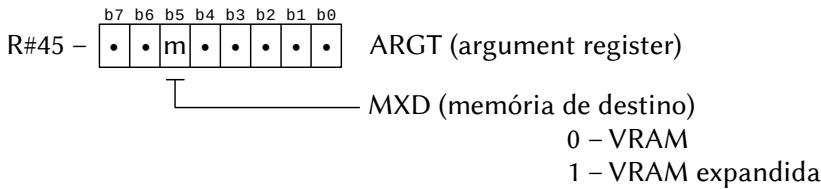
#### 5.5.4.12 – POINT (Lê código de cor de um ponto)

O comando POINT lê um código de cor de um ponto em qualquer coordenada da VRAM.

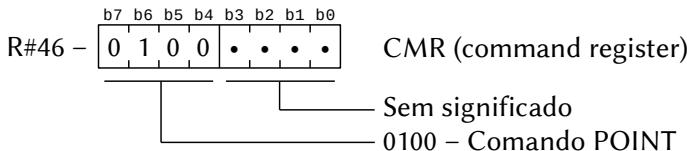


Os seguintes registradores devem ser carregados:

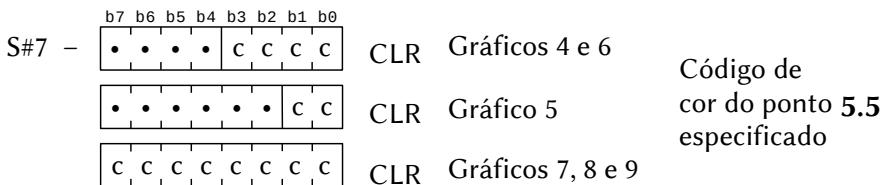
R#32 –  – x7~x0	SX	Coordenada horizontal do ponto a ser lido (0 a 511)
R#33 –  – x8		
R#34 –  – y7~y0	SY	Coordenada vertical do ponto a ser lido (0 a 1023)
R#35 –  – y9~y8		



Para executar o comando POINT, basta escrever 40H (0100B) em R#46:



A cor do ponto especificado pode ser lida em S#7:

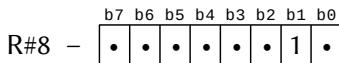


## .5 – Tornando os comandos mais rápidos

A estrutura do VDP permite que várias outras tarefas sejam executadas durante a execução de um comando. Às vezes, a execução de alguns desses comandos fica lenta devido a isso. Se essas funções forem desativadas, a execução do comando ficará mais rápida.

### • Inibição de apresentação dos sprites

Esse meio é muito prático e permite um sensível aumento da velocidade quando os sprites são removidos da tela. Para isso, basta setar o bit 1 de R#8 em 1.



### • Inibição de apresentação da tela

Esse meio possui o inconveniente de que, quando inibida, a tela toda fica com a cor da borda. Para inibir a apresentação da tela, basta resetar o bit 6 de R#1, zerando-o.

R#1 -	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
	•	0	•	•	•	•	•	•

## 5.6 – MISCELÂNEA DE FUNÇÕES DO VDP (V9938/58)

Essa seção descreve várias funções adicionais do VDP.

### 5.6.1 – Ajuste da localização da tela

O registrador R#18 é usado para ajustar a localização da tela. Corresponde à instrução SET ADJUST do BASIC.

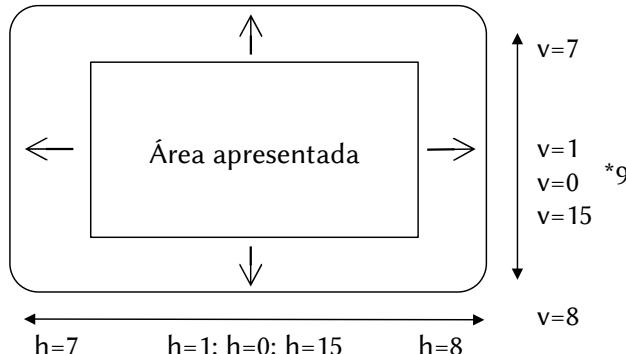
R#18 -	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
	v	v	v	v	h	h	h	h

↓                    ↓

Ajuste horizontal (-8 a +7)  
Ajuste vertical (-8 a +7)

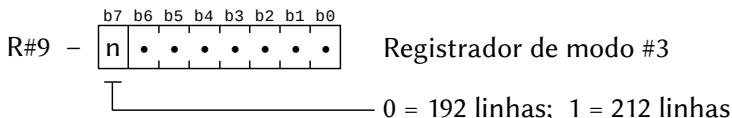
O valor é especificado em complemento de 2 e segue os seguintes valores:

0111	→ 7	→ +7	1111	→ 15	→ -1
0110	→ 6	→ +6	1110	→ 14	→ -2
:	:	:	:	:	:
0001	→ 1	→ +1	1001	→ 9	→ -7
0000	→ 0	→ 0	1000	→ 8	→ -8



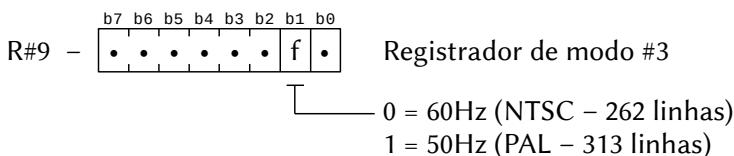
### 5.6.2 – Número de pontos na direção vertical

O número de pontos na direção vertical pode ser escolhido entre 192 ou 212, através do bit 7 de R#9. Essa função só é válida para o modo texto 2 e para os modos gráficos 4 a 9. O modo 212 linhas para o modo texto 2 não é suportado pelo BASIC.



### 5.6.3 – Frequência de interrupção (PAL/NTSC)

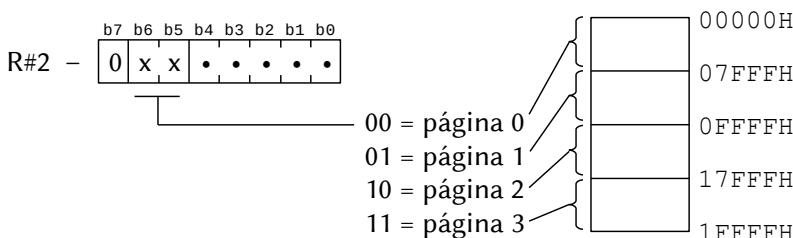
A frequência de interrupção no MSX é controlada pelo VDP e pode ser de 50Hz ou 60Hz. A frequência de 60Hz é usada no sistema NTSC do Japão e no sistema PAL-M brasileiro. A de 50Hz é usada para o sistema PAL-N europeu.



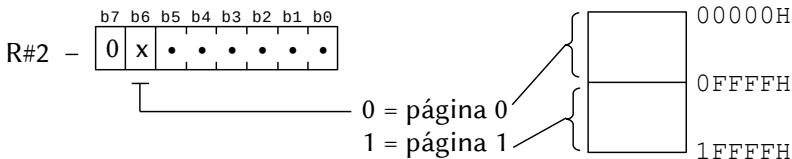
### 5.6.4 – Troca das páginas de vídeo

Nos modos gráficos 4 a 9, as páginas em apresentação podem ser trocadas modificando o endereço de início da tabela de padrões.

Modos gráficos 4 e 5:

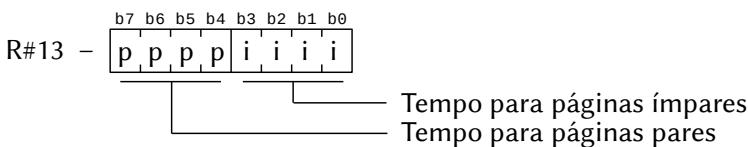


### Modos gráficos 6, 7, 8 e 9:



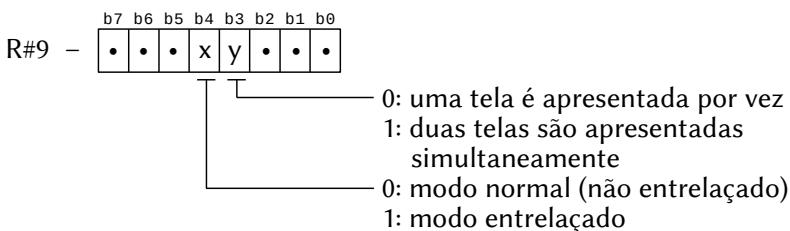
### 5.6.5 – Troca automática de tela

Nos modos gráficos 4 a 9, duas páginas de vídeo podem ser apresentadas alternadamente. As páginas 0 e 1 ou 2 e 3 podem usar esse recurso. Para iniciar a troca automática de telas, a página ímpar deve ser selecionada (1 ou 3); depois é só regular o tempo de troca em R#13. Os quatro bits mais altos definem o tempo para a página par e os quatro bits mais baixos para a página ímpar. O período de tempo é contado em unidades de 1/6 de segundo. Se o valor do período for 0, o “blinking” é desativado; neste caso, qualquer página pode ser selecionada para apresentação. Se for “0001B” será 166 mS para 60 Hz (NTSC) ou 200 mS para 50 Hz (PAL); se for “1111B”, será 2.503 mS (2,5 segundos) para 60 Hz e 3.000 mS (3 segundos) para 50 Hz.

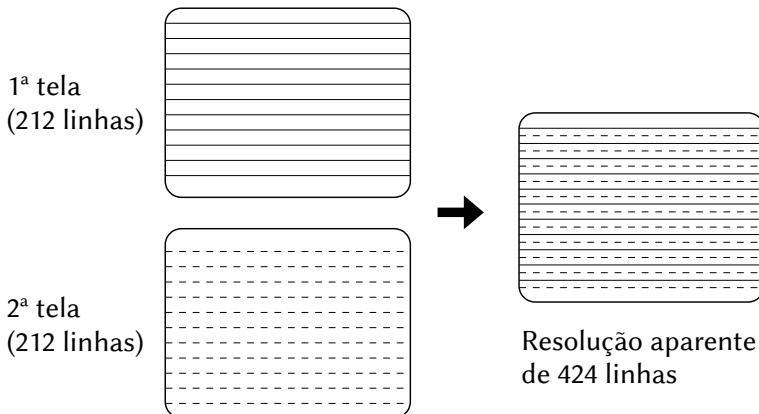


### 5.6.6 – Modo entrelaçado

O modo entrelaçado pode ser usado para se ter uma resolução vertical aparente de 424 linhas. Isso é feito alternando em alta velocidade duas páginas de vídeo e mostrando apenas a metade da altura de cada linha dessas páginas. As duas páginas são trocadas 60 vezes por segundo, o que pode causar cintilação. Esse modo é selecionado por R#9.

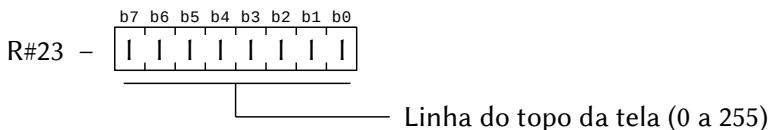


O modo entrelaçado funciona como ilustrado abaixo.



### 5.6.7 – Scroll vertical

O registrador R#23 é usado para indicar a linha inicial a tela. Trocando o valor desse registrador, pode-se fazer um scroll vertical muito suave. Como o scroll é feito para 256 linhas, a tabela de sprites poderá aparecer e ser movida para outra página.



### 5.6.8 – Scroll horizontal (Só V9958)

O scroll horizontal é suportado pelo MSX2+ ou superior. Ele é feito através dos registradores R#26 e R#27, sempre considerando que a tela tem 256 pontos horizontais, mesmo nos modos gráficos 5 e 6.

O registrador R#26 pode variar de 0 a 31 (scroll com uma página de vídeo) ou de 0 a 63 (scroll com duas páginas de vídeo). Cada incremento corresponde ao deslocamento de 8 pontos na tela (16 nos modos gráficos 5 e 6). Já R#27 pode variar de 7 a 0 sendo que cada decremento corresponde ao deslocamento de um ponto na tela (dois para os modos gráficos 5 e 6). É importante frisar que quando um é incrementado, o outro deve ser decrementado.

O bit 0 de R#25 determina se o scroll será feito com duas páginas de vídeo. Se for 0, o scroll será feito com apenas uma página; se for 1, será feito com duas páginas consecutivas, sendo que a página que está sendo exibida deve ser ímpar. O bit 1 de R#25 determina a ligação de uma máscara que cobre as 8 colunas da esquerda da tela. Se for 0, a máscara estará desligada; se for 1, estará ligada. A cor da máscara é igual à cor da borda.

	b7 b6 b5 b4 b3 b2 b1 b0	
R#27 -	0 0 0 0 0   h h h	- h2~h0
R#26 -	0 0   h h h h h h	- h8~h3
R#25 -	0 0 0 0 0 0   m s	<div style="display: flex; align-items: center; justify-content: space-between;"> <span>SP2:</span> <span>0 – scroll com 1 página</span>   <span>1 – scroll com 2 páginas</span>   <span>MSK:</span> <span>0 – máscara desligada</span>   <span>1 – máscara ligada</span> </div>

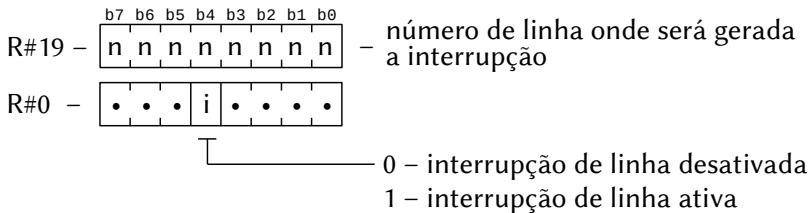
### 5.6.9 – Código de cor 0

Das 16 cores da paleta, a cor 0 é transparente, ou seja, não pode ser definida uma cor para ela e qualquer objeto desenhado com ela não será visto. Entretanto, setando o bit 5 de R#8, a função de transparente será desativada e a cor 0 poderá ser definida por P#0.

	b7 b6 b5 b4 b3 b2 b1 b0	
R#8 -	• •   t   • • • • •	<div style="display: flex; align-items: center; justify-content: space-between;"> <span>0 = cor transparente ativa</span>   <span>1 = código de cor 0 pode ser definido</span> </div>

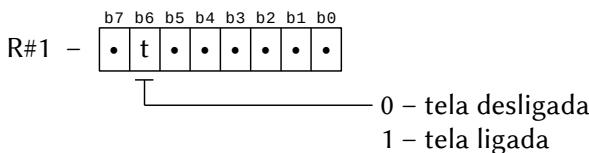
### 5.6.10 – Interrupção por varredura de linha

No MSX-VIDEO, uma interrupção pode ser gerada quando termina a varredura de uma linha específica da tela. Para isso, basta colocar em R#19 o número da linha que deverá gerar a interrupção e setar o bit 4 de R#0 em 1.



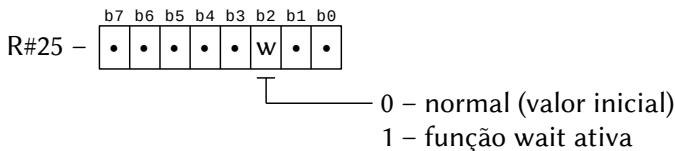
### 5.6.11 – Liga/desliga a tela

A função de ligar e desligar a apresentação na tela é controlada pelo bit 6 de R#1. Quando estiver desligada, a tela inteira fica com a cor especificada pelos quatro bits mais baixos de R#7 (8 bits no modo gráfico 7). Os comandos de hardware do VDP ficam mais rápidos quando a tela estiver desligada.



### 5.6.12 – Comando de espera (Só V9958)

Esta função pode ser usada para aumentar a velocidade da transferência de dados da CPU para a VRAM. Quando ativa, todas as portas de acesso ao VDP são colocadas em modo de espera até que o acesso à VRAM seja completado. Esta função é controlada pelo bit 2 de R#25.



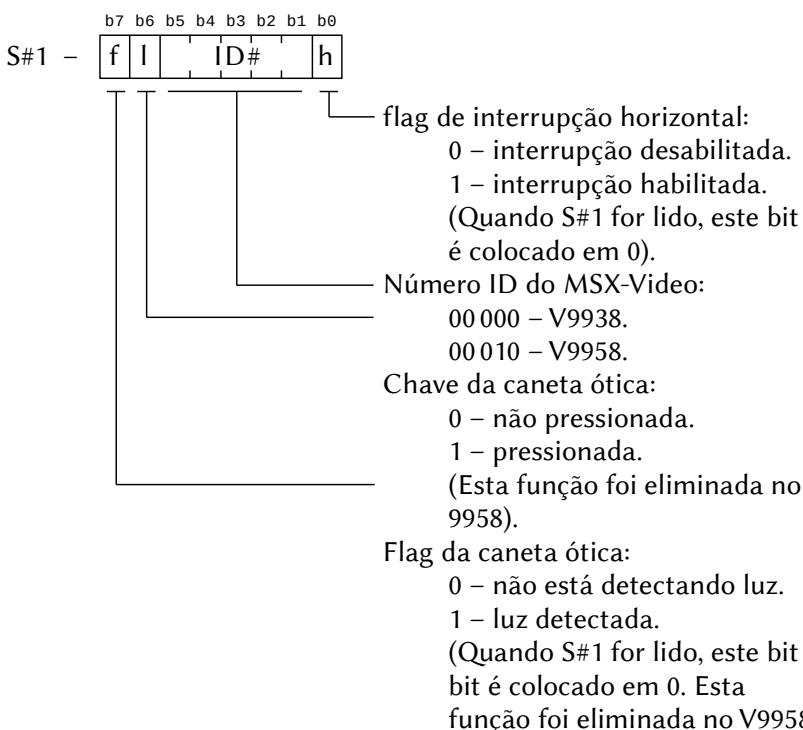
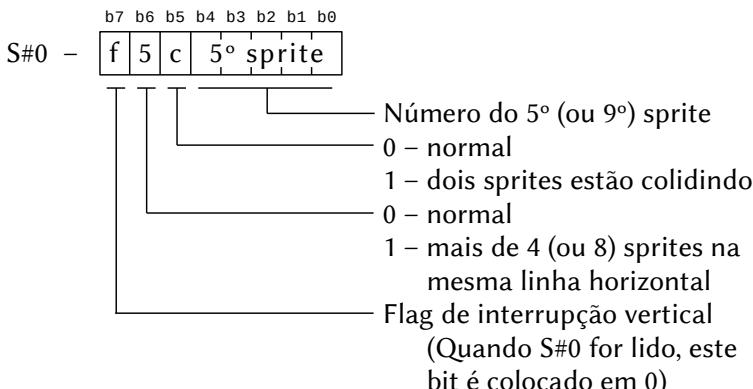
## 5.7 – DESCRIÇÃO DOS REGISTRADORES (V9938/58)

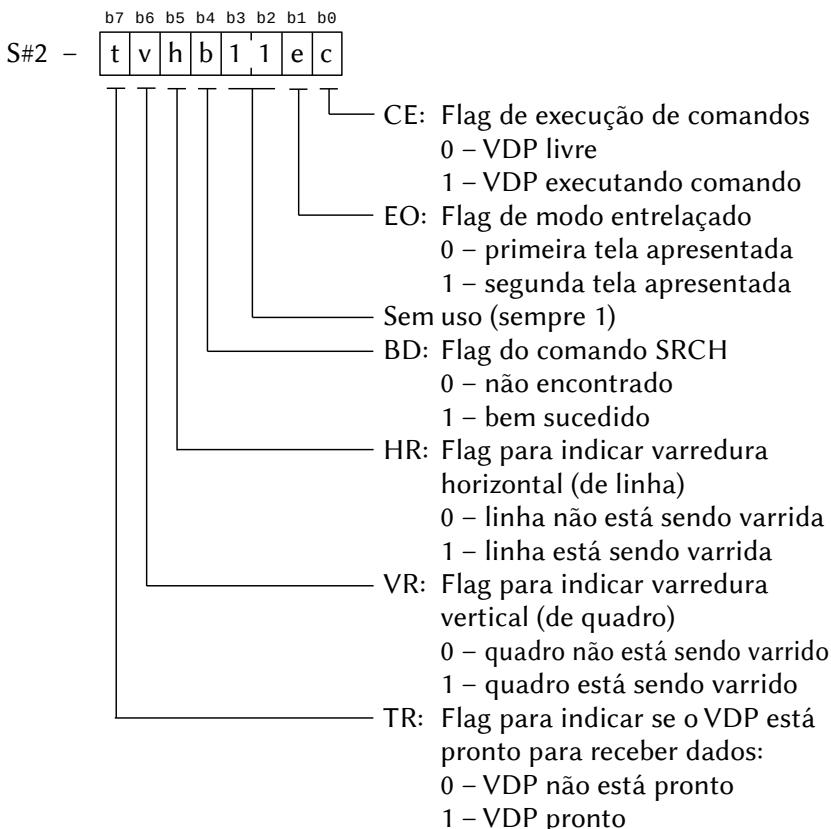
Esta seção descreve resumidamente todos os registradores dos VDP's TMS9918, V9938 e V9958.

### 5.7.1 – Registradores de estado

Os registradores de estado são numerados de S#0 até S#9. São

registradores apenas de leitura e suas funções estão ilustradas abaixo.  
Apenas o registrador S#0 está presente no TMS9918.





Os registradores S#3 a S#6 contêm a localização da colisão entre sprites. Para o V9938 também indicarão a localização da caneta ótica e do movimento relativo do mouse (estas funções foram eliminadas no V9958).

S#3 -	<table border="1"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td> </tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	Coordenada X + 12
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
S#4 -	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td> </tr> </table>	0	0	0	0	0	0	0	x	- x8									
0	0	0	0	0	0	0	x												
S#5 -	<table border="1"> <tr> <td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td> </tr> </table>	y	y	y	y	y	y	y	y	- y7~y0	Coordenada Y + 8								
y	y	y	y	y	y	y	y												
S#6 -	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td> </tr> </table>	0	0	0	0	0	0	y	y	- y9~y8									
0	0	0	0	0	0	y	y												

O registrador S#7 contém o código de cor para os comandos PO-INT e LMCM.

S#7 -	<table border="1"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr> <td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	•	•	•	•	c	c	c	c	CLR Gráficos 4 e 6	Código de cor do ponto especificado
b7	b6	b5	b4	b3	b2	b1	b0												
•	•	•	•	c	c	c	c												
	<table border="1"> <tr> <td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td> </tr> </table>	•	•	•	•	•	•	c	c	CLR Gráfico 5									
•	•	•	•	•	•	c	c												
	<table border="1"> <tr> <td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td> </tr> </table>	c	c	c	c	c	c	c	c	CLR Gráficos 7, 8 e 9									
c	c	c	c	c	c	c	c												

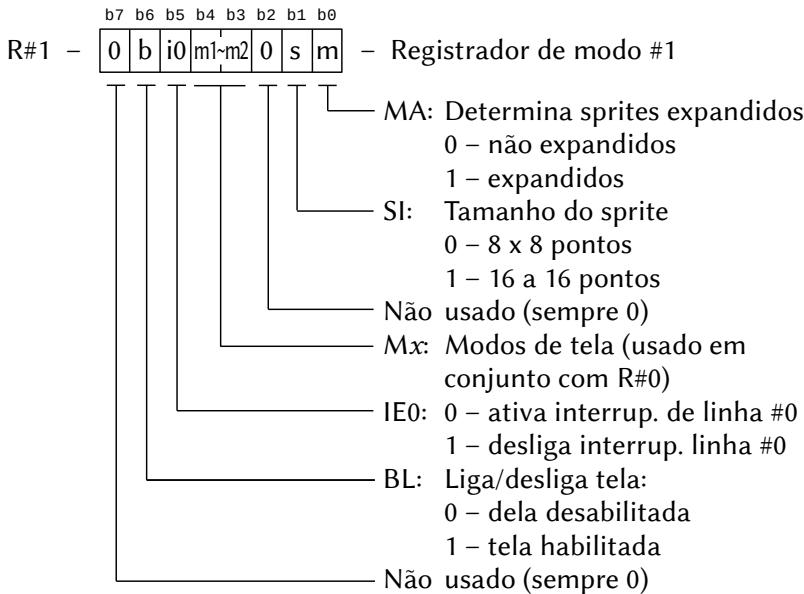
Os registradores S#8 e S#9 contêm a posição horizontal para o comando SRCH.

S#8 -	<table border="1"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	Coordenada horizontal do ponto com a cor especificada, quando encontrado
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
S#9 -	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td> </tr> </table>	0	0	0	0	0	0	0	x	- x8									
0	0	0	0	0	0	0	x												

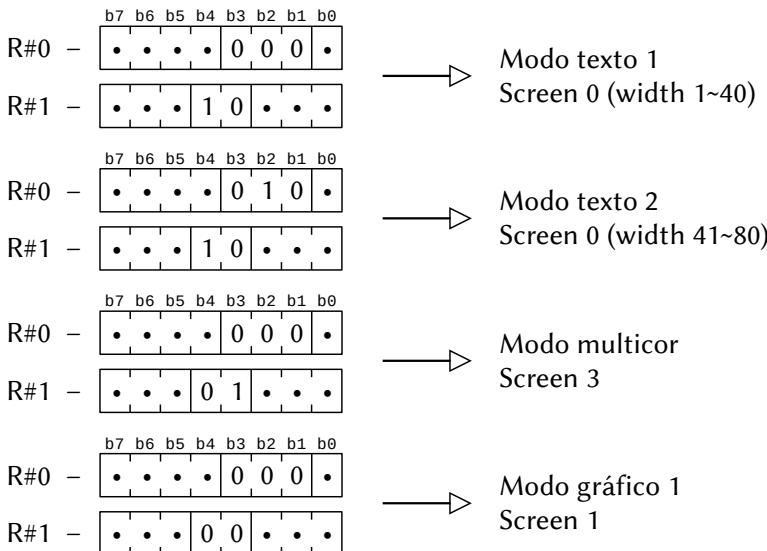
### 5.7.2 – Registradores de modo

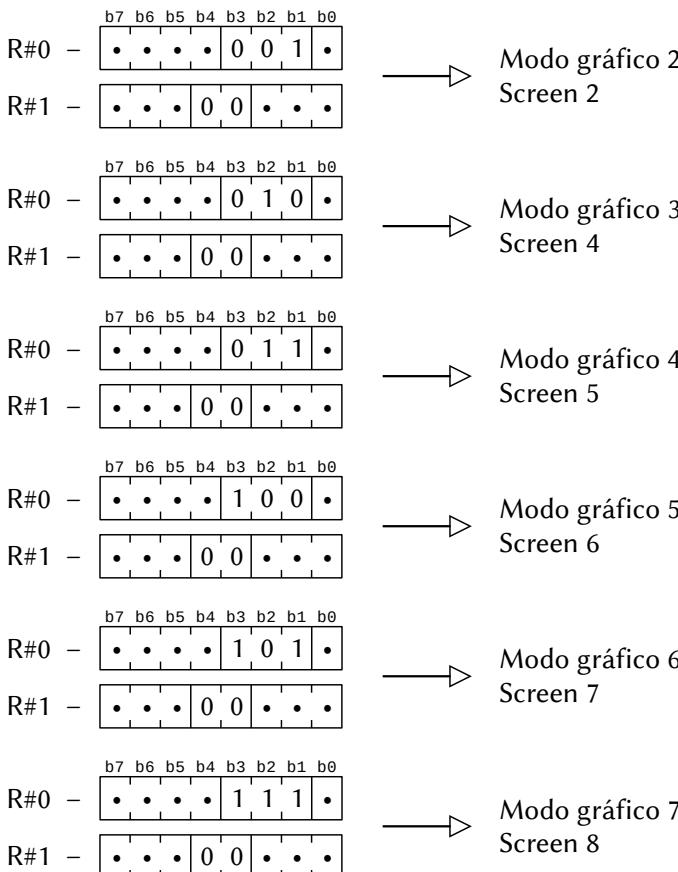
Os registradores de modo são designados por R#0, R#1, R#8, R#9 e R#25, sendo que R#8 e R#9 são exclusivos do V9938/58 e R#25 é exclusivo do V9958. São registradores apenas de escrita e controlam os modos de operação do VDP.

R#0 -	<table border="1"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr> <td>0</td><td>d</td><td>i2</td><td>i1</td><td>m5~m3</td><td>0</td><td>0</td><td>0</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	0	d	i2	i1	m5~m3	0	0	0	- Registrador de modo #0	
b7	b6	b5	b4	b3	b2	b1	b0												
0	d	i2	i1	m5~m3	0	0	0												
		Não usado (sempre 0)																	
		Mx: Modos de tela (usado em conjunto com R#1)																	
		IE1: 0 – ativa interrup. de linha #1 1 – desliga interrup. linha #1																	
		IE2: Ativa interrupção de caneta ótica. Esta função foi eliminada no V9958																	
		DG: 0 – normal 1 – Seta o color bus como entrada																	
		Não usado (sempre 0)																	

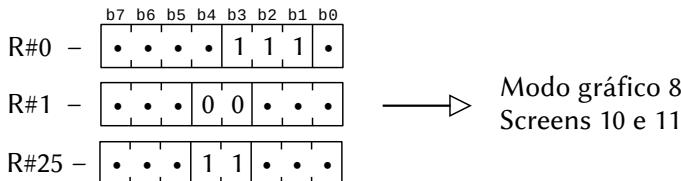


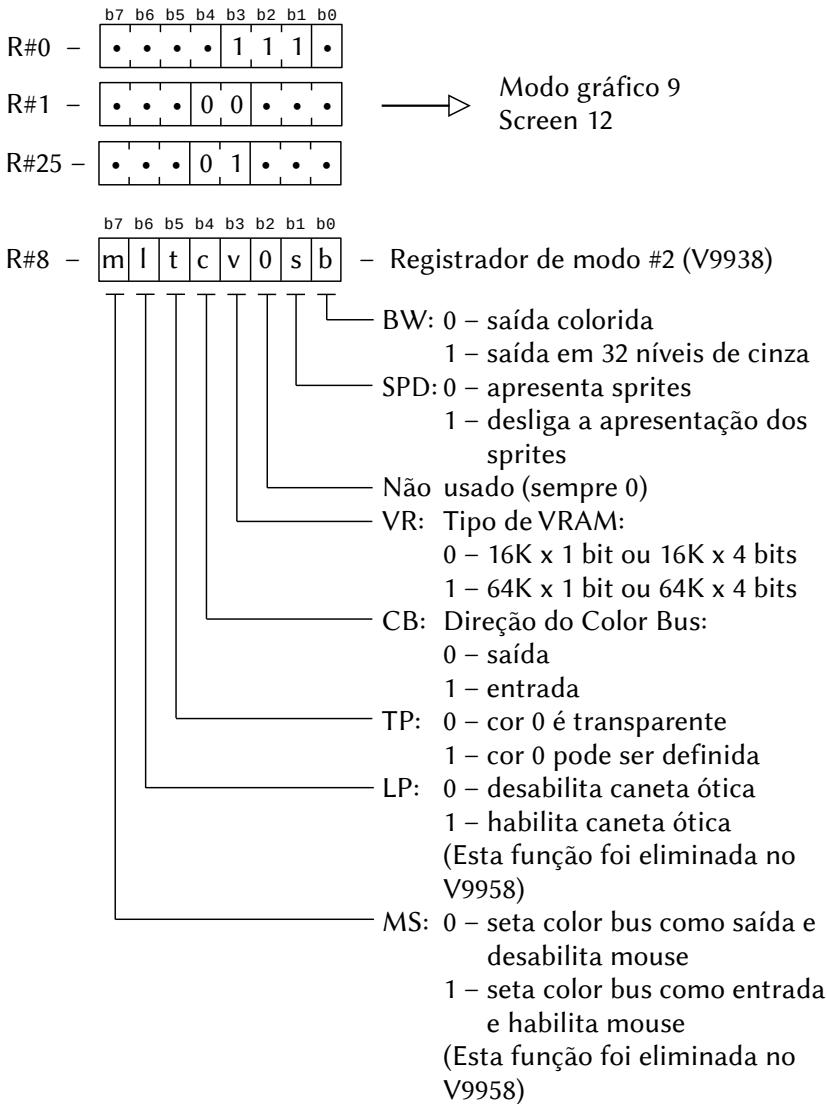
As screens são selecionadas pelos bits m1~m5 de R#0 e R#1 de acordo com as seguintes ilustrações:

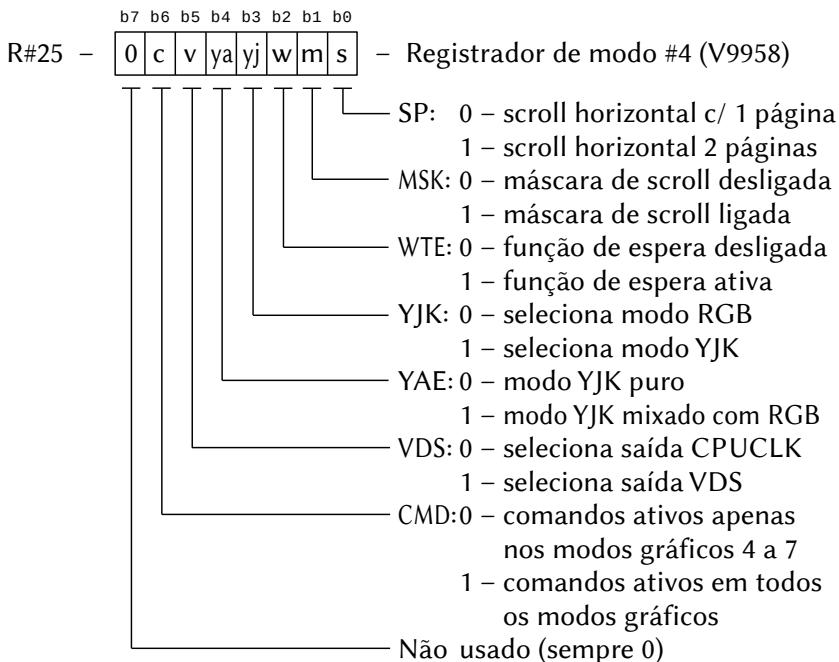
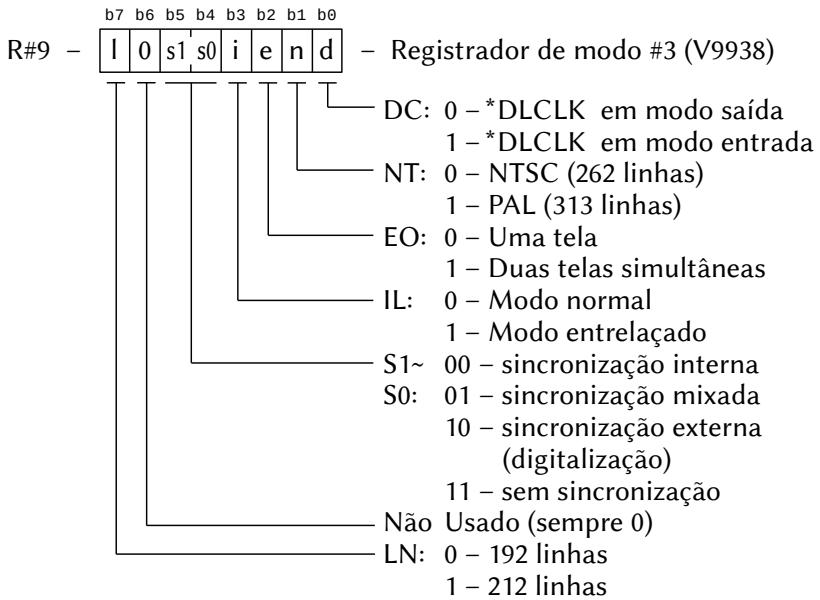




Os modos exclusivos do V9958 são selecionados em conjunto com R#25, que é o registrador de modo #4. Para os modos 1 a 7 os dois bits de R#25 devem estar zerados.







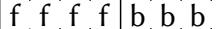
### 5.7.3 – Registradores de endereços das tabelas

São registradores que armazenam os endereços das tabelas na VRAM. Os valores desses registradores dependem da screen em uso.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#2 -	0	a16	a15	a14	a13	a12	a11	a10
R#3 -	a13	a12	a11	a10	a9	a8	a7	a6
R#10 -	0	0	0	0	0	a16	a15	a14
R#4 -	0	0	a16	a15	a14	a13	a12	a11
R#5 -	a14	a13	a12	a11	a10	a9	a8	a7
R#11 -	0	0	0	0	0	0	a16	a15
R#6 -	0	0	a16	a15	a14	a13	a12	a11

#### 5.7.4 – Registradores de cores e display

São usados para ajustar as cores dos e outras características da tela.

R#7 - 

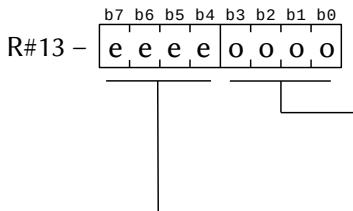
cor de fundo e da borda nos modos texto e da borda em todos os modos

cor dos caracteres nos modos texto

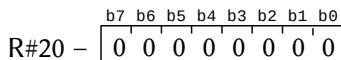
R#12 – 

cor de fundo dos caracteres nos modos texto para a função “blink”

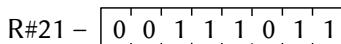
cor de frente dos caracteres nos modos texto para a função “blink”



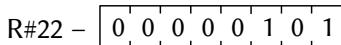
Tempo de “blink” para a cor especificada em R#7 no modo texto 2 ou para a página ímpar nos modos gráficos 4 a 7  
 Tempo de “blink” para a cor especificada em R#7 no modo texto 2 ou para a página par nos modos gráficos 4 a 7



Regitrador de “burst” de cor #1

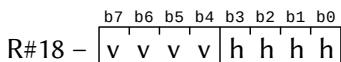


Regitrador de “burst” de cor #2



Regitrador de “burst” de cor #3

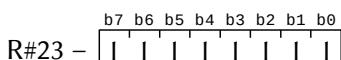
Os valores acima definem o sinal da subportadora de cor para a saída de vídeo composto do V9938. Esta saída não está presente no V9958. Além disso, na maioria dos MSX2, o sinal de vídeo composto é gerado por circuitos externos, de forma que estes registradores se tornaram inúteis.



Registrador de ajuste da imagem

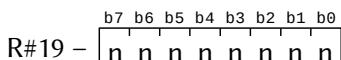
Ajuste horizontal de imagem (-8 a +7)

Ajuste vertical de imagem (-8 a +7)



Registrador de offset da imagem

Número da linha da área de imagem que será apresentada no topo da tela (0 a 255)

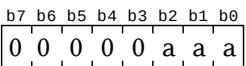


Registrador de interrupção de linha

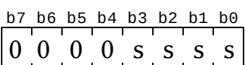
Número da linha da tela em que a interrupção será gerada

### 5.7.5 – Registradores de acesso

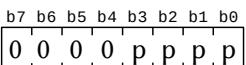
São usados para acessar a VRAM ou outros registradores do VDP.

R#14 –  Registrador de endereço base da VRAM

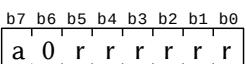
  
a16 ~ a14 (três bits altos de endereço)  
Sempre “00 000”

R#15 –  Apontador para registrador de estado

  
Número do registrador de estado (0~9)  
Sempre “0000”

R#16 –  Apontador para registrador de paleta

  
Número do registrador de paleta (0~15)  
Sempre “0000”

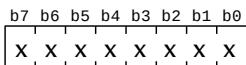
R#17 –  Registrador de controle de ponteiro

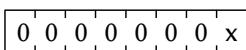
  
Número do registrador (R#0 a R#46)  
Sempre “0”

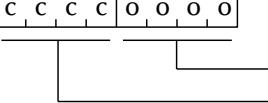
  
0 – autoincremento habilitado  
1 – autoincremento desligado

### 5.7.6 – Registradores de comando

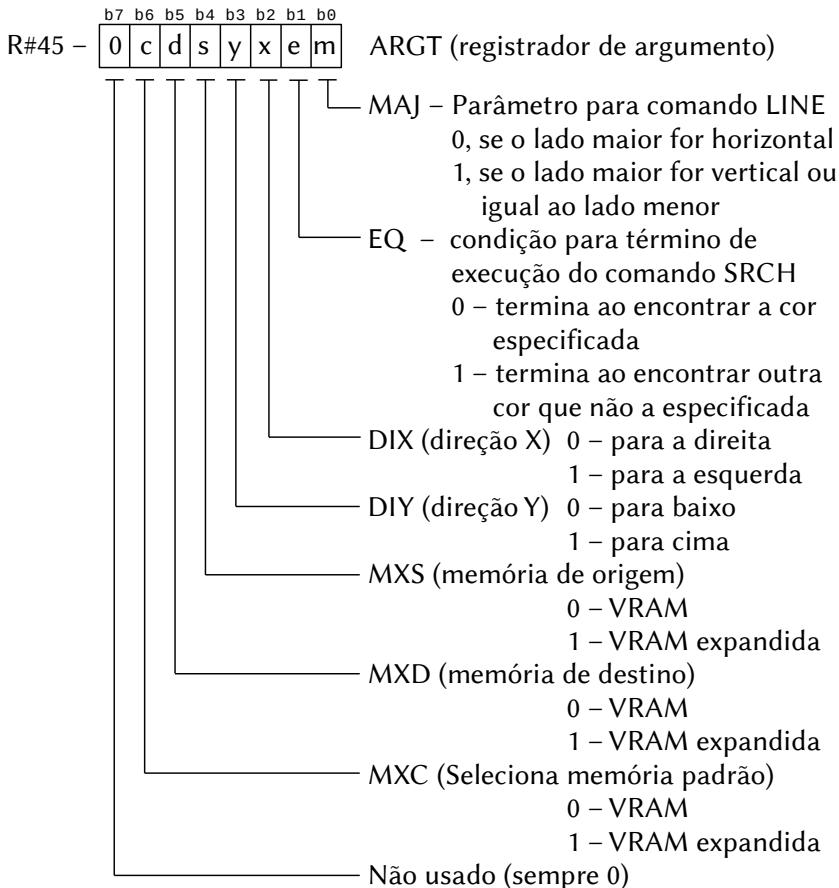
São usados para controlar os comandos de hardware do VDP.

R#32 –  – x7~x0 SX Coordenada horizontal de origem (0 a 511)

R#33 –  – x8

R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY	Coordenada vertical de origem (0 a 1023)								
b7	b6	b5	b4	b3	b2	b1	b0																					
y	y	y	y	y	y	y	y																					
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8																		
0	0	0	0	0	0	y	y																					
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal de destino (0 a 511)								
b7	b6	b5	b4	b3	b2	b1	b0																					
x	x	x	x	x	x	x	x																					
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8																		
0	0	0	0	0	0	0	x																					
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical de destino (0 a 1023)								
b7	b6	b5	b4	b3	b2	b1	b0																					
y	y	y	y	y	y	y	y																					
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8																		
0	0	0	0	0	0	y	y																					
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX	Número de pontos na direção horizontal (0 a 511)								
b7	b6	b5	b4	b3	b2	b1	b0																					
x	x	x	x	x	x	x	x																					
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td></tr></table>	0	0	0	0	0	0	0	x	- x8																		
0	0	0	0	0	0	0	x																					
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY	Número de pontos na direção vertical (0 a 1023)								
b7	b6	b5	b4	b3	b2	b1	b0																					
y	y	y	y	y	y	y	y																					
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td></tr></table>	0	0	0	0	0	0	y	y	- y9~y8																		
0	0	0	0	0	0	y	y																					
R#44 -	<table border="1"><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td><td>c</td><td>c</td></tr><tr><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>c</td><td>c</td></tr><tr><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td><td>c</td></tr></table>	•	•	•	•	c	c	c	c	•	•	•	•	•	•	c	c	c	c	c	c	c	c	c	c	CLR   Gráficos 4 e 6 CLR   Gráfico 5 CLR   Gráficos 7, 8 e 9		Formato dos dados de cores
•	•	•	•	c	c	c	c																					
•	•	•	•	•	•	c	c																					
c	c	c	c	c	c	c	c																					
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>c</td><td>c</td><td>c</td><td>c</td><td>o</td><td>o</td><td>o</td><td>o</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	c	c	c	c	o	o	o	o	CMR (registrador de comando)										
b7	b6	b5	b4	b3	b2	b1	b0																					
c	c	c	c	o	o	o	o																					
		Código de operação lógica Código de comando de hardware																										

(R#45 está descrito na próxima página)



## 5.8 – O V9990 (E-VDP-III)

O VDP V9990 foi lançado em 1992 mas não foi usado em nenhum modelo MSX. Ele tem algumas screens iguais ao V9958, mas não pode ser considerado compatível. Algum tempo depois, foi lançado um cartucho que o utilizava (o GFX9000) mas que também não tinha todos os modos screen do V9958. Mais recentemente foi lançado o cartucho PowerGraph nos mesmos moldes. A tabela abaixo ilustra a “semi-compatibilidade” dos modos de tela do V9990 com os modos do V9958.

Modos de tela		V9958		V9990	
		Modos	Sprites	Modos	Sprites
Screen 0 (40)	Texto 1	Sim	Não	Não	Não
Screen 0 (80)	Texto 2	Sim	Não	Não	Não
Screen 3	Multicor	Sim	256	Não	Não
Screen 1	Gráfico 1	Sim	256	Não	Não
Screen 2	Gráfico 2	Sim	256	Não	Não
Screen 4	Gráfico 3	Sim	256	Não	Não
Screen 5	Gráfico 4	Sim	256	Sim	2
Screen 6	Gráfico 5	Sim	256	Sim	2
Screen 7	Gráfico 6	Sim	256	Sim	2
Screen 8	Gráfico 7	Sim	256	Sim	2
Screen 10/11	Gráfico 8	Sim	256	Sim	2
Screen 12	Gráfico 9	Sim	256	Sim	2

Apesar de não ter todos os modos de tela do V9958, o V9990 tem alguns a mais, além de ser bem mais rápido.

### 5.8.1 – Os registradores do V9990

O V9990 tem 52 registradores de 8 bits para controlar suas operações, numerados de R#0 a R#28 e de R#32 a R#54. O primeiro subgrupo controla as operações de tela e o segundo subgrupo controla os comandos de hardware do VDP. Segue uma descrição resumida de todos os registradores.

R#0	(W)	Endereço de escrita na VRAM (A7 ~ A0)
R#1	(W)	Endereço de escrita na VRAM (A15 ~ A8)
R#2	(W)	Endereço de escrita na VRAM (A18 ~ A16)
R#3	(W)	Endereço de leitura da VRAM (A7 ~ A0)
R#4	(W)	Endereço de leitura da VRAM (A15 ~ A8)
R#5	(W)	Endereço de leitura da VRAM (A18 ~ A16)
R#6	(R/W)	Modo screen #0
R#7	(R/W)	Modo screen #1
R#8	(R/W)	Registrador de controle
R#9	(R/W)	Registrador de interrupção #0
R#10	(R/W)	Registrador de interrupção #1
R#11	(R/W)	Registrador de interrupção #2

R#12	(R/W)	Registrador de interrupção #3
R#13	(W)	Controle de paleta
R#14	(W)	Apontador de paleta
R#15	(R/W)	Back Drop Color (cor de fundo)
R#16	(R/W)	Ajuste de tela
R#17	(R/W)	Scroll vertical da primeira tela (Y7 ~ Y0)
R#18	(R/W)	Scroll vertical da primeira tela (Y12 ~ Y8)
R#19	(R/W)	Scroll horizontal da primeira tela (X2 ~ X0)
R#20	(R/W)	Scroll horizontal da primeira tela (X10 ~ X3)
R#21	(R/W)	Scroll vertical da segunda tela (Y7 ~ Y0)
R#22	(R/W)	Scroll vertical da segunda tela (Y8)
R#23	(R/W)	Scroll horizontal da segunda tela (X2 ~ X0)
R#24	(R/W)	Scroll horizontal da segunda tela (X8 ~ X3)
R#25	(R/W)	Endereço da tabela de padrões dos sprites
R#26	(R/W)	Controle para LCD
R#27	(R/W)	Controle de prioridade
R#28	(W)	Controle da paleta dos sprites
R#32	(W)	Coordenada horizontal / endereço inicial (7 ~ 0)
R#33	(W)	Coordenada horizontal / endereço inicial (10 ~ 8)
R#34	(W)	Coordenada vertical / endereço inicial (7 ~ 0)
R#35	(W)	Coordenada vertical / endereço inicial (11 ~ 8)
R#36	(W)	Coordenada horizontal / endereço final (7 ~ 0)
R#37	(W)	Coordenada horizontal / endereço final (10 ~ 8)
R#38	(W)	Coordenada vertical / endereço final (7 ~ 0)
R#39	(W)	Coordenada vertical / endereço final (11 ~ 8)
R#40	(W)	Contador de transferência horizontal (7 ~ 0)
R#41	(W)	Contador de transferência horizontal (11 ~ 8)
R#42	(W)	Contador de transferência vertical (7 ~ 0)
R#43	(W)	Contador de transferência vertical (11 ~ 8)
R#44	(W)	Registrador de argumento
R#45	(W)	Registrador de operação lógica
R#46	(W)	Máscara de escrita (7 ~ 0)
R#47	(W)	Máscara de escrita (15 ~ 8)
R#48	(W)	Cor de frente (7 ~ 0)
R#49	(W)	Cor de frente (15 ~ 8)
R#50	(W)	Cor de fundo (7 ~ 0)
R#51	(W)	Cor de fundo (15 ~ 8)
R#52	(W)	Registrador de comando
R#53	(R)	Coordenada horizontal da borda (7 ~ 0)

R#54 (R) Coordenada horizontal da borda (10 ~ 8)

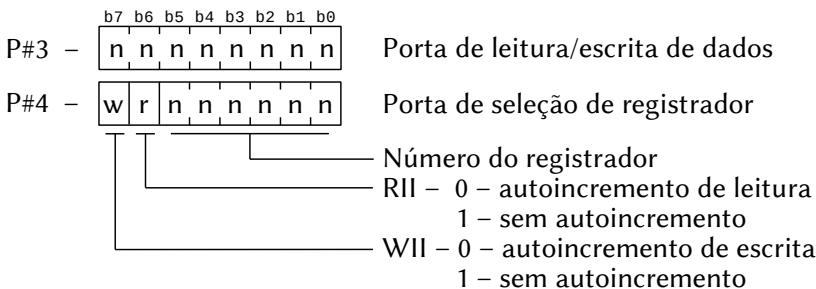
### 5.8.2 – Acesso ao V9990

O acesso ao V9990 é feito diretamente por 12 portas de I/O do Z80, denominadas P#0 a P#B. A função de cada uma está descrita abaixo.

P#0	60H	(R/W)	Acesso à VRAM
P#1	61H	(R/W)	Acesso à paleta de cores
P#2	62H	(R/W)	Acesso aos comandos de hardware
P#3	63H	(R/W)	Acesso aos registradores
P#4	64H	(W)	Seleção de registradores
P#5	65H	(R)	Porta de estado
P#6	66H	(W)	Flag de interrupção
P#7	67H	(W)	Controle do sistema
P#8	68H	(W)	Endereço da Kanji-ROM (low) – 1
P#9	69H	(R/W)	Endereço da Kanji-ROM (high) e dados – 1
P#A	6AH	(W)	Endereço da Kanji-ROM (low) – 2
P#B	6BH	(R/W)	Endereço da Kanji-ROM (high) e dados – 2

#### 5.8.2.1 – Acesso aos registradores

Para escrever um valor num registrador, primeiro é necessário escrever o número do registrador na porta P#4 (64H) e depois o byte de dados na porta P#3 (63H). Para ler o valor de um registrador, basta escrever o número do registrador na porta P#4 (64H) e depois ler o valor respectivo na porta P#3 (63H).

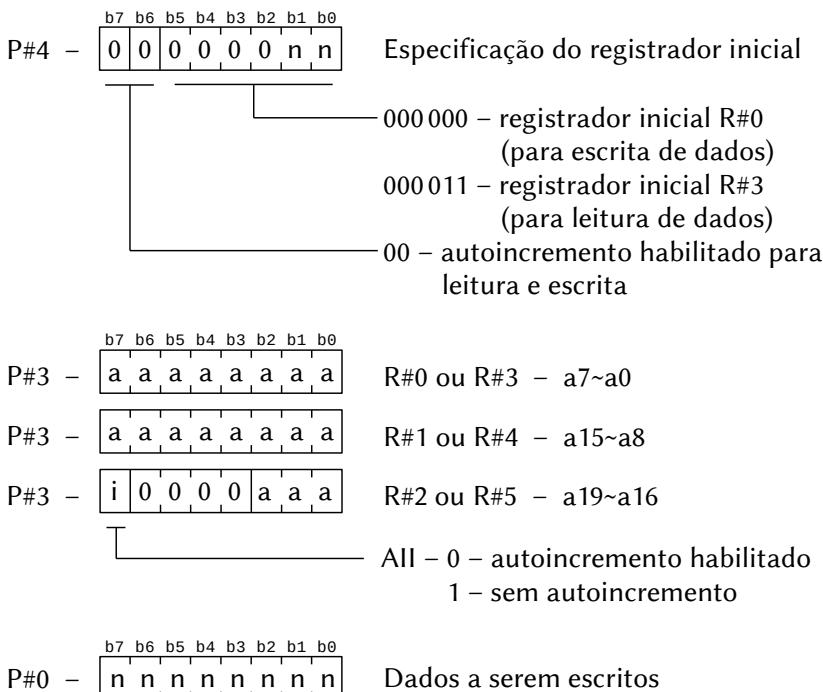


Os bits RII e WII habilitam ou desabilitam o autoincremento de registradores durante a leitura ou escrita, respectivamente. Se forem 0, a função de autoincremento estará ativada e bytes consecutivos escritos ou

lidos através da porta P#3 provocarão o acesso a regis-tradores subsequentes. Se forem 1, os bytes serão sempre enviados para o mesmo regis-trador, indefinidamente. O número do registrador a ser acessado deve ser especificado nos 6 bits mais baixos da porta P#4.

### 5.8.2.2 – Acesso à VRAM

O V9990 pode ser conectado a 128, 256 ou 512 Kbytes de VRAM; por isso o bus de endereços tem 19 bits. Para escrever um byte na VRAM, é preciso carregar os registradores R#0 a R#2 com o endereço a ser escrito e escrever o byte através da porta P#0 (60H). Para ler um byte, os regis-tradores R#3 a R#5 devem ser carregados com o endereço a ser lido e o byte pode ser obtido lendo-se a porta P#0 (60H). A sequência a ser segui-da é a seguinte:



O bit 7 de R#2 ou R#5 habilita ou desabilita autoincremento du-rante a escrita ou leitura da VRAM. Se for 0, ao ser lido ou escrito um

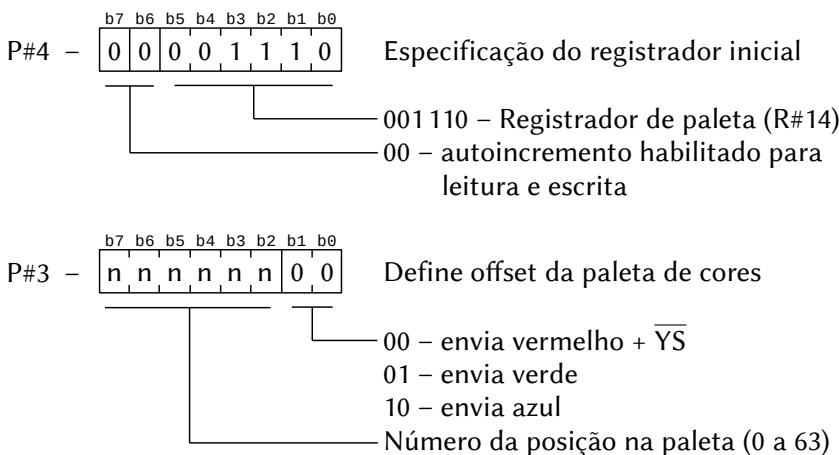
byte pela porta P#0 (60H), o endereço será automaticamente incrementado em 1 e o próximo acesso será no endereço seguinte. Se esse bit for 1, a função de autoincremento será desativada.

### 5.8.2.3 – Acesso à paleta de cores

Para escrever dados nos registradores de paleta, é necessário escrever o número da paleta em R#14 e os valores respectivos de vermelho, verde e azul na porta P#1 (61H). Através da paleta, podem ser definidas até 64 cores escolhidas de 32 768.

O número da posição na paleta deve ser especificado nos 6 bits mais altos de R#14 (0 a 63). Os dois bits mais baixos devem definir qual cor primária será enviada (0=vermelho, 1=verde, 2=azul). Esses dois bits são automaticamente incrementados em 1 cada vez que for escrito um byte de dados na porta P#1; portanto setando-os em 0, basta enviar consecutivamente os valores de vermelho, verde e azul, respectivamente.

A sequência para inserir dados na paleta é a seguinte:



P#1 –		YS + Nível de vermelho (0 a 31)
P#1 –		Nível de verde (0 a 31)
P#1 –		Nível de azul (0 a 31)

O bit YS determina cor transparente, que é um recurso usado para superimpose. Se este bit for 1, a função de transparência estará ativa; se for 0 será apresentada a cor definida na paleta.

Configurações adicionais da paleta devem ser especificadas no Registrador de Controle de Paleta (R#13).

R#13 –		Registrador de controle de paleta
		Offset (PLTO5 ~ PLTO2)
		PLTAIH – controle de incremento
		0 – Ao ler/escrever através de P#1, R#14 será incrementado
		1 – sem incremento
		YAE – 0 – somente modo YJK/YUV
		1 – YJK/YUV mixado com RGB
		PLTM – seleção de modo
		00 – paleta
		01 – 256 cores
		10 – modo YJK
		11 – modo YUV

#### 5.8.2.4 – Acesso à Kanji ROM

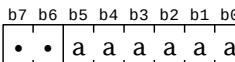
O conteúdo da Kanji-ROM pode ser lido através das portas P#8 (68H) e P#9 (69H) para o conjunto primário (JIS1) e P#A (6AH) e P#B (6BH) para o conjunto secundário (JIS2). Cada conjunto pode conter até 4096 Kanjis ou qualquer outro padrão definido numa célula 16x16.

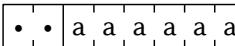
Para realizar a leitura, os seis bits mais altos do código JIS do Kanji respectivo devem ser enviados pela porta P#9/P#B, seguidos dos seis bits mais baixos pela porta P#8/P#A. Em seguida, a porta P#9/P#B

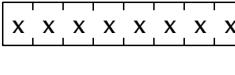
deve ser lida 32 vezes para obter os 32 bytes que compõem o padrão do Kanji.

1ºs 8 bytes	2ºs 8 bytes
3ºs 8 bytes	4ºs 8 bytes

Os 32 bytes que compõem o padrão são obtidos em 4 sequências de 8 bytes, conforme ilustração ao lado, de forma a compor uma célula de 16 x 16 pontos.

P#9 –  a11 ~ a6

P#8 –  a5 ~ a0

P#9 –  A porta P#9 deve ser lida 32 vezes consecutivas para obter o padrão

### 5.8.3 – Modos de tela do V9990

A rigor, o V9990 tem 8 modos de tela, mas cada um deles pode ter várias paletas e modos RGB, YJK ou YUV, de modo que o número de screens acaba sendo bem maior. Com 512K de VRAM, são 20 screens disponíveis (sendo 2 tipo “pattern” e 18 bit-map) mais dois modos não documentados (que desdobraem em 6 screens, conforme a configuração de cores). Existem dois sistemas de screens: Pattern Mode (P1 e P2) e Bit Map Mode (B1 a B6).

#### 5.8.3.1 – Modos por apresentação de padrões

Nome do modo	P1	P2
Frequência mestre	21,5 MHz	21,5 MHz
Frequência de ponto	5,4 MHz	10,7 MHz
Frequência horizontal	15,7 KHz (NTSC)	15,7 KHz (NTSC)
Resolução	256 x 212 pontos	512 x 212 pontos
Número de padrões	32 x 26,5 padrões	64 x 26,5 padrões
Tamanho do padrão	8 x 8 pontos	8 x 8 pontos

Número de screens	2 screens	1 screen
Cores simultâneas	15 + uma transparente	15 + uma transparente
Paletas	4 paletas de 16 cores escolhidas de 32 768	4 paletas de 16 cores escolhidas de 32 768
Área de imagem	64 x 64 padrões	128 x 64 padrões
Padrões selecionáveis	16 384 (máximo)	16 384 (máximo)
Gerador de padrões	128 Kbytes	1535 padr. por screen
	256 Kbytes	3583 padr. por screen
	512 Kbytes	7679 padr. por screen

### 5.8.3.2 – Modos bit-map

Modo	BPP	Tamanho da imagem (128 K)	Tamanho da imagem (256 K)	Tamanho da imagem (512 K)
B1	5,4 MHz	256x256	256x512 512x256	256x1024 512x512 1024x256
		256x512 512x256	256x1024 512x512 1024x256	256x2048 512x1024 1024x512 2048x256
		256x1024 512x512 1024x256	256x2048 512x1024 1024x512 2048x256	256x4096 512x2048 1024x1024 2048x512
		256x2048 512x1024 1024x512 2048x256	256x4096 512x2048 1024x1024 2048x512	256x8192 512x4096 1024x2048 2048x1024
	15,75 KHz (NTSC)			



B6	4	NC	1024x512	1024x1024 2048x512
	2	1024x512	1024x1024 2048x512	1024x2048 2048x1024

Nos modos B1 até B4, a resolução vertical pode ser dobrada usando o recurso de interlace, como no V9938/58. Os modos B2 e B4 são modos overscan, isto é, a imagem apresentada vai além da área de tela; por isso, esses modos não têm borda. Além disso, há duas resoluções verticais possíveis, que também podem ser dobradas pelo recurso de interlace: 240 para NTSC (padrão 525 linhas) e 290 para PAL (padrão 625 linhas). Os modos overscan são úteis para apresentação de animações ou vídeos.

#### 5.8.3.3 – Modos bit-map não documentados

Modo	Freq Ponto	BPP	Tamanho da imagem (128 K)	Tamanho da imagem (256 K)	Tamanho da imagem (512 K)
*B0	Freq Central	16	256x256	256x512 512x256	256x1024 512x512 1024x256
			256x512 512x256	256x1024 512x512 1024x256	256x2048 512x1024 1024x512 2048x256
		4	256x1024 512x512 1024x256	256x2048 512x1024 1024x512 2048x256	256x4096 512x2048 1024x1024 2048x512
			256x2048 512x1024 1024x512 2048x256	256x4096 512x2048 1024x1024 2048x512	256x8192 512x4096 1024x2048 2048x1024
31,5 KHz	Resolução	14,3 MHz	192 x 240/290 (Overscan)		
			15,75 KHz (NTSC)		

<b>*B7</b>	4	NC	1024x512	1024x1024 2048x512
	2	1024x512	1024x1024 2048x512	1024x2048 2048x1024

Os modos acima não foram documentados no “V9990 Application Manual”. Os dois também permitem dobrar a resolução vertical usando o recurso de interlace.

#### 5.8.3.4 – Sistemas de cores

Bits por ponto	b7/b6 de R#13 (PLTM)	Codificação	Número de cores
16 bits / ponto	0 0	RGB YS=1-bit; G=5-bit; R=5-bit; B=5-bit	32.768 simultâneas
8 bits / ponto	0 0	Paleta	64 cores de 32.768
	01	RGB G=3-bit; R=3-bit; B=2-bit	256 simultâneas
	1 0	YJK	19.268 simultâneas
	1 1	YUV	19.268 simultâneas
4 bits / ponto	0 0	Paleta	16 cores de 32.768
2 bits / ponto	0 0	Paleta	4 cores de 32.768

### 5.8.3.5 – Valores dos registradores para cada modo

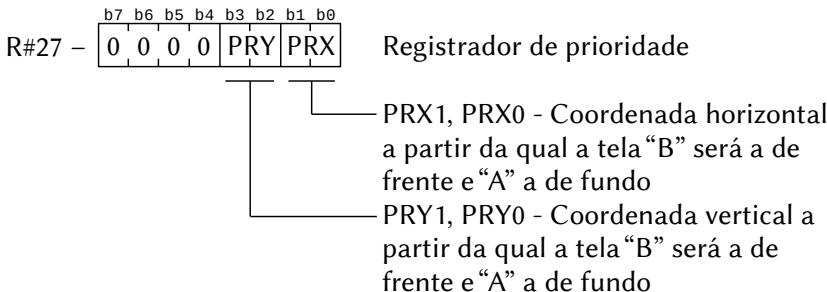
Modo	P#7	R#6				R#7						
	MCS	DSPM	DCKM	XIMM	CLRM	C25M	SM1	SM	PAL	EO	IL	HSCN
P1	0	0	0	1	1	0	0/1	0/1	0/1	0/1	0/1	0
P2	0	1	1	2	1	0	0/1	0/1	0/1	0/1	0/1	0
*B0	1	2	0	0~3	0~3	0	0/1	0/1	0/1	0/1	0/1	0
B1	0	2	0	0~3	0~3	0	0/1	0/1	0/1	0/1	0/1	0
B2	1	2	1	1~3	0~3	0	0/1	0/1	0/1	0/1	0/1	0
B3	0	2	1	1~3	0~3	0	0/1	0/1	0/1	0/1	0/1	0
B4	1	2	2	2~3	0~1	0	0/1	0/1	0/1	0/1	0/1	0
B5	0	2	2	2~3	0~1	0	0	0	0	0	0	1
B6	0	2	2	2~3	0~1	1	0	0	0	0	0	1
*B7	0	2	2	2~3	0~1	0	0/1	0/1	0/1	0/1	0/1	0

Os modos \*B0 e \*B7 não estão documentados no “V9990 Application Manual”

### 5.8.3.6 – Modo P1

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>		
•	•	•	•	•	•	•	0	Controle de sistema	
R#6 –	0	0	0	0	0	1	0	1	Registrador de modo #0
R#7 –	0	0	•	•	•	•	•	0	Registrador de modo #1
R#13 –	0	0	0	•	•	•	•	•	Registrador de paleta

Esse modo tem uma resolução de 256 pontos horizontais por 212 verticais e é mapeado por padrões. Uma característica interessante é que ele tem 2 telas independentes que podem ser sobrepostas (A e B). A área total de imagem, na verdade, é de 512 x 512 pontos para cada tela, mas apenas 256 x 212 são apresentados. O ponto superior esquerdo da tela pode ser localizado na área de imagem pelos registradores de scroll (R#17 a R#20 para a primeira tela, ou “A”, e R#21 a R#24 para a segunda, ou “B”). Existe também um registrador de prioridade (R#27) para essas telas.



Quando PRX e PRY forem 0, a tela “A” será a de frente em toda a área apresentada. Mudando esses valores (00B a 11B), as áreas das telas de frente e de fundo serão movidas em incrementos de 64 pontos. O exemplo abaixo mostra a posição das telas quando PRX=10B e PRY=10B.

		PRX → 01    10    11    00	
PRY	0    64    128    192    256		
↓	0	A   A   B   B	
01	64	A   A   B   B	
10	128	B   B   B   B	
11	192	B   B   B   B	
00	256		

Cada espaço de imagem nas telas “A” e “B” consistem em 4096 (64 x 64) padrões de 8 x 8 pontos, com 16 cores para cada ponto. Para cada um desses padrões, são reservados 2 bytes na Tabela de Nomes para localizá-los na área de imagem.

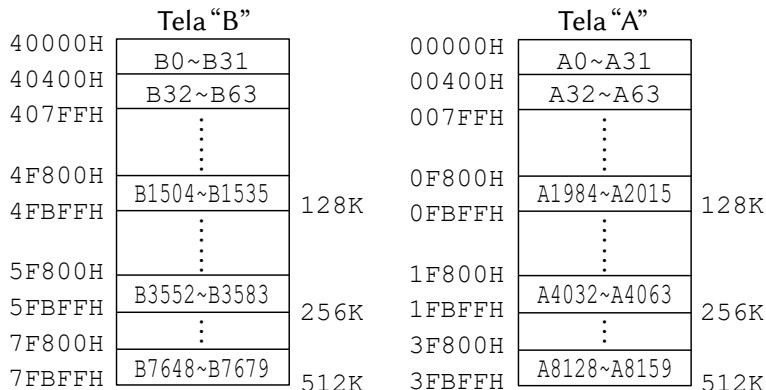
0	8	504	511	0	8	504	511
A0   A1		A62   A63		B0   B1		B62   B63	
A64		A127		B64		B127	
	Tela “A”				Tela “B”		
A3968		A4031		B3968		B4031	
A4032   A4033		A4094   A4095		B4032   B4033		B4094   B4095	
504		511		504		511	
511							

A tabela de nomes começa no endereço 7C000H e vai até o endereço 7DFFFH para a tela “A” e começa em 7E000H indo até 7FFFFH para a tela “B”. Assim, os endereços 7C000H/7C001H correspondem ao padrão A0, 7C000H/7C001H ao padrão A1 e assim sucessivamente, na

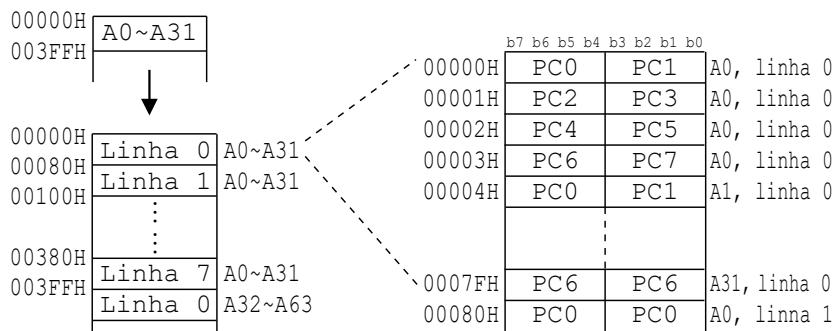
forma low/high. O número máximo de padrões definíveis depende do tamanho da VRAM, conforme tabela abaixo:

VRAM	Tela “A”	Tela “B”
128K	2015	1535
256K	4063	3583
512K	8159	7679

A tabela geradora de padrões começa em 00000H e vai até 3BFFFH para a tela “A” e de 40000H até 7BFFFH para a tela “B”, conforme ilustração abaixo.



A tabela geradora de padrões do modo P1 é organizada como bit map, baseada em 256 pontos horizontais, reservando 4 bits por ponto.



### 5.8.3.7 – Modo P2

O modo P2 é selecionado pelos seguintes registradores:

	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub>	
P#7 –	• • • • • • • 0	Controle de sistema
R#6 –	0 1 0 1 1 0 0 1	Registrador de modo #0
R#7 –	0 0 • • • • • 0	Registrador de modo #1
R#13 –	0 0 0 • • • • •	Registrador de paleta

Esse modo tem uma resolução de 512 pontos horizontais por 212 verticais e é mapeado por padrões. Esse modo tem apenas uma tela, e não duas, como no modo P1. A área total de imagem é de 1024 x 512 pontos embora apenas 512 x 212 sejam apresentados. O ponto superior esquerdo da tela pode ser localizado na área de imagem pelos registradores de scroll (R#17 a R#20). O espaço de imagem consiste em 8192 (128 x 64) padrões de 8 x 8 pontos, com 16 cores para cada ponto. Para cada um desses padrões, são reservados 2 bytes na Tabela de Nomes para localizá-los na área de imagem.

0	8	1008	1023
	P0 P1 P126 P127		
8	P128		P255
	Área de imagem		
504			
511	P8064 P8065	P8190	P8191

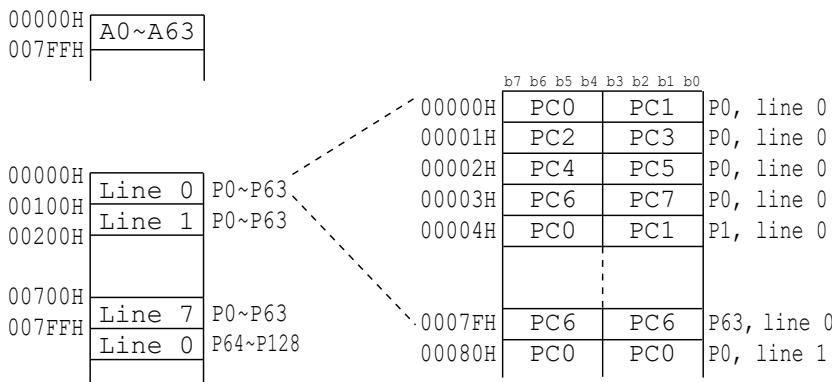
A tabela de nomes começa no endereço 7C000H e vai até o endereço 7FFFFH. Assim, os endereços 7C000H/7C001H correspondem ao padrão P0, 7C000H/7C001H ao padrão P1 assim sucessivamente, na forma low/high. O número máximo de padrões definíveis depende do tamanho da VRAM, conforme tabela abaixo:

VRAM	Padrões
128K	3071
256K	7167
512K	15 359

A tabela geradora de padrões começa em 00000H e vai até 77FFFH, conforme ilustração abaixo.

00000H	P0~P63	
00800H	P64~P127	
00FFFH	:	
17800H		
17FFFH	P3008~P3071	128K
37800H	:	
37FFFH	P7104~P7167	256K
77800H	:	
77BFFH	P15296~P15359	512K

A tabela geradora de padrões do modo P2 é organizada como bit map, baseada em 512 pontos horizontais, reservando 4 bits por ponto.



### 5.8.3.8 – Modo B0

O modo B0 é selecionado pelos seguintes registradores:

	b7 b6 b5 b4 b3 b2 b1 b0	
P#7 –	• • • • • • • 1	Controle de sistema
R#6 –	1 0 0 0 • • • •	Registrador de modo screen #0
R#7 –	0 0 • • • • 0	Registrador de modo screen #1

O modo B0 não é documentado oficialmente, tem uma resolução de 192 x 240 pontos (60Hz) ou 192 x 290 pontos (50Hz) e é um modo overscan. A área de imagem pode variar bastante, indo de 256 até 2048 pontos horizontais, e de 256 a 8192 pontos verticais, dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados nos bits b3~b0 de R#6, conforme ilustração abaixo:

	b7 b6 b5 b4 b3 b2 b1 b0		
R#6 –	1 0 0 0 X C		Registrador de modo screen #0
		CLRM:	00 – 2 bits/ponto 01 – 4 bits/ponto 10 – 8 bits/ponto 11 – 16 bits/ponto
		XIMM:	00 – 256 pontos horizontais 01 – 512 pontos horizontais 10 – 1024 pontos horizontais 11 – 2048 pontos horizontais

O número de pontos verticais da área de imagem é automaticamente setado conforme o valor de XIMM e a quantidade de memória de vídeo disponível.

### 5.8.3.9 – Modo B1

O modo B1 é selecionado pelos seguintes registradores:

P#7 -	<table border="1"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr> <td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>0</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	•	•	•	•	•	0	Controle de sistema
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
•	•	•	•	•	•	•	0											
R#6 -	<table border="1"> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> </table>	1	0	0	0	•	•	•	•	Registrador de modo screen #0								
1	0	0	0	•	•	•	•											
R#7 -	<table border="1"> <tr> <td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>0</td> </tr> </table>	0	0	•	•	•	•	•	0	Registrador de modo screen #1								
0	0	•	•	•	•	•	0											

O modo B1 tem uma resolução de 255 x 212 pontos e é um modo bit map puro. A área de imagem pode variar bastante, indo de 256 até 2048 pontos horizontais, e de 256 a 8192 pontos verticais, dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais da área de imagem é automaticamente setado conforme o valor de XIMM e a quantidade de memória de vídeo disponível.

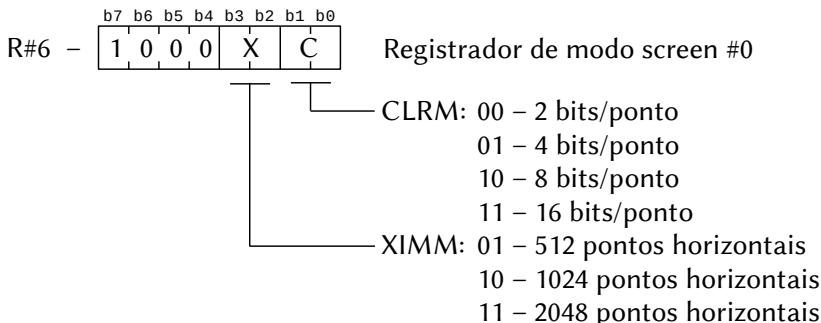
R#6 -	<table border="1"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>X</td><td>C</td><td></td><td></td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	0	0	0	X	C			Registrador de modo screen #0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
1	0	0	0	X	C													
		CLRM: 00 – 2 bits/ponto 01 – 4 bits/ponto 10 – 8 bits/ponto 11 – 16 bits/ponto																
		XIMM: 00 – 256 pontos horizontais 01 – 512 pontos horizontais 10 – 1024 pontos horizontais 11 – 2048 pontos horizontais																

### 5.8.3.9 – Modo B2

O modo B2 é selecionado pelos seguintes registradores:

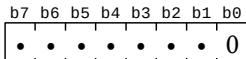
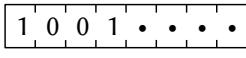
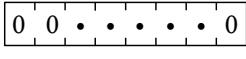
P#7 -	<table border="1"> <tr> <td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr> <td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>1</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	•	•	•	•	•	•	•	1	Controle de sistema
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
•	•	•	•	•	•	•	1											
R#6 -	<table border="1"> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> </table>	1	0	0	0	•	•	•	•	Registrador de modo screen #0								
1	0	0	0	•	•	•	•											
R#7 -	<table border="1"> <tr> <td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>0</td> </tr> </table>	0	0	•	•	•	•	•	0	Registrador de modo screen #1								
0	0	•	•	•	•	•	0											

O modo B2 tem uma resolução de 384 x 240 (60 Hz) ou 384 x 290 (50 Hz) pontos (modo overscan) e sua área de imagem pode variar de 512 a 2048 pontos horizontais e de 256 a 2048 pontos verticais dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.

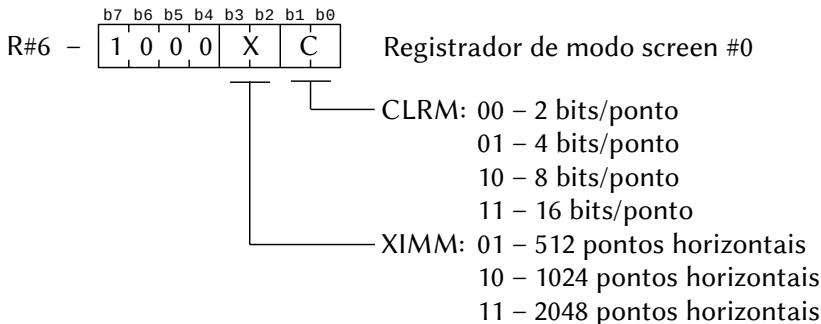


#### 5.8.3.10 – Modo B3

O modo B3 é selecionado pelos seguintes registradores:

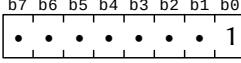
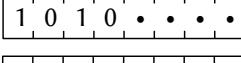
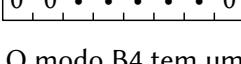
P#7 –		Controle de sistema
R#6 –		Registrador de modo screen #0
R#7 –		Registrador de modo screen #1

O modo B3 tem uma resolução de 512 x 212 pontos. Sua área de imagem varia entre 512 até 2048 pontos horizontais, e de 256 a 4096 pontos verticais, dependendo do tamanho da VRAM e da paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.

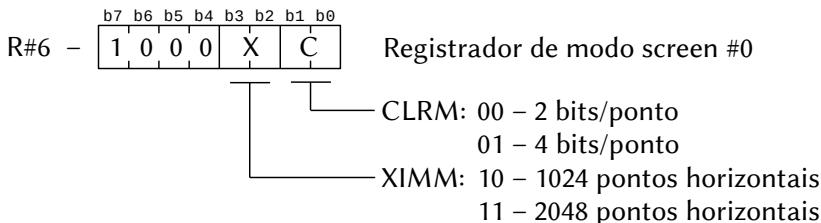


### 5.8.3.11 – Modo B4

O modo B4 é selecionado pelos seguintes registradores:

P#7 –		Controle de sistema
R#6 –		Registrador de modo screen #0
R#7 –		Registrador de modo screen #1

O modo B4 tem uma resolução de 768 x 240 (60 Hz) ou 768 x 290 (50 Hz) pontos (modo overscan) e sua área de imagem pode variar de 1024 a 2048 pontos horizontais e de 256 a 2048 pontos verticais dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.



### 5.8.3.12 – Modo B5

O modo B5 é selecionado pelos seguintes registradores:

	b7 b6 b5 b4 b3 b2 b1 b0	
P#7 –	• • • • • • • 0	Controle de sistema
R#6 –	1 0 1 0 • • • •	Registrador de modo screen #0
R#7 –	0 0 0 0 0 0 0 1	Registrador de modo screen #1

O modo B5 tem 640 x 400 pontos e é um modo de alta resolução, só funcionando em monitores tipo VGA ( $fH=24,8$  KHz). A área de imagem pode variar entre 1024 e 2048 pontos horizontais e 512 e 2048 pontos verticais. dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.

	b7 b6 b5 b4 b3 b2 b1 b0		
R#6 –	1 0 0 0 X C	Registrador de modo screen #0	
			CLRM: 00 – 2 bits/ponto 01 – 4 bits/ponto
			XIMM: 10 – 1024 pontos horizontais 11 – 2048 pontos horizontais

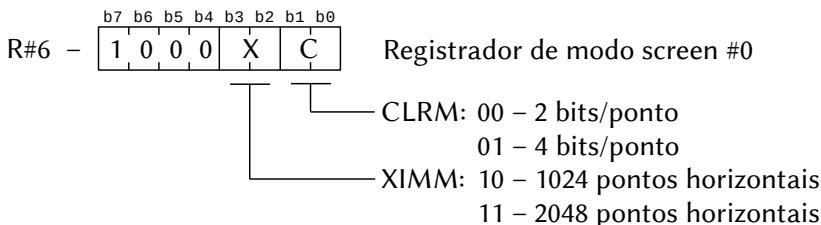
### 5.8.3.13 – Modo B6

O modo B6 é selecionado pelos seguintes registradores:

	b7 b6 b5 b4 b3 b2 b1 b0	
P#7 –	• • • • • • • 0	Controle de sistema
R#6 –	1 0 1 0 • • • •	Registrador de modo screen #0
R#7 –	1 0 0 0 0 0 0 1	Registrador de modo screen #1

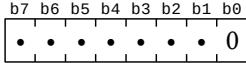
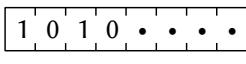
O modo B6 tem 640 x 480 pontos e é um modo de alta resolução, só funcionando em monitores tipo VGA ( $fH=31,5$  KHz). A área de imagem pode variar entre 1024 e 2048 pontos horizontais e 512 e 2048 pontos verticais. dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de

bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.

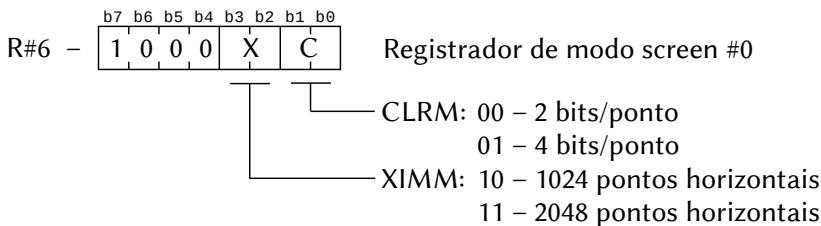


#### 5.8.3.14 – Modo B7

O modo B7 é selecionado pelos seguintes registradores:

P#7 –		Controle de sistema
R#6 –		Registrador de modo screen #0
R#7 –		Registrador de modo screen #1

O modo B7 tem 1024 x 212 pontos e é um modo não documentado oficialmente. Apesar da alta resolução horizontal, funciona em monitores NTSC e PAL, tendo fH = 15,75 KHz. A área de imagem pode variar entre 1024 e 2048 pontos horizontais e 512 e 2048 pontos verticais, dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.



### 5.8.3.15 – Mapas de memória dos modos B0~B7

A área de memória ocupada pelos modos B1~B6 é linear para toda a área de imagem, reservando 512 bytes para a função de cursor, no final da memória disponível, embora mesmo essa área possa ser apresentada. Os pontos são atribuídos linearmente, da esquerda para a direita e depois de cima para baixo, referindo-se à área de imagem e não à área apresentada na tela. A organização dos pontos na memória também depende do número de bits usados para cada ponto. O exemplo abaixo refere-se a uma área de imagem de 511 x 511 pontos, reservando 8 bits para cada ponto.

	0	1	510	511	
0	00000H	00001H	.....	001FEH	0011FFH
1	00200H				003FFH
	.....				.....
511	3FE00H	3FE01H	.....	3FFFFH	3FFFFH

Os pontos são distribuídos na memória de acordo com a paleta usada, conforme ilustrado abaixo.

2 bits por ponto

4 bits por ponto

8 bits por ponto

16 bits por ponto

	b7	b6	b5	b4	b3	b2	b1	b0
00 000H					0,0	low		
00 001H					0,0	high		
00 002H					1,0	low		

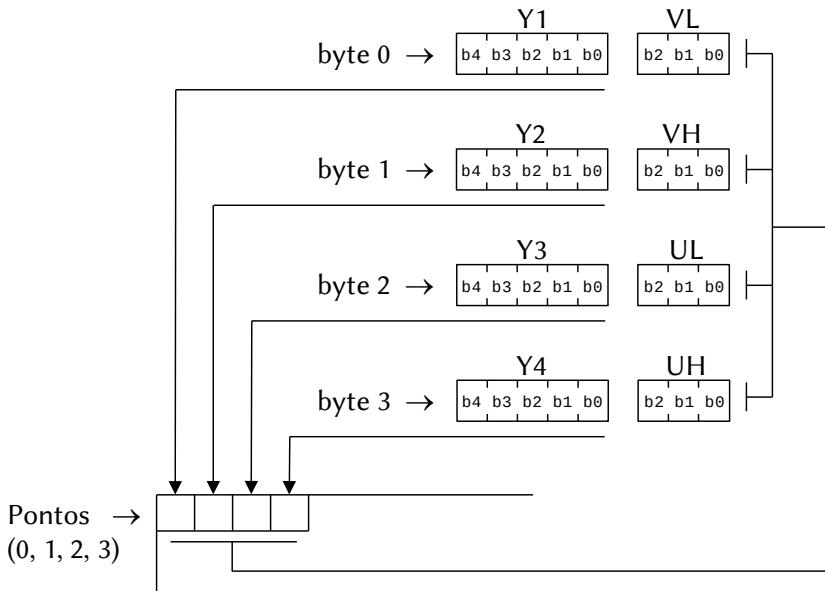
### 5.8.4 – Especificações de cores

Como já descrito, cada ponto pode ocupar 2, 4, 8 ou 16 bits nos modos B1 a B6. Entretanto, existem nove tipos de representações diferentes que podem ser selecionadas. Esses tipos são os seguintes:

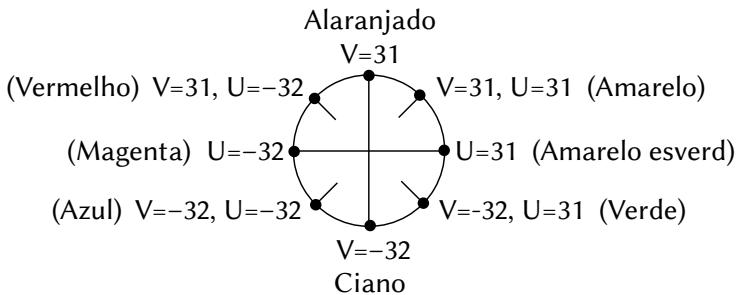
1. BYUV – Modo YUV puro nos modos B1 ~ B6.
2. BYUVP – Modo YUV mais paleta nos modos B1 ~B6.
3. BYJK – Modo YJK puro nos modos B1 ~ B6.
4. BYJKP – Modo YJK mais paleta nos modos B1 ~B6.
5. BD16 – Apresenta 32 768 cores simultâneas sem paleta.
6. BD8 – Apresenta 256 cores simultâneas sem paleta.
7. BP6 – Apresenta 64 cores simultâneas / paleta de 32 768.
8. BP4 – Apresenta 16 cores simultâneas / paleta de 32 768.
9. BP2 – Apresenta 4 cores simultâneas / paleta de 32 768.

#### 5.8.4.1 – Modo BYUV

O modo BYUV apresenta até 19 268 cores simultâneas usando apenas 8 bits por ponto. Para tanto, os pontos são distribuídos em grupos de 4 no sentido horizontal, conforme ilustração abaixo.



As cores são escolhidas pelos vetores U e V, conforme ilustrado logo abaixo.



Os binários correspondentes aos valores mostrados na ilustração são os seguintes:

	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0 →	0	0	0	0	0	0
31 →	0	1	1	1	1	1
-32 →	1	0	0	0	0	0
-1 →	1	1	1	1	1	1

Como há 12 bits para representar a cor, fazemos  $2^{12} = 4096$  cores, que é o número máximo de cores que podem ser definidas. Cada grupo de 4 pontos horizontais só pode ter uma cor escolhida dessas 4096. Entretanto, cada ponto individual desse grupo pode ter uma variação de brilho de 32 níveis, representada pelos bits Y<sub>n</sub>, onde 00 000B representa brilho mínimo e 11 111B brilho máximo.

Os vetores U e V podem variar de -32 a 31, conforme ilustração acima. Com a combinação dos valores extremos, pode-se formar as quatro cores primárias do sistema YUV: verde, vermelho, azul e amarelo. O uso de quatro vetores de cores primárias não altera o sistema de mistura de cores usado pelo sistema RGB; é necessário apenas levar em conta o uso de mais uma cor. Utilizando os valores intermediários, podem ser geradas as 4096 cores. A conversão do sistema YUV para o RGB e vice-versa pode ser feita através das seguintes fórmulas:

$$Y = R/4 + G/2 + B/8$$

$$U = R - Y$$

$$V = G - Y$$

$$R = Y + U$$

$$G = 5/4 Y - 1/2 U - 1/4 V$$

$$B = Y + V$$

Um detalhe importante é quanto ao número de cores. Como há 4096 cores e 32 níveis de brilho para cada uma, na verdade são  $32 * 4096 = 131\,072$  cores possíveis. Acontece que nesse modo as cores não são totalmente independentes para cada ponto (além de características técnicas do V9990 que não vêm ao caso), o que causa uma redução no número de cores apresentadas simultaneamente para 19 268.

O endereço na VRAM de cada ponto da área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X + Y^*(\text{Tamanho horizontal da área em pontos})$$

Onde X é a coordenada horizontal e Y a vertical.

O modo BYUV é selecionado por R#6 e R#13, conforme ilustração abaixo.

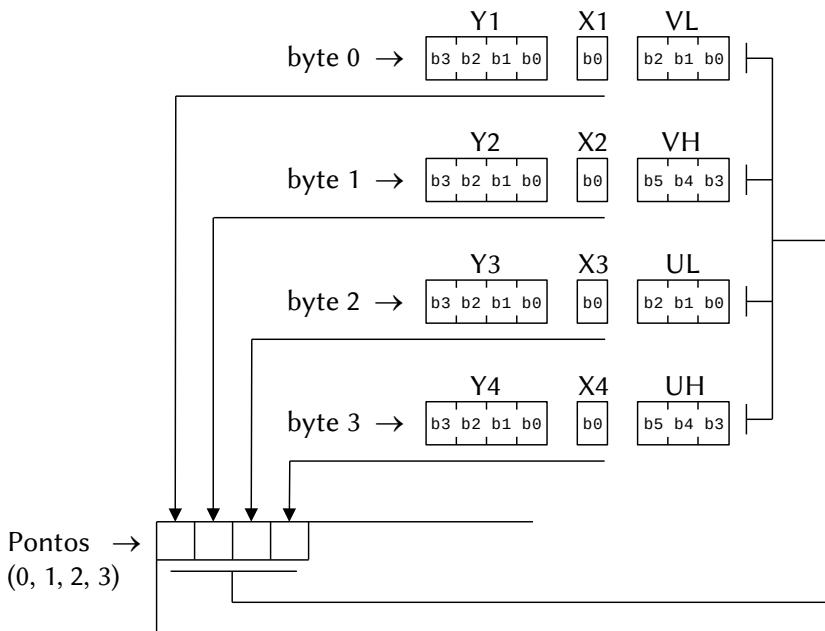
R#6 –  Registrador de modo screen #0

R#13 –  Registrador de paleta

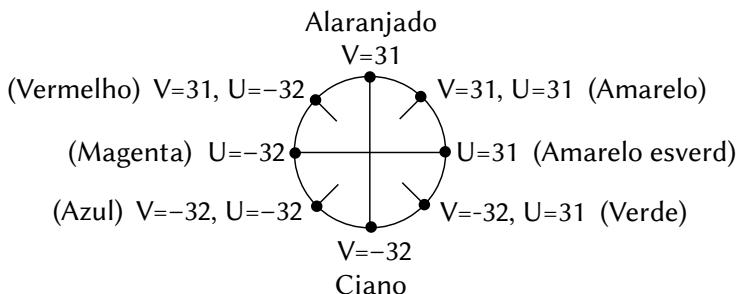
#### 5.8.4.2 – Modo BYUVP

O modo BYUVP é um modo misto, podendo apresentar até 12 499 cores simultâneas, através do sistema YUV, ou então usar a paleta.

Como no sistema YUV, aqui os pontos também estão organizados de quatro em quatro na horizontal. Cada grupo de 4 pontos pode ter uma única cor, escolhidas de 4096, com até 16 níveis de brilho para cada ponto individual. Ou então cada ponto pode ter até 16 cores escolhidas de uma paleta de 32 768. A organização desse modo está ilustrada abaixo.



Quando os bits  $X_n$  forem 0, o sistema usado será o YUV, com a única diferença que a variação de saturação tem apenas 16 níveis, e não 32, como no modo BYUV, já que  $Y_n$  só pode variar de 0 a 15. Já se os bits  $X_n$  forem 1, a cor será escolhida da paleta. Podem ser escolhidas até 16 cores de 32 768. Não é obrigatório que todos os bits  $X_n$  sejam iguais, podendo haver mistura nos 4 pontos que compõem o grupo. Quando o sistema YUV for selecionado, as cores do grupo de pontos são escolhidas pelos vetores  $U$  e  $V$ , de acordo com a ilustração abaixo.



Os binários correspondentes aos valores mostrados na ilustração são os seguintes:

	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0 →	0	0	0	0	0	0
31 →	0	1	1	1	1	1
-32 →	1	0	0	0	0	0
-1 →	1	1	1	1	1	1

Como há 12 bits para representar a cor (6 bits para U e 6 bits para V), fazemos  $2^{12} = 4096$  cores, que é o número máximo de cores que podem ser definidas. Cada grupo de 4 pontos horizontais só pode ter uma cor escolhida dessas 4096. Entretanto, cada ponto individual desse grupo pode ter uma variação de brilho de 16 níveis, representada pelos bits Y<sub>n</sub>, onde 0000B representa brilho mínimo e 1111B brilho máximo.

Um detalhe importante é quanto ao número de cores. Como há 4096 cores e 16 níveis de saturação para cada uma, na verdade são  $16 * 4096 = 65\,536$  cores possíveis. Mas como nesse modo as cores também não são totalmente independentes para cada ponto, (além de características técnicas do V9990 que não vêm ao caso), há uma redução no número de cores apresentadas simultaneamente para 12 499.

O endereço na VRAM de cada ponto da área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X + Y^*(\text{Tamanho horizontal da área em pontos})$$

O modo BYUV é selecionado por R#6 e R#13, conforme ilustração abaixo.

R#6 -	<table border="1"><tr><td>1</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td><td>1</td><td>0</td></tr></table>	1	0	•	•	•	•	1	0	Registrador de modo screen #0
1	0	•	•	•	•	1	0			
R#13 -	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td></tr></table>	1	1	1	•	•	•	•	•	Registrador de paleta
1	1	1	•	•	•	•	•			

### 5.8.4.3 – Modo BYJK

O modo BYJK é exatamente igual ao modo BYUV, exceto pela proporção dos vetores de cor (JK ou UV), onde os canais de cor verde e azul são trocados. No caso do sistema YJK, a conversão para o sistema RGB pode ser feita pelas seguintes fórmulas:

$$Y = R/4 + G/8 + B/2$$

$$R = Y + J$$

$$J = R - Y$$

$$G = Y + K$$

$$K = G - Y$$

$$B = 5/4 Y - J/2 - K/4$$

A conversão entre os sistemas YJK e YUV pode ser feita pelas seguintes fórmulas:

$$Y = Y$$

$$Y = Y$$

$$J = U$$

$$U = J$$

$$K = Y/4 - U/2 - V/4$$

$$V = Y/4 - J/2 - K/4$$

O modo BYJK é selecionado por R#6 e R#13, conforme ilustração abaixo.

R#6 – Registrador de modo screen #0

R#13 – Registrador de paleta

### 5.8.4.4 – Modo BYJKP

O modo BYJKP é exatamente igual ao modo BYUVP, exceto pela proporção dos vetores de cor (JK ou UV), onde os canais de cor verde e azul são trocados e podem ser calculados pelas fórmulas apresentadas acima. Esse modo é selecionado por R#6 e R#13, conforme ilustração abaixo.

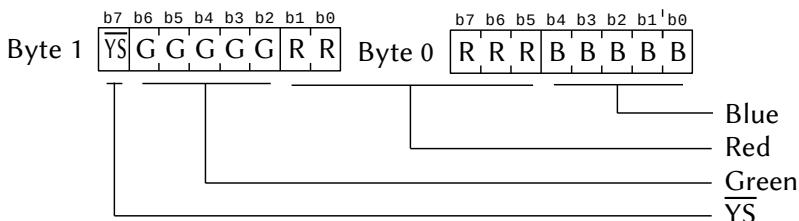
R#6 – Registrador de modo screen #0

R#13 – Registrador de paleta

### 5.8.4.5 – Modo BD16

Nesse modo, podem ser apresentadas até 32 768 cores simultâneas, sem uso de paleta. Dois bytes são reservados para cada ponto, na forma LSB-MSB. São usados 5 bits para cada cor primária, e mais um bit

para a função YS (superimpose). Esses bits são organizados na VRAM de acordo com a ilustração abaixo.



Onde G é a intensidade de verde (00 000B a 11 111B), R a do vermelho e B a do azul. YS é uma flag para indicar superimpose para cada ponto individual. Quando YS for 0, a função de superimpose para o ponto estará desativada; quando for 1 estará ativada. O endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

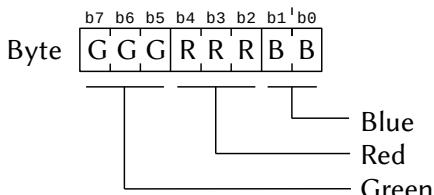
$$\text{ENDEREÇO} = X^*2 + Y^*(\text{tamanho horizontal da área em pontos})^*2$$

O modo BD16 é selecionado pelos seguintes registradores:

R#6 –	<table border="1"><tr><td>1</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td><td>1</td><td>1</td></tr></table>	1	0	•	•	•	•	1	1	Registrador de modo screen #0
1	0	•	•	•	•	1	1			
R#13 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td></tr></table>	0	0	0	•	•	•	•	•	Registrador de paleta
0	0	0	•	•	•	•	•			

#### 5.8.4.6 – Modo BD8

Esse modo é similar ao anterior, mas reserva apenas 8 bits para cada ponto; por isso, podem se apresentadas simultaneamente apenas 256 cores, também sem uso da paleta. Cada byte é organizado conforme ilustração abaixo.



Onde G é a intensidade de verde (0 a 7), R a do vermelho (0 a 7) e B a do azul (0 a 3). O endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X + Y^*(\text{Tamanho horizontal da área em pontos})$$

Esse modo é selecionado pelos seguintes registradores:

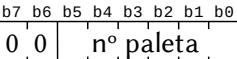
R#6 –  Registrador de modo screen #0

R#13 –  Registrador de paleta

Nesse modo, não há como fazer superimpose.

#### 5.8.4.7 – Modo BP6

Nesse modo, cada ponto pode ter até 64 cores escolhidas de uma paleta de 32 768. Cada ponto é representado por um byte, sendo que apenas os seis bits mais baixos são válidos, conforme ilustração abaixo.

Byte – 

Onde “nº paleta” pode variar de 0 a 63. Esse modo permite superimpose seletivo, desde essa função esteja selecionada para a cor respectiva na paleta (bit  $\overline{YS}$  setado em 1). Nesse caso, todos os pontos com a mesma cor estarão selecionados para superimpose. O endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X + Y^*(\text{Tamanho horizontal da área em pontos})$$

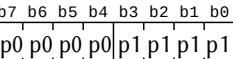
Esse modo é selecionado pelos seguintes registradores:

R#6 –  Registrador de modo screen #0

R#13 –  Registrador de paleta

#### 5.8.4.8 – Modo BP4

Nesse modo, cada ponto pode ter até 16 cores escolhidas de uma paleta de 32 768. A vantagem sobre o modo BP6 é que cada byte é usado para representar dois pontos, ocupando metade da memória do modo BP6, conforme ilustração abaixo.

Byte – 

Onde “p0” representa a paleta do primeiro ponto (0 a 15) e “p1” representa a paleta do ponto seguinte no sentido horizontal, cada uma podendo variar de 0 a 15. Esse modo permite superimpose seletivo, desde essa função esteja selecionada para a cor respectiva na paleta (bit  $\bar{Y}S$  setado em 1). Nesse caso, todos os pontos com a mesma cor estarão selecionados para superimpose. O endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X/2 + Y^*(\text{Tamanho horizontal da área em pontos})/2$$

Ponto par: 4 bits mais altos

Ponto ímpar: 4 bits mais baixos

O modo BP4 é selecionado pelos seguintes registradores:

R#6 – 

1	0	•	•	•	•	0	1
---	---	---	---	---	---	---	---

 Registrador de modo screen #0

R#13 – 

0	0	0	•	•	•	•	•
---	---	---	---	---	---	---	---

 Registrador de paleta

#### 5.8.4.9 – Modo BP2

Nesse modo, cada ponto pode ter até 4 cores escolhidas de uma paleta de 32 768. Ele ocupa apenas metade da memória usada pelo modo BP4, já que cada byte representa 4 pontos no sentido horizontal, reservando apenas 2 bits para cada um, conforme ilustração abaixo.

Byte – 

b7	b6	b5	b4	b3	b2	b1	b0
p0	p0	p1	p1	p2	p2	p3	p3

Onde p0 representa a paleta do primeiro ponto (0 a 3), p1 representa a paleta do segundo (0 a 3), p2 a paleta do terceiro (0 a 3) e p3 a paleta do quarto ponto (0 a 3), sempre no sentido horizontal, da esquerda para a direita. Este modo permite superimpose seletivo, desde essa função esteja selecionada para a cor respectiva na paleta (bit  $\bar{Y}S$  setado em 1). Nesse caso, todos os pontos com a mesma cor estarão selecionados para superimpose. O endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

$$\text{ENDEREÇO} = X/4 + Y^*(\text{Tamanho horizontal da área em pontos})/4$$

Primeiro ponto: bits b7 e b6

Segundo ponto: bits b5 e b4

Terceiro ponto: bits b3 e b2

Quarto ponto: bits b1 e b0

O modo BP2 é selecionado pelos seguintes registradores:

R#6 –  Registrador de modo screen #0

R#13 –  Registrador de paleta

#### 5.8.4.10 – Cores para os modos P1 e P2

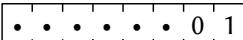
Os modos vistos até agora valem somente para as screens B1 a B6. Para as screens P1 e P2, existe um modo especial, o PP. Existem 4 paletas que podem apresentar, cada uma, 16 cores de 32 768. Duas delas podem ser usadas simultaneamente, de acordo com a tabela abaixo.

Modo P1, plano “A”	bits b1 e b0 de R#13
Modo P1, plano “B”	bits b3 e b2 de R#13
Modo P2, pontos ímpares	bits b1 e b0 de R#13
Modo P2, pontos pares	bits b3 e b2 de R#13

Na verdade, a paleta é uma só e tem 64 posições. O que os 2 bits de R#13 reservados para a seleção de paletas selecionam são os 4 segmentos de 16 posições dentro dessas 64, conforme tabela abaixo.

b1/b3	b0/b2	
0	0	posições 0 a 15
0	1	posições 16 a 31
1	0	posições 32 a 47
1	1	posições 48 a 63

Esses modos também permitem superimpose seletivo, desde essa função esteja selecionada para a cor respectiva na paleta (bit YS setado em 1). Nesse caso, todos os pontos com a mesma cor estarão selecionados para superimpose. O modo PP é selecionado pelos seguintes registradores:

R#6 –  Registrador de modo screen #0

R#13 –  Registrador de paleta

### 5.8.5 – SPRITES E CURSORES

Existem dois modos de sprites que podem ser usados no V9990. Um modo, mais potente, é usado para as screens P1 e P2. O outro é chamado de função de cursor, e é usado para os modos B1 a B6.

#### 5.8.5.1 – Sprites para os modos P1 e P2

Para esses modos, podem ser definidos até 125 sprites 16 x 16 com 16 cores independentes para cada ponto escolhidas de 32 768 (incluindo a cor “transparente”, necessária para dar forma ao sprite). Até 16 sprites podem ser colocados em cada linha horizontal, e todos os 125 podem ser apresentados simultaneamente na tela. No caso do modo P1, a prioridade dos sprites pode ser definida levando-se em conta os dois planos de imagem.

O formato dos sprites é definido através da *Tabela Geradora de Padrões dos Sprites*, e seu endereço inicial é apontado pelo registrador R#25, conforme ilustração abaixo.

R#25 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>a</td><td>a</td><td>a</td><td>0</td></tr></table>	0	0	0	0	a	a	a	0	A17~A15 – (modo P1)
0	0	0	0	a	a	a	0			
R#25 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>a</td><td>a</td><td>a</td><td>a</td></tr></table>	0	0	0	0	a	a	a	a	A18~A15 – (modo P2)
0	0	0	0	a	a	a	a			

A tabela geradora de sprites só pode começar em múltiplos de 16 Kbytes a partir de 00 000H. Para o modo P1, ela tem a seguinte estrutura:

Offset	Offset	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub>
00 000H	00 000H	SC0 SC1
00 080H		⋮
00 100H	00 007H	SC14 SC15
⋮	00 008H	SC0 SC1
00 380H	00 007FH	⋮
003FFH	00 080H	SC14 SC15
Linha 0		SC0 SC1
Linha 1		S0 linha 0
⋮		S0 linha 0
Linha 15		S0 linha 0
Linha 0		S1 linha 0
S0~S15		S1 linha 0
S0~S15		S15 linha 0
S0~S15		S0 linha 1
S16~S31		S0 linha 1

Para o modo P2, há uma ligeira mudança nos endereços.

Offset			Offset	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub>		
00 000H	Linha 0	S0~S31	00 000H	SC0	SC1	S0 linha 0
00 100H	Linha 1	S0~S31				
00 200H	⋮			⋮		
00 F00H	Linha 15	S0~S31	00 007H	SC14	SC15	S0 linha 0
01 000H	Linha 0	S32~S63	00 008H	SC0	SC1	S1 linha 0
				⋮		
			00 0FFH	SC14	SC15	S31 linha 0
			00 100H	SC0	SC1	S0 linha 1

Tabela de Atributos dos Sprites sempre começa em 3FE00H e termina em 3FFFFH para o modo P1 e em 7BE00H/7BFFFH para o modo P2. Ela está organizada conforme a ilustração abaixo.

Offset		Offset	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
3FE00H	Sprite 1	3FE00H	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Coord vert
3FE04H	Sprite 2	3FE00H	P7	P6	P5	P4	P3	P2	P1	P0	Nº padrão
	⋮	3FE00H	X7	X6	X5	X4	X3	X2	X1	X0	Coord hor
3FFECH	Sprite 124	3FE00H	C5	C4	C3	C2			X9	X8	Inform adic
3FFF0H	Sprite 125										

Os sprites podem ser localizados pelos valores de X e Y somente na área de tela (256 x 212 para modo P1 e 512 x 212 para modo P2). A posição vertical do sprite é a posição especificada pelos bits Y mais 1. Para cada sprite, pode ser selecionado um dos 4 segmentos de 16 posições da paleta, através dos bits P1 e P0, conforme tabela abaixo:

P1	P0	
0	0	posições 0 a 15
0	1	posições 16 a 31
1	0	posições 32 a 47
1	1	posições 48 a 63

Conforme ilustrado na tabela acima, toda a paleta pode ser selecionada para os 125 sprites, podendo ter estes até 64 cores de 32 768. En-

tretanto, são possíveis apenas 15 cores por sprite individual mais a cor 0 transparente, necessária para definir o desenho do sprite.

A prioridade de apresentação dos sprites no modo P1 leva em conta os dois planos de tela (A e B), conforme tabela abaixo.

P1	P0	Ordem de prioridade	
0	0	SP > A > B > BD	SP – plano dos sprites A – plano frente imagem B – plano trás imagem BD – plano de fundo
1	0	A > SP > B > BD	
-	1	A > B > BD	<sup>(13)</sup>

Já a prioridade de apresentação no modo P2 segue a tabela abaixo.

P1	P0	Ordem de prioridade	
0	0	SP > IP > BD	SP – plano dos sprites IP – plano de imagem BD – plano de fundo
1	0	IP > SP > BD	
-	1	IP > BD	<sup>(14)</sup>

Em qualquer dos casos, o sprite de número menor tem prioridade de apresentação maior em relação ao de número maior, da mesma forma que para o VDP V9958.

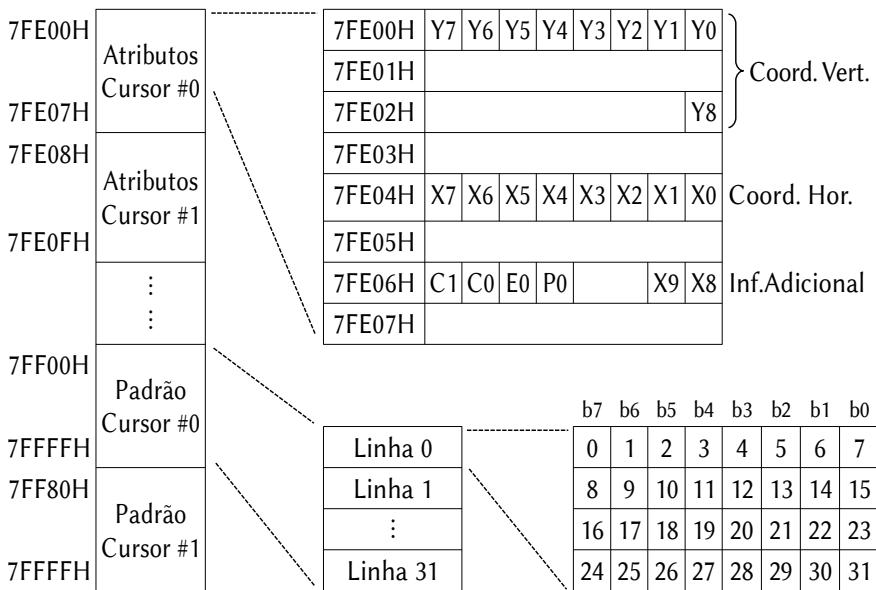
### 5.8.5.2 – Cursores para os modos B0 a B7

Para os modos B1 a B6, existe uma função denominada “cursor”. Na verdade, são sprites com bem menos recursos. Apenas 2 cursores 32 x 32 pontos podem ser definidos, com apenas 1 cor cada. Opcionalmente pode ser feita uma operação XOR entre os pontos do cursor e da imagem. Os cursores são sempre definidos no final da memória, de 7FE00H a 7FFFFH, ocupando 512 bytes. A estrutura dessa tabela está ilustrada abaixo.

---

13 Os sprites não são exibidos quando P0=1

14 Os sprites não são exibidos quando P0=1



A área em que o cursor pode ser apresentado é a área da tela. A posição vertical de apresentação do cursor é igual à coordenada Y definida mais 1 (ou mais 2 no caso de modo entrelaçado). Na verdade, a escolha de cores é bem limitada: apenas uma das 4 cores iniciais da paleta pode ser escolhida, sendo que a cor 0 é transparente. A cor é selecionada pelos bits C1 e C0 (0 a 3). Se o bit EO for setado, uma operação lógica XOR será feita entre os pontos do cursor e os pontos da imagem. P0 é uma flag que indica a apresentação do cursor: se for 0, o cursor será apresentado; se for 1, não será.

### 5.8.6 – COMANDOS DO VDP V9990

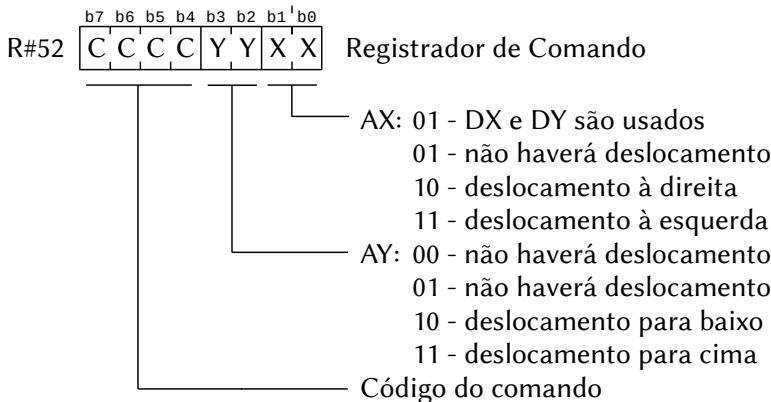
O V9990 também possui comandos de hardware, que funcionam de forma semelhante aos VDPs V9938 e V9958. Esses comandos podem ser executados tanto nos modos P1 e P2 quanto nos modos B1 a B6 e tem por base a área de imagem; portanto seus parâmetros variam conforme a seleção dessa área.

O V9990 tem 15 comandos possíveis de hardware, mais um comando de parada. Quando o respectivo valor for escrito no registrador R#52, o comando começa a ser executado. Os registradores R#32 a R#50 mais R#53 e R#54 devem ser especificados antes da execução do coman-

do. A tabela na página seguinte descreve resumidamente todos os comandos.

0 0 0 0	STOP	Comando de parada
0 0 0 1	LMMC	Transferência da CPU → VRAM (coordenadas)
0 0 1 0	LMMV	Pinta retângulo na VRAM
0 0 1 1	LMCM	Transferência da VRAM → CPU (coordenadas)
0 1 0 0	LMMM	Transferência VRAM ↔ VRAM (coordenadas)
0 1 0 1	CMMC	Transfere caractere da CPU para a VRAM
0 1 1 0	CMMK	Transfere dados da KanjiROM para a VRAM
0 1 1 1	CMMM	Transfere caractere da VRAM para a VRAM
1 0 0 0	BMXL	Transf. VRAM ↔ VRAM (linear → coord.)
1 0 0 1	BMLX	Transf. VRAM ↔ VRAM (coord. → linear)
1 0 1 0	BMLL	Transf. VRAM ↔ VRAM (linear → linear)
1 0 1 1	LINE	Desenha uma linha
1 1 0 0	SRCH	Procura código de cor de um ponto
1 1 0 1	POINT	Lê código de cor de um ponto
1 1 1 0	PSET	Desenha um ponto e avança coordenadas
1 1 1 1	ADVN	Avança coordenadas sem desenhar

O código de comando deve ser escrito em R#52 no seguinte formato (os valores AX e AY só são válidos para os comandos PSET e ADVN):



### 5.8.6.1 – Formatos de dados para os comandos

Para alguns comandos, o formato dos dados a serem enviados para o VDP varia de acordo com a paleta usada. Para os comandos LMMC e LMCM, os dados escritos pela porta P#2 (porta de comando) devem ser especificados conforme ilustração abaixo.

		b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub>
2 bits / ponto	1º byte	1ºpt   2ºpt   3ºpt   4ºpt
	2º byte	5ºpt   6ºpt   7ºpt   8ºpt
4 bits / ponto	1º byte	1ºponto   2ºponto
	2º byte	3ºponto   4ºponto
8 bits / ponto	1º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> 1º ponto
	2º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> 2º ponto
8 bits / ponto	1º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> 1º ponto (low)
	2º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> 1º ponto (high)
	3º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> 2º ponto (low)

Para o comando POINT, o formato para os dados é o seguinte:

		b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub>
2 bits / ponto	byte	cor   inválido
4 bits / ponto	byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> cor   inválido
8 bits / ponto	byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> código de cor
2 bits / ponto	1º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> código de cor (low)
	2º byte	b <sub>7</sub> b <sub>6</sub> b <sub>5</sub> b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub> b <sub>0</sub> código de cor (high)

### 5.8.6.2 – Parâmetros para os comandos

Os parâmetros para a execução de cada um dos comandos devem ser carregados nos registradores R#32 a R#51 antes de enviar o comando respectivo ao registrador R#52. Esses parâmetros estão descritos logo abaixo.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#32 –	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0
	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0
	KA7	KA6	KA5	KA4	KA3	KA2	KA1	KA0
R#33 –	0	0	0	0	0	SX10	SX9	SX8
R#34 –	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8
	KA15	KA14	KA13	KA12	KA11	KA10	KA9	KA8
R#35 –	0	0	0	0	SY11	SY10	SY9	SY8
	0	0	0	0	0	SA18	SA17	SA16
	0	0	0	0	0	0	KA17	KA16

Esses registradores especificam as coordenadas e/ou endereços de início para a execução dos comandos, e podem ser setados de três formas diferentes, dependendo do comando a ser executado.

#### Comandos LMCM, LMMM, BMLX, SRCH e POINT

**SX0~10:** Especifica a coordenada horizontal inicial. Será levada a 0 quando for maior que a largura da área de imagem. No modo P1, o plano “A” será selecionado quando SX9=0 e o plano “B” será selecionado quando SX9=1.

**SY0~10:** Especifica a coordenada vertical inicial. Será levada a 0 quando for maior que a altura da área de imagem.

#### Comandos CMMM, BMXL e BMLL

**SA0~18:** Especifica o endereço inicial da VRAM

## Comando CMMK

**SA0~18:** Especifica o endereço da Kanji-ROM

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#36 –	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
R#37 –	0	0	0	0	0	DX10	DX9	DX8
R#38 –	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
R#39 –	0	0	0	0	DY11	DY10	DY9	DY8
	0	0	0	0	0	DA18	DA17	DA16

Os registradores acima especificam as coordenadas e/ou endereços finais para a execução dos comandos, e podem ser setados de duas formas diferentes, dependendo do comando a ser executado.

**Comandos LMMC, LMMV, LMMM, CMMC, CMMK, CMMM, BMXL, LINE, PSET e ADVN.**

**DX0~10:** Especifica a coordenada horizontal final. Será levada a 0 quando a coordenada especificada for maior que a largura da área de imagem. No modo P1, o plano “A” será selecionado quando DX9=0 e o plano “B” será selecionado quando DX9=1.

**DY0~11:** Especifica a coordenada vertical final. Será levada a 0 quando a coordenada especificada for maior que a altura da área de imagem.

**Comandos BMLX e BMLL**

**DA0~18:** Especifica o endereço final da VRAM.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#40 –	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0
	NA7	NA6	NA5	NA4	NA3	NA2	NA1	NA0
	MJ7	MJ6	MJ5	MJ4	MJ3	MJ2	MJ1	MJ0

R#41 -	0	0	0	0	0	NX10	NX9	NX8
	0	0	0	0	MJ11	MJ10	MJ9	MJ8
R#42 -	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
	NA15	NA14	NA13	NA12	NA11	NA10	NA9	NA8
	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
R#43 -	0	0	0	0	NY11	NY10	NY9	NY8
	0	0	0	0	0	NA18	NA17	NA16
	0	0	0	0	MI11	MI10	MI9	MI8

Os registradores acima especificam o número de pontos ou bytes para a execução dos comandos, e podem ser setados de três formas diferentes, dependendo do comando a ser executado.

### **Comandos LMMC, LMMV, LMCM, LMMM, CMMC, CMMK, CMMM, BMXL e BMLX**

**NX0~10:** Especifica o número de pontos na direção horizontal. Será levada a 0 quando a coordenada especificada for maior que o tamanho da área de imagem. Seu valor máximo é 2048 (todos os bits iguais a 0).

**NY0~11:** Especifica o número de pontos na direção vertical. Será levada a 0 quando a coordenada especificada for maior que a altura da área de imagem. Seu valor máximo é 4096 (todos os bits iguais a 0).

### **Comando BMLL**

**NA0~18:** Especifica o número de bytes a transferir. Será levada a 0 quando seu valor exceder a capacidade da VRAM. Seu valor máximo é 512K (todos os bits iguais a 0).

### **Comando LINE**

**MJ0~11:** Tamanho do lado maior do triângulo retângulo de referência em pontos. Será levada a 0 quando seu valor exceder o tamanho da área de imagem.

**MI0~11:** Tamanho do lado menor do triângulo retângulo de referência em pontos. Será levada a 0 quando seu valor exceder o tamanho da área de imagem.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#44 -	•	•	•	•	DIY	DIX	NEQ	MAJ

Esse é o registrador de argumento.

**DIY:** Direção vertical de transferência. Indica incremento quando for 0 (deslocamento para baixo) e decremento quando for 1 (deslocamento para cima). Com os comandos BMXL e BMLX, o endereço linear é sempre incrementado.

**DIX:** Direção horizontal de transferência. Indica incremento quando for 0 (deslocamento à direita) e decremento quando for 1 (deslocamento à esquerda). Com os comandos BMXL e BMLX, o endereço linear é sempre incrementado e com BMLL, DIX e DIY são especificados igualmente.

**NEQ:** Na especificação de cor da borda para SRCH, 0 indica cor especificada para detecção e 1 cor não especificada.

**MAJ:** Indica a direção do lado maior do triângulo retângulo de referência para o comando LINE. Se for 0, o lado maior será paralelo ao eixo X (horizontal) e se for 1, será paralelo ao eixo Y (vertical).

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#45 -	•	•	•	TP	L11	L10	L01	L00

Esse registrador especifica o código de operação lógica que pode ser feita entre os bits do código de cor da fonte e do destino. Quando o bit TP for 1, os pontos de origem que tiverem o código de cor 0 (transparente) não serão transferidos. Os códigos possíveis estão listados abaixo.

L11-L10-L01-L00	Operação lógica
0 0 0 0	
0 0 0 1	WC = not (SC or DC)
0 0 1 0	
0 0 1 1	WC = not (SC)
0 1 0 0	
0 1 0 1	
0 1 1 0	WC = SC xor DC
0 1 1 1	WC = not (SC and DC)

1 0 0 0	WC = SC and DC
1 0 0 1	WC = not (SC xor DC)
1 0 1 0	
1 0 1 1	
1 1 0 0	WC = SC
1 1 0 1	
1 1 1 0	WC = SC or DC
1 1 1 1	

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#46 –	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0
R#47 –	WM15	WM14	WM13	WM12	WM11	WM10	WM9	WM8

Esses registradores especificam uma máscara de escrita bit a bit. R#46 é a máscara para VRAM0 e R#47 para VRAM1. Para o modo P1, R#46 é a máscara para o plano “A” e R#47 para o plano “B”. Quando o bit desses registradores for 1, a escrita está habilitada para o bit respectivo do dado a ser escrito. Quando o bit dos registradores for 0, a escrita está proibida.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#48 –	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0
R#49 –	FC15	FC14	FC13	FC12	FC11	FC10	FC9	FC8
R#50 –	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
R#51 –	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8

Esses registradores especificam a cor da fonte para os comandos CMMC, CMMK e CMMM e também a cor de desenho para LMMV, LINE e PSET. Para o comando SRCH, especifica a cor da borda através de FC0~FC15. A correspondência na VRAM é a mesma que a da máscara de escrita. FC0~FC15 é o código de cor para a fonte de dados #1 e BC0~BC15 para a fonte de dados #0. O formato dos códigos de cores deve ser setado de acordo com o valor contido em R#6, conforme abaixo:

- 16 bits por ponto: Todos os bits são válidos  
 8 bits por ponto: Mesmo dado para 0~7 e 8~15  
 4 bits por ponto: Mesmo dado para 0~3, 4~7, 8~11 e 12~15  
 2 bits por ponto: Preencher 0~15 oito vezes com os 2 bits

### 5.8.6.3 – Executando os comandos

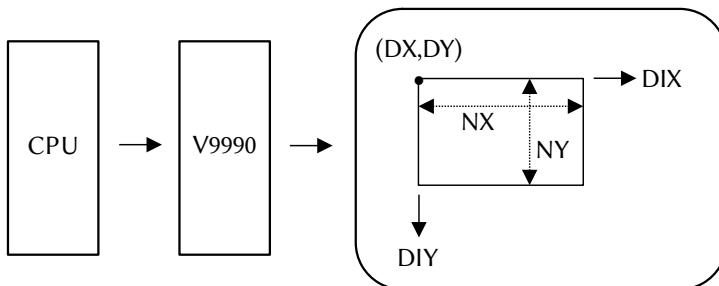
Para executar os comandos de hardware do V9990, primeiro, é necessário setar todos os valores nos registradores adequados. Depois, basta escrever o código do comando em R#52, juntamente com os dados para deslocamento de pontos para os comandos PSET e ADVN. Para interromper o comando, é só escrever o comando de parada em R#52 (00H).

Enquanto o comando estiver sendo executado, o bit CE da porta P#5 ficará setado em 1.



### 5.8.6.4 – LMMC (Transferência Lógica CPU → VRAM)

Nesse comando, os dados são transferidos da CPU para uma área retangular na VRAM.



Os registradores devem ser carregados de acordo com a ilustração abaixo.

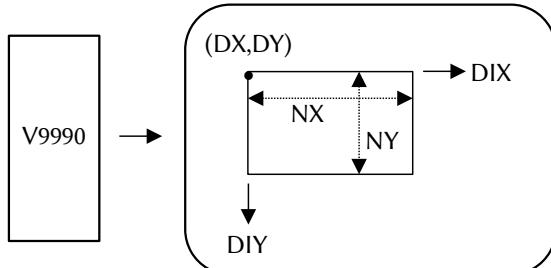
R#36 –	<table border="1"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr> <tr> <td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal de destino (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#37 –	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr> </table>	0	0	0	0	0	x	x	x	- x10~x8										
0	0	0	0	0	x	x	x													

R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical de destino (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX	Número de pontos a transferir na direção horizontal
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x8										
0	0	0	0	0	x	x	x													
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY	Número de pontos a transferir na direção vertical
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y9~y8										
0	0	0	0	y	y	y	y													
R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	- DIY,DIX		Direção da transferência
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	x	0	0													
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	- LOP		Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	t	1	1	1	1													
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0		
b7	b6	b5	b4	b3	b2	b1	b0													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8		Máscara de escrita								
w	w	w	w	w	w	w	w													
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	1	0	0	0	0	- OP-CODE	Cód. comando LMMC	
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	1	0	0	0	0													

Ao executar o comando, o número necessário de bytes a transferir deverá ser enviado pela porta de comando (P#2).

### 5.8.6.5 – LMMV (Desenha retângulo)

Esse comando desenha um retângulo na área de imagem.

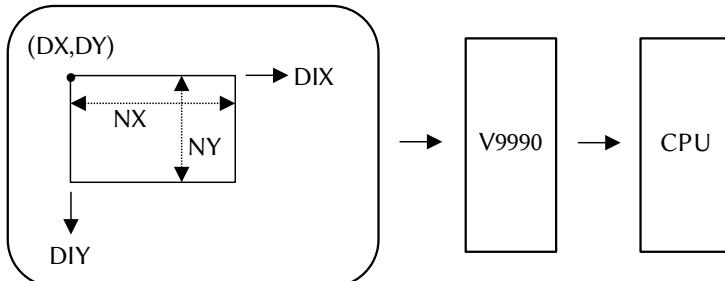


Os registradores devem ser carregados de acordo com a ilustração abaixo.

R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal de destino (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	0	x	x	x	- x10~x8		
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	0	x	x	x													
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical de destino (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	y	y	y	- y11~y8		
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	y	y	y													
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX	Número de pontos a pintar na direção horizontal
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	0	x	x	x	- x8		
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	0	x	x	x													
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY	Número de pontos a pintar na direção vertical
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	y	y	y	- y9~y8		
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	y	y	y													
R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	- D1Y,D1X		Direção da transferência
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	x	0	0													
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	- LOP		Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	t	1	1	1	1													
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0		Máscara de escrita
b7	b6	b5	b4	b3	b2	b1	b0													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm15~wm8		
b7	b6	b5	b4	b3	b2	b1	b0													
w	w	w	w	w	w	w	w													
R#48 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	f	f	f	f	f	f	f	f	- fc7~fc0		
b7	b6	b5	b4	b3	b2	b1	b0													
f	f	f	f	f	f	f	f													
R#49 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	f	f	f	f	f	f	f	f	- fc15~fc8		Cód. de cor para pintura
b7	b6	b5	b4	b3	b2	b1	b0													
f	f	f	f	f	f	f	f													
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	1	0	0	0	0	0	- OP-CODE		Cód. comando LMMV
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	1	0	0	0	0	0													

### 5.8.6.6 – LMCM (Transferência lógica VRAM → CPU)

Nesse comando, os dados de uma área retangular na VRAM são transferidos para a CPU. O comando LMCM está ilustrado abaixo.



Os seguintes registradores devem ser carregados:

R#32 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	– x7~x0	DX	Coordenada horizontal inicial para transferência (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#33 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	– x10~x8										
0	0	0	0	0	x	x	x													
R#34 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	– y7~y0	DY	Coordenada vertical inicial para transferência (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#35 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	– y11~y8										
0	0	0	0	y	y	y	y													
R#40 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	– x7~x0	NX	Número de pontos a transferir na direção horizontal
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	– x8										
0	0	0	0	0	x	x	x													
R#42 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	– y7~y0	NY	Número de pontos a transferir na direção vertical
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	– y9~y8										
0	0	0	0	y	y	y	y													
R#44 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	– DIY,DIX		Direção da transferência
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	x	0	0													
R#45 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	– LOP		Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	t	1	1	1	1													

R#52 - 

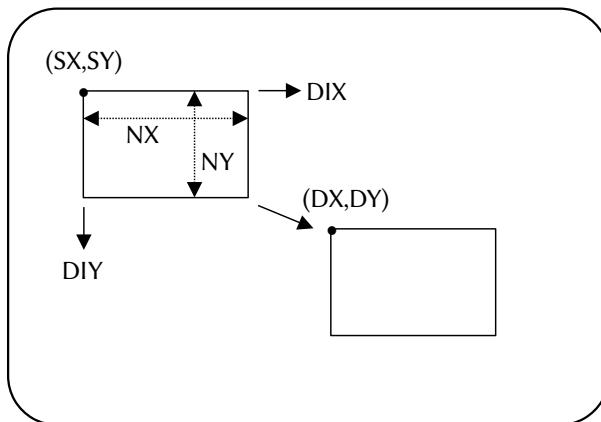
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	1	1	0	0	0	0

 - OP-CODE Cód. comando LMCM

Os bytes de dados a serem transferidos devem ser enviados pela porta de comando (P#2).

#### 5.8.6.7 – LMMM (Transferência lógica VRAM → VRAM)

Nesse comando, uma área retangular da VRAM é transferida para outra posição na VRAM. Operações lógicas no destino são possíveis.



Os seguintes registradores devem ser carregados:

R#32 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	SX	Coordenada horizontal inicial para transferência (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
x	x	x	x	x	x	x	x												
R#33 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- x10~x8	SY	Coordenada vertical inicial para transferência (0 a 4095)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	0	x	x	x												
R#34 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DX	Coordenada horizontal de destino para transferência (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
y	y	y	y	y	y	y	y												
R#35 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	y	y	y	- y11~y8		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	y	y	y	y												
R#36 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
x	x	x	x	x	x	x	x												
R#37 - <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- x10~x8		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	0	x	x	x												

R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7~y0</sub>	DY	Coordenada vertical de destino para transferência (0 a 4095)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y <sub>11~y8</sub>										
0	0	0	0	y	y	y	y													
R#40 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x <sub>7~x0</sub>	NX	Número de pontos a transferir na direção horizontal
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x <sub>8</sub>										
0	0	0	0	0	x	x	x													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y <sub>7~y0</sub>	NY	Número de pontos a transferir na direção vertical
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y <sub>9~y8</sub>										
0	0	0	0	y	y	y	y													
R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	x	0	0	- DIY,DIX		Direção da transferência
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	0	y	x	0	0													
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>I</td><td>I</td><td>I</td><td>I</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	t	I	I	I	I	- LOP		Cód. de operação lógica
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	t	I	I	I	I													
R#46 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	w	w	w	w	w	w	w	w	- w <sub>m7~wm0</sub>		Máscara de escrita
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- w <sub>m15~wm8</sub>										
w	w	w	w	w	w	w	w													
R#52 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	1	0	0	0	0	0	0	- OP-CODE	Cód. comando LM	MMM
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	1	0	0	0	0	0	0													

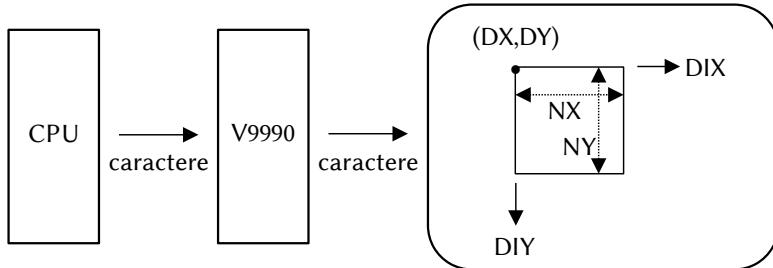
### 5.8.6.8 – CMMC (Transferência de caractere → VRAM)

Nesse comando, caracteres são transferidos da CPU para uma área retangular da VRAM. Operações lógicas no destino são possíveis. O formato do caractere é de 16x16 pontos e sua organização é igual à dos caracteres da Kanji-ROM. Veja a ilustração abaixo.

1ºs 8 bytes	2ºs 8 bytes
3ºs 8 bytes	4ºs 8 bytes

Os 32 bytes que compõem o padrão devem estar organizados em 4 sequências de 8 bytes, conforme ilustração ao lado, de forma a compor uma célula de 16 x 16 pontos.

Logo abaixo está a ilustração de como o comando CMMC é executado.



Os registradores a serem carregados estão ilustrados abaixo.

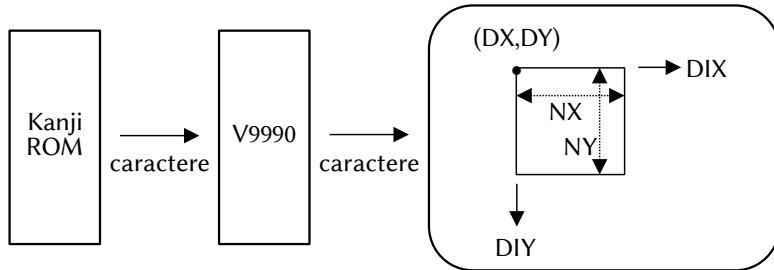
R#36 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal de destino para transferência (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8										
0	0	0	0	0	x	x	x													
R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical de destino para transferência (0 a 4095)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													
R#40 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	NX	Largura da célula do caractere (número de pontos horizontais)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x8										
0	0	0	0	0	x	x	x													
R#42 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	NY	Altura da célula do caractere (número de pontos verticais)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y9~y8										
0	0	0	0	y	y	y	y													
R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	x	0	0	- DIY,DIX	Direção da transferência	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	0	y	x	0	0													
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	t	1	1	1	1	- LOP	Cód. de operação lógica	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	t	1	1	1	1													
R#46 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	w	w	w	w	w	w	w	w	- wm7~wm0	Máscara de escrita	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8										
w	w	w	w	w	w	w	w													

R#48 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	f	f	f	f	f	f	f	f	- fc7~fc0	Código de cor do caractere a ser transferido para a fonte #1
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
f	f	f	f	f	f	f	f												
R#49 -	<table border="1"><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	f	f	f	f	f	f	f	f	- fc15~fc8									
f	f	f	f	f	f	f	f												
R#50 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	b	b	b	b	b	b	b	b	- bc7~bc0	Código de cor do caractere a ser transferido para a fonte #0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
b	b	b	b	b	b	b	b												
R#51 -	<table border="1"><tr><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td></tr></table>	b	b	b	b	b	b	b	b	- bc15~bc8									
b	b	b	b	b	b	b	b												
R#52 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	1	0	1	0	0	0	0	- OP-CODE Cód. comando CMMC	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	1	0	1	0	0	0	0												

Os bytes de dados a serem transferidos devem ser enviados pela porta de comando (P#2).

#### 5.8.6.9 – CMMK (Transf. de caractere Kanji → VRAM)

Os caracteres Kanji estão organizados da forma mostrada no comando CMMC anterior.



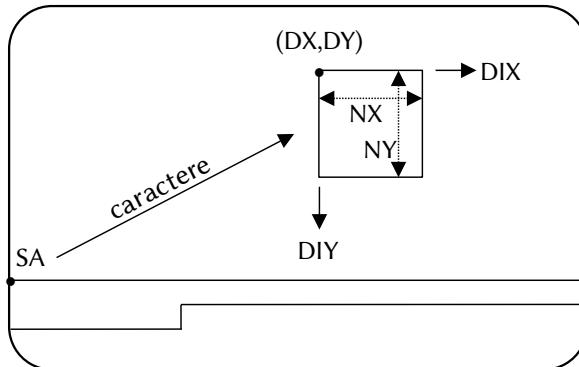
Os registradores a serem carregados estão ilustrados abaixo.

R#32 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	k	k	k	k	k	k	k	k	- ka7~ka0	Endereço do caractere Kanji a ser transferido para a VRAM
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
k	k	k	k	k	k	k	k												
R#34 -	<table border="1"><tr><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td><td>k</td></tr></table>	k	k	k	k	k	k	k	k	- ka15~ka8									
k	k	k	k	k	k	k	k												
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>k</td><td>k</td></tr></table>	0	0	0	0	0	0	k	k	- ka17~ka16									
0	0	0	0	0	0	k	k												
R#36 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	Coordenada horizontal de destino para transferência (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
x	x	x	x	x	x	x	x												
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8	DX								
0	0	0	0	0	x	x	x												

R#38 -		- y7~y0	DY	Coordenada vertical de destino para transferência (0 a 4095)
R#39 -		- y11~y8		
R#40 -		- x7~x0	NX	Largura da célula do caractere (número de pontos horizontais)
R#41 -		- x8		
R#42 -		- y7~y0	NY	Altura da célula do caractere (número de pontos verticais)
R#43 -		- y9~y8		
R#44 -		- DIY,DIX		Direção da transferência
R#45 -		- LOP		Cód. de operação lógica
R#46 -		- wm7~wm0		
R#47 -		- wm15~wm8		Máscara de escrita
R#48 -		- fc7~fc0		Código de cor do caractere kanji a ser transferido p/ a fonte #1
R#49 -		- fc15~fc8		
R#50 -		- bc7~bc0		Código de cor do caractere kanji a ser transferido p/ a fonte #0
R#51 -		- bc15~bc8		
R#52 -		- OP-CODE		Cód. comando CMMK

### 5.8.6.10 – CMMM (Transf. de caractere VRAM → VRAM)

Nesse comando, um caractere é transferido de uma área linear da VRAM para uma área retangular na VRAM. Operações lógicas no destino são possíveis. O caractere tem o mesmo formato mostrado no comando CMMC.



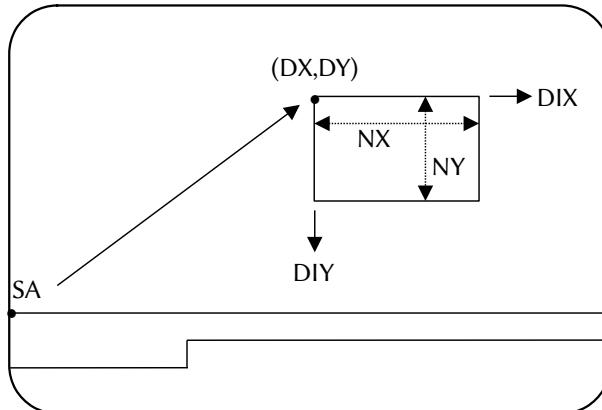
Para executar o comando CMMM, os seguintes registradores devem ser carregados:

R#32 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	s	s	s	s	s	s	s	s	- sa7~sa0	Endereço do caractere na VRAM a ser transferido para as coordenadas
b7	b6	b5	b4	b3	b2	b1	b0												
s	s	s	s	s	s	s	s												
R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	s	s	s	s	s	s	s	s	- sa15~sa8	
b7	b6	b5	b4	b3	b2	b1	b0												
s	s	s	s	s	s	s	s												
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>s</td><td>s</td><td>s</td></tr></table>	0	0	0	0	0	0	s	s	s	- sa17~sa16								
0	0	0	0	0	0	s	s	s											
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	Coordenada horizontal de destino para transferência (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	0	x	x	x	- x10~x8	DX							
0	0	0	0	0	0	x	x	x											
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8	Coordenada vertical de destino para transferência (0 a 4095)								
0	0	0	0	y	y	y	y												
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	0	x	x	x	- x8	Largura da célula do caractere (número de pontos horizontais)							
0	0	0	0	0	0	x	x	x											
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y9~y8	Altura da célula do caractere (número de pontos verticais)								
0	0	0	0	y	y	y	y												

R#44 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	x	0	0	– DIY,DIX	Direção da transferência
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	y	x	0	0												
R#45 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	t	1	1	1	1	– LOP	Cód. de operação lógica
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	t	1	1	1	1												
R#46 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	w	w	w	w	w	w	w	w	– wm7~wm0	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
w	w	w	w	w	w	w	w												
R#47 –	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	– wm15~wm8	Máscara de escrita								
w	w	w	w	w	w	w	w												
R#48 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	f	f	f	f	f	f	f	f	– fc7~fc0	Código de cor do caractere a ser transferido para a fonte #0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
f	f	f	f	f	f	f	f												
R#49 –	<table border="1"><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	f	f	f	f	f	f	f	f	– fc15~fc8									
f	f	f	f	f	f	f	f												
R#50 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	b	b	b	b	b	b	b	b	– bc7~bc0	Código de cor do caractere a ser transferido para a fonte #1
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
b	b	b	b	b	b	b	b												
R#51 –	<table border="1"><tr><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td><td>b</td></tr></table>	b	b	b	b	b	b	b	b	– bc15~bc8									
b	b	b	b	b	b	b	b												
R#52 –	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	1	1	1	0	0	0	0	– OP-CODE	Cód. comando CMMM
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	1	1	1	0	0	0	0												

### 5.8.6.11 – BMXL (Transf. de bytes linear → coordenadas)

Nesse comando, bytes de dados são transferidos de uma área linear da VRAM para uma área retangular na VRAM. Operações lógicas no destino são possíveis.

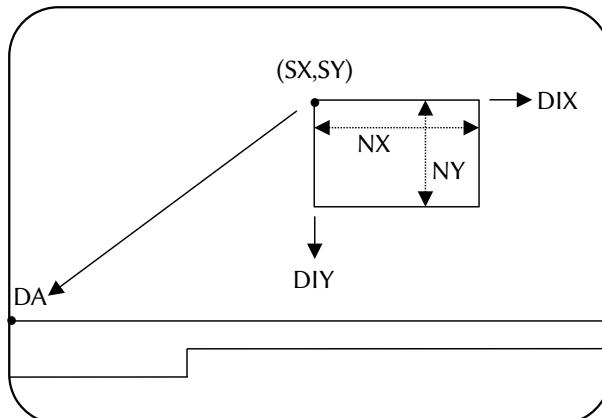


Os seguintes registradores devem ser carregados:

R#32 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	s	s	s	s	s	s	s	s	- sa7~sa0	
b7	b6	b5	b4	b3	b2	b1	b0												
s	s	s	s	s	s	s	s												
R#34 -	<table border="1"><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	s	s	s	s	s	s	s	s	- sa15~sa8	Endereço linear na VRAM								
s	s	s	s	s	s	s	s												
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>s</td><td>s</td><td>s</td></tr></table>	0	0	0	0	0	s	s	s	- sa18~sa16									
0	0	0	0	0	s	s	s												
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8	Coordenada horizontal de destino para transferência (0 a 2047)								
0	0	0	0	0	x	x	x												
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8	Coordenada vertical de destino para transferência (0 a 4095)								
0	0	0	0	y	y	y	y												
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	NX
b7	b6	b5	b4	b3	b2	b1	b0												
x	x	x	x	x	x	x	x												
R#41 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x8	Número de pontos a transferir na direção horizontal								
0	0	0	0	0	x	x	x												
R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY
b7	b6	b5	b4	b3	b2	b1	b0												
y	y	y	y	y	y	y	y												
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y9~y8	Número de pontos a transferir na direção vertical								
0	0	0	0	y	y	y	y												
R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	- DIY,DIX	Direção da transferência
b7	b6	b5	b4	b3	b2	b1	b0												
0	0	0	0	y	x	0	0												
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	- LOP	Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0												
0	0	0	t	1	1	1	1												
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0	
b7	b6	b5	b4	b3	b2	b1	b0												
w	w	w	w	w	w	w	w												
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8	Máscara de escrita								
w	w	w	w	w	w	w	w												
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	1	0	0	0	0	0	0	0	- OP-CODE	Cód. comando BMXL
b7	b6	b5	b4	b3	b2	b1	b0												
1	0	0	0	0	0	0	0												

### 5.8.6.12 – BMLX (Transf. de bytes coordenadas → linear)

No comando BMLX, bytes de dados são transferidos de uma área retangular da VRAM para uma área linear na VRAM. Operações lógicas no destino são possíveis.



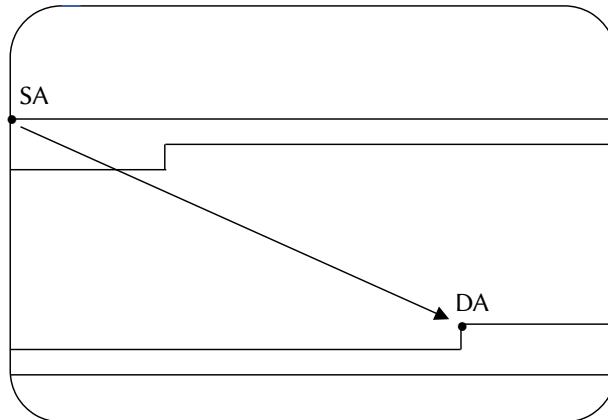
Os seguintes registradores devem ser carregados:

R#32 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	– x7~x0	SX	Coordenada horizontal de origem para transferência (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#33 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	– x10~x8										
0	0	0	0	0	x	x	x													
R#34 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	– y7~y0	SY	Coordenada vertical de origem para transferência (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#35 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	– y11~y8										
0	0	0	0	y	y	y	y													
R#36 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	d	d	d	d	d	d	d	d	– da7~da0		
b7	b6	b5	b4	b3	b2	b1	b0													
d	d	d	d	d	d	d	d													
R#38 –	<table border="1"><tr><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td></tr></table>	d	d	d	d	d	d	d	d	– da15~da8		Endereço linear na VRAM								
d	d	d	d	d	d	d	d													
R#39 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>d</td><td>d</td><td>d</td></tr></table>	0	0	0	0	0	d	d	d	– da18~da16										
0	0	0	0	0	d	d	d													
R#40 –	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	– x7~x0	NX	Número de pontos a transferir na direção horizontal
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#41 –	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	– x8										
0	0	0	0	0	x	x	x													

R#42 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	NY	Número de pontos a transferir na direção vertical
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y9~y8										
0	0	0	0	y	y	y	y													
R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	- DIY,DIX	Direção da transferência	
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	0	y	x	0	0													
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>I</td><td>I</td><td>I</td><td>I</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	I	I	I	I	- LOP	Cód. de operação lógica	
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	t	I	I	I	I													
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0		Máscara de escrita
b7	b6	b5	b4	b3	b2	b1	b0													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8										
w	w	w	w	w	w	w	w													
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	1	0	0	1	0	0	0	0	- OP-CODE	Cód. comando BMLX	
b7	b6	b5	b4	b3	b2	b1	b0													
1	0	0	1	0	0	0	0													

### 5.8.6.13 – BMLL (Transf. de bytes linear → linear)

Nesse comando, um bloco de dados de uma área linear da VRAM é transferido para outra área linear. Operações lógicas no destino são possíveis.

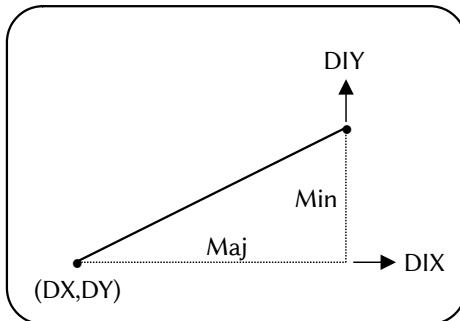


Os seguintes registradores devem ser carregados:

R#32 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	s	s	s	s	s	s	s	s	- sa7~sa0	
b7	b6	b5	b4	b3	b2	b1	b0												
s	s	s	s	s	s	s	s												
R#34 -	<table border="1"><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr></table>	s	s	s	s	s	s	s	s	- sa15~sa8	Endereço linear de origem na VRAM								
s	s	s	s	s	s	s	s												
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>s</td><td>s</td><td>s</td></tr></table>	0	0	0	0	0	s	s	s	- sa18~sa16									
0	0	0	0	0	s	s	s												
R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	d	d	d	d	d	d	d	d	- da7~da0	
b7	b6	b5	b4	b3	b2	b1	b0												
d	d	d	d	d	d	d	d												
R#38 -	<table border="1"><tr><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td><td>d</td></tr></table>	d	d	d	d	d	d	d	d	- da15~da8	Endereço linear de destino na VRAM								
d	d	d	d	d	d	d	d												
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>d</td><td>d</td><td>d</td></tr></table>	0	0	0	0	0	d	d	d	- da18~da16									
0	0	0	0	0	d	d	d												
R#40 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	n	n	n	n	n	n	n	n	- na7~na0	
b7	b6	b5	b4	b3	b2	b1	b0												
n	n	n	n	n	n	n	n												
R#42 -	<table border="1"><tr><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td><td>n</td></tr></table>	n	n	n	n	n	n	n	n	- na15~na8	Número de bytes a transferir								
n	n	n	n	n	n	n	n												
R#43 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>n</td><td>n</td><td>n</td></tr></table>	0	0	0	0	0	n	n	n	- na18~na16									
0	0	0	0	0	n	n	n												
R#44 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	y	x	0	0	- DIY,DIX	Direção da transferência
b7	b6	b5	b4	b3	b2	b1	b0												
0	0	0	0	y	x	0	0												
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	- LOP	Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0												
0	0	0	t	1	1	1	1												
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0	
b7	b6	b5	b4	b3	b2	b1	b0												
w	w	w	w	w	w	w	w												
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8	Máscara de escrita								
w	w	w	w	w	w	w	w												
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	1	0	1	0	0	0	0	0	- OP-CODE	Cód. comando BMLL
b7	b6	b5	b4	b3	b2	b1	b0												
1	0	1	0	0	0	0	0												

### 5.8.6.14 – LINE (Desenha uma linha)

Este comando desenha uma linha entre coordenadas da área de imagem. Operações lógicas no destino são possíveis. Os parâmetros são especificados incluindo a coordenada (X,Y) de início da linha e o comprimento horizontal e vertical até o ponto final, conforme a ilustração abaixo:



Os seguintes registradores devem ser carregados:

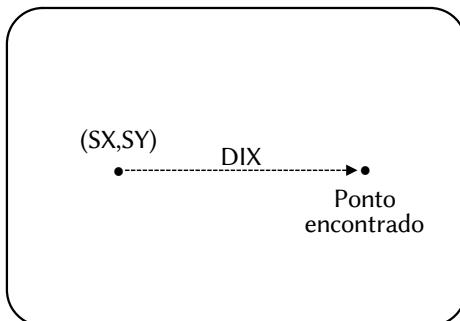
R#36 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal inicial a partir da qual a linha será desenhada (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8	DY	Coordenada vertical inicial a partir da qual a linha será desenhada (0 a 4095)								
0	0	0	0	0	x	x	x													
R#38 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	Maj	Número de pontos do cateto maior do triângulo retângulo de referência para desenho
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8	Min	Número de pontos do cateto menor do triângulo retângulo de referência para desenho								
0	0	0	0	y	y	y	y													
R#40 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>j</td><td>j</td><td>j</td><td>j</td><td>j</td><td>j</td><td>j</td><td>j</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	j	j	j	j	j	j	j	j	- x7~x0		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
j	j	j	j	j	j	j	j													
R#41 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>j</td><td>j</td><td>j</td></tr></table>	0	0	0	0	0	j	j	j	- x8										
0	0	0	0	0	j	j	j													
R#42 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>i</td><td>i</td><td>i</td><td>i</td><td>i</td><td>i</td><td>i</td><td>i</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	i	i	i	i	i	i	i	i	- y7~y0		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
i	i	i	i	i	i	i	i													
R#43 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>i</td><td>i</td><td>i</td><td>i</td></tr></table>	0	0	0	0	i	i	i	i	- y9~y8										
0	0	0	0	i	i	i	i													
R#44 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>0</td><td>m</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	x	0	m	- DIY,DIX,Maj		Direção do desenho <sup>*obs</sup>
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	0	y	x	0	m													
R#45 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	t	1	1	1	1	- LOP		Cód. de operação lógica
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	t	1	1	1	1													
R#46 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	w	w	w	w	w	w	w	w	- wm7~wm0		Máscara de escrita
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8										
w	w	w	w	w	w	w	w													

R#48 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	f	f	f	f	f	f	f	f	- fc7~fc0	Código de cor da linha a ser desenhada
b7	b6	b5	b4	b3	b2	b1	b0												
f	f	f	f	f	f	f	f												
R#49 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	f	f	f	f	f	f	f	f	- fc15~fc8	
b7	b6	b5	b4	b3	b2	b1	b0												
f	f	f	f	f	f	f	f												
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	1	0	1	1	0	0	0	0	- OP-CODE Cód. comando BMXL	
b7	b6	b5	b4	b3	b2	b1	b0												
1	0	1	1	0	0	0	0												

**Obs:** para Maj=0, o lado maior do triângulo retângulo de referência é paralelo ao eixo X (horizontal) e para Maj=1, o lado maior é paralelo ao eixo Y (vertical).

#### 5.8.6.15 – SRCH (Procura código de cor de um ponto)

Esse comando procura a existência de um ponto com uma cor específica na área de imagem, sempre na direção horizontal, para a esquerda ou direita. O comando termina quando o ponto é encontrado, quando um ponto com a cor da borda é encontrado ou quando o limite da área de imagem é atingido.



Os seguintes registradores devem ser carregados:

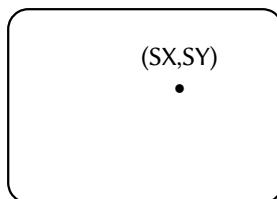
R#32 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	SX	Coordenada horizontal de início da busca (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#33 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8										
0	0	0	0	0	x	x	x													
R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY	Coordenada vertical de início da busca (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													

R#44 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>x</td><td>n</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	y	x	n	0	- DIY,DIX,NEQ <sup>*obs</sup>	Direção de busca e especificação de cor
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	y	x	n	0												
R#45 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	t	1	1	1	1	- LOP	Cód. de operação lógica
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	t	1	1	1	1												
R#48 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	f	f	f	f	f	f	f	f	- fc <sub>7</sub> ~fc <sub>0</sub>	Código de cor do ponto a ser encontrado
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
f	f	f	f	f	f	f	f												
R#49 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	f	f	f	f	f	f	f	f	- fc <sub>15</sub> ~fc <sub>8</sub>	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
f	f	f	f	f	f	f	f												
R#52 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	0	0	0	0	0	0	- OP-CODE	Cód. comando SRCH
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
1	1	0	0	0	0	0	0												
R#53 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x <sub>7</sub> ~x <sub>0</sub>	Coordenada horizontal do ponto, se encontrado
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
x	x	x	x	x	x	x	x												
R#54 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- x <sub>10</sub> ~x <sub>8</sub>	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
0	0	0	0	0	x	x	x												

**Obs:** para NEQ=0, a cor para detecção é especificada; para NEQ=1 a cor para detecção não é especificada.

#### 5.8.6.16 – POINT (Lê código de cor de um ponto)

Esse comando lê o código de cor de um ponto qualquer na área de imagem. O código de cor lido fica disponível na porta P#2.



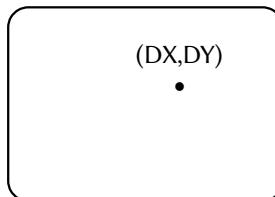
Os seguintes registradores devem ser carregados para a execução do comando POINT:

R#32 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x <sub>7</sub> ~x <sub>0</sub>	SX	Coordenada horizontal do ponto (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#33 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- x <sub>10</sub> ~x <sub>8</sub>		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
0	0	0	0	0	x	x	x													

R#34 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	SY	Coordenada vertical do ponto (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#35 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													
R#52 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	1	1	0	1	0	0	0	0	- OP-CODE Cód. comando POINT		
b7	b6	b5	b4	b3	b2	b1	b0													
1	1	0	1	0	0	0	0													

### 5.8.6.17 – PSET (Desenha um ponto e avança)

Esse comando desenha um ponto na área de imagem e depois avança coordenadas de acordo com o valor passado em R#52. Operações lógicas no destino são possíveis.



Os seguintes registradores devem ser carregados:

R#36 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal do ponto a ser desenhado (0 a 2047)
b7	b6	b5	b4	b3	b2	b1	b0													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8										
0	0	0	0	0	x	x	x													
R#38 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical do ponto a se desenhado (0 a 4095)
b7	b6	b5	b4	b3	b2	b1	b0													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													
R#45 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>t</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	t	1	1	1	1	- LOP		Cód. de operação lógica
b7	b6	b5	b4	b3	b2	b1	b0													
0	0	0	t	1	1	1	1													
R#46 -	<table border="1"><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	b7	b6	b5	b4	b3	b2	b1	b0	w	w	w	w	w	w	w	w	- wm7~wm0		Máscara de escrita
b7	b6	b5	b4	b3	b2	b1	b0													
w	w	w	w	w	w	w	w													
R#47 -	<table border="1"><tr><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td><td>w</td></tr></table>	w	w	w	w	w	w	w	w	- wm15~wm8										
w	w	w	w	w	w	w	w													

R#48 -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	f	f	f	f	f	f	f	f	- fc7~fc0	Código de cor do ponto a ser desenhado
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
f	f	f	f	f	f	f	f												
R#49 -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td><td>f</td></tr> </table>	f	f	f	f	f	f	f	f	- fc15~fc8									
f	f	f	f	f	f	f	f												
R#52 -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>AY</td><td>AX</td><td></td><td></td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	1	0	AY	AX			- OP-CODE Cód. comando PSET	
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>												
1	1	1	0	AY	AX														

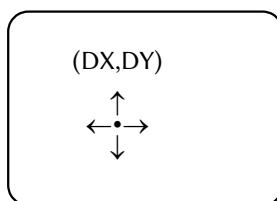
AX: 00 – DX e DY são usados  
 01 – não haverá deslocamento  
 10 – deslocamento à direita  
 11 – deslocamento à esquerda  
 AY: 00 – não haverá deslocamento  
 01 – não haverá deslocamento  
 10 – deslocamento para baixo  
 11 – deslocamento para cima

Alguns cuidados devem ser observados para a execução desse comando. Quando o ponto for desenhado na posição corrente, os registradores R#36 a R#39 não devem ser carregados. Após a execução do comando, o ponteiro avança de acordo com os valores de AY e AX e o próximo ponto poderá ser desenhado nessa posição. Oficialmente, os valores AY e AX são nomeados conforme a ilustração abaixo:

R#52 -	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>AYM</td><td>AYE</td><td>AXM</td><td>AXE</td></tr> </table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	0	1	AYM	AYE	AXM	AXE
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>										
1	1	0	1	AYM	AYE	AXM	AXE										

#### 5.8.6.18 – ADVN (Avança coordenadas)

Esse comando simplesmente avança coordenadas na área de imagem sem desenhar.



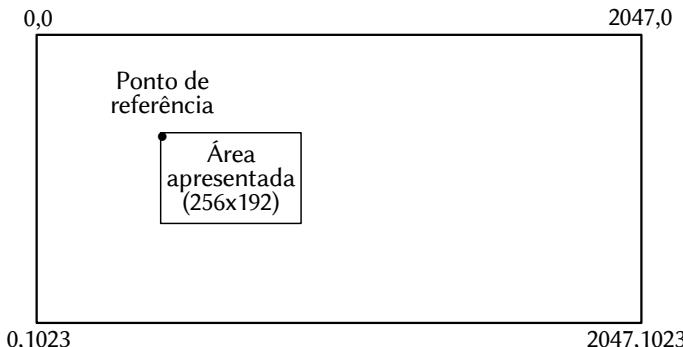
Os registradores a serem carregados estão ilustrados abaixo.

R#36 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- x7~x0	DX	Coordenada horizontal a partir da qual ocorrerá o avanço (0 a 2047)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
x	x	x	x	x	x	x	x													
R#37 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	0	0	0	0	0	x	x	x	- x10~x8										
0	0	0	0	0	x	x	x													
R#38 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- y7~y0	DY	Coordenada vertical a partir da qual ocorrerá o avanço (0 a 4095)
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
y	y	y	y	y	y	y	y													
R#39 -	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	0	0	0	0	y	y	y	y	- y11~y8										
0	0	0	0	y	y	y	y													
R#52 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>AY</td><td>AX</td><td></td><td></td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	1	1	1	1	AY	AX			- OP-CODE Cód. comando ADVN		
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>													
1	1	1	1	AY	AX															

Os valores de deslocamento (AY e AX) são os mesmos usados para o comando PSET, descritos acima. Como no comando PSET, os registradores R#36 a R#39 não devem ser carregados quando o avanço deve ocorrer a partir da posição atual.

### 5.8.7 – Scroll e área de imagem

No V9990, o tamanho da imagem é, normalmente, maior que a área apresentada na tela. Exemplificando, para o modo B1 com 512 Kbytes de VRAM e 4 cores, podemos ter uma imagem de até 2048 x 1024 pontos. Entretanto, na tela aparecem apenas 256 x 192 pontos. O ponto superior esquerdo apresentado pode ser definido pelos registradores de scroll. Assim, pode-se “varrer” toda a área de imagem, e o efeito na tela será de um scroll suave em todas as direções. Abaixo há uma ilustração do exemplo citado.



O ponto de referência pode ser deslocado livremente pela área de imagem através dos registradores R#17 a R#24, conforme descrito abaixo.

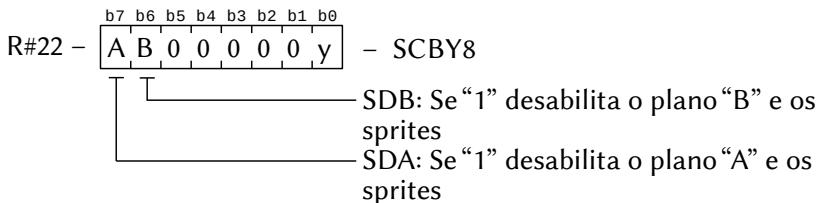
R#17 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- SCAY7~SCAY0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
y	y	y	y	y	y	y	y											
R#18 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>R</td><td>R</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	R	R	0	y	y	y	y	y	- SCAY12~SCAY8
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
R	R	0	y	y	y	y	y											
R#19 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- SCAX2~SCAX0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
0	0	0	0	0	x	x	x											
R#20 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	x	x	x	x	x	x	x	x	- SCAX10~SCAX3
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
x	x	x	x	x	x	x	x											
R#21 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	y	y	y	y	y	y	y	y	- SCBY7~SCBY0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
y	y	y	y	y	y	y	y											
R#22 -	<table border="1"><tr><td>A</td><td>B</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>y</td></tr></table>	A	B	0	0	0	0	0	y	- SCBY8								
A	B	0	0	0	0	0	y											
R#23 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	0	0	0	x	x	x	- SCBX2~SCBX0
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
0	0	0	0	0	x	x	x											
R#24 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	0	0	x	x	x	x	x	x	- SCBX8~SCBX3
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
0	0	x	x	x	x	x	x											

Os valores SCAY e SCAX correspondem às coordenadas relativas ao plano “A” do modo P1 e a todos os outros modos de tela. Quando forem usados 16 bits por ponto nos modos B2 e B3, o bit menos significativo (SCAX0) é ignorado, e a coordenada horizontal é especificada em incrementos de 2 pontos. Já os valores SCBY e SCBX correspondem, exclusivamente, às coordenadas para o plano “B” do modo P1.

O número de pontos verticais que poderão ser usados para scroll é especificado em R#18, conforme ilustrado abaixo.

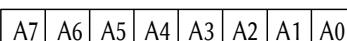
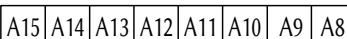
R#18 -	<table border="1"><tr><td>b<sub>7</sub></td><td>b<sub>6</sub></td><td>b<sub>5</sub></td><td>b<sub>4</sub></td><td>b<sub>3</sub></td><td>b<sub>2</sub></td><td>b<sub>1</sub></td><td>b<sub>0</sub></td></tr><tr><td>R</td><td>R</td><td>0</td><td>y</td><td>y</td><td>y</td><td>y</td><td>y</td></tr></table>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	R	R	0	y	y	y	y	y	- SCAY12~SCAY8
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>											
R	R	0	y	y	y	y	y											
	<hr/>	00: scroll em toda a área de imagem 01: scroll em 256 linhas 10: scroll em 512 linhas 11: sem significado																

Os bits b7 e b6 de R#22 podem ser usados para habilitar ou desabilitar os planos P1 e P2, conforme ilustração abaixo:

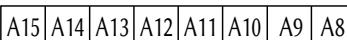
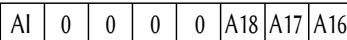


## 5.8.8 – Descrição dos registradores do V9990

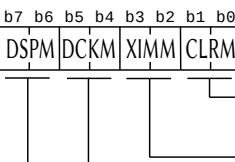
### 5.8.8.1 – Endereço de escrita na VRAM (Só escrita)

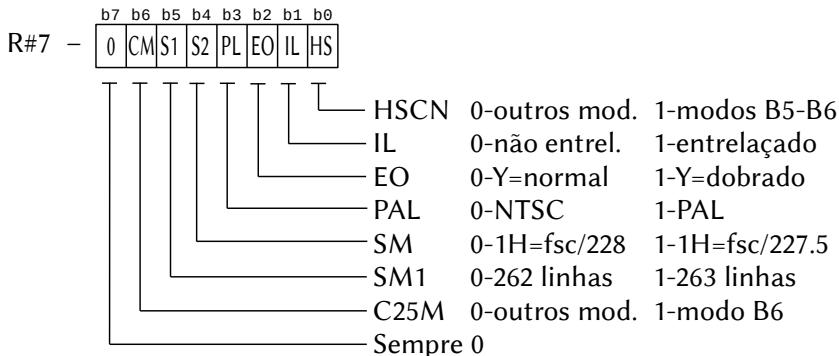
R#0 – 	Endereço de escrita na VRAM (low)
R#1 – 	Endereço de escrita na VRAM (mid)
R#2 – 	Endereço de escrita na VRAM (high) AI = 0 → autoincremento

### 5.8.8.2 – Endereço para leitura da VRAM (Só escrita)

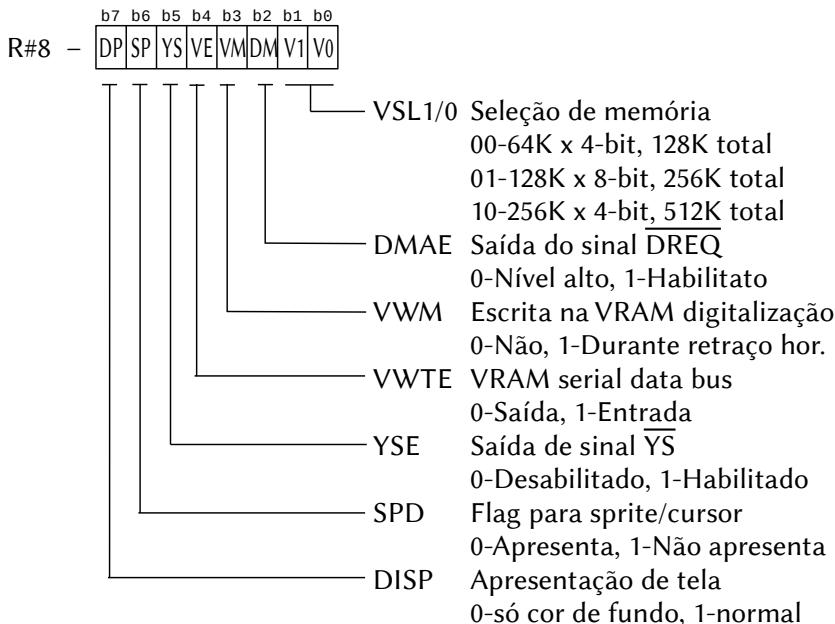
R#3 – 	Endereço de leitura da VRAM (low)
R#4 – 	Endereço de leitura da VRAM (mid)
R#5 – 	Endereço de leitura da VRAM (high) AI = 0 → autoincremento

### 5.8.8.3 – Modo de tela (Leitura/escrita)

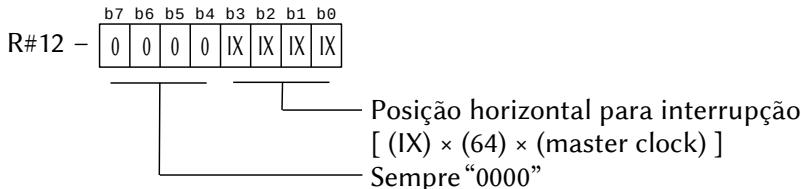
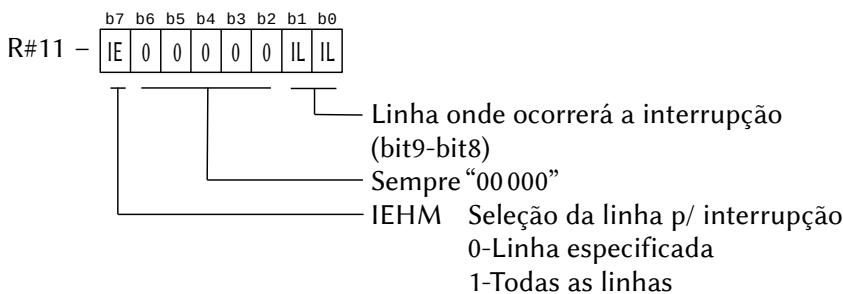
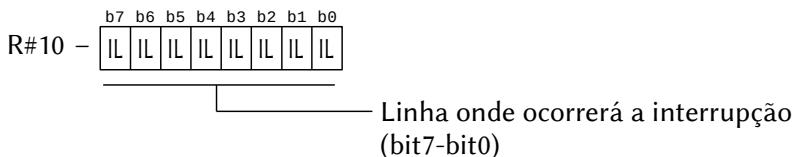
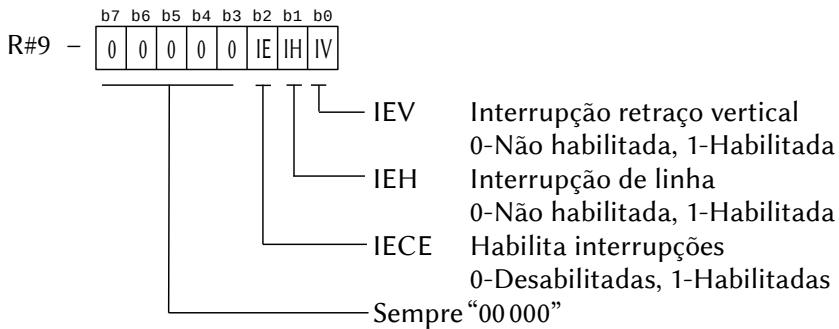
R#6 – 	CLRM	00-2bits/pixel	10-8bits/pixel
		01-4bits/pixel	11-16bits/pixel
	XIMM	00-Y=256pixels	10-Y=1024pixels
		01-Y=512pixels	11-Y=2048pixels
	DCKM	00-XTAL=1/4	10-XTAL=1/1
		01-XTAL=1/2	11 – N/A
	DSPM	00-modo P1	10-Bit Map
		01-modo P2	11-Stand-by



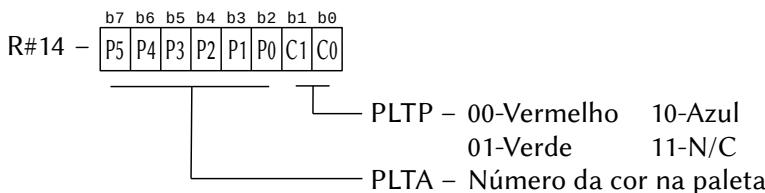
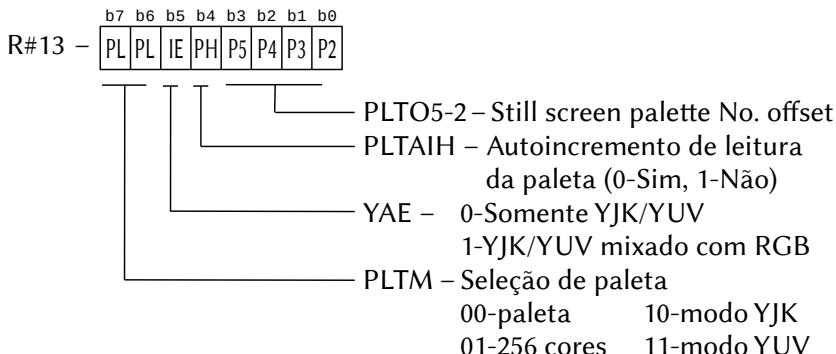
#### 5.8.8.4 – Controle do sistema (Leitura/escrita)



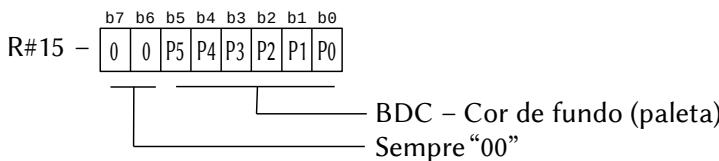
### 5.8.8.5 – Controle de interrupções (Leitura/escrita)



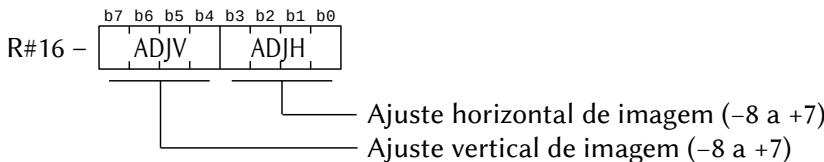
### 5.8.8.6 – Controle de paleta (Só escrita)



### 5.8.8.7 – Cor de fundo (Leitura/escrita)



### 5.8.8.8 – Ajuste de tela (Leitura/escrita)



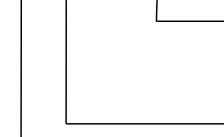
### 5.8.8.9 – Controle de scroll (Leitura/escrita)

R#17 – 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
SCAY – bits 7~0							

R#18 – 

ROLL	0	SCAY – bits 12~8					
------	---	------------------	--	--	--	--	--

  
 SCAY – Coordenada Y de início de exibição para o plano "A" do modo P1 e das telas dos outros modos  
 Sempre "0"  
 ROLL – Tipo de rolagem na direção Y  
 00-tela inteira 10-512 linhas  
 01-256 linhas 11-N/C

R#19 – 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	0	0	0	SX-bit2~0		

R#20 – 

SCAX – bits 10~3							
------------------	--	--	--	--	--	--	--

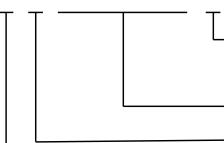
  
 SCAX – Coordenada X de início de exibição para o plano "A" do modo P1 e das telas dos outros modos (para 16 bpp nos modos B2 e B3, o bit 0 é ignorado e a unidade será de 2 pixels).

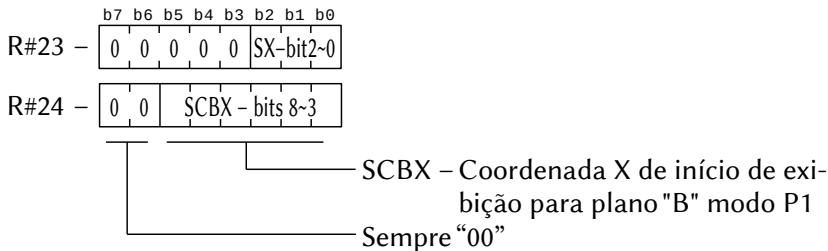
R#21 – 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
SCBY – bits 7~0							

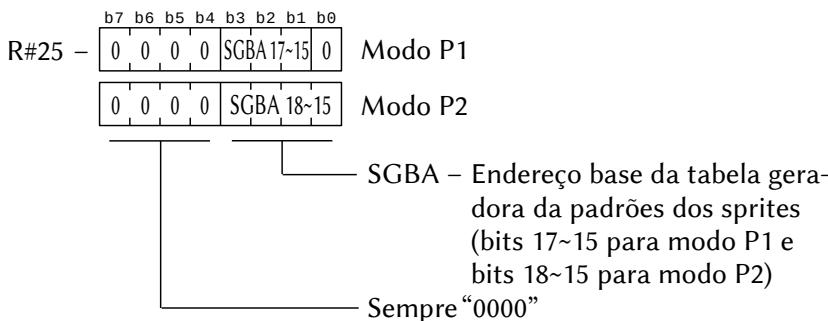
R#22 – 

A	B	0	0	0	0	0	Y
---	---	---	---	---	---	---	---

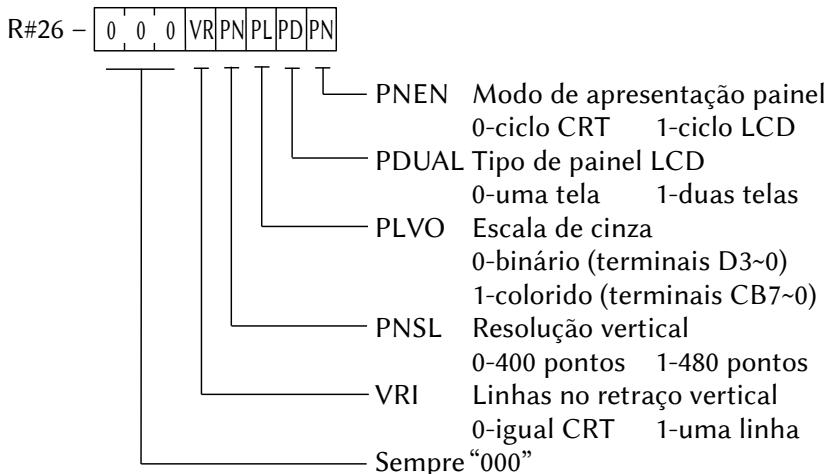
  
 SCBY – Coordenada Y de início de exibição para plano "B" modo P1  
 Sempre "00 000"  
 SDB – Se 1, desabilita plano "B" e sprites  
 SDB – Se 1, desabilita plano "A" e sprites



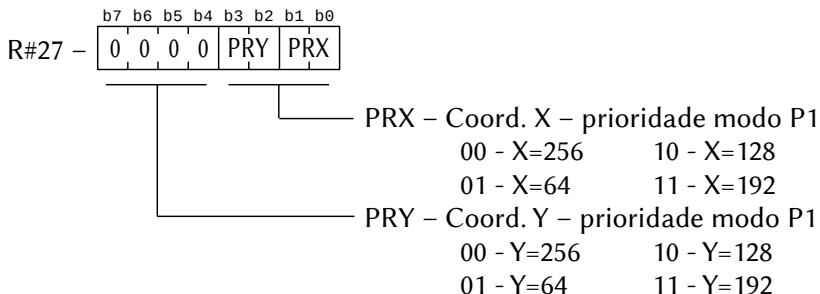
#### 5.8.8.10 – Endereço da tabela de sprites (Leitura/escrita)



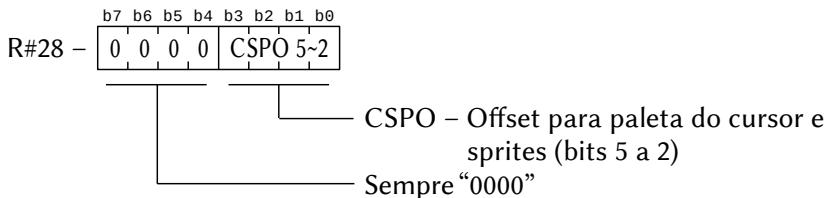
#### 5.8.8.11 – Controle do painel LCD (Leitura/escrita)



### 5.8.8.12 – Controle de prioridade (Leitura/escrita)



### 5.8.8.13 – Offset da paleta para sprite/cursor (Só escrita)



Os registradores R#19 a R#31 não existem

### 5.8.8.14 – Comandos do VDP – Fonte: Endereço linear e Kanji-ROM (Só escrita)

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#32 –	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0
	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0
	KA7	KA6	KA5	KA4	KA3	KA2	KA1	KA0
R#33 –	0	0	0	0	0	SX10	SX9	SX8
R#34 –	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8
	KA15	KA14	KA13	KA12	KA11	KA10	KA9	KA8

R#35 -	0	0	0	0	SY11	SY10	SY9	SY8
	0	0	0	0	0	SA18	SA17	SA16
	0	0	0	0	0	0	KA17	KA16

SX10 ~ SX0 → Coordenada horizontal da fonte

SY11 ~ SY0 → Coordenada vertical da fonte

SA18 ~ SA0 → Endereço linear da fonte

KA18 ~ KA0 → Endereço da Kanji-ROM

#### 5.8.8.15 – Comandos do VDP – Destino: Coordenada XY / Endereço linear (Só escrita)

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#36 -	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

R#37 -	0	0	0	0	0	DX10	DX9	DX8
--------	---	---	---	---	---	------	-----	-----

R#38 -	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8

R#39 -	0	0	0	0	DY11	DY10	DY9	DY8
	0	0	0	0	0	DA18	DA17	DA16

DX10 ~ DX0 → Coordenada horizontal do destino

DY11 ~ DY0 → Coordenada vertical do destino

DA18 ~ DA0 → Endereço linear do destino

#### 5.8.8.16 – Comandos do VDP – Tamanho (Número de pontos a transferir XY / Linear / Linha menor/maior) (Só escrita)

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#40 -	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0
	NA7	NA6	NA5	NA4	NA3	NA2	NA1	NA0
	MJ7	MJ6	MJ5	MJ4	MJ3	MJ2	MJ1	MJ0

R#41 -	0	0	0	0	0	NX10	NX9	NX8
	0	0	0	0	MJ11	MJ10	MJ9	MJ8

R#42 -	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
	NA15	NA14	NA13	NA12	NA11	NA10	NA9	NA8
	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0

R#43 -	0	0	0	0	NY11	NY10	NY9	NY8
	0	0	0	0	0	NA18	NA17	NA16
	0	0	0	0	MI11	MI10	MI9	MI8

NX10 ~ NX0 → Número de pontos a transferir na horizontal

NY11 ~ NY0 → Número de pontos a transferir na vertical

NA18 ~ NA0 → Número de bytes a transferir

MJ11 ~ MJ0 → Lado maior da linha (comando LINE)

MI11 ~ MI0 → Lado menor da linha (comando LINE)

#### 5.8.8.17 – Comandos do VDP – Argumentos (Só escrita)

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
R#44 -	0	0	0	0	DY	DX	NQ MJ

MAJ – Lado maior comando LINE  
 0-horizontal 1-vertical

EQ – cond. término comando SRCH  
 0-cor especificada  
 1-outra cor

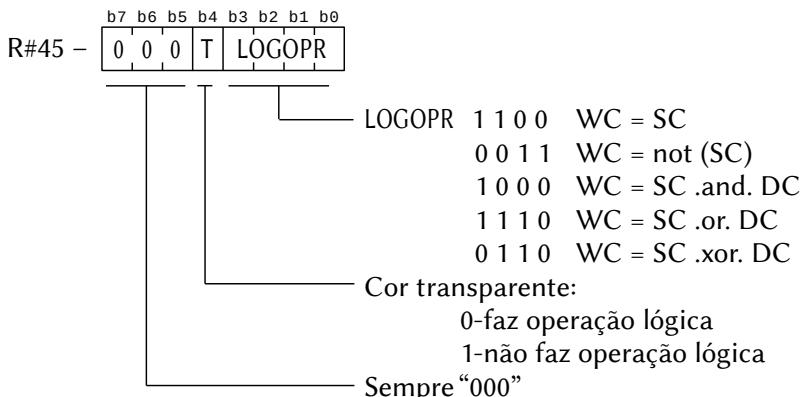
DIX (direção X) 0 – para a direita  
 1 – para a esquerda

DIY (direção Y) 0 – para baixo  
 1 – para cima

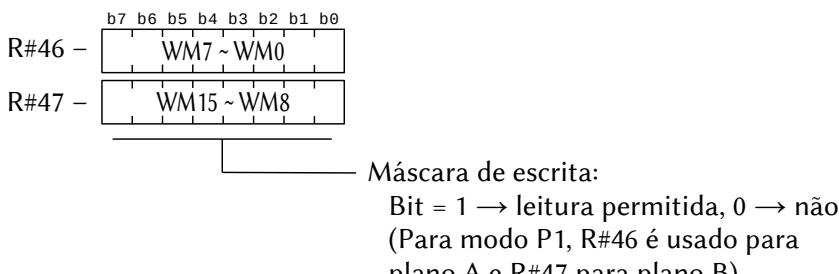
Obs. BMXL e BMLX são fixados em “+” e para BMLL, X e Y são fixados na mesma direção.

Sempre “0000”

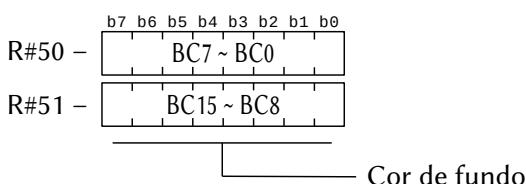
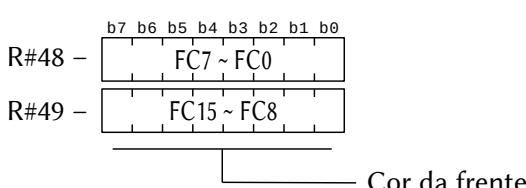
### 5.8.8.18 – Comandos do VDP – Operação lógica (Só escrita)



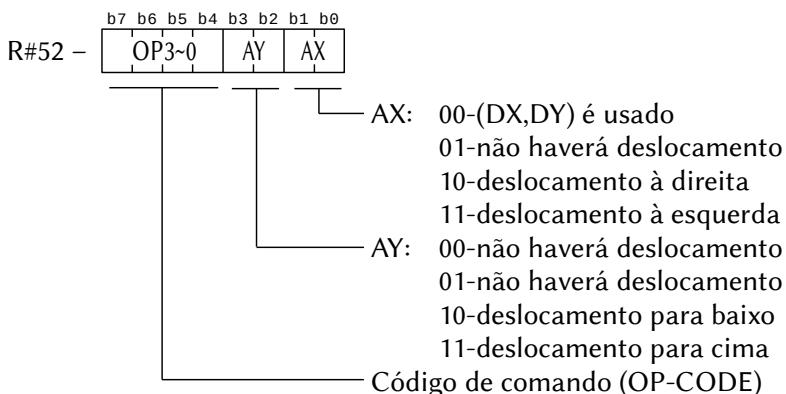
### 5.8.8.19 – Comandos do VDP – Máscara de escrita (Só escrita)



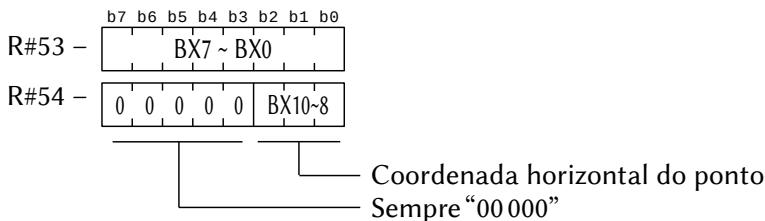
### 5.8.8.20 – Comandos do VDP – Cor da fonte (Só escrita)



### 5.8.8.21 – Comandos do VDP – Controle de operação (Só escrita)



### 5.8.8.22 – Comandos do VDP – Coord horizontal (Só leitura)



## **Capítulo 6**

### **GERADORES DE ÁUDIO**

Os micros MSX têm várias opções para a geração de sons, incluindo desde geradores de AM simples até digitalizadores sofis-ticados. Essas opções estão listadas abaixo:

- 1 – PSG (padrão do MSX1)
- 2 – 1-bit I/O port (padrão do MSX1)
- 3 – OPLL (opcional MSX2, padrão MSX2+)
- 4 – PCM (padrão MSX turbo R)
- 5 – MSX-Audio (opcional)
- 6 – SCC (para alguns jogos da Konami)
- 7 – OPL4 (opcional, só em cartucho de expansão)
- 8 – Covox (opcional)

#### **6.1 – O PSG**

PSG significa “Programmable Sound Generator”, ou seja, Gedor de Sons Programável. O PSG pode gerar até 3 vozes em até 4096 escalas (equivalente a 8 oitavas) e 16 níveis de volume independente para cada voz. Adicionalmente, possui um gerador de ruído branco (chiado) que deve estar presente em uma das 3 vozes. O chip responsável é o AY-3-8910A.

O PSG tem 16 registradores de 8 bits para a geração de sons. Eles estão descritos na tabela abaixo. Os registradores 14 e 15 são usados para operações de I/O e não para a especificação de sons.

##### **6.1.1 – Descrição dos registradores**

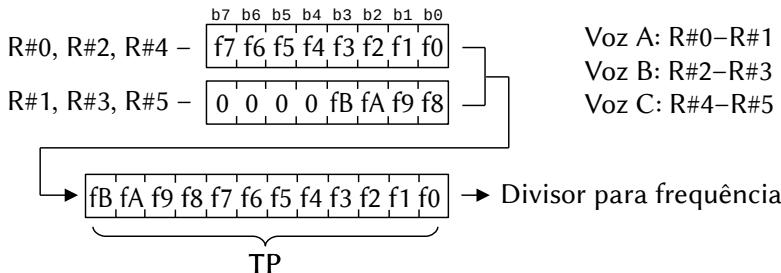
Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
R#0	7	6	5	4	3	2	1	0	Frequência da voz A
R#1	•	•	•	•	11	10	9	8	111860,87 / f_num (b11~b0)
R#2	7	6	5	4	3	2	1	0	Frequência da voz B
R#3	•	•	•	•	11	10	9	8	111860,87 / f_num (b11~b0)
R#4	7	6	5	4	3	2	1	0	Frequência da voz C
R#5	•	•	•	•	11	10	9	8	111860,87 / f_num (b11~b0)

R#6	•	•	•	4	3	2	1	0	Frequência do ruído branco 111860,87 / f_num (b11~b0)
R#7	ioB	ioA	rC	rB	rA	tC	tB	tA	habilita/desabilita sons
	b0~b2		Habilita/desabilita tons (0=habilita)						
	b5~b4		Habilita/desabilita ruído branco (0=habilita)						
	b6		Configura porta A de I/O (0=in, 1=out)						
	b7		Configura porta B de I/O (0=in, 1=out)						
R#8	•	•	•	m	v	v	v	v	Volume da voz A
R#9	•	•	•	m	v	v	v	v	Volume da voz B
R#10	•	•	•	m	v	v	v	v	Volume da voz C
	b7~b5		Não utilizados (sempre “000”)						
	b4		0=não usa a envoltória; 1=usa a envoltória						
	b3~b0		0000=volume mínimo; 1111=volume máximo						
R#11	7	6	5	4	3	2	1	0	Frequência da envoltória
R#12	15	14	13	12	11	10	9	8	6983,3 / f_num (b15~b0)
R#13	•	•	•	•	e	e	e	e	Forma da envoltória
	b7~b4		Não usados (sempre “0000”)						
	b3~b0		Define a forma da envoltória						
	00xx	=			1011	=			
	01xx	=			1100	=			
	1000	=			1101	=			
	1001	=			1110	=			
	1010	=			1111	=			
R#14	a7	a6	a5	a4	a3	a2	a1	a0	Envia/recebe porta A de I/O
R#15	b7	b6	b5	b4	b3	b2	b1	b0	Envia/recebe porta B de I/O

A operação dos registradores do PSG é muito simples. Basta escrever os valores adequados para que o som seja gerado. Os registradores estão descritos detalhadamente abaixo.

#### 6.1.1.1 – Especificação da frequência

A frequência central usada pelo PSG para comandar o divisor de frequências é de 111860,78 Hz. Assim, para obter a frequência de saída do gerador de tons, basta dividir 111860,78 pelo valor TP, representado pelos pares de registradores R#0-R#1, R#2-R#3 e R#4-R#5.

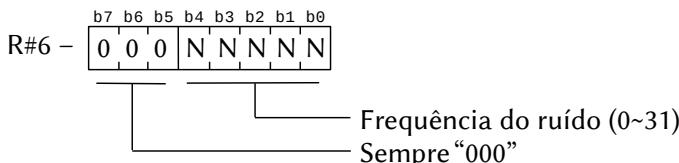


Os valores de cada registro TP para as 8 oitavas dos três geradores de tom com a nota LÁ central de 440 Hz estão listados abaixo.

Oitava →	1 <sup>a</sup>	2 <sup>a</sup>	3 <sup>a</sup>	4 <sup>a</sup>	5 <sup>a</sup>	6 <sup>a</sup>	7 <sup>a</sup>	8 <sup>a</sup>
Dó	C	D5D	6AF	357	1AC	0D6	06B	035
	C#	C9C	64E	327	194	0CA	085	032
Ré	D	BE7	5F4	2FA	17D	0BE	05F	030
	D#	B3C	59E	2CF	168	0B4	05A	02D
Mi	E	A9B	54E	2A7	153	0AA	055	02A
Fá	F	A02	501	281	140	0A0	050	028
	F#	973	4BA	25D	12E	097	04C	026
Sol	G	8EB	476	23B	11D	08F	047	024
	G#	86B	436	21B	10D	087	043	022
Lá	A	7F2	3F9	1FD	0FE	07F	040	020
	A#	780	3C0	1E0	0F0	078	03C	01E
Si	B	714	38A	1C5	0E3	071	039	01C
								00E

### 6.1.1.2 – Gerador de ruído branco

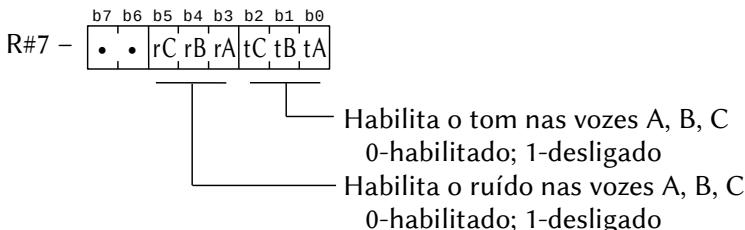
O gerador de ruído branco (chiado) é útil para gerar sons de explosões e outros. O PSG gera o chiado através de uma das três vozes de tom e sua frequência é especificada no registrador R#6.



A frequência central usada pelo gerador de ruído também é de 111860,78 Hz. Como o valor de R#6 pode variar de 1 a 31, a frequência do ruído varia de 3,6 KHz a 111,8 KHz (divisão de 111860,78 pelo valor contido em R#6).

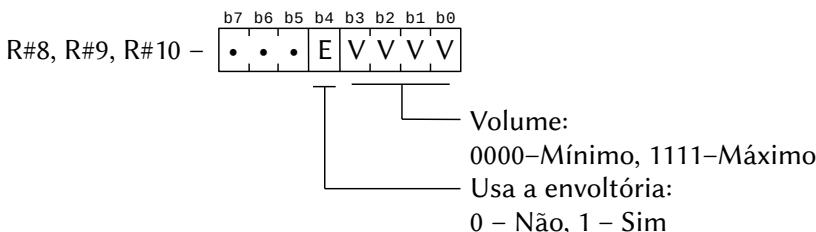
#### 6.1.1.3 – Mixando os sons

O registrador R#7 é usado para habilitar ou desabilitar o tom ou ruído de cada uma das três vozes. Os bits b7 e b6 controlam operações de I/O e não interferem com a geração de sons.



#### 6.1.1.4 – Ajuste de volume

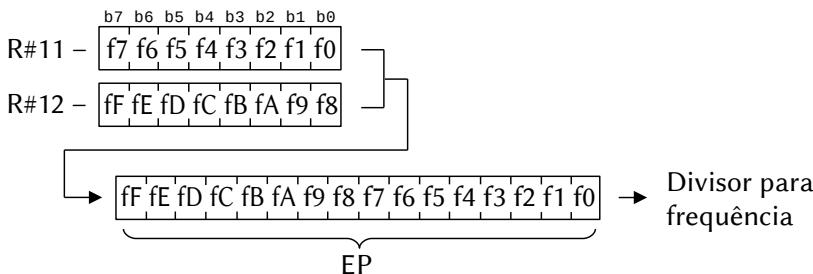
Os registradores R#8 a R#10 são usados para especificar o volume de cada uma das três vozes e podem variar de 0 (volume mínimo) a 15 (volume máximo), ou entregar o controle de volume ao gerador de envoltória (R#8 - voz A; R#9 - voz B; R#10 - voz C).



Quando o bit E for 0, o volume é especificado pelos bits V. Quando o bit E for 1, o volume é controlado pelo gerador de envoltória e os bits V são solenemente ignorados.

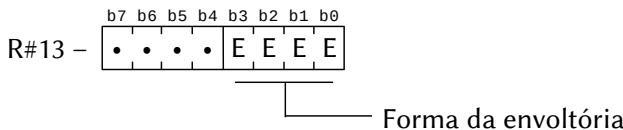
### 6.1.1.5 – Frequência da envoltória

Os registradores R#11 e R#12 são usados como divisor de frequência para o gerador de envoltória. Todos os bits são válidos. A frequência central usada pelo gerador de envoltória para comandar o divisor de frequências é de 6983,3 Hz; portanto, a frequência da envoltória pode variar de 6983,3 Hz a 0,107 Hz.

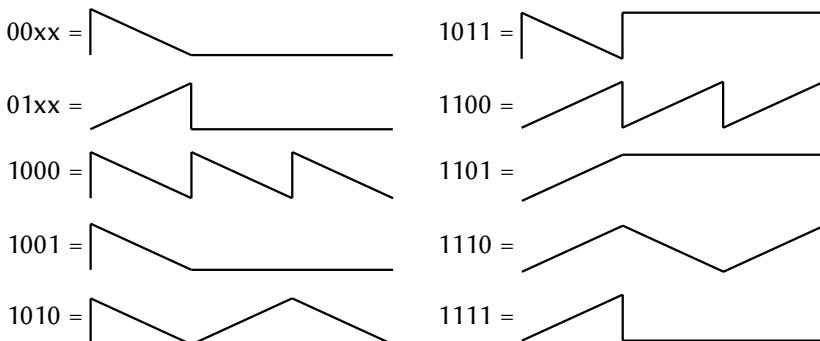


### 6.1.1.6 – Forma da envoltória

A forma da envoltória é especificada nos quatro primeiros bits de R#13, conforme ilustração abaixo.



As formas possíveis da envoltória estão ilustradas abaixo.



### 6.1.2 – Acesso ao PSG

O acesso ao PSG é feito através de portas de I/O. Entretanto, o padrão MSX determina que todos os acessos ao PSG devem ser feitos através de rotinas da BIOS para evitar problemas de sincronização.

As rotinas da BIOS destinadas ao acesso ao PSG são as seguintes:

#### GICINI (0090H/Main)

**Função:** Inicializa o PSG e seta os valores iniciais para o comando PLAY.

**Entrada:** Nenhuma.

**Saída:** Nenhuma.

**Registradores:** Todos.

#### WRTPSG (0093H/Main)

**Função:** Escreve um byte de dados em um registrador do PSG.

**Entrada:** A – Número do registrador do PSG.

E – Byte de dados a ser escrito.

**Saída:** Nenhuma.

**Registradores:** Nenhum.

#### RDPSC (0096H/Main)

**Função:** Lê o conteúdo de um registrador do PSG.

**Entrada:** A – Número do registrador do PSG.

**Saída:** A – Byte lido.

**Registradores:** Nenhum.

Também é possível o acesso direto. Existem três portas destinadas ao acesso ao PSG. Essas portas são:

Porta A0H: porta de endereço (0 a 15)

Porta A1H: porta de escrita de dados (0 a 255)

Porta A2H: porta de leitura de dados (0 a 255)

O acesso por essas portas é bem simples: basta enviar pela porta de endereço (A0H) o número do registrador a ser acessado (0 a 15). Depois, podem haver acessos repetidos ao mesmo registrador através das portas A1H (escrita) ou A2H (leitura).

## 6.2 – GERAÇÃO DE SONS PELA PORTA 1-bit

O padrão MSX dispõe de outro método padrão para a geração de sons, embora bastante limitado. Estes são gerados ligando e desligando repetidamente uma porta de I/O de 1 bit. O som de “click” das teclas é gerado dessa forma. O acesso a esse bit pode ser feito através de uma rotina da BIOS:

**CHGSND (0135H/Main)**

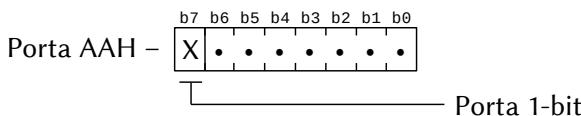
Função: Altera o estado da porta de 1 bit geradora de som.

Entrada: A = 0 desliga o bit, outro valor liga o bit.

Saída: Nenhuma.

Registradores: AF.

Aqui também é possível o acesso direto. A porta de 1-bit é acessada pelo bit b7 da porta C da PPI (porta de I/O AAH). Neste caso, devem ser tomadas a precauções para não modificar os outros bits dessa porta.



Ligando e desligando repetidamente este bit, podem ser gerados diversos tipos de efeitos sonoros, inclusive reprodução grosseira da voz humana.

## 6.3 – O OPLL (MSX-MUSIC)

O MSX-Music (FM-OPLL) pode gerar 9 vozes simultâneas ou 6 vozes mais 5 peças de bateria. Sua qualidade sonora é muito superior à do PSG. O gerador FM também é conhecido como OPLL, do inglês “FM OPerator type LL”. O chip responsável é o YM2413 e surgiu como alternativa barata ao MSX-Audio, e é padrão do MSX2+ em diante.

O FM-OPLL tem integrados no próprio chip um conversor DA de 9 bits e uma ROM com definições de instrumentos, incorporando 15 instrumentos e 5 tons de ritmo, bem como todos os tons usados para CAPTAIN e TELETEXT. Tem também um registrador que permite a criação de efeitos sonoros e instrumentos originais.

### 6.3.1 – Descrição da síntese FM

O FM-OPLL usa faz uso de harmônicas geradas por modulação de frequência para sintetizar sons musicais, chamado por “síntese FM”. Esse tipo de síntese é expressado por 3 parâmetros:

$$F = A \sin(\omega_c t + I \sin \omega_m t)$$

Onde  $A$  é a amplitude de saída,  $I$  é o índice de modulação,  $\omega_c$  e  $\omega_m$  as frequências angulares da portadora e da moduladora, respectivamente. A equação 1 pode ser expressada alternativamente como abaixo:

$$A [J_0(I) \sin \omega_c t + J_1(I) (\sin(\omega_c + \omega_m)t - \sin(\omega_c - \omega_m)t) + J_2(I) (\sin(\omega_c + 2\omega_m)t + \sin(\omega_c - 2\omega_m)t + \dots)]$$

Onde  $J_n(I)$  é a enésima ordem da função Bessel de primeiro tipo. A amplitude de cada componente da harmônica é expressada como a função Bessel do índice de modulação. Os sons sintetizados pelo FM podem ser usados para obter sons musicais específicos ou diversos tipos de efeitos sonoros. Sons em série, entretanto, não podem ser obtidos uma vez que a distribuição das harmônicas não é uniforme. O método de “feedback” ou realimentação é usado para resolver o problema. Ele é caracterizado pela seguinte equação:

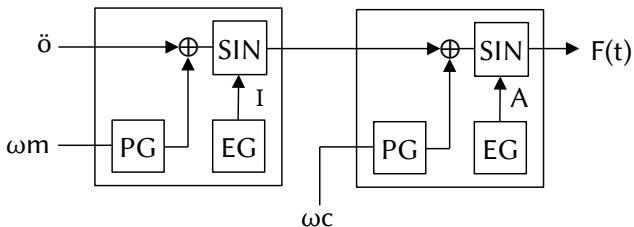
$$F = A \sin(\omega_c t + \beta F)$$

Onde  $\beta$  é a taxa de realimentação. O espectro de harmônicas produzido tem a forma de onda dente-de-serra.

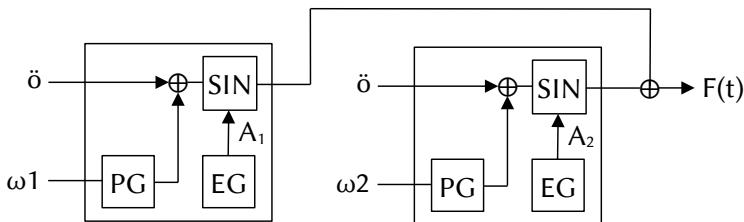
Três blocos são usados para sintetizar os sons FM:

1. Gerador de fase (PG) para gerar  $\omega t$ ;
2. Gerador de envoltória (EG) para gerar a amplitude  $A$  e o índice de modulação ( $I$ );
3. Tabela SIN (seno).

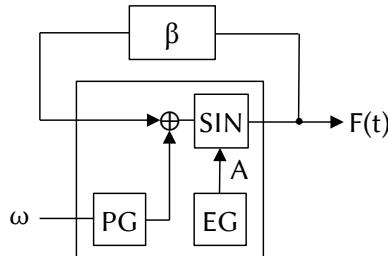
A síntese FM pode ser realizada como mostrado na figura abaixo, em células que combinam as funções dos três blocos. Só é necessário definir os parâmetros da frequência e da envoltória. Abaixo está ilustrada a geração FM através de células individuais.



$$F(t) = A \sin(\omega ct + I \sin \omega mt)$$



$$F(t) = A_1 \sin \omega_1 t + A_2 \sin \omega_2 t$$



$$F(t) = A \sin(\omega t + \beta \cdot F(t))$$

### **6.3.2 – Mapa dos registradores do OPLL**

\$02H	KSL(m)	Nível total modul. (c)				Definição de instrumento			
	b6~b7 b6~b0	KSL (m): 00=0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB Nível total: b0=0,75dB, b1=1,5dB, ...., b5=24dB							
\$03H	KSL(c)	•	DC	DM	Feedback	Definição de instrumento			
	b6~b7 b5 b4 b3 b2~b0	KSL (c): 00=0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB Não usado (sempre 0) DC: 0=onda portadora inteira, 1=retif. p/ meia onda DM: 0=onda moduladora inteira, 1=retif. p/ meia onda Realimentação: (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π)							
\$04H	Attack (m)			Decay (m)		Attack (0dB a 48dB → mín. 0,14 mS; máx 1730 mS) Decay (0dB a 48dB → mín. 1,27 mS; máx 20 926 mS)			
\$05H	Attack (c)			Decay (c)					
\$06H	Sustain (m)			Release (m)		Sustain (b7=24dB, b6=12dB, b5=6dB, b4=3dB) Release (0dB a 48dB → mín. 1,27 mS; máx 28 926 mS)			
\$07H	Sustain (c)			Release (c)					
\$0EH	•	•	R	BD	SD	TOM	TCY	HH	Controle das peças de bateria
	b7~b6 b5 b0~b4	Não usados (sempre "00") 0=modo "melodia"; 1=modo "bateria" 0=desliga instrumentos da bateria; 1=liga BD=Bass Drum SD=Snare Drum TOM=Tom tom TCY=Top cymbal HH=Hi-hat							
\$0FH	b7	•	b5	•	b3	b2	b1	b0	Registrador de teste do OPLL
	b7 b5 b3 b2 b1 b0	1=Reseta os LFOs (amplitude máx. / desvio fase zero) 1=Seleciona forma de onda 1=Atualiza o LFO a cada amostra (a frequência do tremolo é multiplicada por 64 e do vibrato por 1024) 1=Mantém mas coloca a forma de onda em "0". 1=Mantém mas coloca a fase do LFO em "0". 1=Coloca os geradores de envoltória (modulador e portador) em volume total.							

\$10H : \$18H	Frequência LSB (8 bits)					Registradores usados para a seleção de frequências do gerador de tons	
\$20H : \$28H	•	•	Sustain	Key	Oitava	Freq.	Frequência MSB 1 bit Oitava Key / Sustain on/off
	b7~b6	Não usados (sempre "00")					
	b5	0=Sem "sustain"; 1=Release Rate decairá gradativam.					
	b4	0=Voz respectiva desligada (key off); 1=voz ativa					
	b3~b1	Define a oitava. A quarta é 011.					
	b0	Frequência MSB 1 bit. A Nota Lá central de 440 Hz é obtida com b0=1 e \$10H~18H=00 100 000					
\$30H : \$38H	Instrumentos			Volume		Registradores usados para seleção de instrumentos e de volume	
	b7~b4	Definição de instrumento:					
		0000 – A ser definido	1000 – Órgão				
		0001 – Violino	1001 – Piston				
		0010 – Violão	1010 – Sintetizador				
		0011 – Piano	1011 – Cravo				
		0100 – Flauta	1100 – Vibrafone				
		0101 – Clarinete	1101 – Baixo elétrico				
		0110 – Oboé	1110 – Baixo acústico				
		0111 – Trompete	1111 – Guitarra elétrica				
	b3~b0	Volume (0000=mínimo; 1111=máximo)					

Mapa dos registradores para o modo bateria (\$0EH, b5=1)

\$36H	•	•	•	•	BD volume	Registradores de volume para as peças de bateria	
\$37H	HH volume			SD volume			
\$38H	TOM volume			TCY volume			

Regist.	Bit	Conteúdo
00: (m)	b7	Liga/desl. a modulação de amplitude (tremolo)
01: (c)	b6	Liga/desl. a modulação de frequência (vibrato)

	b5 b4 b0~b3	0=tom percussivo; 1=tom constante Razão da key scale Controle multi-sample e harmônicos
02: (m) 03: (c)	b6~b7	Nível da “Key Scale”
02:	b0~b5	Nível total de modulação
03:	b4(c) b3(m) b0~b2(m)	Distorção da forma de onda portadora Distorção da forma de onda moduladora Constante de realimentação FM
04: (m) 05: (c)	b4~b7 b0~b3	Controle de nível de “attack” da envoltória Controle de nível de “decay” da envoltória
06: (m) 07: (c)	b4~b7 b0~b3	Indicação de “decay”; nível de “sustain” Controle do nível “release” da envoltória
0E:	b5 b0~b4	1-modo bateria; 0-modo melodia Liga/desliga instrumentos da bateria
0F:	b5~b4 b3 b2 b1 b0	Não usados (sempre “0000”) 1=Atualiza o LFO a cada amostra 1=Mantém mas coloca a forma de onda em “0”. 1=Mantém mas coloca a fase do LFO em “0”. 1=Coloca o EG em volume total.
10~18:	b0~b7	Frequência (LSB 8 bits)
20~28:	b5 b4 b1~b3 b0	Liga/desliga o “sustain” Liga/desliga a “key” Seleciona a oitava Frequência (MSB 1 bit)
30~30:	b4~b7 b0~b3	Seleção de instrumentos Controle de volume

O OPLL possui internamente 15 instrumentos pré-programados e mais um que pode ser definido pelo usuário, além de cinco peças de bateria. O instrumento que pode se programado é o de número 0 (original). Os instrumentos disponíveis são os seguintes:

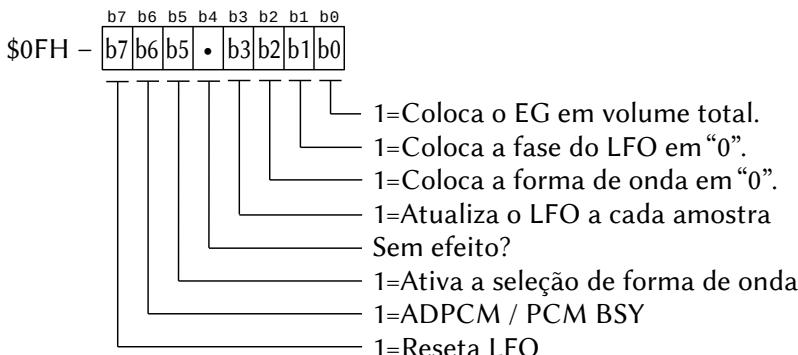
0: original	8: órgão	Bateria:
1: violino	9: piston	
2: violão	10: sintetizador	BD: bass drum
3: piano	11: cravo	SD: snare drum
4: flauta	12: vibrafone	TOM: tom-tom
5: clarinete	13: baixo elétrico	TCY: top cymbal
6: oboé	14: baixo acústico	HH: high hat
7: trompete	15: guitarra elétrica	

### 6.3.3 – Descrição dos registradores

Essa seção descreve detalhadamente os diversos registradores do YM2413 e seu funcionamento.

#### 6.3.3.1 – Registrador de teste

O registrador \$0FH é o registrador de teste. Normalmente seu valor é 0. Quando em modo teste, um DAC interno de 4 bits pode ser acessado. O modo para acessá-lo está descrito mais adiante.



Bit 0: Se for “1”, a saída dos geradores de envoltória é colocada em “0” (volume total) tanto para o modulador quanto para a portadora. A envoltória continua em execução.

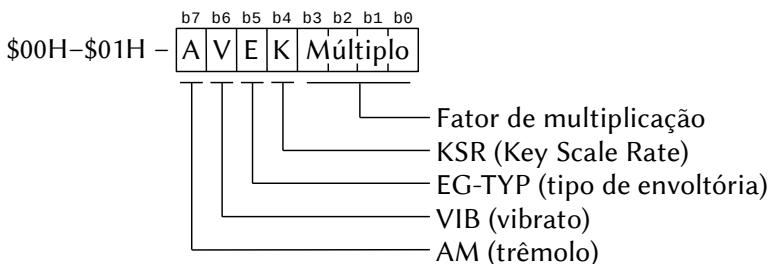
Bit 1: Se for “1”, mantém a fase do LFO em zero. Isso para, desativa e redefine o LFO do tremolo e do vibrato.

- Bit 2: Se for “1”, mantém a fase da forma de onda em zero. As envoltórias continuam em execução, mas a saída é silenciada.
- Bit 3: Se for “1”, atualiza os LFOs de tremolo e vibrato a cada amostra em vez de uma vez a cada várias amostras (o tremolo é 64 vezes mais rápido e vibrato é 1024 vezes).
- Bit 5: Se for “1”, ativa a seleção de forma de onda.
- Bit 6: ADPCM. Afeta o bit 0 (PCMBSY) do registrador de estado.
- Bit 7: Se for “1”, coloca os LFOs nos valores iniciais (amplitude máxima, desvio de fase zero).

### 6.3.3.2 – Registradores para definição de instrumentos

- AM/VIB/EG-TYP/KSR/MÚLTIPLA (\$00H e \$01H)

Esses registradores especificam o fator de multiplicação para as frequências do modulador (\$00H) e da portadora (\$01H) com seus respectivos componentes, como a envoltória e demais.



#### MÚLTIPLA (b0~b3)

As frequências da onda portadora e da onda moduladora, que geram a envoltória, são controladas de acordo com certos fatores de multiplicação, que podem ser vistos na tabela abaixo:

Valor do registro:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fator de multiplicação:	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	10	12	12	15	15

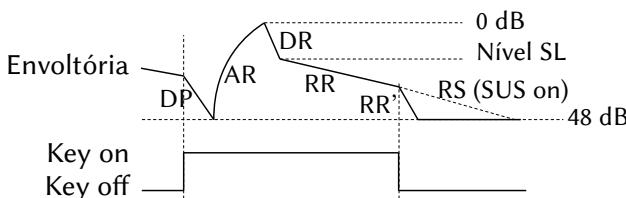
### KSR (b4)

Esse bit é uma flag que indica se será usada ou não a “Key Scale Rate”, especificada pelos bits KSL. Após setar os tons musicais, estes podem ter seus níveis alterados. Se KSR for igual a 0, o nível será o mesmo para todas as frequências. Se KSR for igual a 1, haverá atenuação do som conforme a frequência; quanto mais alta a frequência gerada, maior será o nível de atenuação. Esse nível é especificado nos bits KSL.

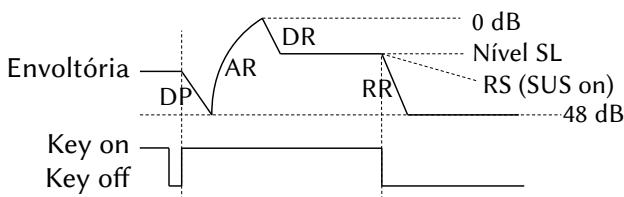
### EG-TYP (b5)

Esse bit seleciona o tipo de envoltória, que pode ser tom constante ou tom percussivo. Se o bit for 0, o tom será percussivo e se for 1 o tom será constante, conforme ilustração abaixo.

#### Tom Percussivo (b5=0)



#### Tom Constante (b5=1)



### VIB (b6)

Flag usada para ativar ou desativar o vibrato. Se for 1, o vibrato estará ativo e se for 0, estará desligado. A frequência do vibrato é de 6,4 Hz.

### AM (b7)

Flag usada para ativar ou desativar a modulação de amplitude ou tremolo. Se for 1, a modulação de amplitude estará ativa e se for 0 estará desligada. A frequência para a modulação de amplitude é de 3,7 Hz.

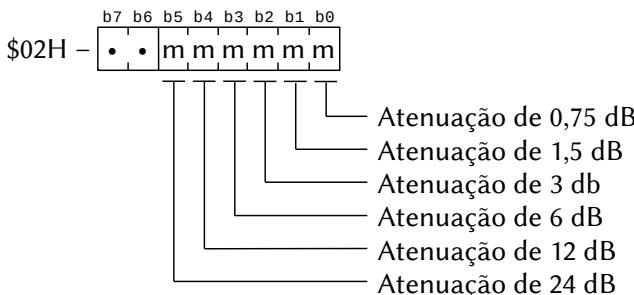
- KSL/NÍVEL TOTAL/DISTORÇÃO/NÍVEL REALIMENT. (\$02H,\$03H)

Esses registradores são usados para regular a saída de modo que o som gerado pelo OPLL se aproxime dos instrumentos musicais reais.

	b7	b6	b5	b4	b3	b2	b1	b0
\$02H	KSL(m)							Nível total (c)
\$03H	KSL(c)	•	DC	DM				Realimentação

### NÍVEL TOTAL (b0~b5)

Esse valor permite controlar o nível de modulação através da atenuação do mesmo (envoltória). Com o valor 000 000, não haverá atenuação e a modulação será máxima. Já com o valor 111 111, a atenuação será máxima, de aproximadamente 48 dB.



Para obter o valor de atenuação correto, basta somar os valores quando o bit respectivo for 1.

### KSL (b6~b7)

Esses bits controlam o nível da “key scaling”. No modo “key scale” (KSR = 1), o nível de atenuação progressiva do som pode variar de 0 dB por oitava até 6 dB por oitava, conforme a tabela abaixo:

b7	b6	Atenuação
0	0	0 dB / oitava
0	1	1,5 dB / oitava
1	0	3 dB / oitava
1	1	6 dB / oitava

DM (b3, \$03H)

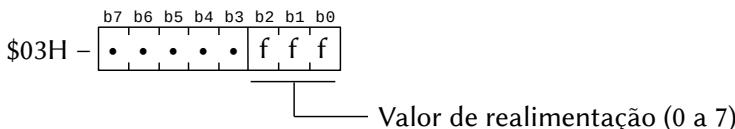
Quando esse bit for igual a 1, a onda moduladora é retificada para meia onda.

DC (b4, \$03H)

Quando esse bit for igual a 1, a onda portadora é retificada para meia onda.

REALIMENTAÇÃO (FEEDBACK) (b0~b2, \$03H)

Esses bits definem o índice de realimentação (porção do sinal de saída que é reinjetado na entrada) para a onda moduladora.



Valor do registrador: 000 001 010 011 100 101 110 111

Valor de realimentação: 0  $\pi/16$   $\pi/8$   $\pi/4$   $\pi/2$   $\pi$   $2\pi$   $4\pi$

- RELAÇÃO ATTACK/DECAY (\$04H e \$05H)

As relações de “attack” e “decay” são definidas pelos registradores \$04H e \$05H, conforme a ilustração abaixo. Quando maior o valor, menor o tempo de “attack” (AR) e/ou “decay” (DR). A variação de tempo obedece, aproximadamente, a uma progressão geométrica.

	b7	b6	b5	b4	b3	b2	b1	b0
\$04H					Attack (moduladora)		Decay (moduladora)	
\$05H					Attack (portadora)		Decay (portadora)	

Os valores de “attack” e “decay” estão descritos na tabela abaixo. Os tempos estão expressos em milissegundos (mS).

RATE		EG decay time (mS)		EG attack time (mS)	
RM	RL	0dB~48dB	10% ~ 90%	0dB~48dB	10% ~ 90%
15	3	1.27	0.52	0.00	0.00
15	2	1.27	0.52	0.00	0.00
15	1	1.27	0.52	0.00	0.00
15	0	1.27	0.52	0.00	0.00
14	3	1.47	0.60	0.14	0.10

14	2	1.71	0.60	0.18	0.12
14	1	2.05	0.82	0.22	0.14
14	0	2.55	1.03	0.28	0.18
13	3	2.94	1.21	0.30	0.22
13	2	3.42	1.37	0.34	0.22
13	1	4.10	1.65	0.42	0.26
13	0	5.11	2.05	0.50	0.32
12	3	5.87	2.41	0.54	0.36
12	2	6.84	2.74	0.60	0.38
12	1	8.21	3.30	0.70	0.44
12	0	10.22	4.10	0.84	0.54
11	3	11.75	4.03	0.97	0.64
11	2	13.60	5.47	1.13	0.72
11	1	16.41	6.60	1.37	0.84
11	0	20.44	8.21	1.69	1.09
10	3	23.49	9.65	1.93	1.29
10	2	27.36	10.94	2.25	1.45
10	1	32.03	13.19	2.74	1.69
10	0	40.07	16.41	3.30	2.17
9	3	46.99	19.31	3.86	2.57
9	2	54.71	12.80	4.51	2.98
9	1	65.65	26.39	5.47	3.30
9	0	81.74	32.03	6.76	4.34
8	3	93.97	38.62	7.72	5.15
8	2	109.42	43.77	9.01	5.79
8	1	131.31	52.78	10.94	6.76
8	0	163.49	65.65	13.52	8.69
7	3	187.95	77.24	15.45	10.30
7	2	218.84	87.54	18.02	11.59
7	1	262.61	185.56	21.00	13.52
7	0	326.98	131.31	27.03	17.30
6	3	375.98	154.40	30.90	20.60
6	2	437.69	175.07	36.04	23.17
6	1	525.22	211.12	43.77	27.03
6	0	653.95	262.61	54.87	34.76
5	3	751.79	308.96	61.79	41.19
5	2	875.37	350.15	72.89	46.34

5	1	1050.45	422.24	87.54	54.07
5	0	1307.91	525.22	108.13	69.51
4	3	1503.58	617.91	123.50	82.39
4	2	1750.75	700.30	144.48	92.69
4	1	2180.89	844.40	175.07	108.13
4	0	2615.82	1050.45	216.27	139.63
3	3	3007.16	1235.82	247.16	164.70
3	2	3501.49	1400.60	280.36	185.37
3	1	4201.79	1680.95	358.15	216.27
3	0	5231.64	1280.89	432.54	278.06
2	3	6014.32	2471.64	494.33	329.55
2	2	7002.98	2801.19	576.72	370.75
2	1	8403.58	3377.91	780.30	432.54
2	0	10463.30	4201.79	865.88	556.12
1	3	12078.66	4943.20	988.66	659.11
1	2	14606.80	5602.39	1153.43	741.49
1	1	16807.20	6755.82	1400.60	865.80
1	0	20926.60	8403.58	1730.15	1112.24

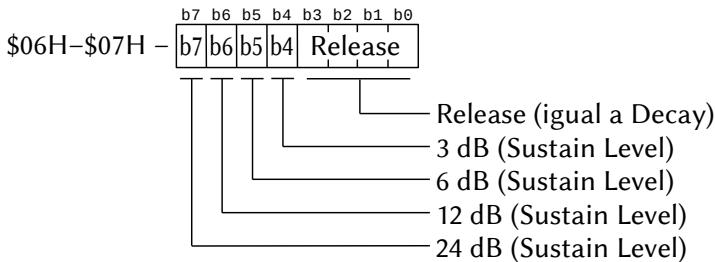
- SUSTAIN LEVEL / RELEASE RATE (\$06H e \$07H)

“Sustain level” é o nível no qual a envoltória permanece após ter sido atenuada pelo “decay rate”. Para o tom percussivo, é o ponto de troca do modo “decay” para o modo “release”. Quanto maior o valor do registrador, mais baixo será o nível de “sustain”.

“Release rate” é a relação de desaparecimento do som após a “key off”. Para o tom percussivo, é expressada pela atenuação após o “sustain level”. Quanto maior o valor do registrador, menor será a duração do “release rate”.

	b7	b6	b5	b4	b3	b2	b1	b0
\$06H	Sustain (moduladora)				Release (moduladora)			
\$07H		Sustain (portadora)				Release (portadora)		

Os valores para “sustain level” são calculados da seguinte forma:

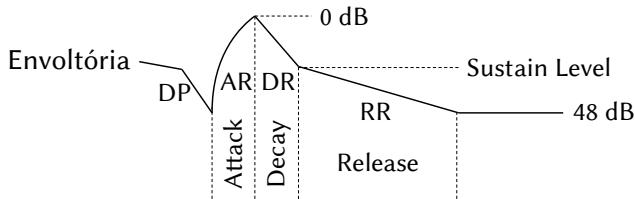


Para obter o valor correto de “sustain”, é preciso somar os valores quando o bit respectivo for “1”.

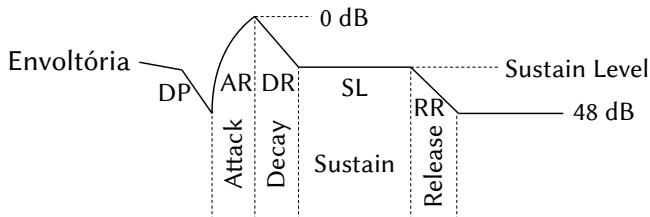
Os tempos referentes ao “release rate” são os mesmos do “decay rate” ilustrados na tabela acima.

Abaixo, há uma ilustração dos valores de “attack”, “decay”, “sustain level” e “release rate” na forma de onda.

Tom Percussivo (b5=0)



Tom Constante

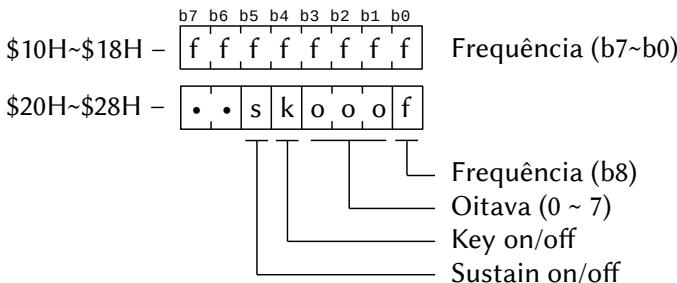


### 6.3.3.3 – Registradores de seleção

- OITAVA/FREQUÊNCIA/KEY/SUSTAIN (\$10H~\$18H, \$20H~\$28H)

São nove grupos de dois registradores de 8 bits cada, formando pares, sendo numerados de \$10H~\$18H a \$20H~\$28H. Assim, os registradores \$10H e \$20H controlam a primeira voz, os registradores \$11H e

\$21H controlam a segunda voz e assim por diante. São esses registradores que definem a frequência de cada uma das 9 vozes que podem ser geradas pelo OPLL.



### FREQUÊNCIA (\$1xH e bit 0 de \$2xH)

Esses 9 bits definem um escala de frequências para cada oitava. Na tabela abaixo estão especificados os valores dos registradores para a quarta oitava (de um total de 8 oitavas), com a nota LÁ central de 440 Hz.

Cifrado	Frequência	Decimal	\$2xH,b0	\$1xH
Dó C#	277.2 Hz	181	0	1 0 1 1 0 1 0 1
Ré D	293.7 Hz	192	0	1 1 0 0 0 0 0 0
D#	311.1 Hz	204	0	1 1 0 0 1 1 0 0
Mi E	329.6 Hz	216	0	1 1 0 1 1 0 0 0
Fá F	349.2 Hz	229	0	1 1 1 0 0 1 0 1
F#	370.0 Hz	242	0	1 1 1 1 0 0 1 0
Sol G	392.0 Hz	257	1	0 0 0 0 0 0 0 1
G#	415.3 Hz	272	1	0 0 0 1 0 0 0 0
Lá A	440.0 Hz	288	1	0 0 1 0 0 0 0 0
A#	466.2 Hz	305	1	0 0 1 1 0 0 0 1
Si B	493.9 Hz	323	1	0 1 0 0 0 0 0 1
Dó C	523.3 Hz	343	1	0 1 0 1 0 1 1 1

Os valores das frequências guardam entre si uma relação geométrica igual à 12<sup>a</sup> raiz de 2, que vale 1,0594630943592. Pode-se usar esse número para alterar os valores dos registradores a fim de aumentar ou diminuir a frequência gerada dentro da escala musical. Os valores dos registradores também guardam entre si a mesma relação.

### OITAVA (\$2xH, b3~b1)

Esses três bits definem a oitava. Podem ser definidas até 8 oitavas, de 000 a 111, sendo que a quarta oitava é a 011.

### KEY (\$2xH, b4)

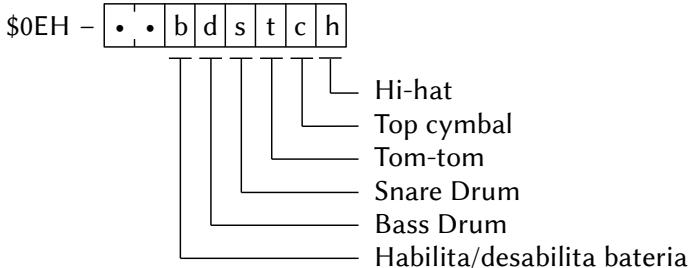
Esse bit deve ser setado em 1 para que o som de cada uma das nove vozes seja habilitado. Quando for 0, o som da voz respectiva estará desligado (key off).

### SUSTAIN (\$2xH, b5)

Quando esse bit estiver setado em 1, o valor de “release rate - RR” decairá gradativamente quando o bit “key” respetivo for desligado; caso contrário, o som será cortado abruptamente.

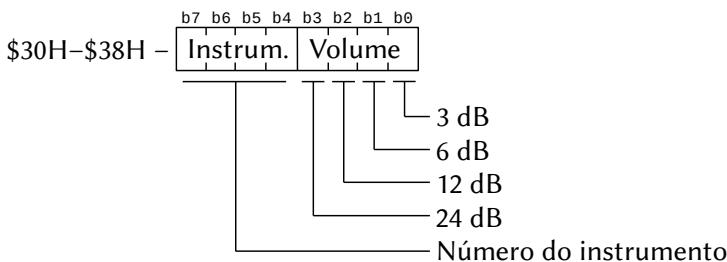
- **CONTROLE DA BATERIA (\$0EH)**

O registrador \$0EH controla o modo de bateria do OPLL. Para ativá-lo, basta setar seu bit 5 em 1. Os bits 0 a 4 habilitam ou desabilitam cada uma das 5 peças de bateria disponíveis. Quando estiver no modo bateria, somente as seis primeiras vozes do OPLL estarão disponíveis para a geração de sons de outros instrumentos musicais.



- **SELEÇÃO DE INSTRUMENTOS E VOLUME (\$30H~\$38H)**

Esses registradores selecionam o instrumento e o volume para cada uma das nove vozes disponíveis. Assim, \$30H é usado para a primeira voz, \$31H para a segunda e assim por diante.



Os bits b3~b0 determinam o volume. A menor resolução é 3 dB e a maior 45 dB, obtida somando os valores quando o bit respectivo for “1”.

Os bits b7~b4 selecionam o instrumento, sendo que o valor 0000B seleciona o instrumento definido pelo usuário. Os 15 instrumentos disponíveis são os seguintes:

0001 – violino	0110 – oboé	1011 – cravo
0010 – violão	0111 – trompete	1100 – vibrafone
0011 – piano	1000 – órgão	1101 – baixo elétrico
0100 – flauta	1001 – piston	1110 – baixo acústico
0101 – clarinete	1010 – sintetizador	1111 – guitarra elétrica

No modo bateria, os registradores \$36H, \$37H e \$38H determinam apenas o volume de cada uma das peças de bateria disponíveis, mas os registradores \$30H a \$35H mantêm suas funções inalteradas. Nesse modo, o OPLL pode gerar seis instrumentos mais cinco peças de bateria.

\$36H	•	•	•	•	Bass Drum	Registradores de volume para as peças de bateria
\$37H	Hi Hat				Snare Drum	
\$38H	Tom tom				Top Cymbal	

### 6.3.4 – A FM-BIOS

Normalmente, o OPLL vem acompanhado de uma ROM que permite acessar, através do BASIC, todos os registradores do mesmo. Esse BASIC ampliado denomina-se MSX-MUSIC. Adicionalmente, há a definição de mais 48 instrumentos nessa ROM. Assim, tem-se acesso a até 63 instrumentos. Porém apenas os 15 instrumentos internos do

OPLL podem ser mixados entre si livremente em quaisquer das nove vozes. Os instrumentos selecionados da FM-BIOS não podem ser mixados uns com os outros, já que o OPLL aceita a definição externa de apenas um instrumento. A FM-BIOS reserva o número 63 (silence) para a definição externa de instrumento pelo MSX-MUSIC. Para acesso direto, o instrumento definido pelo usuário é o de número 0. A tabela abaixo traz os valores de definição de todos os instrumentos da FM-BIOS.

	Instrumento	\$0	\$1	\$2	\$3	\$4	\$5	\$6	\$7
00	Piano 1	Dados do OPLL (3)							
01	Piano 2	30	10	0F	04	D9	B2	10	F4
02	Violin	Dados do OPLL (1)							
03	Flute 1	Dados do OPLL (4)							
04	Clarinet	Dados do OPLL (5)							
05	Oboe	Dados do OPLL (6)							
06	Trumpet	Dados do OPLL (7)							
07	Pipeorgan	34	30	37	06	50	30	76	06
08	Xylophone	17	52	18	05	88	D9	66	24
09	Organ	Dados do OPLL (8)							
10	Guitar	Dados do OPLL (2)							
11	Santool 1	19	53	0C	06	C7	F5	11	03
12	Electric guitar	Dados do OPLL (15)							
13	Clavicode 1	03	09	11	06	D2	B4	F5	F6
14	Harpsicode 2	Dados do OPLL (11)							
15	Harpsicode 2	01	01	11	06	C0	B4	01	F7
16	Vibraphone	Dados do OPLL (12)							
17	Koto 1	13	11	0C	06	FC	D2	33	84
18	Taiko	01	10	0E	07	CA	E6	44	24
19	Engine 1	E0	F4	1B	87	11	F0	04	08
20	UFO	FF	70	19	07	50	1F	05	01
21	Synthesizer bell	13	11	11	07	FA	F2	21	F5
22	Chime	A6	42	10	05	FB	B9	11	02

23	Synthesizer bass	Dados do OPLL (13)							
24	Synthesizer	Dados do OPLL (10)							
25	Synthesizer percussion	01	03	0B	07	BA	D9	25	06
26	Synthesizer rhythm	40	00	00	07	FA	D9	37	04
27	Harmdrum	02	03	09	07	CB	FF	39	06
28	Cowbell	18	11	09	05	F8	F5	26	26
29	Close hi-hat	0B	04	09	07	F0	F5	01	27
30	Snare drum	40	40	07	07	D0	D6	01	27
31	Bass drum	00	01	07	06	CB	E3	36	25
32	Piano 3	11	11	08	04	FA	B2	20	F5
33	Wood bass	Dados do OPLL (14)							
34	Santool 2	19	53	15	07	E7	95	21	03
35	Brass	30	70	19	07	42	62	26	24
36	Flute 2	62	71	25	07	64	43	12	26
37	Clavicode 2	21	03	0B	05	90	D4	02	F6
38	Clavicode 3	01	03	0A	05	90	A4	03	F6
39	Koto 2	43	53	0E	85	B5	E9	85	04
40	Pipe organ	34	30	26	06	50	30	76	06
41	RhodsPLA	73	33	5A	06	99	F5	14	15
42	RhodsPRA	73	13	16	05	F9	F5	33	03
43	Orch L	61	21	15	07	76	54	23	06
44	Orch R	63	70	1B	07	75	4B	45	15
45	Synthesizer violin	61	A1	0A	05	76	54	12	07
46	Synthesizer organ	61	78	0D	05	85	F2	14	03
47	Synthesizer brass	31	71	15	07	B6	F9	03	26
48	Tube	Dados do OPLL (9)							
49	Shamisen	03	0C	14	06	A7	FC	13	15
50	Magical	13	32	81	03	20	85	03	B0
51	Huwawa	F1	31	17	05	23	40	14	09
52	Wander flat	F0	74	17	47	5A	43	06	FD

53	Hardrock	20	71	0D	06	C1	D5	56	06
54	Machine	30	32	06	06	40	40	04	74
55	Machine V	30	32	03	03	40	40	04	74
56	Comic	01	08	0D	07	78	F8	7F	FA
57	SE-Comic	C8	C0	0B	05	76	F7	11	FA
58	SE-Laser	49	40	0B	07	B4	F9	00	05
59	SE-Noise	CD	42	0C	06	A2	F0	00	01
60	SE-Star 1	51	42	13	07	13	10	42	01
61	SE-Star 2	51	42	13	07	13	10	42	01
62	Engine 2	30	34	12	06	23	70	26	02
63	Silence	00	00	00	00	00	00	00	00

A tabela relaciona os 63 instrumentos do MSX-MUSIC. Os oito bytes relacionados devem preencher, respectivamente, os oito primeiros registradores do OPLL (\$00H a \$07H), que são os responsáveis pela definição do instrumento criado pelo programador. A tabela traz também os instrumentos internos do OPLL e, nesse caso, ao invés dos bytes, traz a expressão “dados do OPLL”, seguido do número do instrumento.

### 6.3.5 – O FM estéreo

Embora não previsto oficialmente para o padrão MSX, o FM estéreo acabou sendo padronizado pelo mercado devido a uma característica do OPLL: o chip responsável, o YM2413, possui duas saídas separadas para os sons; uma é denominada “melody output” e por ela o OPLL gera as seis primeiras vozes; a outra é denominada “rhythm output” e por ela o OPLL gera as três vozes restantes ou as cinco peças de bateria. Convencionou-se então que a “melody output” seria um dos canais estéreo e a “rhythm output” mais o som do PSG seria o outro canal estéreo.

Assim, o FM estéreo pode gerar dois canais de seis vozes cada. Muitos programas, especialmente jogos, podem fazer uso do FM estéreo mesmo que não tenham sido programados para usá-lo, o que, na maioria das vezes, gera um belíssimo som de suaves nuances.

### 6.3.6 – Acesso ao OPLL

O acesso ao OPLL pode ser feito diretamente ou através da FM-BIOS, que é uma ROM específica instalada no cartucho FM-OPLL e também em todos os MSX que vêm com o OPLL. Esta ROM está localizada na página física 1 (entre 4000H e 7FFFH) e possui algumas rotinas que podem ser acessadas pelo usuário.

Para acessar o OPLL, o primeiro passo é verificar a presença da FM-BIOS. Para isso, é preciso fazer uma varredura procurando as seguintes strings em todos os slots e subslots na ordem a seguir:

1º – “APRLOPLL” em 4018H ~ 401FH (Interno ou clone)

2º – “OPLL” em 401CH ~ 401FH (Panasoft FM-PAC)

A ordem das verificações é importante porque, se for feita na ordem errada, o Panasoft FM-PAC tocará simultaneamente com o FM-PAC interno. Isso também significa que um MSX-MUSIC interno (ou cartucho clone) tem precedência sobre um Panasoft FM-PAC.

Se o primeiro for encontrado, você poderá usar diretamente o OPLL nas portas de E/S. Se a primeira não for encontrada, mas a segunda for, então você precisa primeiro habilitar as portas de E/S configurando o sinalizador de habilitação de E/S no endereço 7FF6H (veja abaixo), ou usar a E/S mapeada na memória.

#### Acesso direto ao OPLL interno

Porta I/O	R/W	Uso	Espera (μsec)	Espera (ciclos relógio)
7CH	W	Seleção	3.4	12
7DH	W	Dados	23.5	84

#### Acesso direto ao OPLL externo (em cartucho)

Porta I/O	Endereço	R/W	Uso	Espera (μsec)	Espera (ciclos relógio)
7CH	7FF4H	W	Seleção	3.4	12
7DH	7FF5H	W	Dados	23.5	84

Para ativar as portas de E/S para acesso ao OPLL em cartucho, é preciso setar o bit “0” do endereço 7FF6H no slot FM-PAC, caso contrário o chip OPLL estará disponível para uso somente através de endereços memória. Para evitar o acesso simultâneo a dois dispositivos nas mesmas portas de E/S, não é recomendado ativar o FM-PAC quando houver MSX-MUSIC interno presente.

Se houver uma string “APRLOPLL”, o bit0 de #7FF6 não deve ser setado, pois isso quebra a compatibilidade com computadores Panasonic MSX2+. Nos Panasonic A1WX e A1WSX, setar o bit 0 em qualquer endereço entre #7FF0 e #7FFF desabilitará a FM-BIOS completamente.

### **6.3.6.1 – Acesso através da FM-BIOS**

Se você usar MSX-BASIC ou FM-BIOS, nenhuma atenção extra será necessária porque o comando CALL MUSIC e a rotina OPLINI ativarão automaticamente o OPLL quando necessário. Essas rotinas sempre procurarão primeiro o MSX-MUSIC interno (APRLOPLL) e depois o externo (PAC2OPLL) e ativará o externo somente se o interno não for encontrado.

As seguintes rotinas estão disponíveis na FM-BIOS:

WRTOPL (4110H) – Escreve um byte de dados em um registrador do OPLL.

INIOPL (4113H) – Inicializa a área de trabalho da FM-BIOS/OPLL.

MSTART (4116H) – Começa a tocar a música.

MSTOP (4119H) – Parar a música.

RDDATA (411CH) – Retorna os dados dos instrumentos da ROM.

OPLDRV (411FH) – Entrada para o driver OPLL. É a rotina que toca a música, devendo ser chamada pelo manipulador de interrupções através do hook HTIMI.

TSTBGM (4122H) – Verifica se ainda há dados na fila musical.

Estas rotinas estão detalhadamente descritas no Apêndice, item “8.7 – ROTINAS DA MSX-MUSIC (FM/OPLL)”.

### **6.3.6.2 – Acesso direto**

O acesso direto é feito por duas portas de I/O, a 7CH e a 7DH ou pelos endereços 7FF4H e 7FF5H. A porta 7CH (ou 7FF4H) seleciona os registradores e a porta 7DH (ou 7FF5H) escreve os bytes de dados nos

mesmos. Entretanto, o OPLL é lento. Entre um acesso e outro deve haver uma pausa, conforme tabela abaixo.

Seleção de registradores	(7CH)	12 ciclos (master clock – 3,58 MHz)
Escrita de dados	(7DH)	84 ciclos (master clock – 3,58 MHz)

Estes valores correspondem aos seguintes tempos:

	Tempo	Ciclos T Z80	Ciclos T R800
Seleção (7CH)	3,4 µs	12	24
Escrita (7DH)	23,5 µs	84	168

Observe que o clock interno do R800 é 7,16 MHz; portanto o número de ciclos T deve ser o dobro. Ao contrário do VDP, não são geradas pausas por hardware para o acesso ao YM2413; portanto haverá dessincronização e os dados serão corrompidos quando o R800 estiver ativo no MSX turbo R, caso o programa não tenha previsto isso.

Para o Z80, podem ser usadas pausas tipo “EX (SP),HL” ou “NOP” até que o OPLL esteja pronto para novo acesso.

Primeiramente, deve-se selecionar o registrador a ser escrito através da porta 7CH. Após a escrita, deve-se dar uma pausa de, no mínimo, 12 ciclos T no caso de um MSX padrão (Z80 a 3,58 MHz). A instrução “OUT (07CH),A” demora 11 ciclos T para ser processada; portanto é necessária, ao menos, mais 1 ciclo T. Pode ser usada uma instrução NOP (que demora 4 ciclos T para ser processada) para isso, conforme ilustração abaixo:

```
LD      A,REG      ; Número do registrador em A
OUT    (07CH),A   ; Seleciona o registrador
NOP                ; Pausa
```

Logo em seguida, escreve-se o dado no registrador selecionado através da porta 7DH. A pausa agora deve ser de, no mínimo, 84 ciclos T numa máquina MSX padrão (Z80 a 3,58 MHz). Para isso, podem ser usadas 4 instruções “EX (SP),HL” (que demoram 19 ciclos T cada para serem processadas), resultando numa pausa de 76 ciclos T que, somados aos 11 ciclos da instrução OUT, resultam em 87 ciclos T. Então, o OPLL estará pronto para receber novo dado. É essencial que as instruções “EX

(SP),HL” estejam em dupla; caso contrário o conteúdo da pilha será alterado, provocando corrupção de dados.

```

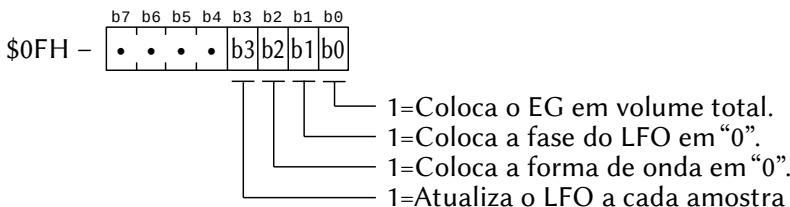
LD   A,REG      ; Número do registrador em A
OUT (07CH),A   ; Seleciona o registrador
NOP           ; Pausa
LD   A,DADO     ; Dado a ser escrito
OUT (07DH),A   ; Escreve o dado no registrador
EX  (SP),HL    ; Pausa
EX  (SP),HL    ; Pausa
EX  (SP),HL    ; Pausa
EX  (SP),HL    ; Pausa

```

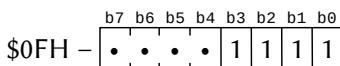
O tempo de pausa também pode ser usado para processar alguma tarefa, desde que este processamento demore ao menos 84 ciclos T para o Z80 ou 168 para o R800 e não interfira nos valores a serem escritos no OPLL.

### 6.3.7 – Acesso ao DAC interno

O OPLL possui internamente um DAC de 4 bits que pode ser acessado quando o chip é colocado em modo de teste. Funciona da mesma forma que o PCM descrito mais adiante no item 4. Para acessá-lo, é necessário usar o registrador de teste (\$0FH).



Para habilitar o DAC interno, é preciso colocar os bits 0 a 3 do registrador de teste em “1”:



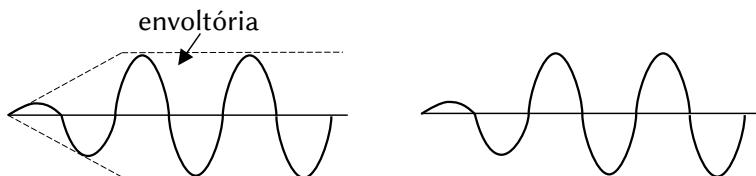
Então os dados podem ser enviados para os 4 bits mais altos do registrador \$10H. É necessário usar uma máscara do tipo “AND 0F0H” para que os quatro bits mais baixos permaneçam em “0”. Antes de enviar os dados, é necessário setar alguns registradores:

$\$0FH \leftarrow 15$  ou 255  
 $\$30H \leftarrow 140$  ou 255  
 $\$20H \leftarrow 31$  ou 255

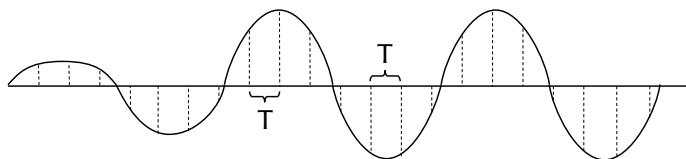
Em seguida podem ser enviados dos dados PCM para  $\$10H$ , usando a máscara “AND  $0F0H$ ”. A temporização deve ser controlada por software.

#### 6.4 – O PCM

PCM significa “pulse code modulation”, ou “modulação por código de pulsos”. O PCM não é um gerador de sons propriamente dito; funciona como digitalizador ou “sampler” de sons. Dessa forma, é possível reproduzir sons de qualquer natureza, inclusive a voz humana, de forma praticamente perfeita. O PCM não tem registradores para especificar os sons; estes são obtidos por amostragem. Um sinal típico gerado pelo PCM, no caso uma onda senoidal, está ilustrado abaixo:



A forma de atuação do PCM para reproduzir essa mesma onda está ilustrada abaixo:



A cada período  $T$ , o PCM faz uma coleta do nível de som. O período  $T$  é fixo e a frequência com a qual se processa a amostragem é chamada de “sampling rate”. Para reproduzir o som, basta repetir os dados na mesma velocidade em que foram coletados. Quando maior a frequência de amostragem (sampling rate), melhor a qualidade do som reproduzido.

Outra característica do PCM é a resolução. No caso do MSX, a resolução é de 8 bits, sou seja, cada coleta tem 8 bits e por isso a ampli-

tude da onda sonora coletada tem uma variação de 256 níveis. Cada amostra ocupa, portanto, um byte de memória.

No MSX, há 4 taxas de amostragem (sampling rate) padronizadas: 3,9375 KHz, 5,25 KHz, 7,875 KHz e 15,75 KHz. Isso quer dizer que, por exemplo, na taxa de 5,25 KHz, há 5250 coletas de 8 bits a cada segundo, e para o armazenamento de 1 segundo de som, são gastos mais de 5 Kbytes de memória. Por isso, os dados para o PCM são armazenados na própria RAM do micro e não em registradores.

#### 6.4.1 – Acesso ao PCM

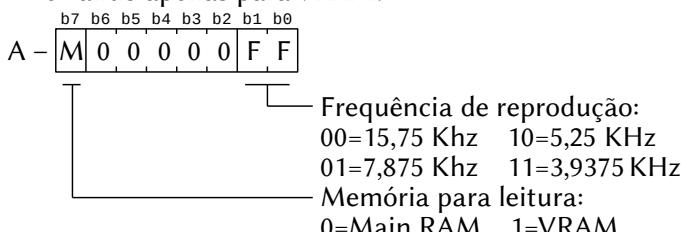
O PCM pode ser acessado tanto pela BIOS quanto diretamente. No caso de acesso pela BIOS, há duas rotinas disponíveis:

**PCMPLY** (0186H/Main)

Função: Reproduzir sons através do PCM.

Entrada: EHL – Endereço para início da leitura. “E” é válido apenas para VRAM.

DBC – Tamanho do bloco a reproduzir (comprimento). “D” é válido apenas para VRAM.



Obs.: Usar 15,75 KHz apenas no modo R800 DRAM.

Saída: CY = 0 → Reprodução OK; 1 → Erro na reprodução.

A = 0 → Erro na especificação da frequência.

A = 1 → Interrupção por CTRL+STOP.

EHL – Endereço até onde efetivamente reproduziu. “E” é válido apenas para VRAM.

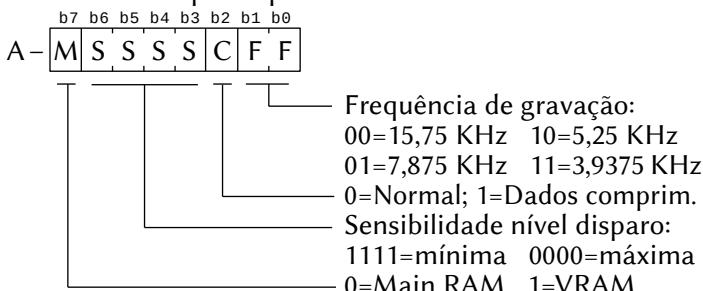
Registradores: Todos.

**PCMREC** (0189H/Main)

Função: Digitalizar sons através do PCM.

Entrada: EHL – Endereço para início da gravação. “E” é válido apenas para VRAM.

DBC – Tamanho do bloco a gravar (comprimento). “D” é válido apenas para VRAM.



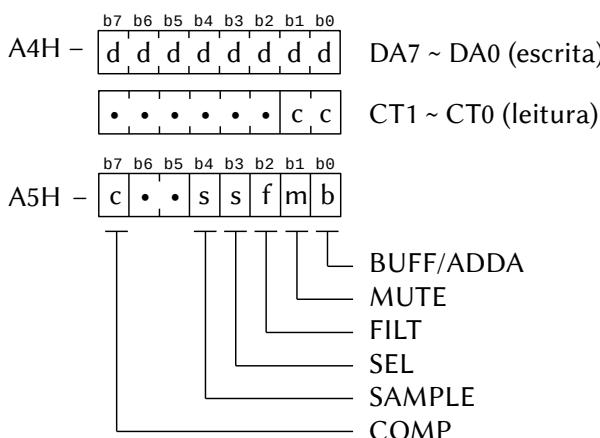
Obs.: A frequência de 15,75 Khz só pode ser usada no modo R800 DRAM.

Saída: CY = 0 → Gravação OK; 1 → Erro na gravação.  
A = 0 → Erro na especificação da frequência.  
1 → Interrupção por CTRL+STOP.

EHL – Endereço até onde efetivamente gravou. “E” é válido apenas para VRAM.

Registradores: Todos.

O acesso direto ao PCM é feito por duas portas de I/O. A porta A4H é a porta de dados e a A5H é a de comando.



BUFF/ADDA: define a direção de conversão. Para geração de som (saída), esse bit deve ser 0 (conversão D/A). Para digitalização de som (entrada), esse bit deve ser 1 (conversão A/D).

MUTE: liga ou desliga a saída de som de todo o sistema. Se for 0, a saída estará desligada (modo selecionado no reset). Se for 1, estará ligada.

FILT: define o tipo de sinal para a conversão A/D. Se for 0, será usado do sinal normal (selecionado no reset). Se for 1, o sinal passará pelo filtro.

SEL: seleciona o sinal de entrada do filtro. Se for 0, será usado o filtro passa-baixa (conversão D/A). Se for 1, será usado o sinal do microfone.

SAMPLE (sample hold): define como será tratado o sinal de entrada. Se for 0, a conversão A/D estará ativa (modo selecionado no reset). Se for 1, estará desligada.

COMP: esse bit só é válido para a leitura da porta. Ele define o nível do sinal do comparador de saída (conversão D/A). Se esse bit for 1, o sinal do conversor D/A será maior que o sinal do “sample hold”. Se for 0, será menor.

DA7 ~ DA0: dados de saída do conversor D/A. O formato dos dados é em binário absoluto, onde o valor 127 corresponde ao nível 0.

CT1 ~ CT0 (counter data): é um contador de referência. A cada 63,5 $\mu$ s, o contador é incrementado. Como esse período corresponde à frequência de 15,75 KHz, o contador serve como referência para os quatro “sampling rates” disponíveis, como ilustrado abaixo:

00 – 15,75 KHz	01 – 7,875 KHz
10 – 5,25 KHz	11 – 3,9375 KHz

Ao se escrever um dado em A4H, o contador é resetado.

Para digitalizar um som usando acesso direto, é necessário ler os dados bit a bit. Segue uma rotina em assembler que faz a digitalização de sons através do PCM.

```
PMDAC EQU 0A4H
PMCNT EQU 0A4H
PMCNTL EQU 0A5H
PMSTAT EQU 0A5H
SYSTML EQU 0E6H
```

```
REC: LD A, 00001100B
```

```

        OUT  (PMCTRL),A ; MODO A/D
        DI
        XOR  A
        OUT  (SYSTML),A ; CONTADOR 0
REC1:   IN   A, (SYSTML)
        CP   E
        JR   C, REC1
        XOR  A
        OUT  (SYSTML),A
        PUSH BC
        LD   A, 00011100B
        OUT  (PMCTRL),A ; SEGURA DADO
        LD   A, 80H
        LD   C, PMSTAT
        OUT  (PMDAC),A ; CONVERSÃO DO BIT
        DEFB 0EDH,70H ; IN F, (C)
        JP   M, RECAD0
        AND  01111111B
RECAD0: OR   01000000B ; BIT 7
        OUT  (PMDAC),A
        DEFB 0EDH,70H ; IN F, (C)
        JP   M, RECAD1
        AND  10111111B
RECAD1 :OR   00100000B ; BIT 6
        OUT  (PMDAC),A
        DEFB 0EDH,70H ; IN F, (C)
        JP   M, RECAD2
        AND  11011111B
RECAD2: OR   00010000B ; BIT 5
        OUT  (PMDAC),A
        DEFB 0EDH,70H ; IN F, (C)
        JP   M, RECAD3
        AND  11101111B
RECAD3: OR   00001000B ; BIT 4
        OUT  (PMDAC),A
        DEFB 0EDH,70H ; IN F, (C)
        JP   M, RECAD4
        AND  11110111B

```

```

RECAD4: OR   00000100B      ; BIT 3
        OUT  (PMDAC),A
        DEFB 0EDH,70H      ; IN F, (C)
        JP   M,RECAD5
        AND  11111011B
RECAD5: OR   00000010B      ; BIT 2
        OUT  (PMDAC),A
        DEFB 0EDH,70H      ; IN F, (C)
        JP   M,RECAD6
        AND  11111101B
RECAD6: OR   00000001B      ; BIT 1
        OUT  (PMDAC),A
        DEFB 0EDH,70H      ; IN F, (C)
        JP   M,RECAD7
        AND  11111110B
RECAD7: OR   00000000B      ; BIT 0
        LD   (HL),A          ; SALVA DADO LIDO
        LD   A,00001100B
        OUT  (PMCTRL),A
        POP BC
        INC HL
        DEC BC
        LD   A, C
        OR   B
        JR   NZ,REC1
        LD   A,00000011B
        OUT  (PMCTRL),A      ; MODO D/A
        EI
;
        RET

```

Para reproduzir os sons, é preciso setar o PCM para reprodução (porta A5H) e depois enviar os bytes de dados na velocidade correta através da porta A4H. A rotina a seguir reproduz os sons através do PCM.

PMDAC	EQU	0A4H
PMCNT	EQU	0A4H
PMCTRL	EQU	0A5H
PMSTAT	EQU	0A5H
SYSTML	EQU	0E6H

```

PLAY: LD A,0000001B
      OUT (PMCNTL),A ; MODO D/A
      DI
      XOR A
      OUT (SYSTML),A ; CONTADOR 0
PLAY1 : IN A,(SYSTML)
      CP E
      JR C,PLAY1
      XOR A
      OUT (SYSTML),A
      LD A,(HL) ; LÊ BYTE DE DADOS
      OUT (PMDAC),A ; REPRODUZ O DADO
      INC HL
      DEC BC
      LD A,C
      OR B
      JR NZ,PLAY1
      EI
      ;
      RET

```

## 6.5 – O MSX-AUDIO

O MSX-Audio foi lançado juntamente com o MSX2 em 1985 como periférico opcional padronizado, mas, provavelmente por seu preço elevado, acabou não se tornando padrão, cedendo lugar mais tarde ao OPLL. Foi utilizado em apenas alguns cartuchos de som.

O chip responsável é o Y8950. De todos os geradores de som criados para o MSX, o MSX-Audio é o mais completo. Como o OPLL, possui 9 vozes de som FM, mas todas são redefiníveis. Também possui um canal ADPCM (Adaptive Differential Pulse Code Modulation - Modulação por Códigos de Pulses por Adaptação Diferencial), que funciona de forma semelhante ao PCM, mas armazena os dados de forma compactada. Além disso, também pode ser conectado a 256 Kbytes de memória externa, liberando a Main RAM e a CPU e também a um teclado musical externo diretamente. Essas e outras características fazem do MSX-Audio o melhor e mais completo gerador de áudio já criado para o sistema MSX.

### 6.5.1 – Descrição da análise e síntese ADPCM

O ADPCM é um método de análise ou síntese de sons no qual é obtida diferença relativa entre o dado PCM atual e o dado PCM subsequente. Esse método previne a deterioração do som sintetizado e reduz a quantidade de memória requerida. De fato, o ADPCM converte um dado PCM de 8 bits de resolução em um dado ADPCM de 4 bits. A codificação dos dados durante a digitalização se processa da seguinte forma:

1. O dado é digitalizado em 8 bits de resolução ( $X_n$ );
2. O resultado é multiplicado por 256 para convertê-lo em 16 bits e é então comparado com o dado subsequente ( $X_{n+1}$ ) para obter a diferença ( $d_n$ );
3. Quando a diferença resultar num valor positivo, o bit  $L_4$  do dado ADPCM será 0; quando negativo, será 1. Ao mesmo tempo, o valor absoluto da diferença é calculado ( $|d_n|$ ).
4. Então, os três bits restantes são setados de acordo com a tabela abaixo ( $\Delta_n$  é a variação relativa calculada).

$L_4$		$L_3$	$L_2$	$L_1$	Valor analisado
$d_n \geq 0$	$d_n < 0$				
0	1	0	0	0	$ d_n  < \Delta_n / 4$
		0	0	1	$\Delta_n / 4 \leq  d_n  < \Delta_n / 2$
		0	1	0	$\Delta_n / 2 \leq  d_n  < \Delta_n * \frac{3}{4}$
		0	1	1	$\Delta_n * \frac{3}{4} \leq  d_n  < \Delta_n$
		1	0	0	$\Delta_n \leq  d_n  < \Delta_n * \frac{5}{4}$
		1	0	1	$\Delta_n * \frac{5}{4} \leq  d_n  < \Delta_n * \frac{7}{4}$
		1	1	0	$\Delta_n * \frac{7}{4} \leq  d_n  < \Delta_n * \frac{9}{4}$
		1	1	1	$\Delta_n * \frac{9}{4} \leq  d_n $

5. Depois que o dado ADPCM for obtido, novo dado subsequente ( $X_{n+2}$ ) e nova variação ( $\Delta_{n+1}$ ) são obtidos.

$$X_{n+2} = (1 - 2 * L_4) * (L_3 + L_2 / 2 + L_1 / 4 + 1/8) * \Delta_n + X_{n+1}$$

$$\Delta_{n+1} = f(L_3, L_2, L_1) * \Delta_n$$

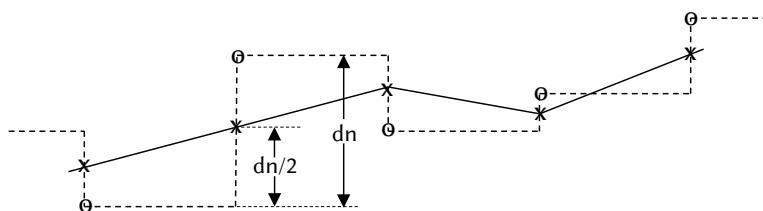
A tabela da página seguinte apresenta os fatores de multiplicação usados para ajustar a variação relativa ( $\Delta n$ ).

$L_3$	$L_3$	$L_3$	$f$
0	0	0	0,9
0	0	1	0,9
0	1	0	0,9
0	1	1	0,9
1	0	0	1,2
1	0	1	1,6
1	1	0	2,0
1	1	1	2,4

Os dados ADPCM completos são obtidos repetindo os passos 1 a 5 para cada amostra PCM de 8 bits.

A síntese de som usando os dados ADPCM é feita lendo-se os dados obtidos no passo 5 e calculando a variação relativa entre eles. A reprodução direta desses dados, entretanto, provoca grande distorção da forma de onda e ruído. Por isso, o MSX-Audio incorpora um procedimento para suavizar a forma de onda.

Primeiramente, os dados são processados através do circuito de suavização. É como um filtro passa-baixa para eliminar o ruído de alta frequência. Depois é feita uma interpolação linear e os dados digitalizados são repetidos a uma frequência de 50 KHz nos intervalos entre as amostras originais. O resultado é mostrado abaixo.



- o : amostra de dados ADPCM
- x : dado após a suavização
- : forma de onda sem a suavização
- : forma de onda após suavização e aplicação de 50 KHz

### 6.5.2 – Mapa dos registradores do MSX-Audio

O MSX-Audio possui 141 registradores de 8 bits para especificar todas as suas funções, numerados de \$01H a \$1AH, \$20H a \$35H, \$40H a \$55H, \$60H a \$75H, \$80H a \$95H, \$A0H a \$A8H, \$B0H a \$B8H, \$BDH e \$C0H a \$C8H. As funções desses registradores estão descritas resumidamente na tabela abaixo.

Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
\$01H	b7	b6	•	•	b3	b2	b1	b0	Registrador de teste
	b7		1=Reseta os LFOs (amplitude máx. / desvio fase zero)						
	b6		1=Afeta o bit 0 (PCMBSY) do registrador de estado.						
	b3		1=Atualiza o LFO a cada amostra (a frequência do tremolo é multiplicada por 64 e do vibrato por 1024)						
	b2		1=Mantém mas coloca a forma de onda em “0”.						
	b1		1=Mantém mas coloca a fase do LFO em “0”.						
	b0		1=Coloca os geradores de envoltória (modulador e portador) em volume total.						
\$02H	1º Temporizador (80 µs)						Registradores de tempo		
\$03H	2º Temporizador (320 µs)								
\$04H	IRQ	T1M	T2M	EOS	BR	•	ST2	ST1	Registrador de sinalizadores
	b7		IRQ – Se colocado em 1, reseta todas as flags.						
	b6		T1M – Se colocado em 1, b0 será colocado em 0.						
	b5		T2M – Se colocado em 1, b1 será colocado em 0.						
	b4		EOS – Máscara p/ b3, indicando fim da operação atual						
	b3		BR – Máscara para ADPCM / Mem. Áudio (1=ativada)						
	b2		Não usado (sempre 0)						
	b1		ST2 – Controla início/parada de \$03 (1=inicia cont.)						
	b0		ST1 – Controla início/parada de \$02 (1=inicia cont.)						
\$05H	Teclado externo (entrada)						Registradores para acesso ao teclado externo		
\$06H	Teclado externo (saída)								
\$07H	STA	REC	MEM	REP	OFF	•	•	RST	Registrador de controle (1)
	b7		STA – Deve ser 1 para iniciar leitura/gravação de dados						
	b6		REC – Deve ser 1 para gravação de dados na memória						
	b5		MEM – Deve ser 1 ao acessar a memória de áudio						
	b4		REP – Quando 1, habilita repetição de dados ADPCM						





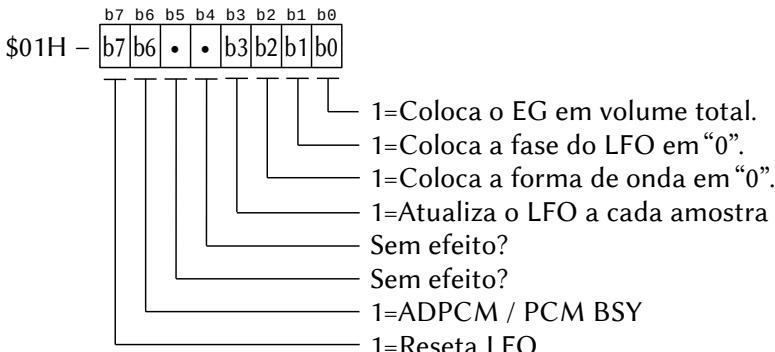
\$BDH	AM	VIB	BAT	BD	SD	TOM	TCY	HH	Controle da bateria do FM
	b7		Grau do tremolo (0=1dB. 1=4,8dB)						
	b6		Grau do vibrato (0=7%; 1=14%)						
	b5		0=Modo Melodia; 1=Modo Bateria						
	b4		1=Bass Drum						
	b3		1=Snare Drum						
	b2		1=Tom-tom						
	b1		1=Top Cymbal						
	b0		1=High-Hat						
\$C0H : \$C8H	•	•	•	•	Feedback	CON	Fator de realimentação para o FM e tipo de conexão		
	b7~b4		Não usados (sempre “0000”)						
	b3~b1		Realimentação (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π)						
	b0		Tipo de conexão dos operadores (0=série; 1=paralelo)						
STAT	INT	T1	T2	EOS	BUF	•	•	PCM	Registrador de estado
	b7		Será 1 quando um ou mais bits b3 a b6 forem 1						
	b6		Será 1 após o tempo contado pelo timer 1 (\$02)						
	b5		Será 1 após o tempo contado pelo timer 2 (\$03)						
	b4		Será 1 quando análise/síntese ADPCM completar						
	b3		Será 1 no fim da leitura/gravação/análise/síntese						
	b2-b1		Não usados (sempre “00”)						
	b0		Será 1 durante a análise/síntese ADPCM (se b7 de \$07 for 1)						

### 6.5.3 – Descrição dos registradores

Essa seção descreve detalhadamente os diversos registradores do Y8950 e seu funcionamento.

#### 6.5.3.1 – Registrador de teste

O registrador \$1H é o registrador de teste. É usado somente para teste do MSX-Audio. Normalmente seu valor é 0.



Bit 0: Se for “1”, a saída dos geradores de envoltória é colocada em “0” (volume total) tanto para o modulador quanto para a portadora. A envoltória continua em execução.

Bit 1: Se for “1”, mantém a fase do LFO em zero. Isso para, desativa e redefine o LFO do tremolo e do vibrato.

Bit 2: Se for “1”, mantém a fase da forma de onda em zero. As envoltórias continuam em execução, mas a saída é silenciada.

Bit 3: Se for “1”, atualiza os LFOs de tremolo e vibrato a cada amostra em vez de uma vez a cada várias amostras (o tremolo é 64 vezes mais rápido e vibrato é 1024 vezes).

Bit 6: ADPCM. Afeta o bit 0 (PCMBSY) do registrador de estado.

Bit 7: Se for “1”, coloca os LFOs nos valores iniciais (amplitude máxima, desvio de fase zero).

### 6.5.3.2 – Registradores de tempo

Existem dois registradores de tempo: \$02H, com resolução de 80  $\mu$ s e \$03H com resolução de 320  $\mu$ s. Eles são contadores de tempo de 8 bits e podem realizar operações de início, parada e sinalização. Se o sinalizador for setado, uma interrupção de hardware será enviada à CPU. Eles também podem controlar a modulação senoidal composta do gerador FM. Esses registradores podem ser carregados com qualquer valor entre 0 e 255. Quando a contagem excede o valor máximo do registrador, o sinalizador será setado e o valor inicial será recarregado. Então uma interrupção de hardware será gerada e/ou todas as 9 vozes FM

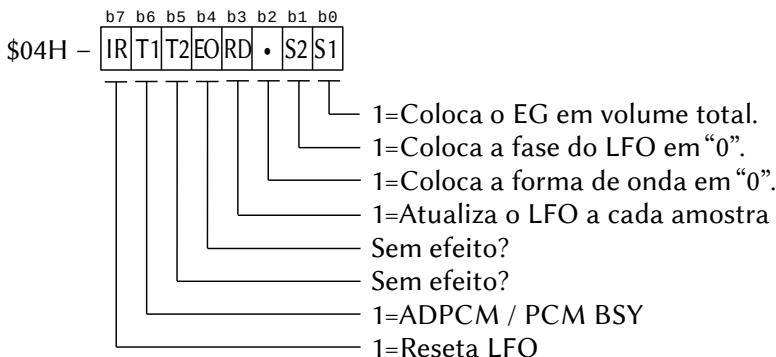
terão o Key-on ativado e, logo após, desativado (Key-off). O tempo de contagem em milissegundos de cada registrador, pode ser calculado pelas seguintes fórmulas:

$$T_0 \text{ (ms)} = (256 - (\$02H)) * 0,08$$

$$T_1 \text{ (ms)} = (256 - (\$03H)) * 0,32$$

#### 6.5.3.3 – Controle de flags (sinalizadores)

O registrador de flags (\$04H) é usado para controle de início, parada e interrupções dos registradores \$02H e \$03H, do ADPCM e da memória externa de áudio. Cada bit desse registrador habilita ou desabilita uma função, conforme descrito abaixo.



b0 (ST1): Esse bit controla as operações de início e parada de \$02H.

Quando for 0, \$02H estará desativado. Quando for 1, \$02H será carregado e iniciará a contagem.

b1 (ST2): Esse bit controla as operações de início e parada de \$03H, da mesma forma que b0 para \$02H.

b2: Não usado.

b3 (MASKBUF RDY): Esse bit controla o ADPCM e a memória de áudio. Quando for 0, a função estará desativada. Quando for 1, os dados de escrita e leitura estarão maskados durante a transferência de dados entre o processador e o ADPCM e a memória de áudio.

B4 (MASK EOS): Esse bit é usado para mascarar o bit b3, indicando o fim da leitura/escrita do ADPCM ou armazenamento externo, ou o fim da conversão AD.

B5 (MASK T2): Quando esse bit for setado em 1, b1 será setado em 0.

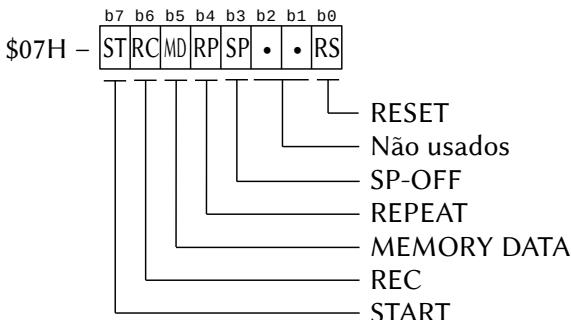
B6 (MASK T1): Esse bit é usado para mascarar b0.

B7 (IRQ RESET): Cada flag do MSX-Audio é setada em 1 quando o respectivo evento ocorre e IRQ fica no nível 0 (as interrupções ficam desabilitadas). Esse bit é usado para reabilitar as interrupções. Quando esse bit for 1, todas as flags (sinalizadores) serão colocadas em 0. Se somente algumas flags devem ser resetadas, basta setar em 1 o bit MASK correspondente. Após todas as flags serem resetadas, b7 é automaticamente resetado em 0.

#### 6.5.3.4 – Controle de teclado, memória e ADPCM

	b7	b6	b5	b4	b3	b2	b1	b0
\$05H –	b7	b6	b7	b6	b3	b2	b1	b0
\$06H –	b7	b6	b7	b6	b3	b2	b1	b0

Esses dois registradores são usados para acessar o teclado musical externo, sendo que \$05H é configurado como entrada e \$06H como saída. Assim, o sinal emitido por cada bit de \$06H pode ser lido por cada bit de \$05H, formando uma matriz de 8 x 8 para o teclado.



Esse registrador é usado para controlar o início e parada do ADPCM e também para setar o acesso à memória de áudio. Cada bit é

uma flag (sinalizador) que realiza uma operação diferente, conforme descrito a seguir.

b0: (RESET): Quando esse bit for setado em 1 durante a síntese de som pelo ADPCM usando a memória de áudio como fonte, o circuito de síntese ADPCM e a memória externa de áudio são levados ao estado inicial. Nesse caso, b4 (REPEAT) deve ser setado em 0. Esse bit pode ser usado quando da perda de controle do circuito ADPCM ou da memória externa.

b1: Não usado

b2: Não usado

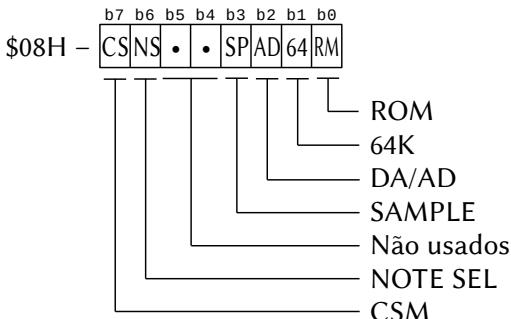
b3 (SP-OFF): Quando esse bit for setado em 1, o terminal de saída de som estará desativado. Esse bit deve ser usado para proteger o alto-falante durante a gravação pelo ADPCM.

b4 (REPEAT): Durante a síntese de som pelo ADPCM usando a memória de áudio, esse bit pode ser setado em 1 para habilitar a repetição de dados de uma mesma área (do endereço inicial ao endereço final).

b5 (MEMORY DATA): Esse bit deve ser setado em 1 quando a memória de áudio for ser acessada.

b6 (REC): Esse bit deve ser setado em 1 para a digitalização de sons pelo ADPCM ou para transferência de dados da CPU para a memória de áudio.

b7 (START): Esse bit deve ser setado em 1 para leitura ou gravação de dados pelo ADPCM. O procedimento difere de acordo com a localização dos dados (Main RAM ou memória de áudio). Se os dados estiverem na Main RAM, o ADPCM começará a leitura ou escrita através do registrador \$0FH. Se estiverem na memória de áudio, o ADPCM começará acessando o endereço inicial especificado. Conseqüentemente, devem ser carregados todos os registradores necessários antes de setar esse bit em 1.



Este registrador é usado junto com \$08H para controle do ADPCM e memória de audio.

b0: (ROM): Esse bit é usado para identificar o tipo de memória de áudio: 0=RAM; 1=ROM.

b1: (64K): Esse bit é usado para especificar a quantidade de memória de áudio disponível (0 = 256K DRAM; 1 = 64K DRAM). Quando esse bit for 1, a linha de endereço A8 é ignorada. Para ROM, esse bit deve ser 0.

b2: (DA/AD): Esse bit é usado em conjunto com b3 (SAMPLE) e controla a conversão DA/AD. Quando for 1, os dados especificados em \$15H e \$16H são enviados para a saída. Quando for 0, estará habilitada a conversão AD (b3=1) ou a saída de música (b3=0).

b3: (SAMPLE): Esse bit é usado para habilitar o timer para a conversão AD/DA. Quando for 1, a conversão AD se inicia; quando for 0, a conversão D/A é iniciada.

b4: Não usado.

b5: Não usado.

b6 (NOTE SEL): Esse bit é usado para especificar os pontos de separação de uma oitava para o teclado musical externo. Quando for 0, o ponto de separação é especificado pelos dois bits MSB de frequência. Quando for 1, o ponto de separação é especificado pelo bit MSB de frequência (registradores \$B0H a \$B8H), conforme a tabela abaixo.

**b6=1**

0	1	2	3	4	5	6	7	Oitava
0	1	2	3	4	5	6	7	Bloco de dados
1	1	1	1	1	1	1	1	F-num MSB
0	1	0	1	0	1	0	1	0
0	1	2	3	4	5	6	7	2º F-num
0	1	2	3	4	5	6	7	Nº sep. teclado

**b6=1**

0	1	2	3	4	5	6	7	Oitava
0	1	2	3	4	5	6	7	Bloco de dados
1	1	1	1	1	1	1	1	F-num MSB
*	*	*	*	*	*	*	*	2º F-num
0	1	2	3	4	5	6	7	Nº sep. teclado

b7 (CSM): Este bit deve ser setado em 1 para ativar o modo de modulação senoidal composta. Para isso, todas as vozes devem estar setadas em Key-off.

#### 6.5.3.5 – Endereços de acesso

\$09H – 

b7	b6	b5	b4	b3	b2	b1	b0
b7	b6	b7	b6	b3	b2	b1	b0

 Endereço inicial (low)

\$0AH – 

b15	b14	b13	b12	b11	b10	b9	b8
b15	b14	b13	b12	b11	b10	b9	b8

 Endereço inicial (high)

Esses registradores especificam os endereços inicial e final da memória de áudio a serem acessados. O valor desses registradores difere um pouco de acordo com o tipo de memória (ROM ou DRAM). A maneira pela qual eles especificam os endereços está ilustrada abaixo.

#### 64K RAM (Endereço inicial)

Banco	Endereço CAS	Endereço RAS
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0	a8 a7 a6 a5 a4 a3 a2 a1 a0
- \$0AH -	- \$09H -	
b7 b6 b5 0 b3 b2 b1 b0	b7 b6 b5 b4 0 b2 b1 b0	0 0 0 0 0

Obs: b4 de \$0AH e b3 de \$09H devem ser “0”.

## 256K RAM (Endereço inicial)

Banco	Endereço CAS								Endereço RAS							
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0								a8 a7 a6 a5 a4 a3 a2 a1 a0							
- \$0AH -	b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
*	* * * b4 b3 b2 b1 b0								0 0 0 0 0							

## 64K ROM (Endereço inicial)

Banco	Endereço CAS								Endereço RAS							
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0								a8 a7 a6 a5 a4 a3 a2 a1 a0							
- \$0AH -	b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
*	* * * b4 b3 b2 b1 b0								0 0 0 0 0							

Obs: Os bits b7, b6 e b5 de \$0AH devem ser iguais aos de \$0CH.

\$0BH - Endereço final (low)

\$0CH - Endereço final (high)

## 64K RAM (Endereço final)

Banco	Endereço CAS								Endereço RAS							
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0								a8 a7 a6 a5 a4 a3 a2 a1 a0							
- \$0CH -	b7 b6 b5 0 b3 b2 b1 b0								b7 b6 b5 b4 0 b2 b1 b0							
*	* * * b4 b3 b2 b1 b0								1 1 1 1 1							

Obs: b4 de \$0AH e b3 de \$09H devem ser “0”.

## 256K RAM (Endereço final)

Banco	Endereço CAS								Endereço RAS							
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0								a8 a7 a6 a5 a4 a3 a2 a1 a0							
- \$0CH -	b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
*	* * * b4 b3 b2 b1 b0								1 1 1 1 1							

## 64K ROM (Endereço final)

Banco	Endereço CAS								Endereço RAS							
b2 b1 b0	a8 a7 a6 a5 a4 a3 a2 a1 a0								a8 a7 a6 a5 a4 a3 a2 a1 a0							
- \$0CH -	b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
*	* * * b4 b3 b2 b1 b0								1 1 1 1 1							

Obs: Os bits b7, b6 e b5 de \$0CH devem ser iguais aos de \$0AH.

### 6.5.3.6 – Acesso ao ADPCM e I/O 4 bits

Os registradores \$0DH e \$0EH especificam a taxa de amostragem (sampling rate) para a conversão AD e DA do ADPCM. A taxa máxima é de 16 KHz e a mínima de 1,8 KHz. O valor a ser carregado nos registradores pode ser obtido usando a fórmula abaixo.

$$\text{Taxa (KHz)} = 3580 / \text{NPRE} \quad (@ \varnothing m = 3,58 \text{ MHz})$$

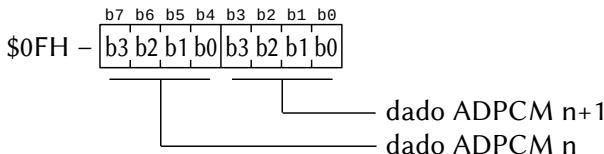
O valor NPRE é o valor contido em 11 bits dos registradores \$0DH e \$0EH, conforme ilustrado abaixo.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
\$0DH	b <sub>7</sub>	b <sub>6</sub>	b <sub>7</sub>	b <sub>6</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
\$0EH	0	0	0	0	0	b <sub>10</sub>	b <sub>9</sub>	b <sub>8</sub>

16 KHz → \$0DH = E0H e \$0EH = 00H (freq max)

1,8 KHz → \$0DH = C4H e \$0EH = 07H (freq min)

O registrador \$0FH é usado para intercâmbio dos dados do ADPCM com a CPU. Ele também é usado como buffer quando a memória de áudio é usada pela CPU. Esse registrador contém normalmente 2 dados, visto que os dados do ADPCM são compactados em 4 bits cada. Os 4 bits mais altos contêm o dado n e os 4 bits mais baixos contêm o dado n+1.



Os registradores \$10H e \$11H especificam o fator para a interpolação linear com a frequência de 50 KHz do gerador FM durante os intervalos da síntese de sons pelo ADPCM. Esse fator também é usado como taxa de amostragem para a síntese e, nesse caso, os registradores \$0DH e \$0EH são ignorados. A fórmula para calcular o valor desses registradores é a seguinte:

$$\Delta n = k * 2^{16}, \quad k = \left( \frac{3.58 \text{ MHz}}{50 \text{ KHz}} \right) / \left( \frac{3.58 \text{ MHz}}{\text{fsample}} \right), \quad (@ \varnothing m = 3,58 \text{ MHz})$$

$$\text{VOICE}_{n,i} = \text{VOICE}_n + (\text{Noff}_n + i_n * k) * (\text{VOICE}_{n+1} - \text{VOICE}_n)$$

$$\begin{cases} 0 \leq \text{Noff}_n + i_n * k < 1 \\ \text{Noff}_n < k, \quad \text{Noff}_n = \text{Noff}_{n-1} + i_{n-1} * k + k - 1 \end{cases}$$

(i'n-1 é o valor máximo da (n-1)<sup>a</sup> amostragem)

Mas pode ser usada a seguinte fórmula simplificada:

$$\Delta n = 1310,72 * \text{Taxa de amostragem (Khz)}$$

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
\$10H -	b <sub>7</sub>	b <sub>6</sub>	b <sub>7</sub>	b <sub>6</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	$\Delta n$ (low)
\$11H -	b <sub>15</sub>	b <sub>14</sub>	b <sub>13</sub>	b <sub>12</sub>	b <sub>11</sub>	b <sub>10</sub>	b <sub>9</sub>	b <sub>8</sub>	$\Delta n$ (high)

$$16 \text{ KHz} \rightarrow \$10H = ECH \text{ e } \$11H = 51H$$

$$1,8 \text{ KHz} \rightarrow \$10H = 37H \text{ e } \$11H = 09H$$

O registrador \$12H é o controle de volume de saída do ADPCM. Ele tem 256 níveis, sendo que o volume máximo é obtido quando o registrador contiver o valor 255. O valor 0 indica volume nulo (não há saída de som). O valor desse registrador não afeta a saída de som do gerador FM.

$$\text{AUDIO OUT} = \text{VOICE}_n * \text{EG}$$

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
\$12H -									Controle EG

Os registradores \$15H a \$17H são usados para especificar os dados digitais para a conversão DA. Nos três registradores, apenas 13 bits são usados. Os valores iniciais devem ser escritos nesses registradores antes de setar o bit 2 do registrador \$08H. Os valores desses registradores podem ser calculados pelas seguintes fórmulas:

$$\text{Saída} = \frac{V_{cc}}{2} + \frac{V_{cc}}{4} * (-1 + f_9 + f_8 * 2^{-1} + \dots + f_1 * 2^{-8} + f_0 * 2^{-9} + 2^{10}) * 2^{-E}$$

$$E = S_2 * 2^2 + S_1 * 2^1 + S_0 * 2^0 \quad @ S_0 + S_1 + S_2 > 0$$

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
\$15H -	f <sub>9</sub>	f <sub>8</sub>	f <sub>7</sub>	f <sub>6</sub>	f <sub>5</sub>	f <sub>4</sub>	f <sub>3</sub>	f <sub>2</sub>
\$16H -	f <sub>1</sub>	f <sub>0</sub>	•	•	•	•	•	•
\$17H -	•	•	•	•	•	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>

Os registradores \$18H e \$19H são registradores de 4 bits usados para controlar as portas de I/O de uso geral do MSX-Audio. \$18H especifica se é entrada ou saída; deve ser setado em 1 para saída e em 0 para entrada. \$19H é usado para transferir os dados pela porta I/O de 4 bits.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
\$18H -	•	•	•	•	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Controle de I/O
\$19H -	•	•	•	•	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Dados de I/O

O registrador \$1AH é usado para armazenar os dados processados pela conversão A/D. O código PCM é expressado em complemento de dois, ou seja, o valor 127 corresponde ao nível 0.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
\$1AH -	b <sub>7</sub>	b <sub>6</sub>	b <sub>7</sub>	b <sub>6</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Dados PCM

### 6.5.3.7 – Acesso ao gerador FM

O MSX-Audio pode gerar até 9 vozes de som FM ou 6 vozes mais 5 peças de bateria, como o OPLL, mas todas as 9 vozes devem ser definidas pelo usuário.

Cada voz utiliza dois operadores (designados como onda moduladora e onda portadora) para a geração de sons, resultando num total de 18 operadores. A tabela abaixo mostra a relação entre operadores, vozes e registradores.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	*1
1	2	3	1	2	3	4	5	6	4	5	6	7	8	9	7	8	9	*2
20	21	22	23	24	25	28	29	2A	2B	2C	2D	30	31	32	33	34	35	*3
A0	A1	A2	A0	A1	A2	A3	A4	A5	A3	A4	A5	A6	A7	A8	A6	A7	A8	*4

\*1 - número do operador utilizado

\*2 - número da voz gerada

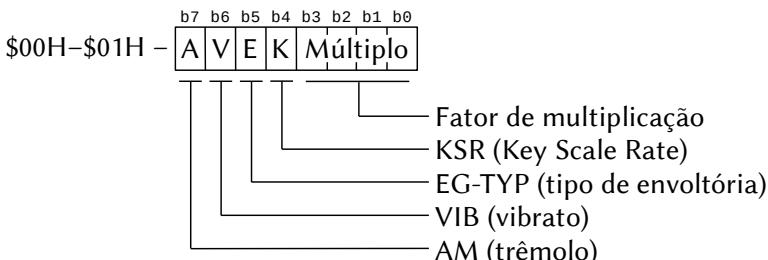
\*3 - registrador correspondente (no exemplo, \$20H a \$35H)

\*4 - registrador correspondente (no exemplo, \$A0H a \$A8H)

Os registradores usados para a geração de sons FM estão descritos detalhadamente abaixo.

#### • AM/VIB/EG-TYP/KSR/MÚLTIPLO

Esses registradores são usados para especificar a forma da envoltória e os fatores de multiplicação para as ondas portadora e moduladora.



#### MÚLTIPLO (b0~b3)

Esses bits especificam os fatores de multiplicação usados para converter as ondas moduladora e portadora. Os fatores de multiplicação podem ser vistos na tabela abaixo.

Valor do registro: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Fator de multiplicação:  $\frac{1}{2}$  1 2 3 4 5 6 7 8 9 10 10 12 12 15 15

Por exemplo, se F-num for wf, o fator para a onda portadora for 1 e para a onda moduladora for 7, F(t) será calculado pela seguinte fórmula:

$$F(t) = E \sin (\omega ft + 1 \sin (7\omega ft))$$

KSR (b4)

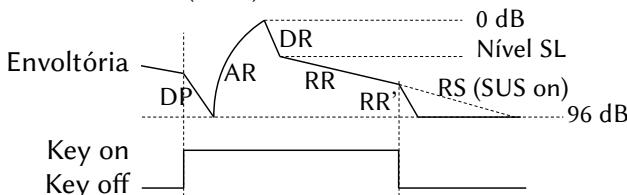
KSR significa “key scale rate”. Ele especifica as razões de “attack” e “decay”. A “key scale” é usada para fazer com que o som gerado pelo FM se aproxime do dos instrumentos musicais acústicos. O significado desse bit está ilustrado na tabela abaixo:

Key Scale		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fatores	b4=0	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
	b4=1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

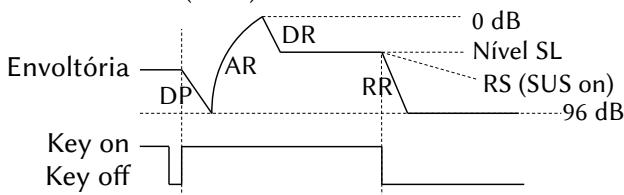
### EG-TYP (b5)

Esse bit seleciona entre tom constante ou tom percussivo. Se for 0, o tom será percussivo e se for 1 o tom será constante. A forma de onda gerada varia conforme a ilustração abaixo.

#### Tom Percussivo (b5=0)



#### Tom Constante (b5=1)



AR = Attack Rate

DR = Decay Rate

SR = Sustain Level

RR = Release Rate

### VIB (b6)

Esse bit liga ou desliga o vibrato. Se for 1, o vibrato estará ligado e se for 0 estará desligado. A frequência do vibrato é de 6,4 Hz. O grau do vibrato é setado pelo bit VIB-DEPTH do registrador \$BDH.

### AM (b7)

Esse bit liga ou desliga a modulação de amplitude (tremolo). Se for 1, a modulação de amplitude estará ligada e se for 0 estará desligada. A frequência do tremolo é de 3,7 Hz. O grau do tremolo é setado pelo bit AM-DEPTH do registrador \$BDH.

- KSL / NÍVEL TOTAL

	b7	b6	b5	b4	b3	b2	b1	b0
\$40H~\$55H -	KSL	Nível total						

### NÍVEL TOTAL (b0~b5)

Os seis bits do nível total não usados para controlar o grau de modulação da envoltória (nível da envoltória). A tabela abaixo mostra todos os graus de modulação possíveis.

b5	b4	b3	b2	b1	b0
24dB	12dB	6dB	3dB	1,5dB	0,75dB

Para obter o valor correto, deve-se somar os graus quando o bit respectivo for 1. Assim, a resolução máxima de “decay” é de 0,75 dB e o nível de saída pode ser reduzido até 47,25 dB.

### KSL (b6~b7)

Esses bits controlam o nível de saída através da atenuação progressiva do som (key scale level). O nível de atenuação progressiva do som pela frequência pode variar de 0 db/oitava até 6 dB/oitava.

b6	b7	Atenuação
0	0	0 dB/oitava
0	1	1,5 dB/oitava
1	0	3 dB/oitava
1	1	6 dB/oitava

A tabela abaixo descreve os fatores de decay em dB/oitava de acordo com F-number.

F-num ↓	Oitava							
	0	1	2	3	4	5	6	7
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000	0.000	3.000	6.000	9.000
2	0.000	0.000	0.000	0.000	3.000	6.000	9.000	12.000
3	0.000	0.000	0.000	1.875	4.875	7.875	10.875	13.875
4	0.000	0.000	0.000	3.000	6.000	9.000	12.000	15.000
5	0.000	0.000	1.125	4.125	7.125	10.125	13.125	16.125
6	0.000	0.000	1.875	4.875	7.875	10.875	13.875	16.875
7	0.000	0.000	2.625	5.625	8.625	11.625	14.625	17.625

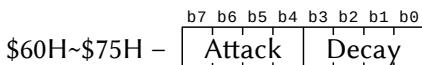
8	0.000	0.000	3.000	6.000	9.000	12.000	15.000	18.000
9	0.000	0.750	3.750	6.750	9.750	12.750	15.750	18.750
10	0.000	1.125	4.125	7.125	10.125	13.125	16.125	19.125
11	0.000	1.500	4.500	7.500	10.500	13.500	16.500	19.500
12	0.000	1.875	4.875	7.875	10.875	13.875	16.875	19.875
13	0.000	2.250	5.250	8.250	11.250	14.250	17.250	20.250
14	0.000	2.625	5.625	8.625	11.625	14.625	17.625	20.625
15	0.000	3.000	6.000	9.000	12.000	15.000	18.000	21.000

Obs: F-num indica os 4 bits mais altos.

Multiplicar cada valor por 1/2 para decay de 1,5 dB/oitava.

Multiplicar cada valor por 2 para decay de 6 dB/oitava.

#### • RELAÇÃO ATTACK/DECAY



As razões de “attack” e “decay” são definidas pelos registradores \$60H a \$75H, sendo que os bits b0~b3 definem o nível de “decay” e os bits b4~b7 o de attack. Quanto maior o valor, menor o tempo de “attack” e “decay”. A variação de tempo obedece, aproximadamente, a uma progressão geométrica, e seus valores estão listados na tabela abaixo.

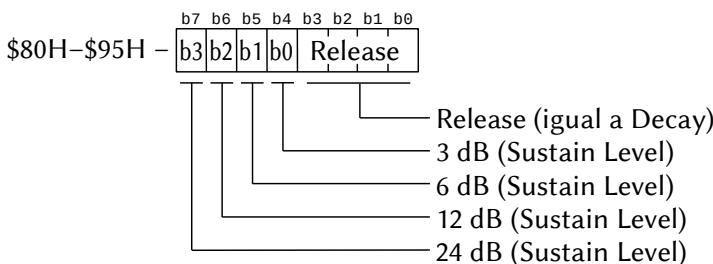
RATE		EG decay time (mS)		EG attack time (mS)	
RM	RL	0dB~96dB	10% ~ 90%	0dB~96dB	10% ~ 90%
15	3	2.40	0.51	0.00	0.00
15	2	2.40	0.51	0.00	0.00
15	1	2.40	0.51	0.00	0.00
15	0	2.40	0.51	0.00	0.00
14	3	2.74	0.58	0.20	0.11
14	2	3.20	0.63	0.24	0.11
14	1	3.84	0.81	0.30	0.14
14	0	4.80	1.01	0.38	0.19
13	3	5.48	1.15	0.42	0.22
13	2	6.40	1.55	0.46	0.26
13	1	7.68	1.62	0.56	0.31
13	0	9.60	2.02	0.70	0.37
12	3	10.96	2.32	0.80	0.43

12	2	12.80	2.68	0.92	0.49
12	1	15.36	3.22	1.12	0.61
12	0	19.20	4.02	1.40	0.73
11	3	21.92	4.62	1.56	0.85
11	2	25.56	5.38	1.84	0.97
11	1	30.68	6.42	2.20	1.13
11	0	38.36	8.02	2.76	1.45
10	3	43.84	9.24	3.12	1.70
10	2	51.12	10.76	3.68	1.94
10	1	61.36	12.84	4.40	2.26
10	0	76.72	16.04	5.52	2.90
9	3	87.68	18.48	6024	3.39
9	2	102.24	21.52	7.36	5.87
9	1	122.72	25.68	8.80	4.51
9	0	153.44	32.08	11.04	5.79
8	3	175.36	36.96	12.48	6.78
8	2	204.48	43.04	14.72	7.74
8	1	245.44	51.36	17.60	9.02
8	0	306.88	64.16	22.08	11.58
7	3	350.72	73.92	24.96	13.57
7	2	408.96	86.08	29.44	15.49
7	1	490.88	102.72	35.20	18.05
7	0	613.76	128.32	44.16	22.17
6	3	701.44	147.84	49.92	27.14
6	2	817.92	172.16	58.88	30.98
6	1	981.76	205.44	70.40	36.10
6	0	1227.52	256.64	88.32	45.74
5	3	1402.88	295.68	99.84	554.27
5	2	1635.84	344.32	117.76	61.95
5	1	1963.52	410.88	140.80	72.19
5	0	2455.04	513.28	176.64	92.67
4	3	2805.76	591.36	199.68	108.54
4	2	3271.68	688.64	235.52	123.90
4	1	3927.08	821.76	281.60	144.38
4	0	4910.08	1026.56	353.28	185.34

3	3	5611.52	1182.72	399.36	217.03
3	2	6543.36	1377.28	471.04	247.81
3	1	7854.08	1643.52	563.20	288.77
3	0	9820.16	2053.12	706.56	370.69
2	3	11223.04	2365.44	798.72	434.18
2	2	13086.72	2754.56	942.08	495.62
2	1	18708.16	3287.24	1126.40	577.54
2	0	19640.32	4106.24	1413.12	741.38
1	3	22446.08	4730.88	1597.44	868.35
1	2	26173.44	5509.12	1884.16	991.23
1	1	31416.32	6574.08	2252.80	1155.07
1	0	39280.64	8212.48	2826.24	1482.75

#### • SUSTAIN LEVEL / RELEASE RATE

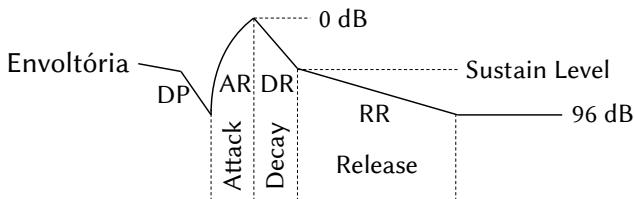
“Sustain level” especifica o nível em que o som ficará depois do “decay rate”. Para o tom percussivo, especifica o ponto de troca do modo “decay” para o modo “release”. Quanto maior o valor desse registrador, menor será o nível de “sustain”.



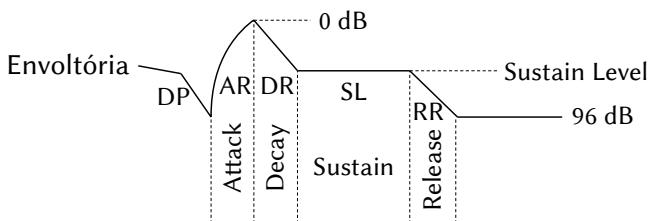
Para obter o valor correto de “sustain”, é preciso somar os valores quando o bit respectivo for “1”.

“Release rate”, para tom constante, especifica a razão de “decay” após a “key off”. Para tom percussivo, especifica a razão de “decay” após o “sustain level”. Quanto maior o valor do registrador, menor será a duração da “release rate”. O valor “release” é especificado com os mesmos valores usados para o “decay”, descritos na tabela acima.

Abaixo, há uma ilustração dos valores de “attack”, “decay”, “sustain level” e “release rate” na forma de onda da envoltória.

Tom Percussivo ( $b_5=0$ )

## Tom Constante



## • OITAVA/FREQUÊNCIA

São nove grupos de dois registradores cada, sendo que os registradores \$A0H e \$B0H controlam a primeira voz, \$A1H e \$B1H controlam a segunda e assim por diante.

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
\$A0H~\$A8H -	f	f	f	f	f	f	f	f	Frequência (b <sub>7</sub> ~b <sub>0</sub> )
\$B0H~\$B8H -	.	.	k	o	o	o	f	f	Frequência (b <sub>9</sub> ~b <sub>8</sub> ) Oitava (0 ~ 7) Key on/off

## FREQUÊNCIA (\$AxH e bits 1 e 0 de \$BxH)

Esses 10 bits definem uma escala de frequências para cada oitava. Na tabela abaixo, estão especificados os valores dos registradores para a quarta oitava, de um total de 8 oitavas, com a nota LÁ central de 440 Hz.

Cifrado	Frequência	Decimal	\$2xH,b1~b0	\$1xH
Dó C#	277.2 Hz	363	0 1	0 1 1 0 1 0 1 1
Ré D	293.7 Hz	385	0 1	1 0 0 0 0 0 0 1
	D#	311.1 Hz	408	1 0 0 1 1 0 0 0
Mi E	329.6 Hz	432	0 1	1 0 1 1 0 0 0 0
Fá F	349.2 Hz	458	0 1	1 1 0 0 1 0 1 0
	F#	370.0 Hz	485	1 1 1 0 0 1 0 1
Sol G	392.0 Hz	514	1 0	0 0 0 0 0 0 1 0
	G#	415.3 Hz	544	0 0 1 0 0 0 0 0
Lá A	440.0 Hz	577	1 0	0 1 0 0 0 0 0 1
	A#	466.2 Hz	611	0 1 1 0 0 0 1 1
Si B	493.9 Hz	647	1 0	1 0 0 0 0 1 1 1
Dó C	523.3 Hz	686	1 0	1 0 1 0 1 1 1 0

Os valores das frequências guardam entre si uma relação geométrica igual à 12ª raiz de 2, que vale 1,0594630943592. Pode-se usar esse número para alterar os valores dos registradores a fim de aumentar ou diminuir a frequência gerada dentro da escala musical. Os valores dos registradores também guardam entre si a mesma relação.

Os valores especificados pelos bits f0 ~ f9 são chamados de “F-number” e seus valores podem ser calculados pelas seguintes fórmulas:

$$F\text{-num} = (fmus * 2^{19} / fsam) / 2^{b-1}$$

onde:

F-num = dado F-number

fmus = Frequência desejada

fsam = Frequência de amostragem (50 KHz)

b = Bloco (oitava)

A frequência de amostragem corresponde à frequência do relógio (3,58 MHz em um MSX padrão) dividida por 72, o que resulta mais precisamente em 49,722 KHz.

OITAVA (\$BxH, b4~b2)

Esses três bits definem a oitava. Podem ser definidas até 8 oitavas, de 000 a 111, sendo que a quarta oitava é a 011.

### KEY (\$BxH, b5)

Esse bit deve ser setado em 1 para que o som de cada uma das nove vozes seja habilitado. Quando for 0, o som da voz respectiva estará desligado (key off).

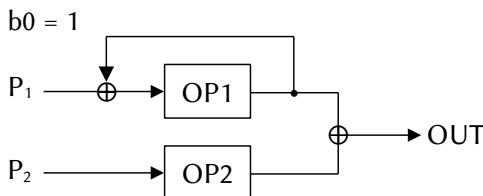
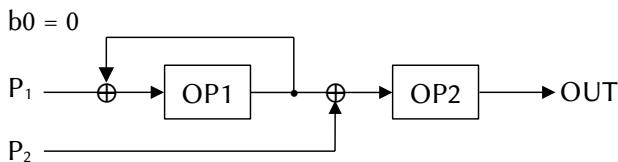
- REALIMENTAÇÃO/CONEXÃO

$\$C0H \sim \$C8H$  - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
•	•	•	•	R	R	R	C

### CONEXÃO (\$CxH, b0)

Esse bit é usado para especificar o tipo de conexão entre os dois operadores FM (onda moduladora e onda portadora). Se for 0, os operadores estarão no modo FM. Se for 1, estarão no modo de modulação seinaloidal composta (em paralelo). Esses modos estão ilustrados abaixo.

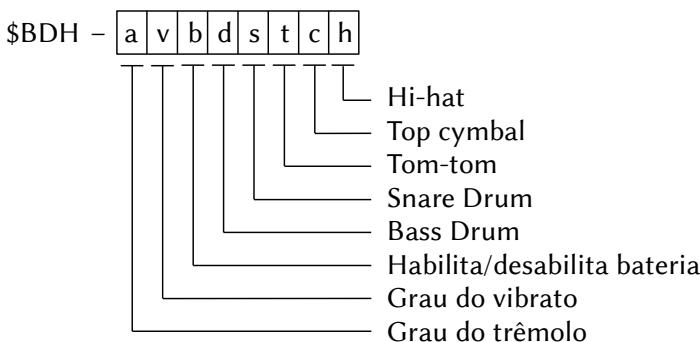


### REALIMENTAÇÃO (\$CxH, b1~b3)

Esses bits especificam o índice de realimentação (porção do sinal de saída que é reinjetado na entrada) para a onda moduladora. Quanto maior o valor do registrador, maior o fator de realimentação (feedback). Os fatores estão mostrados na tabela abaixo.

Valor do registrador:	000	001	010	011	100	101	110	111
Valor de realimentação:	0	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	$\pi$	$2\pi$	$4\pi$

- BATERIA / AM.VIB-DEPTH



#### MODO DE BATERIA (b0~b5)

Para ativar o modo de bateria do MSX-Audio, basta setar em 1 o bit 5 de \$BDH. Os bits b0 a b4 ativam (1) ou desativam (0) cada uma das cinco peças de bateria disponíveis. No modo bateria, apenas as seis primeiras vozes do MSX-Audio ficam disponíveis para o programador.

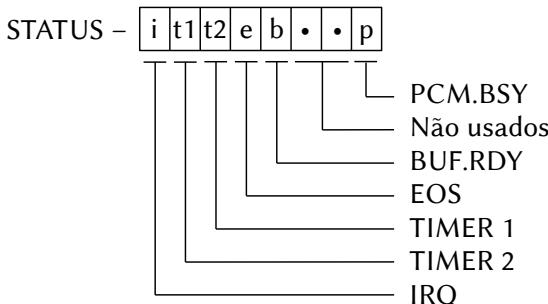
#### VIB-DEPTH (b6)

Esse bit é usado para selecionar o grau do vibrato. Se for 0, o grau será de 7% e se for 1, de 14%.

#### AM-DEPTH (b7)

Esse bit é usado para selecionar o grau da modulação de amplitude (tremolo). Se for 0, o grau será de 1 dB e se for 1, de 4,8 dB.

#### 6.5.3.8 – O registrador de estado



O MSX-Audio possui um registrador de estado com flags para controlar dois timers e a memória de áudio, usadas durante a síntese ou análise de sons pelo ADPCM. É um registrador só de leitura.

b0 (PCM.BSY): Durante a análise ou síntese de sons pelo ADPCM, esse bit será setado em 1 se o bit b7 de \$07H. Não é gerado sinal de interrupção.

b1: Não usado

b2: Não usado

b3 (BUF.RDY): Esse bit será setado em 1 nos seguintes casos:

- Fim da análise de sons pelo ADPCM (\$07H, b5=0).
- Fim da síntese de sons pelo ADPCM (\$07H, b5=0).
- Fim da escrita na memória de áudio.
- Fim da leitura da memória de áudio.

b4 (EOS): Esse bit será setado em 1 quando a análise ou síntese de sons pelo ADPCM for completada ou quando houver lapso de tempo durante a conversão AD/DA.

b5 (Timer 2): Esse bit é setado em 1 após o lapso de tempo gerado pelo timer 2.

b6 (Timer 1): Igual a b5, mas setado pelo timer 1.

b7 (IRQ): Este bit será setado em 1 quando um ou mais dos bits b3 a b6 for ou forem 1.

#### **6.5.4 – Acesso à memória de áudio E ADPCM**

Os protocolos recomendados para acesso à memória de áudio e ao ADPCM estão descritos em seguida.

##### **6.5.4.1 – Análise do som (MSX-Audio → CPU)**

Reg.	Dado	R/W	Comentários
\$04H	00H	W	• Inicialização Todas as flags são habilitadas
\$04H	80H	W	Todas as flags são resetadas
\$07H	C8H	W	ADPCM é habilitado e saída de som é desligada
\$08H	00H	W	“Sampling rate” = 8 KHz (NPRE = 450)
\$0DH	C2H	W	

\$0EH	01H	W	• Inicia a análise
\$0FH	80H	R	Inicia com a leitura do “dummy”
\$0FH	48H	R	• Análise
(\$04H	00H	W)	Quando BUF.RDY for 1, \$0FH é lido, o dado é armazenado e a flag resetada. Quando BUF.RDY for 0, espera.
\$07H		W	• Fim da análise
\$07H		W	A análise pelo ADPCM foi completada O registrador \$07H é resetado

#### 6.5.4.2 – Síntese de som (CPU → MSX-Audio)

Reg.	Dado	R/W	Comentários
			• Inicialização
\$04H	00H	W	Todas as flags são habilitadas
\$04H	80H	W	Todas as flags são resetadas
\$07H	80H	W	Síntese de som pelo ADPCM é habilitada
\$08H	00H	W	
\$10H	F6H	W	“Sampling rate” = 8 KHz ( $\Delta n = 10\,486$ )
\$11H	28H	W	
\$12H	xxH	W	Especificar o volume de saída
			• Início da síntese
\$0FH	yyH	W	Escreve dado para o ADPCM em \$0FH
			• Síntese
\$0FH	zzH	W	Quando BUF.RDY for 1, o dado de síntese é escrito em \$0FH e a flag resetada. Quando flag for 0, esperar.
(\$04H	80H	W)	• Final da síntese
\$07H	80H		A síntese pelo ADPCM foi completada.

#### 6.5.4.3 – Análise do som (MSX-Audio → Memória de Áudio)

Reg.	Dado	R/W	Comentários
			• Inicialização
\$04H	08H	W	Somente a flag BUF.RDY é mascarada
\$04H	80H	W	Todas as flags são resetadas
\$07H	68H	W	A análise pelo ADPCM é habilitada
\$08H	02H/00H	W	Especificar o tipo de memória de áudio
\$09H	xxH	W	Endereço inicial da memória de áudio

\$0AH	xxH	W	Endereço final da memória de áudio
\$0BH	yyH	W	“Sampling rate” = 16 KHz (NPRE = 225)
\$0CH	yyH	W	• Início da análise
\$0DH	E1H	W	Iniciar quando o bit b0 de \$07H for 1.
\$0EH	00H	W	• Análise
\$07H	E8H	W	A flag EOS fica setada em 1 até o final da análise
\$07H	68H	W	• Fim da análise
\$07H	00H	W	A análise pelo ADPCM foi completada O endereço \$07H é resetado

#### 6.5.4.4 – Síntese de som (Memória de Áudio → MSX-Audio)

Reg.	Dado	R/W	Comentários
\$04H	08H	W	• Inicialização Somente a flag BUF.RDY é mascarada
\$04H	80H	W	Todas as flags são resetadas
\$07H	20H/30H	W	A síntese pelo ADPCM é habilitada
\$08H	00H-02H	W	Especificar o tipo de memória de áudio
\$09H	xxH	W	Endereço inicial da memória de áudio
\$0AH	xxH	W	
\$0BH	yyH	W	Endereço final da memória de áudio
\$0CH	yyH	W	
\$10H	ECH	W	“Sampling rate” = 16 KHz ( $\Delta n = 20\,992$ )
\$11H	51H	W	
\$12H	xxH	W	Especificar o volume de saída • Início da síntese
\$07H	A0H/B0H	W	Iniciar quando b7 de \$07H for 1 • Síntese A flag EOS fica setada em 1 até o final da síntese
(\$07H)	A0H	W)	(Modo de repetição é estabelecido)
(\$07H)	A1H	W)	(Força interrupção da síntese) • Fim da síntese
\$07H	20H	W	A síntese pelo ADPCM foi completada
\$07H	00H	W	O registrador \$07H é resetado

#### 6.5.4.5 – Escrita na RAM de áudio (CPU → Memória de Áudio)

Reg.	Dado	R/W	Comentários
\$04H	00H	W	• Inicialização Todas as flags são habilitadas
\$04H	80H	W	Todas as flags são resetadas
\$07H	60H	W	Modo de escrita na memória é estabelecido
\$08H	00H/02H	W	Especifica o tipo de RAM
\$09H	xxH	W	Endereço inicial da memória de áudio
\$0AH	xxH	W	
\$0BH	yyH	W	Endereço final da memória de áudio
\$0CH	yyH	W	
\$0FH	zzH	W	• Escrita na memória Byte de dados a ser escrito
(\$04H)	80H	W	(Quando BUF.RDY for 1, o dado é escrito; quando for 0, esperar. Quando o fim da memória for atingido, a flag EOS será 1)
			• Reset
\$07H	00H	W	O registrador \$07H é resetado

#### 6.5.4.6 – Leitura da RAM/ROM de áudio (Mem. de Áudio → CPU)

Reg.	Dado	R/W	Comentários
\$04H	00H	W	• Inicialização Todas as flags são habilitadas
\$04H	80H	W	Todas as flags são resetadas
\$07H	20H	W	Modo de leitura da memória é estabelecido
\$08H	00H-02H	W	Especifica o tipo de memória
\$09H	xxH	W	Endereço inicial da memória de áudio
\$0AH	xxH	W	
\$0BH	yyH	W	Endereço final da memória de áudio
\$0CH	yyH	W	
\$0FH	zzH	R	• Leitura da memória Iniciar após ler o “dummy” duas vezes
\$0FH		R	(Necessário para checar a flag)
\$0FH		R	Leitura do byte de dados

\$04H	80H	W	Quando BUF.RDY for 1, o dado é lido; quando for 0, esperar. Quando o fim da memória for atingido, a flag EOS será 1) • Reset
\$07H	00H	W	O registrador \$07H é resetado

## 6.5.5 – Acesso ao MSX-AUDIO

O acesso ao MSX-Audio pode ser feito diretamente ou através da MBIOS (Music BIOS).

### 6.5.5.1 – Acesso Direto

O acesso direto ao MSX-Audio é feito através de duas portas de I/O da CPU, a C0H e a C1H. A porta C0H seleciona os registradores ou lê o registrador de estado e a porta C1H lê ou escreve os dados nos outros registradores. Entretanto, tal qual o OPLL, o MSX-Audio é lento. Deve haver uma pausa entre um acesso e outro. O tempo de cada pausa está descrito na tabela abaixo.

Seleção de registradores (C0H)	3,4 µs	12 ciclos T (3,58 MHz)
Acesso aos regs \$00H a \$1AH (C1H)	3,4 µs	12 ciclos T (3,58 MHz)
Acesso aos regs \$20H a \$C8H (C1H)	23,5 µs	84 ciclos T (3,58 MHz)

Recomenda-se o uso de pausas tipo “EX (SP),HL” ou “NOP” até que o MSX-Audio esteja pronto para novo acesso.

Primeiramente, deve-se selecionar o registrador a ser escrito através da porta 7CH. Após a escrita, deve-se dar uma pausa de, no mínimo, 12 ciclos T no caso de um MSX padrão (Z80 a 3,58 MHz). A instrução “OUT (07CH),A” demora 11 ciclos T para ser processada; portanto é necessária, ao menos, mais 1 ciclo T. Pode ser usada uma instrução NOP (que demora 4 ciclos T para ser processada) para isso, conforme ilustração abaixo:

```
LD   A,REG      ;número do registrador em A
OUT (07CH),A    ;seleciona o registrador
NOP            ;pausa
```

Logo em seguida, escreve-se ou lê-se o dado no registrador selecionado através da porta C1H, observando as pausas. Para leitura do registrador de estado não há necessidade de especificar endereços. Para

escrever um byte de dados nos registradores \$00H a \$1AH, deve-se proceder da forma ilustrada abaixo.

```
LD A,REG ;nº do registrador (00H a 1AH)
OUT (0C0H),A ;seleciona o registrador
NOP ;pausa
LD A,DADO ;byte de dados a ser escrito
OUT (0C1H),A ;escreve o dado no registrador
NOP ;pausa
```

Para escrever nos registradores \$20H a \$C8H, que são bem mais lentos, é necessária uma pausa maior, de 84 ciclos T para um clock de 3,58 MHz. Para isso, podem ser usadas 4 instruções “EX (SP),HL” (que demoram 19 ciclos T cada para serem processadas), resultando numa pausa de 76 ciclos T que, somados aos 11 ciclos da instrução OUT, resultam em 87 ciclos T. Então, o MSX-Audio estará pronto para receber novo dado.

```
LD A,REG ;nº do registrador (00H a 1AH)
OUT (0C0H),A ;seleciona o registrador
NOP ;pausa
LD A,DADO ;byte de dados a ser escrito
OUT (0C1H),A ;escreve o dado no registrador
EX (SP),HL ;pausa
EX (SP),HL ;pausa
EX (SP),HL ;pausa
EX (SP),HL ;pausa
```

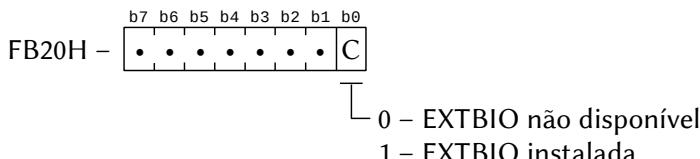
Ao ser usada em pausas, a instrução EX (SP),HL deve sempre vir em duplas, para evitar que o conteúdo da pilha seja alterado.

Os registradores \$00H a \$1AH e o registrador de estado podem ser lidos. Nesse, deve-se proceder como ilustrado abaixo. Aqui deve haver uma pausa de 12 ciclos T, conforme ilustrado abaixo.

```
LD A,REG ;número do registrador (00H a 1AH)
OUT (0C0H),A ;seleciona o registrador
NOP ;pausa
IN A,(0C1H) ;lê o valor do reg. especificado
NOP ;pausa
IN A,(0C0H) ;lê o valor do reg. de estado
NOP ;pausa
```

### 6.5.5.2 – Acesso através da Music BIOS

O acesso através da MBIOS, ou Music BIOS, é feito através da rotina EXTBIO (FFCAH) residente na área de trabalho, usada para expansão da BIOS residente em ROM. Inicialmente é necessário consultar a variável de sistema HOKVLD (FB20H, 1) para confirmar se a BIOS estendida está instalada. O bit 0 deste byte indica a presença de uma BIOS estendida. Se for 0, não há BIOS estendida. Se for 1 há, pelo menos, uma BIOS que pode ser chamada no endereço 0FFCAH (EXTBIO).



O próximo passo é verificar se existem cartuchos MSX-Audio conectados. Isso é feito através da própria rotina EXTBIO:

**EXTBIO (FFCAH/Work Area)**

**Função:** Acessa funções estendidas da BIOS.

**Entrada:** A = 00H.

D = 00H – Comando interno.

E = 00H – Examina os dispositivos presentes no sistema.

B – ID do slot onde será colocada a tabela.

HL – Endereço da tabela.

**Saída:** B – ID do slot da tabela.

HL – Endereço do próximo byte depois da tabela.

CY = 1 se não houver dispositivos.

**Registradores:** Todos.

Em seguida a tabela apontada pelos registradores B e HL deve ser examinada para verificar a presença do identificador do dispositivo MSX Audio, que é 0AH. A tabela inicia no endereço apontado por HL na entrada da rotina e termina um byte antes do endereço apontado por HL no retorno. Se houver o ID, devemos obter quantos cartuchos MSX Audio estão disponíveis, da seguinte forma:

**EXTBIO (FFCAH/Work Area)**

**Função:** Acessa funções estendidas da BIOS

**Entrada:** A = 00H.  
 D = 0AH – Dispositivo de manipulação do MSX-Audio.  
 E = 01H – Retorna quantos cartuchos MSX-Audio estão conectados ao MSX (máximo de 2).

**Saída:** A – 0 → Não há MSX-Audio conectado.  
 1 → Há um cartucho MSX-Audio conectado.  
 2 → Há dois cartuchos MSX-Audio conectados.

**Registradores:** BC, DE, HL.

Em seguida devemos obter a lista de pontos de entrada das rotinas MBIOS disponíveis, o que também é feito através de EXTBIO da seguinte forma:

#### EXTBIO (FFCAH/Work Area)

**Função:** Acessa funções estendidas da BIOS

**Entrada:** A = 00H.  
 D = 0AH – Dispositivo de manipulação do MSX-Audio.  
 E = 00H – Retorna o apontador para a tabela de informações do MSX-Audio.  
 B – ID do slot da tabela de endereços.  
 HL – Endereço de um buffer de 64 bytes para a tabela (deve estar na página 3).

**Saída:** B – ID do slot da tabela de informações.  
 HL – HL é incrementado de 4 e apontará para o final de uma tabela que reserva 4 bytes para o MSX-Audio. O valor original de HL aponta para o início da tabela, que tem a seguinte estrutura:

- +00H – ID do slot
- +01H – Endereço mais baixo
- +02H – Endereço mais alto
- +03H – Reservado para expansão

O ID de slot (+00H) e o endereço (+01H,+02H) apontarão para uma tabela com a seguinte estrutura:

+00H	VERSION	Versão do software
+03H	MBIOS	Music BIOS
+06H	AUDIO	Inicialização do MSX-Audio
+09H	SYNTHE	Chama o aplicativo SYNTHE
+0CH	PLAYF	Estado instrução PLAY
+0FH	BGM	Habilita/cancela modo BGM
+12H	MKTEMP	Definir tempo de gravação/

+15H	PLAYMK	reprodução teclado musical
+18H	RECMK	Toca pelo teclado musical
		Grava as Notas tocadas no teclado musical
+1BH	STOPM	Reprodução/gravação teclado /ADPCM; pára instr. PLAY
+1EH	CONTMK	Continua gravação pelo teclado musical
+21H	RECMOD	Configura modo de gravação do teclado musical
+24H	STPPLY	Pára a instrução PLAY
+27H	SETPCM	Área protegida ADPCM/PCM
+2AH	RECPCM	Gravação ADPCM/PCM
+2DH	PLAYPCM	Reprodução ADPCM/PCM
+30H	PCMFREQ	Alteração da frequência de reprodução ADPCM/PCM
+33H	MKPCM	Configura/cancela dados ADPCM p/ teclado musical
+36H	PCMVOL	Configura o volume de reprodução ADPCM/PCM
+39H	SAVEPCM	Salvar dados ADPCM/PCM
+3CH	LOADPCM	Carrega dados ADPCM/PCM
+3FH	COPYPCM	Transfere dados ADPCM/PCM
+42H	CONVP	Converte dados ADPCM > PCM
+45H	CONVA	Converte dados PCM > ADPCM
+48H	VOICE	Configura dados FM
+4BH	VOICECOPY	Movimenta dados FM

Registradores: F.

No Apêndice, item 8.5.5 – MSX-Audio, estão descritas todas as rotinas MBIOS disponíveis.

## 6.6 – O SCC

O SCC (Sound Creative Chip) é um gerador de áudio criado pela softhouse japonesa Konami para equipar seus cartuchos de jogos megaram. Existem dois tipos de SCC, o SCC “simples” (2212) e o SCC+ (2312).

O SCC gera sons mediante gravação da forma de onda em sua memória interna e sua reprodução se dá tal qual o PCM. A diferença é

que a memória reservada para os sons é muito limitada, de apenas 32 bytes por voz, que devem ser repetidos continuamente durante a síntese.

### 6.6.1 – O SCC “simples” (2212)

Esse SCC possui internamente 256 bytes de memória para armazenar os dados referentes a cada voz, que são em número de cinco. Abaixo está relacionada a memória reservada para cada voz.

- a – 32 bytes → forma de onda
- b – 12 bits → frequência de reprodução
- c – 4 bits → volume de saída
- d – 1 bit → liga/desliga a voz

A distribuição na memória dos dados referentes a cada uma das cinco vozes está ilustrada na tabela abaixo.

Endereços	Descrição resumida (SCC)
9800H~981FH	Forma de onda da voz #1
9820H~983FH	Forma de onda da voz #2
9840H~985FH	Forma de onda da voz #3
9860H~987FH	SCC : Esc/leit: Forma de onda das vozes #4 e #5 SCC+: Leitura: Forma de onda da voz #4
9880H~9881H	Frequência da voz #1
9882H~9883H	Frequência da voz #2
9884H~9885H	Frequência da voz #3
9886H~9887H	Frequência da voz #4
9888H~9889H	Frequência da voz #5
988AH	Volume da voz #1 (0 a 15)
988BH	Volume da voz #2 (0 a 15)
988CH	Volume da voz #3 (0 a 15)
988DH	Volume da voz #4 (0 a 15)
988EH	Volume da voz #5 (0 a 15)
998FH	•   •   •   v5   v4   v3   v2   v1   v5=1 → liga voz 5 v4=1 → liga voz 4, etc
9890H~989FH	Espelho de 9880H~988FH

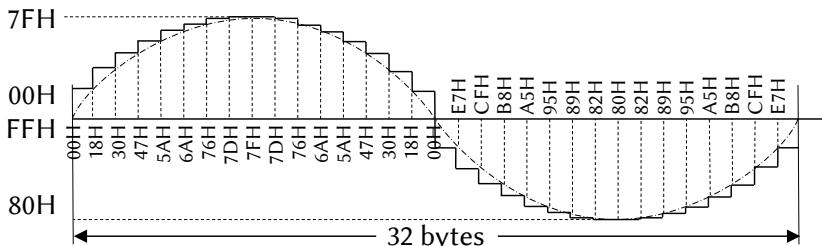
98A0H	SCC: sem função SCC+: leitura forma de onda voz #5 (escrita proibida)
98A1H~98BFH	Espelhos de 98A0H
98C0H	SCC: Espelho de 98A0H SCC+: Registrador de deformação
98C1H~98DFH	SCC: Espelho de 98A0H SCC+: Espelho de 98C0H
98E0H	SCC: Registrador de deformação SCC+: Sem função
98E1H~98FFH	Espelhos de 98E0H

#### 6.6.1.1 – Forma de onda

Os 32 bytes reservados para a forma de onda armazenam a mesma em complemento de dois: de 0 a 127 (00H a 7FH) a amplitude é incrementada; já de -1 a -128 (FFH a 80H) a amplitude é decrementada. A sequência de bytes a serem armazenados para uma forma de onda se- noidal está ilustrada na tabela abaixo.

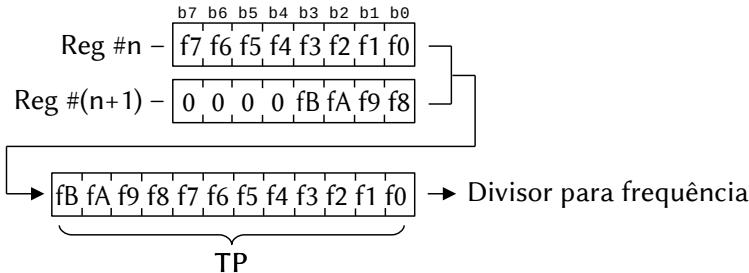
byte	valor	byte	valor	byte	valor	byte	valor
0	00H	8	7FH	16	00H	24	80H
1	18H	9	7DH	17	E7H	25	82H
2	30H	10	76H	18	CFH	26	89H
3	47H	11	6AH	19	B8H	27	95H
4	5AH	12	5AH	20	A5H	28	A5H
5	6AH	13	47H	21	95H	29	B8H
6	76H	14	30H	22	89H	30	CFH
7	7DH	15	18H	23	82H	31	E7H

A forma de onda resultante será a seguinte:



### 6.6.1.2 – Ajuste da frequência

A frequência da onda é definida no mesmo formato que para o PSG, conforme ilustração abaixo.



O valor armazenado em TP é o período. Assim, quando maior o valor de TP, menor será a frequência. A fórmula usada para o cálculo da frequência é a seguinte:

$$\text{F-tone} = \frac{\text{F-clock}}{32 * (\text{TP} + 1)}$$

Onde: F-clock é o clock presente no barramento interno (normalmente 3,579545 MHz);

Ftone é a frequência gerada na respectiva voz;

TP é o valor armazenado nos registradores de frequência.

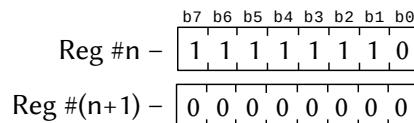
A fórmula derivada para calcular o valor de TP a partir da frequência é:

$$\text{TP} = \frac{\text{F-clock}}{\text{F-tone} * 32} - 1$$

Assim, para obter a nota LÁ central de 440 Hz, temos:

$$\text{TP} = \frac{\text{F-clock}}{\text{F-tone} * 32} - 1 = \frac{3579\,545}{440 * 32} = \frac{3579\,545}{14\,080} = 254$$

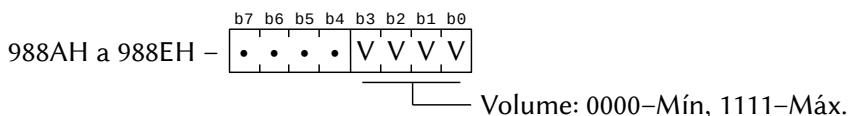
ou 0FEH em hexadecimal. Os registradores ficariam assim:



Cabe aqui ressaltar que o ciclo da frequência é uma passagem completa pelos 32 bytes definidores da forma de onda. Assim, se F-tone for igual a 10, serão feitas 10 passagens por segundo em todos os 32 bytes do registrador da forma de onda.

### 6.6.1.3 – Ajuste do volume

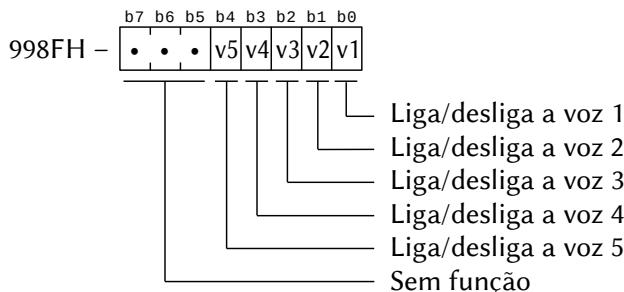
O volume também é armazenado no mesmo formato que para o PSG, conforme ilustrado abaixo.



Quando esse registrador for 0, o som será ausente; quando for 15, o volume será máximo.

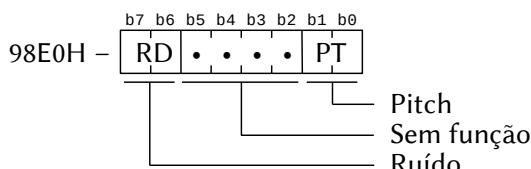
### 6.6.1.4 – Registrador de chaves

O registrador de chaves é mapeado como ilustrado abaixo (se o bit respectivo for 0, desliga a voz; se for 1, liga a voz).



### 6.6.1.5 – Registrador de deformação

Todos os endereços de 98E0H até 98FFH se referem ao mesmo registrador. Sua estrutura é a seguinte:

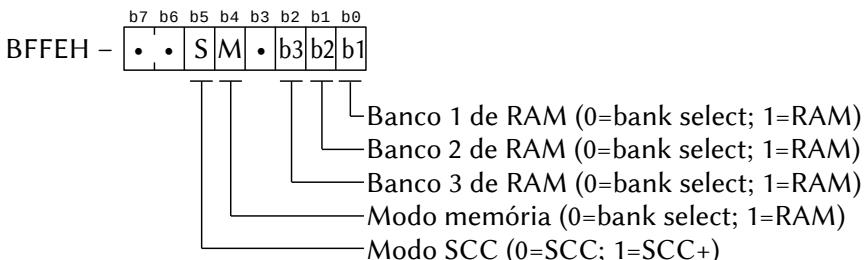


Os bits pitch alteram a frequência de todas as vozes. Se forem 11B ou 10B, as frequências serão multiplicadas por 16. Se forem 01B, as frequências serão multiplicadas por 256. Se forem 00B, as frequências não serão afetadas.

Os bits de ruído fazem com que as vozes 4 e 5 produzam ruído branco. Se forem 01B, produzirão ruído contínuo. Se forem 11B, produzirão ruído de acordo com a forma de onda definida (com envoltória). Se forem 00B, não haverá ruído (as vozes 4 e 5 gerarão somente a forma de onda).

### 6.6.2 – O SCC+ (2312)

O SCC+ é mais elaborado que o SCC “simples”. Além de gerar sons, ele também mapeia até 128 Kbytes de RAM. Nessa seção, só será tratada a parte de geração de sons. O SCC+ possui um registrador de modo que pode ser acessado nos endereços BFFEH e BFFFH. A sua estrutura é a seguinte:



Para usar o SCC+ no modo SCC “simples”, deve ser selecionado SCC no registrador de modo, bem como modo banco 3 no seletor de bancos de RAM. Basta escrever 00H no registrador de modo (valor selecionado do reset) e escrever o valor 00 111111B no registrador seletor de banco 3 (qualquer dos endereços 9000H ou 97FFH). Se o banco 3 estiver no modo RAM, pode-se ler o SCC, mas não escrever. Mesmo assim, há algumas diferenças no mapa de memória, conforme descrito abaixo.

Endereço	Função
9800H a 981FH	Forma de onda da voz 1
9820H a 983FH	Forma de onda da voz 2
9840H a 985FH	Forma de onda da voz 3
9860H a 987FH	Leitura: forma de onda da voz 4 Escrita: forma de onda das vozes 4 e 5
9880H a 9881H	Frequência da voz 1
9882H a 9883H	Frequência da voz 2
9884H a 9885H	Frequência da voz 3
9886H a 9887H	Frequência da voz 4
9888H a 9889H	Frequência da voz 5
988AH	Volume da voz 1
988BH	Volume da voz 2
988CH	Volume da voz 3
988DH	Volume da voz 4
988EH	Volume da voz 5
988FH	Registrador de chaves (liga/desliga)
9890H a 989FH	Espelho de 9880H a 988FH
98A0H a 98BFH	Leitura: forma de onda da voz 5 (escrita proibida)
98C0H a 98DFH	Registrador de deformação
98E0H a 98FFH	Sem função

No modo SCC, a compatibilidade com o SCC “simples” é total, exceto pelo endereço do registrador de deformação.

Para usar o modo SCC+, é necessário ligar o bit respectivo no registrador de modo e escrever o valor 10 000 000B (80H) no registrador seletor de banco 4 (qualquer dos endereços B000H ou B7FFH). Se o banco 4 estiver no modo RAM, o SCC+ pode ser lido, mas não escrito. O SCC+ aparecerá na área de memória B800H a B8FFH conforme a tabela abaixo.

Endereço	Função
B800H a B81FH	Forma de onda da voz 1
B820H a B83FH	Forma de onda da voz 2
B840H a B85FH	Forma de onda da voz 3
B860H a B87FH	Forma de onda da voz 4
B880H a B89FH	Forma de onda da voz 5

B8A0H a B8A1H	Frequência da voz 1
B8A2H a B8A3H	Frequência da voz 2
B8A4H a B8A5H	Frequência da voz 3
B8A6H a B8A7H	Frequência da voz 4
B8A8H a B8A9H	Frequência da voz 5
B8AAH	Volume da voz 1
B8ABH	Volume da voz 2
B8ACH	Volume da voz 3
B8ADH	Volume da voz 4
B8AEH	Volume da voz 5
B8AFH	Registrador de chaves (liga/desliga)
9890H a 989FH	Espelho de 9880H a 988FH
B8B0H a B8BFH	Mesmo que B8A0H a B8AFH
B8C0H a B8DFH	Registrador de deformação
B8E0H a B8FFH	Sem função

O conteúdo dos registradores é exatamente o mesmo do SCC “simples”. A diferença entre um e outro são os endereços de acesso e o fato de que as vozes 4 e 5 possuem registradores separados para a definição da forma de onda, no caso do SCC+.

### 6.6.3 – Acesso ao SCC

Para acessar o SCC, basta selecionar o slot onde ele está instalado e escrever ou ler os dados diretamente nos endereços de memória. O SCC não é acessado por portas de I/O. Entretanto, a área de memória entre 9880H e 98FFH é somente de escrita; se esta for lida, sempre retornará FFH. A área de memória entre 9900H e 99FFH é um espelho da área entre 9800H e 98FFH. O mesmo se dá entre as áreas 9A00H a 9AFFH e 9F00H a 9FFFH. Isso se deve ao fato do SCC não utilizar as linhas de endereço A8~A10; por isso, não consegue distinguir o endereço 9900H de 9800H ou 9F00H de 9800H. A área de memória de 8000H a 97FFH geralmente é uma parte da ROM que vem normalmente nos cartuchos.

### 6.6.4 – Detectando o SCC

Para verificar se existe algum cartucho SCC conectado, siga os seguintes passos:

1. Selecione o slot na página 2;
2. Selecione a ROM página 63;
3. Verifique se você pode ler/escrever em um local de RAM de forma de onda;
4. selecione uma página ROM diferente de 63 (2 por exemplo);
5. Verifique se a localização do RAM da onda é somente leitura.

Se tudo for bem sucedido, este é um slot com um SCC. Repita este teste para todos os slots possíveis.

A seguinte rotina para detecção do SCC foi escrita pelo Nyryikki e foi publicada na msx.org:

```
;-----  
; SCCDETECT (Made by : NYYRIKKI)  
;  
; Input: (None)  
;  
; Output:  
;  
; Success:  
;  
;     CF = NC  
;  
;     SCC SLOT #8000-#BFFF  
;  
;     A = SlotID  
;  
;  
; Fail:  
;  
;     CF = C  
;  
;     (Random slot on #8000-#BFFF)  
;  
; Changes: All registers  
;  
;  
SCCDETECT:  
    LD D, #FF  
    LD HL, #FCC1  
  
.MAINL:  
    INC D  
    LD A, 4  
    CP D  
    SCF  
    RET Z  
  
    LD A, (HL)  
    INC HL  
    AND #80  
    JR Z, .MAINTST
```

```
CALL .SUBTST
JR C,.MAINL
RET

.MAINTST:
LD A,D
CALL .TESTSLOT
JR C,.MAINL
LD A,D
RET
;-----
.SUBTST
LD E,#FC
.SUBLOOP:
LD A,4
ADD A,E
CP 16
SCF
RET Z
LD E,A
OR D
OR #80
LD C,A
CALL .TESTSLOT
JR C,.SUBLOOP
LD A,C
RET

.TESTSLOT:
PUSH BC
PUSH DE
PUSH HL

LD H,#80
CALL #24

LD A,(#9000)
LD D,A
LD A,#3F
LD (#9000),A
LD HL,#9800
LD E,(HL)
XOR A
```

```

LD (HL),A
LD A,(HL)
OR A
JR NZ,.NOSCC
DEC A
LD (HL),A
LD A,(HL)
INC A
JR NZ,.NOSCC
LD A,(#9000)
CP #3F
JR Z,.NOSCC
XOR A
JR .EXIT
.NOSCC
LD (HL),E
LD A,D
LD (#9000),A
SCF
.EXIT
POP HL
POP DE
POP BC
RET

```

## 6.7 – O OPL4

O OPL4 não é padrão no MSX, mas, em termos de reprodução sonora, é o mais perfeito, tendo qualidade de CD de áudio. Atualmente, pode ser encontrado em vários cartuchos, mas o pioneiro foi o Moonsound, desenvolvido pela Sunrise. Até o momento (2022), foram desenvolvidos os seguintes clones da Moonsound:

- Wozblaster, Argentina, por Gustavo Iriarte
- Shockwave/Shockwave 2, Brasil, por Tecnobytes
- Repro Factory Monster Sound FM Blaster, França
- JunSoft DalSoRi / JunSoft DalSoRi R2.0, Coréia
- 8bits4ever MSX-Blaster, Espanha
- MSX Calamar Wozblaster Enhanced, Espanha

O chip responsável é o YM278B. O OPL4 tem 18 vozes FM, todas redefiníveis, ou 15 vozes mais 5 peças de bateria, e 24 vozes PCM com qualidade CD (16 bits de resolução com “sampling rate” de 44,1 KHz). Também possui saída totalmente estéreo. O OPL4 não possui digitalizador interno de sons.

O OPL4 possui 250 registradores de 8 bits, numerados de \$00H a \$F9H, mas sua configuração varia conforme o tipo de reprodução (FM ou Wave).

### 6.7.1 – Descrição dos registradores para síntese wave

Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
\$00H \$01H	Teste								Registradores de teste
\$02H	ID disp		Cabeç. Wave		MT	MM	Funções especiais		
	b7~b5 b4~b2		ID do OPL4 (b7=0; b6=0; b7=1) Cabeçalho da tabela wave: 000=0 a 511 (000 000H)      100=384 a 511 (200 000H) 001=384 a 511 (080 000H)      101=384 a 511 (280 000H) 010=384 a 511 (100 000H)      110=384 a 511 (300 000H) 011=384 a 511 (180 000H)      111=384 a 511 (380 000H)						Endereço da memória de áudio
			b1      Tipo de memória de áudio (0=ROM; 1=RAM) b0      Acesso à memória de áudio (0=OPL4; 1=CPU)						
\$03H	•	•	a21	a20	a19	a18	a17	a16	Endereço da memória de áudio
\$04H	a15	a14	a13	a12	a11	a10	a9	a8	
\$05H	a7	a6	a5	a4	a3	a2	a1	a0	
\$06H	Dados da memória								Registrador de dados
\$08H : \$1FH	Número da tabela Wave LSB (n7~n0)								24 registradores com o número LSB (n7~n0) da tabela Wave reproduzida
\$20H : \$37H	F_number (f6~f0)							Tab Wave (n8)	24 registradores com a Frequência 7 bits LSB e n° da tabela Wave MSB (n8)

\$38H ⋮ \$4FH	Oitava (o3~o0)		Pseudo-rev	F_number (f9~f7)	Oitava (-7 a +7) Pseudo-reverberação Frequência (3 bits MSB)
	b7~b0 b7~b6 b3 b7~b4	Com “b0” (n8) seleciona até 512 samples (0~511) Com “b2-b1-b0” (f9~f7) define a frequência Se “1” liga a pseudo-reverberação; se “0”, desligada Oitava. Varia de -7 a +7 (-8 não é permitido). Em conjunto com F_number define a frequência. Para oitava = 1 e F_number = 0, a frequência é de 44,1 KHz. $f(\text{¢}) = 1200 * (\text{oitava} - 1) + 1200 * \log_2 \frac{1024 + \text{F\_number}}{1024}$			
\$50H ⋮ \$5FH	Nível total (l6~l0)			ND	Nível total 7 bits (l6~l0) Nível direto
\$60H ⋮ \$67H	b7-b1 b0	Nível total (b7=-24dB, b6=-12dB, ... b1=-0,375dB) Nível direto (0=altera envoltória durante a interpolação; 1=altera imediatamente)			
\$68H ⋮ \$7FH	Key on Damp	LFO RST CH	Panpot	Funções diversas e balanceamento estéreo (Panpot)	
	b7 b6 b5 b4 b3~b0	0=key on; 1=key off 0=Damp desligado; 1=Damp ativo LFO RST (0=liga o LFO; 1=desliga o LFO) 0=Wave mixado com FM; 1=Sem mixagem Panpot: 0 1 2 ... 6 7 8 9 ... 13 14 15 Esq(dB) 0 -3 -6 ... -18 -∞ -∞ 0 ... 0 0 0 Dir(dB) 0 0 0 ... 0 0 -∞ -18 ... -9 -6 -3			
\$80H ⋮ \$97H	• •	LFO (s2~s0)	VIB (v2~v0)	Frequência do tremolo e do vibrato (LFO) Grau do vibrato (VIB)	
	b7~b6 b5~b3 b2~b0	Não usados LFO (0=0,168Hz, 1=2,019Hz, ... 7=7,066Hz) Grau Vibrato (0=off, 1=3,378; 2=5,065, ... 7=79,31)			
\$98H ⋮ \$AFH	Attack Rate		Decay Rate (1)	Attack Rate 10-90% → (1=3715 mS; 14=0,23 mS) Decay 1 Rate 10-90% → (1=19 040 mS; 14=1,18 mS)	

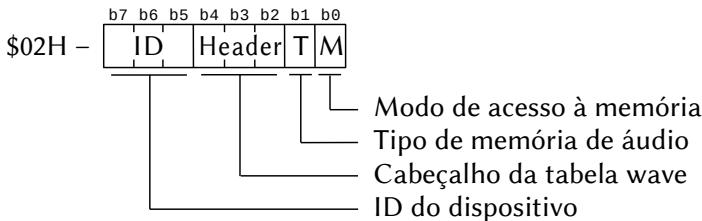
\$B0H : \$C7H	Decay Level	Decay Rate (2)	Decay Level (b7=-24; b6=-12; b5=-6; b4=-3 dB) Decay 2 Rate 10-90% → (1=19 040 mS; 14=1,18 mS)	
\$C8H : \$DFH	Rate Correction	Release Rate	Rate Correction Release Rate	
	b7~b4	Rate Correction: ( RATE = (OCT + RC)*2 + f9 + RD ) OCT = Oitava (-7 a +7 em \$38H~\$4FH) RC = Rate correction (0 ~ 14 em \$C8H~\$DFH) f9 = bit “f9” de F_number (\$38H~\$4FH) RD = valores de AR, D1R, D2R e RR (0001=04; 0010=08; ...; 1111=63)		
	b3~b0	Release Rate 10-90% → (1=19 040 mS; 14=1,18 mS)		
\$E0H : \$F7H	• • . . .	AM(a2~a0)	Grau do tremolo	
	b7~b3	Não usados		
	b2~b0	Amplitude do tremolo (0=off; 1=1,781; ...; 7=11,91)		
\$F8H	• •	Mix FM_R	Mix FM_L	Nível de saída FM
	b7~b6	Não usados (sempre “00”)		
	b5~b3	Nível FM direito (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
	b2~b0	Nível FM esquerdo (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
\$F9H	• •	Mix PCM_R	Mix PCM_L	Nível de saída PCM
	b7~b6	Não usados (sempre “00”)		
	b5~b3	Nível PCM direito (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
	b2~b0	Nível PCM esquerdo (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		

Os registradores para a síntese wave (Wave Table Synthesis) estão descritos logo em seguida. No final dessa seção, será descrito o modelo da síntese wave. Para a síntese FM, os registradores diferem um pouco. Serão descritos mais adiante.

Os registradores \$00H e \$01H são usados apenas para teste do YM278B. Devem sempre ser setados em 00H.

#### 6.7.1.1 – Acesso à memória de áudio

O registrador \$02H controla o acesso à memória:



### Modo de acesso à memória

Quando esse bit for 0, há geração normal de sons. Quando for 1, a CPU poderá ler ou escrever dados na memória de áudio e não haverá geração de sons.

### Tipo de memória de áudio

Quando esse bit for 0, somente ROM pode ser conectada. Quando for 1, podem ser conectadas SRAM mais ROM.

### Cabeçalho (cabeçalho) da tabela wave

O cabeçalho deve ser setado a partir do endereço 000 000H da memória de áudio para os números wave de 0 a 511, e em incrementos de 4 Mbit para os números wave de 384 a 511, conforme ilustrado na tabela abaixo.

cabeçalho b4 b3 b2	Área de memória de áudio
0 0 0	waves número 0 a 511 setados em 000 000H
0 0 1	waves número 384 a 511 setados em 080 000H
0 1 0	waves número 384 a 511 setados em 100 000H
0 1 1	waves número 384 a 511 setados em 180 000H
1 0 0	waves número 384 a 511 setados em 200 000H
1 0 1	waves número 384 a 511 setados em 280 000H
1 1 0	waves número 384 a 511 setados em 300 000H
1 1 1	waves número 384 a 511 setados em 380 000H

### ID do dispositivo

Registrador de identificação do OPL4. Sempre retorna o valor 001B, mesmo que seja escrito com outros valores.

### Endereços da memória de áudio

	b7 b6 b5 b4 b3 b2 b1 b0	
\$03H -	0 0 Endereço (high)	A21 ~ A16
\$04H -	Endereço (mid)	A15 ~ A8
\$05H -	Endereço (low)	A7 ~ A0

Esses registradores especificam o endereço da memória de áudio a ser escrito ou lido. O endereço é setado ao escrever o valor em \$05H; por isso, é necessário sempre preencher os registradores a partir de \$03H. Esses registradores são incrementados a cada acesso à memória de áudio.

### Registrador de dados da memória

	b7 b6 b5 b4 b3 b2 b1 b0	
\$06H -	Dados	Data register

Esse registrador é usado para a transferência de dados entre a CPU e a memória de áudio. Porém é um registrador lento. Deve haver uma pausa de 28 ciclos de relógio antes do próximo dado ser escrito e de 38 ciclos de relógio antes do próximo dado ser lido.

#### 6.7.1.2 – Acesso ao modo wave

	b7 b6 b5 b4 b3 b2 b1 b0	
\$08H ~ \$1FH -	Nº tabela (low)	N7 ~ N0
\$20H ~ \$37H -	• • • • • • n8	N8

O OPL4 suporta até 512 tabelas wave definidas e pode reproduzir até 24 delas simultaneamente. Os registradores funcionam aos pares; assim, \$08H forma par com o \$20H e assim sucessivamente. Deve-se sempre preencher o registrador de menor número antes (N7~N0 antes de N8). Como o cabeçalho é armazenado na memória de áudio, durante o carregamento do mesmo não se pode acessar LFO, VIB, AR, D1R, DL, D2R, Rate Correction, RR ou AM ou algum problema pode ocorrer. Os registradores de outras vozes podem ser acessados normalmente. O carregamento do cabeçalho demora cerca de 300µs após a escrita de N8. O bit b1 do registrador de estado indica quando um cabeçalho está sendo carregado.

## Frequência e oitava

$\$20H \sim \$37H$  - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
f <sub>6</sub>	f <sub>5</sub>	f <sub>4</sub>	f <sub>3</sub>	f <sub>2</sub>	f <sub>1</sub>	f <sub>0</sub>	•

 F\_number ( $f_6 \sim f_0$ )

$\$38H \sim \$4FH$  - 

•	•	•	•	•	f <sub>9</sub>	f <sub>8</sub>	f <sub>7</sub>
---	---	---	---	---	----------------	----------------	----------------

 F\_number ( $f_9 \sim f_7$ )

$\$38H \sim \$4FH$  - 

o <sub>3</sub>	o <sub>2</sub>	o <sub>1</sub>	o <sub>0</sub>	•	•	•	•
----------------	----------------	----------------	----------------	---	---	---	---

 Oitava (o<sub>3</sub> ~ o<sub>0</sub>)

Esses registradores são usados para controlar o “pitch” (tom) da voz. Também funcionam aos pares (\$20H e \$38H, \$21H e \$39H, etc). F\_number é um número positivo (0 a 1023) e a oitava é complemento de 2 (-7 a +7). O valor -8 não deve ser usado. Quando F\_number for 0 e a oitava for 1, o dado wave é reproduzido com o “sampling rate” de 44,1 KHz. Esse é o “pitch” normal ( $F(\$)=0$ , onde  $\$=1\%$ ).

O offset a partir do “pitch” normal pode ser calculado pela seguinte expressão:

$$F(\%) = 1200 * (\text{oitava} - 1) + 1200 * \log_2 \frac{1024 + \text{F-number}}{1024}$$

(1 oitava = 1200%)

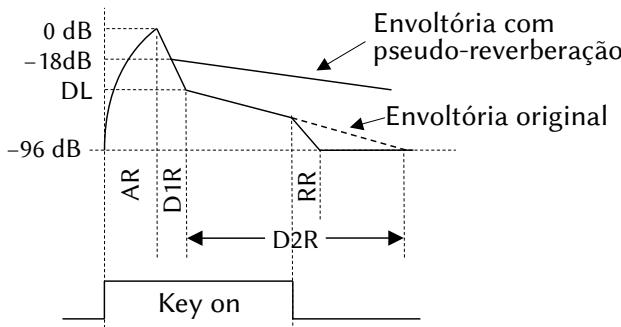
## Pseudo reverberação

$\$38H \sim \$4FH$  - 

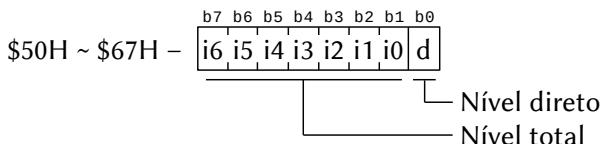
•	•	•	•	r	•	•	•
---	---	---	---	---	---	---	---

 Pseudo-reverberação

Quando esse bit for 0, a pseudo-reverberação estará desligada; quando for 1, estará ligada.



### Nível total / nível direto

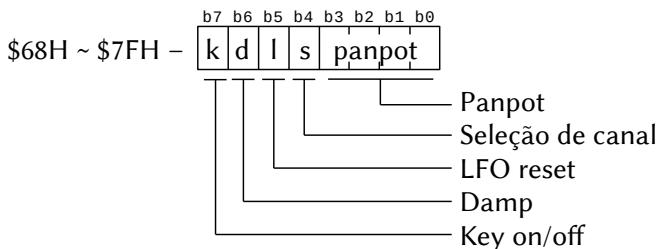


O nível total define o nível de atenuação do som reproduzido. A atenuação é a soma dos valores descritos abaixo quando o bit respectivo no registrador for 1.

$$\begin{array}{ll}
 i6 = -24 \text{ dB} & i2 = -1,5 \text{ dB} \\
 i5 = -12 \text{ dB} & i1 = -0,75 \text{ dB} \\
 i4 = -6 \text{ dB} & i0 = -0,375 \text{ dB} \\
 i3 = -3 \text{ dB} &
 \end{array}$$

O nível direto seleciona a maneira como o nível total modifica a envoltória. Se for 0, o nível total altera a envoltória durante a interpolação; se for 1, o nível total altera a envoltória imediatamente. Quando o nível for modificado durante a interpolação, o tempo de subida do volume mínimo para o máximo será de 78,2 ms e de descida do máximo para o mínimo será de 156,4 ms.

### Key on, DAMP, reset do LFO, seleção de canal, panpot



A função panpot controla o balanceamento estéreo de cada uma das vozes wave. O nível sonoro dos canais direito e esquerdo são definidos de acordo com a tabela abaixo.

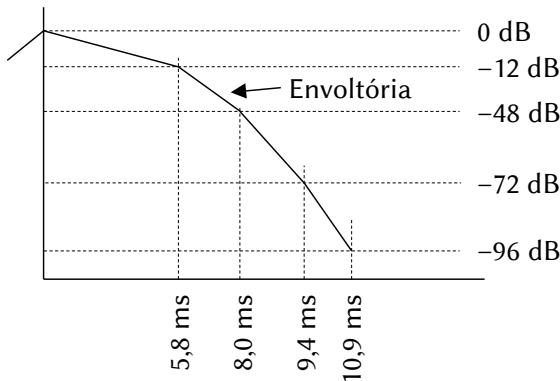
Panpot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Esq. (dB)	0	-3	-6	-9	-12	-15	-18	-∞	-∞	0	0	0	0	0	0	0
Dir. (dB)	0	0	0	0	0	0	0	0	-∞	-∞	-18	-15	-12	-9	-6	-3

O canal de saída pode ser selecionado de acordo com o bit b4 (seleção de canal) desses registradores. Se for 0, a saída será mixada com o gerador FM no pino DO2 do chip. Se for 1, não haverá mixagem com o gerador FM e a saída será direcionada para o pino DO1 do chip.

A flag “LFO Reset” ativa ou desativa o LFO (Low Frequency Oscillattor – Oscilador de Baixa Frequência) que é usado para os efeitos de vibrato e tremolo. Se for 0, o LFO estará ativo; se for 1, estará desligado.

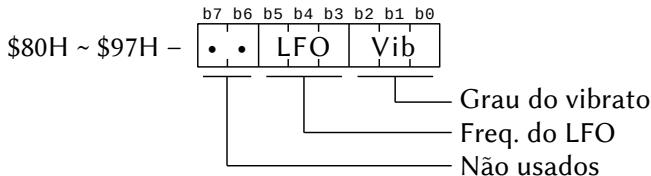
A flag “Damp” faz com que o tempo de “decay” e “release” fiquem mais curtos, quando ativa. Se for 0, o efeito “damp” estará desligado; se for 1 estará ligado. A pseudo-reverberação é desativada durante o “damping”. O efeito será aplicado conforme ilustrado abaixo.

Tempo (ms)	5,8	8,0	9,4	10,9
Atenuação (dB)	-12	-48	-72	-96



A flag “key on” controla a reprodução de sons. Se for 0, será selecionada a “key off” se for 1, será selecionada a “key on”.

### LFO, vibrato



Os bits b2~b0 determinam o grau do vibrato, conforme a tabela abaixo.

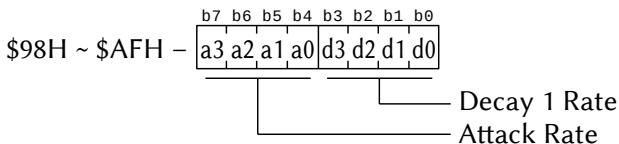
Bits b2~b0:	0	1	2	3	4	5	6	7
Grau (\$):	off	3,378	5,065	6,760	10,11	20,17	40,11	79,31

Os bits b5~b3 determinam a frequência do tremolo e do vibrato, conforme a tabela abaixo:

Bits b5~b3:	0	1	2	3	4	5	6	7
Frequência (Hz):	0,168	2,019	3,196	4,206	5,215	5,888	6,224	7,066

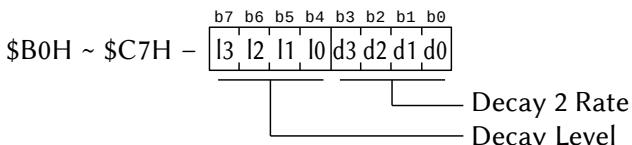
Os bits b7~b6 não são usados.

### Attack rate, decay 1 rate



Esse registrador define o “Decay 1 Rate” e o “Attack Rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

### Decay level, decay 2 rate



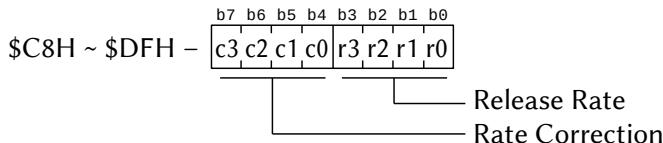
Os bits b0~b3 desse registrador definem o “Decay 2 Rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

Já os bits b7~b4 definem o “Decay Level”. O nível de “decay” pode ser calculado de acordo com a tabela abaixo, somando-se os valores quando o bit respectivo for 1. Entretanto, quando todos os bits forem

um, o “decay level” será levado a -93 dB, e não a -45 dB que corresponde à soma dos valores.

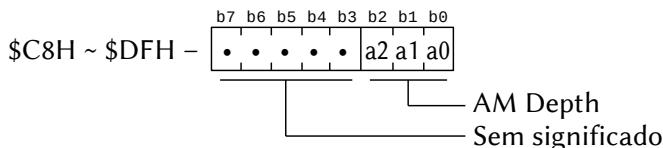
Bit	I3	I2	I1	I0
nível (dB)	-24	-12	-6	-3

### Release rate, rate correction



Os bits b3~b0 desse registrador definem o “Release Rate” e os bits b7~b4 definem o “Rate Correction”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

### AM depth (tremolo)



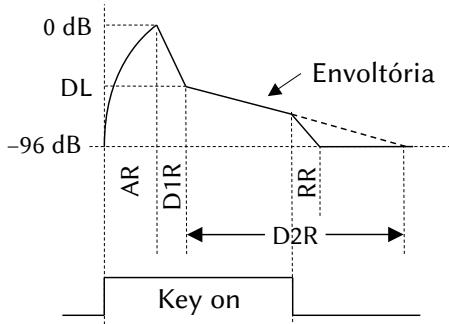
Este registrador determina o grau do tremolo, de acordo com a tabela abaixo:

Registrador:	0	1	2	3	4	5	6	7
Grau(dB):	off	1,781	2,906	3,656	4,406	5,906	7,406	11,91

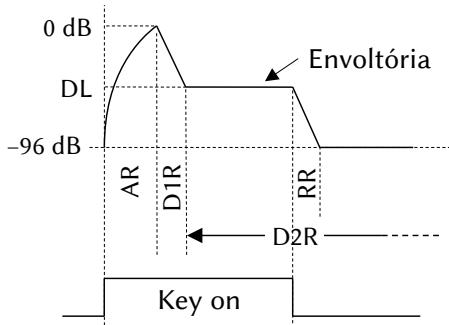
### Forma de onda da envoltória

A envoltória pode ter duas formas distintas, designadas por tom percussivo e tom constante. Elas estão ilustradas abaixo.

### Tom percussivo ( $D2R > 0$ )



### Tom constante ( $D2R = 0$ )



### Calculando os “rates”

O “rate” atual pode ser calculado pela seguinte fórmula:

$$\text{RATE} = (\text{OCT} + \text{Rate Correction}) * 2 + f_9 + RD$$

Onde: OCT: oitava (-7 a +7) especificada em \$38H~\$4FH

f9 : bit f9 do registrador F\_number (\$38H~\$4FH)

Rate Correction: valor de \$C8H~\$DFH (0 a 14)

O valor RD é determinado pelos valores especificados em AR, D1R, D2R e RR. A relação entre estes registradores e RD está ilustrada na tabela abaixo.

AR,D1R,D2R,RR	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Valor de RD	04	08	12	16	20	24	28	32	36	40	44	48	52	56	63

Sempre que o valor for maior que 63 na equação, o valor considerado sempre será 63. Quando AR=D1R=D2R=RR=0, RATE será 0; Quando AR=D1R=D2R=RR=15, RATE será 63.

As razões de “attack”, “decay” e “release” são definidas em 4 bits. Quanto maior o valor, menor o tempo de “attack” “decay” e “release”. Seus valores estão descritos na tabela abaixo.

### Attack Rate (tempo em milissegundos – ms)

RATE	Tempo (0–100%)	Tempo (10–90%)
0	∞	∞
1	∞	∞
2	∞	∞
3	∞	∞
4	6222.95	3715.19
5	4978.37	2972.20
6	4148.66	2476.83
7	3556.01	2122.99
8	3111.47	1857.60
9	2489.21	1486.12
10	2074.33	1238.41
11	1778.00	1061.50
12	1555.74	928.80
13	1244.63	743.08
14	1037.19	619.23
15	889.02	530.75
16	777.87	464.40
17	622.31	371.56
18	518.59	309.61
19	444.54	265.40
20	388.93	232.20
21	311.16	185.80
22	259.32	154.83
23	222.27	132.70
24	194.47	116.10
25	155.60	92.93
26	129.66	77.41

RATE	Tempo (0–100%)	Tempo (10–90%)
32	48.62	29.02
33	38.91	23.27
34	32.43	19.37
35	27.80	16.60
36	24.31	14.51
37	19.46	11.66
38	16.24	9.70
39	13.92	8.30
40	12.15	7.26
41	9.75	5.85
42	8.12	4.85
43	6.98	4.17
44	6.08	3.63
45	4.90	2.95
46	4.08	2.45
47	3.49	2.09
48	3.04	1.81
49	2.49	1.45
50	2.13	1.22
51	1.90	1.09
52	1.72	0.95
53	1.41	0.77
54	1.18	0.63
55	1.04	0.54
56	0.91	0.50
57	0.73	0.36
58	0.59	0.27

27	111.16	66.35
28	97.23	58.05
29	77.82	46.49
30	64.85	38.73
31	55.60	33.20

59	0.50	0.27
60	0.45	0.23
61	0.45	0.23
62	0.45	0.23
63	0.00	0.00

### Decay e Release Rate (tempo em milissegundos – ms)

RATE	Tempo (0–100%)	Tempo (10–90%)
0	∞	∞
1	∞	∞
2	∞	∞
3	∞	∞
4	89 164.63	19 040.36
5	71 331.75	15 278.73
6	59 443.13	12 724.54
7	50 951.25	10 890.16
8	44 582.31	9 520.18
9	35 665.90	7 639.37
10	29 721.59	6 362.27
11	25 475.65	5 445.08
12	22 291.16	4 760.09
13	17 832.97	3 819.68
14	14 860.82	3 181.13
15	12 737.82	2 722.54
16	11 145.58	2 380.05
17	8 916.51	1 909.84
18	7 430.43	1 590.57
19	6 368.93	1 361.27
20	5 572.79	1 190.02
21	4 458.28	954.92
22	3 715.24	795.28
23	3 184.49	680.63
24	2 786.39	595.01
25	2 229.16	477.46

RATE	Tempo (0–100%)	Tempo (10–90%)
32	696.60	148.75
33	557.32	119.37
34	464.44	99.41
35	398.10	85.08
36	348.30	74.38
37	278.68	59.68
38	232.24	49.71
39	199.05	42.54
40	174.15	37.19
41	139.37	29.84
42	116.15	24.85
43	99.55	21.32
44	87.07	18.59
45	69.71	14.92
46	58.10	12.43
47	49.80	10.66
48	43.54	9.23
49	34.83	7.44
50	29.02	6.08
51	24.90	5.31
52	21.77	4.67
53	17.41	3.72
54	14.51	3.13
55	12.43	2.68
56	10.08	2.36
57	8.71	1.95

26	1857.64	397.64
27	1592.24	340.32
28	1393.20	297.51
29	1114.60	238.73
30	928.84	198.82
31	796.15	170.16

58	7.23	1.59
59	6.21	1.36
60	5.44	1.18
61	5.44	1.18
62	5.44	1.18
63	5.44	1.18

### 6.7.1.3 – Formato da “wave table synthesis”

A “Wave Table Synthesis” é uma maneira de reprodução do PCM que utiliza samples, geralmente curtos, que reproduz a parte inicial e repete continuamente a parte seguinte. Para isso, dispõe de um cabeçalho (header) que é gravado na memória externa (de áudio). A estrutura do cabeçalho está descrita abaixo.

Reg	7	6	5	4	3	2	1	0	
00H	d1	d0	s21	s20	s19	s18	s17	s16	Data bit (d1 e d0)
01H	s15	s14	s13	s12	s11	s10	s9	s8	Endereço inicial (s21~s0)
02H	s7	s6	s5	s4	s3	s2	s1	s0	
03H	l15	l14	l13	l12	l11	l10	l9	l8	Endereço de loop (l15~l0)
04H	l7	l6	l5	l4	l3	l2	l1	l0	
05H	e15	e14	e13	e12	e11	e10	e9	e8	Endereço final (e15~e0)
06H	e7	e6	e5	e4	e3	e2	e1	e0	
07H	•	•	f2	f1	f0	v2	v1	v0	Frequência LFO e grau vibrato
08H	ar3	ar2	ar1	ar0	dr3	dr2	dr1	dr0	Attack Rate; Decay 1 Rate
09H	dl3	dl2	dl1	dl0	dr3	dr2	dr1	dr0	Decay Level; Decay 2 Rate
0AH	rc3	rc2	rc1	rc0	rr3	rr2	rr1	rr0	Rate Correction; Release Rate
0BH	•	•	•	•	am2	am1	am0		Grau da AM (tremolo)

Todos os cabeçalhos ficam em sequência na memória de áudio, antes da área de dados (samples), de 0 a 383 ou de 384 a 511, ainda que não utilizados.

#### Resolução de bits

Os bits d1 e d0 especificam a resolução em bits dos dados a serem reproduzidos, conforme a tabela abaixo.

d1	d0	Resolução	d1	d0	Resolução
0	0	8 bits	1	0	16 bits
0	1	12 bits	1	1	Proibido

Os respectivos formatos na área de dados são os seguintes:

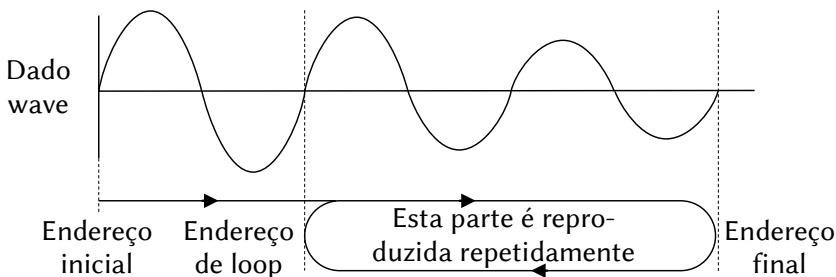
16 bits	d15	d14	d13	d12	d11	d10	d9	d8	+00H
	d7	d6	d5	d4	d3	d2	d1	d0	+01H
12 bits	d11	d10	d9	d8	d7	d6	d5	d4	+00H
	d3	d2	d1	d0	d3 d2 d1 d0				+01H
8 bits	d11	d10	d9	d8	d7	d6	d5	d4	+02H
	d7	d6	d5	d4	d3	d2	d1	d0	+00H

### Endereço inicial

O endereço inicial dos dados wave é especificado de forma absoluta. Para resolução de 12 bits, o endereço inicial sempre deve ser especificado a partir do bit 8 do byte de ordem mais alta.

### Endereço de loop

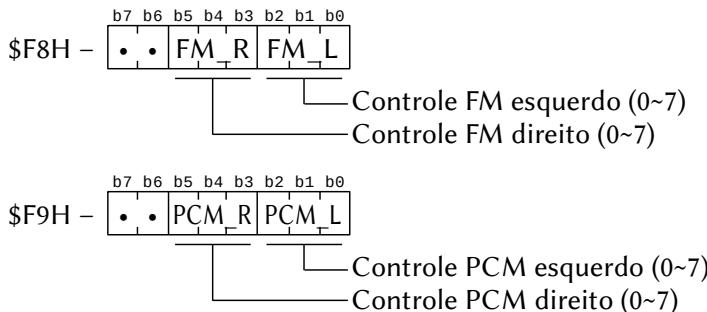
O endereço de loop especifica o endereço a partir do qual os dados serão repetidos. Ele é relativo ao endereço inicial.



### Endereço final

O endereço final de reprodução é especificado relativamente ao endereço inicial. Como apenas 16 bits são usados para o endereço final, cada sample pode ter no máximo 64 Kbytes.

#### **6.7.1.4 – Controle de mixagem wave/FM**

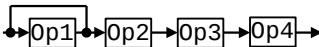
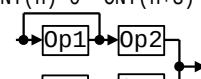
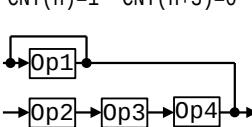
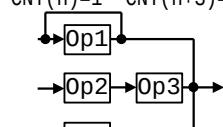
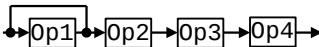
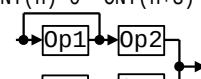
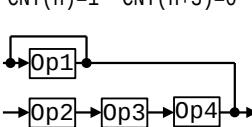
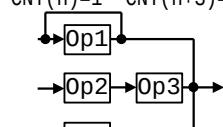
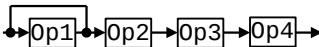
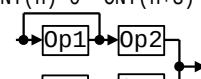
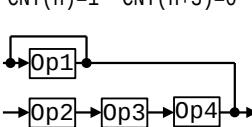
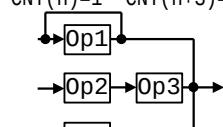


Esses registradores especificam o nível de mixagem da saída do gerador FM e do PCM no pino DO2. Ao ser resetado, o OPL4 coloca o nível de mixagem do FM (\$F8H) em -9 dB como default e do PCM (\$F9H) em 0 dB, balanceando o volume das saídas FM e PCM. Os níveis estão descritos na tabela abaixo.

Registrador:	0	1	2	3	4	5	6	7
Nívelmix(dB):	0	-3	-6	-9	-12	-15	-18	$-\infty$

#### 6.7.2 – Descrição dos registradores para o gerador FM

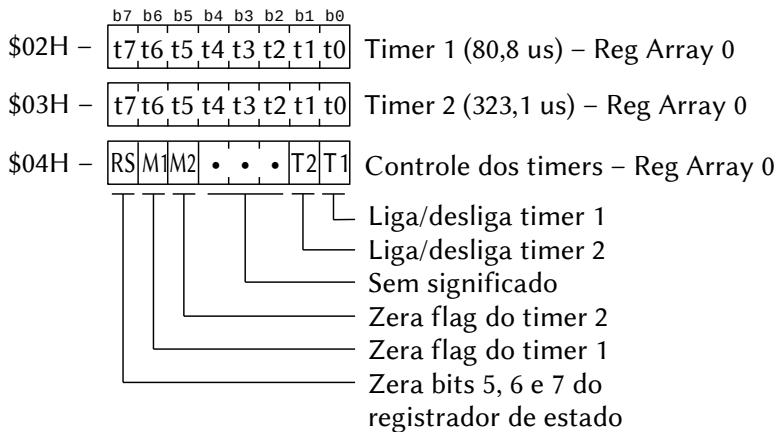
\$05H	Somente no Register Array 1											
\$08H	•	NTS	•	•	•	•	Configuração de teclado					
	b7	Não usado (sempre “0”)										
	b6	NTS – Se 0, as oitavas serão determinadas pelos 2 bits mais altos de F_number. Se 1, serão determinadas apenas pelo bit mais alto de F_number.										
	b5~b0	Não usados (sempre “000 000”)										
\$20H ⋮ \$35H	AM VIB	EGT KSR	Múltiplo			Definição de instrumentos						
	b7	AM (1=liga trêmolo – Variação de amplitude 3,7Hz)										
	b6	VIB (1=liga vibrato – Variação de frequência 6,4Hz)										
	b5	EG-TYP (0=tom percussivo; 1=tom constante)										
	b4	Se 0, KSR→0~3; Se 1, KSR→0~15										
	b3~b0	Fator de multiplicação (0=1/2, 1=1, 2=2, 3=3, ..., 15=15)										
\$40H ⋮ \$55H	KSL	Nível total			KSL (00= 0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB) Nível total (b0=0,75dB, b1=1,5dB ..... b5=24dB)							
\$60H ⋮ \$75H	Attack Rate (AR)		Decay Rate (DR)		Attack (0dB a 96dB → mín. 0,2 mS; máx 2826 mS) Decay (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)							
\$80H ⋮ \$95H	Sustain Level (SL)		Release Rate (RL)		Sustain (b7=24dB, b6=12dB, b5=6dB, b4=3dB) Release (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)							
\$A0H ⋮ \$A8H	Frequência (LSB 8 bits)				Frequência (b7~b0)							
\$B0H ⋮ \$B8H	•	•	KEY	Oitava	Freq. MSB 2 bits	Freq. MSB 2 bits (b9~b8) Oitava (FM) Key on/off (FM)						
	b7~b6	Não usados (sempre “00”)										

	b5 b4~b2 b1~b0	0=Voz respectiva desligada (key off); 1=voz ativa Define a oitava. A quarta é 011. Frequência MSB 2 bits. A Nota Lá central de 440 Hz é obtida com b1~b0=10 e \$A0H~A8H=01000 110																																												
\$BDH	AM   VIB	BAT   BD   SD   TOM   TCY   HH   Controle da bateria do FM																																												
	b7 b6 b5 b4 b3 b2 b1 b0	Grau do trémolo (0=1dB. 1=4,8dB) Grau do vibrato (0=7%; 1=14%) 0=Modo Melodia; 1=Modo Bateria 1=Bass Drum 1=Snare Drum 1=Tom-tom 1=Top Cymbal 1=High-Hat																																												
\$C0H : \$C8H	•   •   •   •   Feedback   CON	Fator de realim. tipo conexão																																												
	b7~b4 b3~b1 b0	Não usados (sempre “0000”) Realimentação (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π) Tipo de conexão (0=série; 1=paralelo) Para 4 operadores: <table style="margin-left: 20px;"> <tr><td>A1</td><td>Canal</td><td>CNT(n)</td><td>CNT(n+3)</td></tr> <tr><td>0</td><td>1</td><td>C0H</td><td>C3H</td></tr> <tr><td>0</td><td>2</td><td>C1H</td><td>C4H</td></tr> <tr><td>0</td><td>3</td><td>C2H</td><td>C5H</td></tr> <tr><td>1</td><td>4</td><td>C0H</td><td>C3H</td></tr> <tr><td>1</td><td>5</td><td>C1H</td><td>C4H</td></tr> <tr><td>1</td><td>6</td><td>C2H</td><td>C5H</td></tr> </table> Para 4 operadores: <table style="margin-left: 20px;"> <tr><td>CNT(n)=0</td><td>CNT(n+3)=0</td><td>CNT(n)=0</td><td>CNT(n+3)=1</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td>CNT(n)=1</td><td>CNT(n+3)=0</td><td>CNT(n)=1</td><td>CNT(n+3)=1</td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>	A1	Canal	CNT(n)	CNT(n+3)	0	1	C0H	C3H	0	2	C1H	C4H	0	3	C2H	C5H	1	4	C0H	C3H	1	5	C1H	C4H	1	6	C2H	C5H	CNT(n)=0	CNT(n+3)=0	CNT(n)=0	CNT(n+3)=1					CNT(n)=1	CNT(n+3)=0	CNT(n)=1	CNT(n+3)=1				
A1	Canal	CNT(n)	CNT(n+3)																																											
0	1	C0H	C3H																																											
0	2	C1H	C4H																																											
0	3	C2H	C5H																																											
1	4	C0H	C3H																																											
1	5	C1H	C4H																																											
1	6	C2H	C5H																																											
CNT(n)=0	CNT(n+3)=0	CNT(n)=0	CNT(n+3)=1																																											
																																														
CNT(n)=1	CNT(n+3)=0	CNT(n)=1	CNT(n+3)=1																																											
																																														

O “Register Array 0” é compatível com o OPL3; já o “Register Array 1” foi expandido para o modo OPL4. Esses modos são selecionados pelos bits “NEW” e “NEW2”. As diferenças entre o “Register Array 0” e entre o “Register Array 1” são de alguns poucos registradores e estão ilustradas abaixo.

\$05H	• • • • • NEW2 NEW	Registrador de expansão
	b7-b2 b1 b0	Não usados (sempre “000 000”) Se 1, ativa o modo OPL4 (Register Array 1) Se 1, ativa o modo OPL3 (Register Array 0)
\$08H	Somente no Register Array 0	
\$BDH	Somente no Register Array 0	

### 6.7.2.1 – Timers



Existem dois timers no OPL4. A resolução do timer 1 é de 80,8 µs e a do timer 2 é de 323,1 µs. As fórmulas que permitem calcular o tempo de cada um, em milissegundos, são as seguintes:

$$t1(\text{ms}) = (256 - n1) * 0,0808 \quad (\text{timer 1})$$

$$t2(\text{ms}) = (256 - n2) * 0,3231 \quad (\text{timer 2})$$

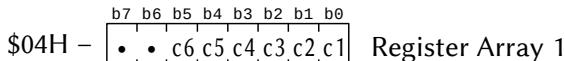
Onde n1 e n2 representam o valor de cada contador (0 ~ 255). Quando o tempo de cada contador for atingido, um sinal de interrupção é enviado para a CPU.

Os bits T1 e T2 de \$04H ativa ou desativam os timers 1 ou 2, respectivamente. Quando o bit for 0, o timer estará desativado; quando for 1, estará ativo. Quando os bits M1 ou M2 forem setados em 1, a flag do timer respectivo sempre será 0, independente da operação dos timers. Nesse caso, não será gerada interrupção. Quando o bit RS for colocado

em 1, os bits d5, d6 e d7 do registrador de estado serão zerados. Depois, RS retornará automaticamente a 0.

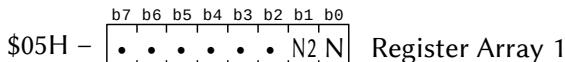
### 6.7.2.2 – Acesso ao modo FM

#### Seleção de modo 4 operadores



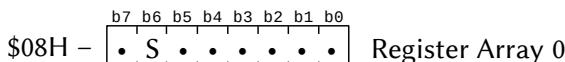
Quando qualquer dos bits c1~c6 for setado em 1, o canal correspondente poderá ser usado no modo 4 operadores. Maiores detalhes estão descritos na seção “CANAIS E SLOTS”.

#### Registrador de expansão



Esses registradores permitem a expansão dos modos OPL2 e OPL3 para OPL4. Se os dois bits forem 0, estará ativo o modo OPL2. Se o bit N for 1, estará ativo o modo OPL3 (Register array 0). Se o bit N2 for 1, estará ativo o modo OPL4 (Register Array 1). Como esses dois bits são zerados no reset, devem ser colocados em 1 para ativar o modo OPL4 antes da utilização do array 1 ou do PCM.

#### Seleção de separação de teclado



Até 8 oitavas podem ser selecionadas de um total de 16, para todas as vozes FM. O bit b6 de \$08H (NTS) determina quais oitavas estarão ativas, conforme a tabela abaixo:

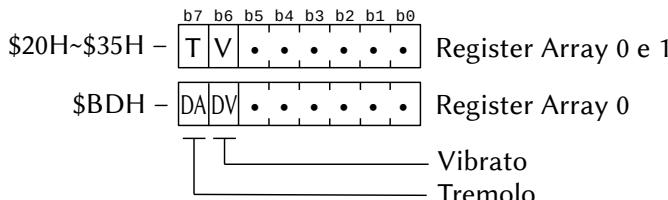
NTS = 0

Oitava	0	1	2	3	4	5	6	7
F_num msb	•	•	•	•	•	•	•	•
F_num 2º	0	1	0	1	0	1	0	1
Nº Key Scale	0	1	2	3	4	5	6	7

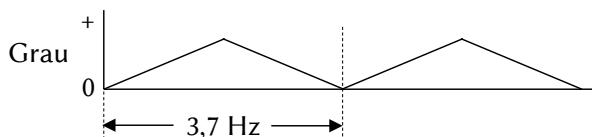
NTS = 1

Oitava	0	1	2	3	4	5	6	7
F_num msb	0	1	0	1	0	1	0	1
F_num 2º	•	•	•	•	•	•	•	•
Nº Key Scale	0	1	2	3	4	5	6	7

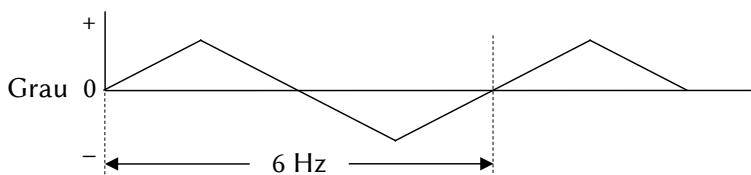
### Trêmolo e vibrato



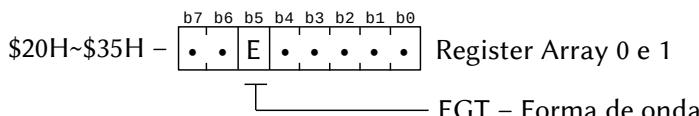
Quando o bit b7 de \$20H~\$35H for 1, o tremolo para a voz respectiva será ativado. A frequência do tremolo é de 3,7 Hz e o grau é determinado pelo bit DA (DA=0, 1db; DA=1, 4,8 dB), conforme ilustração abaixo.



O bit b6 de \$20H~\$35H liga ou desliga o vibrato para a voz respectiva; se for 0, estará desligado; se for 1 estará ligado. A frequência do vibrato é de 6 Hz e seu grau é determinado pelo bit DV (DV=0, 7%, DV=1, 14%), conforme ilustração abaixo.

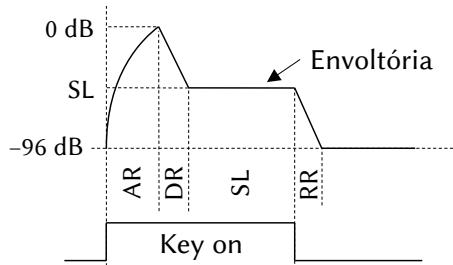


### Forma de onda da envoltória

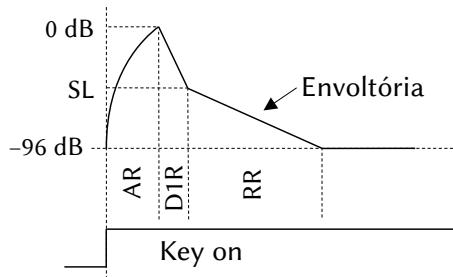


Esse bit determina a forma de onda da envoltória, conforme abaixo.

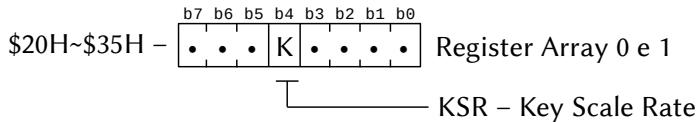
### **Tom constante (EGT = 0)**



### **Tom percussivo (EGT = 1)**



### **KSR (key scale rate)**



Esse bit é usado para regular o tempo de ausência de som no intervalo de mudança de tom, simulando instrumentos musicais reais. Os valores obedecem à tabela abaixo.

Key Scale Value		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Rof	KSR=0	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	
	KSR=1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Para os números da “key scale” ver o item “\$08H – Seleção de se-paração de teclado”.

O RATE pode ser calculado pela seguinte expressão:

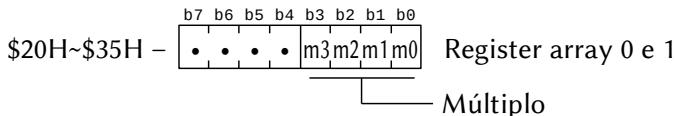
$$\text{RATE} = (\text{Rate value}) * 4 + \text{Rof}$$

Quando “rate value” for 0, RATE será 0. Quando RATE exceder a 63, sempre será setado em 63. As razões de “attack”, “decay” e “release” são definidas em 4 bits. Quanto maior o valor, menor o tempo de “attack” “decay” e “release”. Na tabela abaixo, estão descritos seus valores extremos em milissegundos. A variação de tempo obedece, aproximadamente, a uma progressão geométrica entre esses valores. Consulte a tabela do item “Calculando os rates” na seção “7.1.2 – Acesso ao modo wave” para obter os valores exatos.

	Mínimo	Máximo
Attack (0dB a 96dB)	0,20 ms	2826 ms
Attack (10% a 90%)	0,11 ms	1482 ms
Decay (0dB a 96dB)	2,40 ms	39 280 ms
Decay (10% a 90%)	0,51 ms	8212 ms
Release (0dB a 96dB)	2,40 ms	39 280 ms
Release (10% a 90%)	0,51 ms	8212 ms

Nota: Para RATE=60~63, o tempo será nulo e para RATE=0~3, o tempo será infinito.

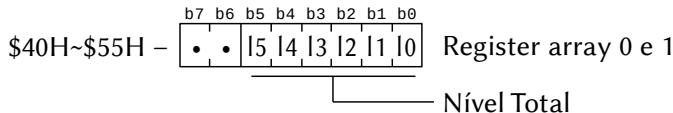
### Múltiplo



Esse registrador especifica o multiplicador para as frequências especificadas por BLOCK e F\_number. Os fatores de multiplicação estão ilustrados abaixo.

Valor do registrador:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fator de multiplicação:	½	1	2	3	4	5	6	7	8	9	10	10	12	15	15	

### Nível total



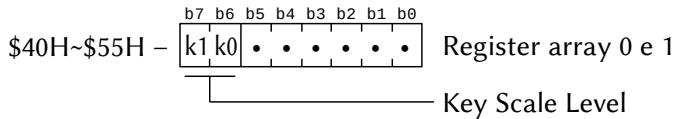
O nível total define o amortecimento para a envoltória. Ele controla o volume e a taxa de modulação. Seu valor corresponde à soma dos valores listados na tabela abaixo quando o bit respectivo for 1.

$l_5 = -24 \text{ dB}$	$l_2 = -3 \text{ dB}$
$l_4 = -12 \text{ dB}$	$l_1 = -1,5 \text{ dB}$
$l_3 = -6 \text{ dB}$	$l_0 = -0,75 \text{ dB}$

O nível real de amortecimento é determinado pela divisão de uma oitava em 16 partes pelo valor dos quatro bits mais altos de F-number. A taxa de amortecimento mostrada abaixo é para 3dB/oct. Para 1,5dB/octava, o amortecimento é metade e para 6,0dB/octava, o amortecimento é o dobro da tabela.

Oct. F_num \	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	3.000	6.000	9.000
2	0	0	0	0	3.000	6.000	9.000	12.000
3	0	0	0	1.875	4.875	7.875	10.875	13.875
4	0	0	0	3.000	6.000	9.000	12.000	15.000
5	0	0	1.125	4.125	7.125	10.125	13.125	16.125
6	0	0	1.875	4.875	7.875	10.875	13.875	16.875
7	0	0	2.625	5.625	8.625	11.625	14.625	17.625
8	0	0	3.000	6.000	9.000	12.000	15.000	18.000
9	0	0.750	3.750	6.750	9.750	12.750	15.750	18.750
10	0	1.125	4.125	7.125	10.125	13.125	16.125	19.125
11	0	1.500	4.500	7.500	10.500	13.500	16.500	19.500
12	0	1.875	4.875	7.875	10.875	13.875	16.875	19.875
13	0	2.250	5.250	8.250	11.250	14.250	17.250	20.250
14	0	2.625	5.625	8.625	11.625	14.625	17.625	20.625
15	0	3.000	6.000	9.000	12.000	15.000	18.000	21.000

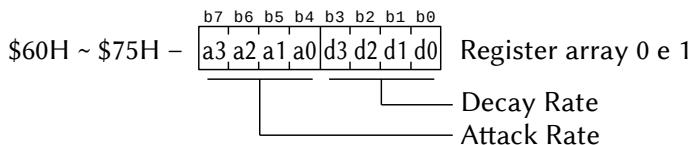
### KSL (key scale level)



Este registrador especifica a atenuação progressiva do som gerado de forma a aproximá-lo do som dos instrumentos musicais acústicos. Essa atenuação se dá de acordo com a tabela abaixo.

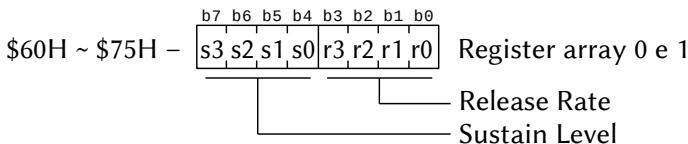
KSL	0	1	2	3
Atenuação	0 dB/oitava	3 dB/oitava	1,5 dB/oitava	6 dB/oitava

### Attack rate (AR) / decay rate (DR)



Este registrador define o “Attack Rate” e o “Decay Rate”. Maiores detalhes podem ser vistos no item “\$20H~\$35H – KSR (KEY SCALE RATE)”.

### Releade (RR) / SUSTAIN LEVEL (SL)

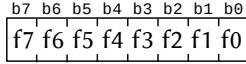


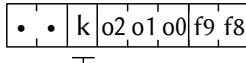
Para o “release rate” ver mais detalhes podem ser vistos no item “\$20H~\$35H – KSR (KEY SCALE RATE)”.

O “Sustain Level” especifica o nível em que a envoltória permanece após a “Decay Rate”. Para tom percussivo, especifica o ponto de transição da “Decay Rate” para a “Release Rate”. Seu valor corresponde à soma dos valores listados abaixo, quando o bit respectivo for 1. Quando todos os bits forem 1, o valor de SL será colocado em -93dB.

s3	s2	s1	s0
-24dB	-12dB	-6dB	-3 dB

### Ajuste da frequência / key on/off

\$A0H~\$A8H –  Register Array 0 e 1

\$B0H~\$B8H –  Register Array 0 e 1

Key on/off

O bit “k” (Key on/off) controla a geração do som. Veja o item “Forma de onda da envoltória” mais acima.

Os valores f0~f9 são chamados de F\_number e especificam a frequência gerada e os valores o0~o2 especificam a oitava (também chamada “bloco”). F\_number pode variar de 0 a 1023 e a oitava de 0 a 7.

A relação entre a frequência, F\_number e oitava é dada pela seguinte equação:

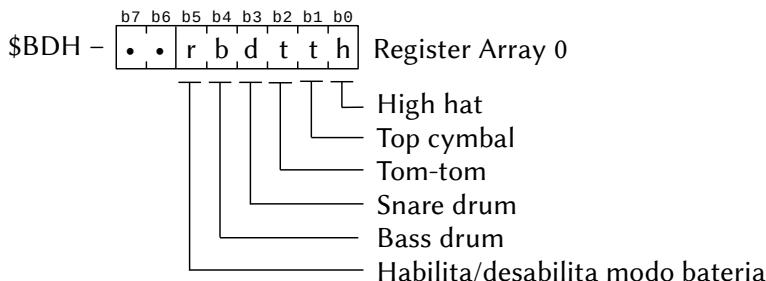
$$F\_number = \frac{(Frequência) \times 2^{19} \div (Freq. amostragem / 684)}{2^{(oitava-1)}}$$

Os valores para a oitava de número 4 (na escala 0 a 7) estão listados abaixo.

Cifrado	Frequência	Decimal	\$BxH,b1-b0	\$AxH
Dó	C	261,6 Hz	346	0 1
	C#	277,2 Hz	367	0 1
Ré	D	293,7 Hz	389	0 1
	D#	311,1 Hz	412	0 1
Mi	E	329,6 Hz	436	0 1
Fá	F	349,2 Hz	462	0 1
	F#	370,0 Hz	490	0 1
Sol	G	392,0 Hz	519	1 0
	G#	415,3 Hz	550	1 0
Lá	A	440,0 Hz	582	1 0
	A#	466,2 Hz	617	1 0
Si	B	493,9 Hz	654	1 0
Dó	C	523,3 Hz	693	1 0

Os valores das frequências guardam entre si uma relação geométrica igual à 12<sup>a</sup> raiz de 2, que vale 1,0594630943592. Pode-se usar esse número para alterar os valores dos registradores a fim de aumentar ou diminuir a frequência gerada dentro da escala musical. Os valores dos registradores também guardam entre si a mesma relação.

### Modo bateria

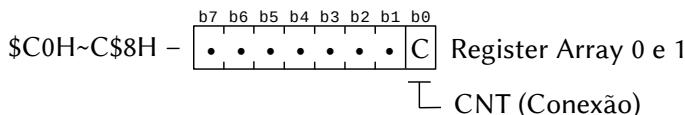


Esse registrador controla o modo bateria. Quando o bit “r” for 1, o modo bateria estará ativo e as três últimas vozes do gerador FM ficarão indisponíveis. Entretanto, poderão ser geradas até 5 peças de bateria, conforme descrito acima. Os slots usados são os seguintes:

Instrumento	Slot
Bass drum (BD)	13,16
Snare drum (SD)	17
Tom-tom (TOM)	15
Top cymbal (TC)	18
High hat (HH)	14

Os valores “Rate” e outros podem ser setados para manipular o som das peças de bateria. Quando em modo bateria, o bit “Key” deve ser colocado em 0 para os slots 13 a 18.

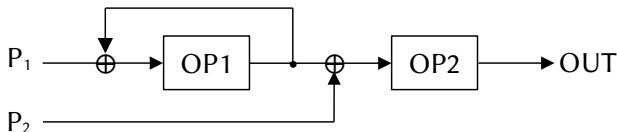
### CNT (conexão)



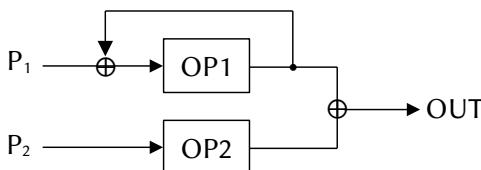
### • Modo 2 Operadores

No modo 2 operadores, quando o esse bit for 0, é selecionado o algoritmo 1. Quando for 1, é selecionado o algoritmo 2.

#### Algoritmo 1 (CNT=0)

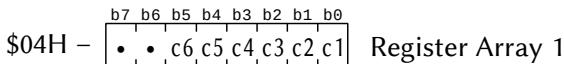


#### Algoritmo 2 (CNT=1)



### • Modo 4 Operadores

Para selecionar o modo 4 operadores, é necessário colocar o bit respectivo no registrador \$04H (Connection SEL) e depois usar os dois bits CNT disponíveis para aplicar os algoritmos.

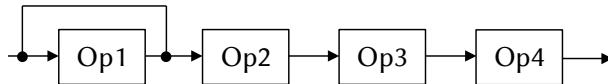


O valor dos bits CNT necessários para a seleção de algoritmos estão ilustrados na tabela abaixo.

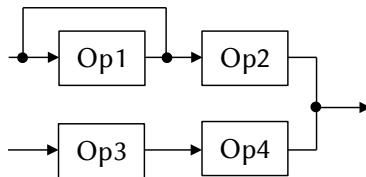
A1	Canal nº	Fazer bit CNT = 1	
		CNT <sub>n</sub>	CNT <sub>n+3</sub>
0	1	\$C0H	\$C3H
	2	\$C1H	\$C4H
	3	\$C2H	\$C5H
1	1	\$C0H	\$C3H
	2	\$C1H	\$C4H
	3	\$C2H	\$C5H

Os 4 algoritmos possíveis, usando os bits CNT<sub>n</sub> e CNT<sub>n+3</sub> estão ilustrados abaixo.

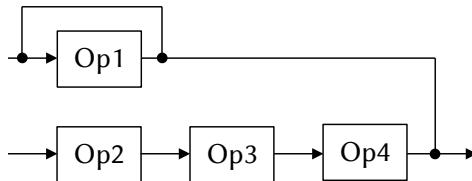
Algoritmo 1 (CNT<sub>n</sub>=0, CNT<sub>n+3</sub>=0)



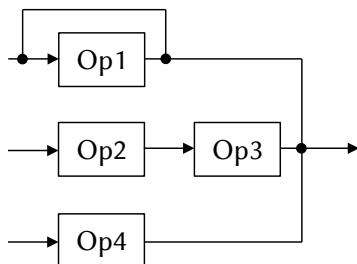
Algoritmo 2 (CNT<sub>n</sub>=0, CNT<sub>n+3</sub>=1)



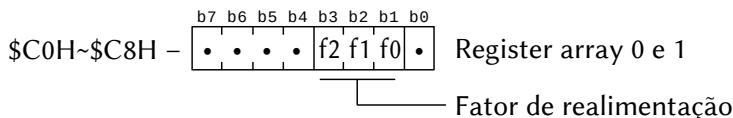
Algoritmo 3 (CNT<sub>n</sub>=1, CNT<sub>n+3</sub>=0)



Algoritmo 4 (CNT<sub>n</sub>=1, CNT<sub>n+3</sub>=1)



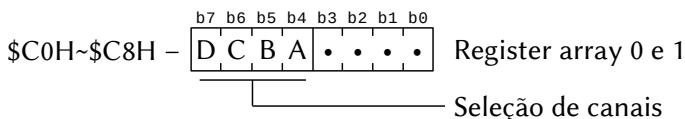
### Realimentação (feedback)



Esse registrador define o fator de realimentação (porção do sinal de saída que é reinjetado na entrada). Os valores de realimentação estão descritos na tabela abaixo.

Valor do registrador:	0	1	2	3	4	5	6	7
Taxa de modulação:	0	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	$\pi$	$2\pi$	$4\pi$

### Seleção de canais de saída

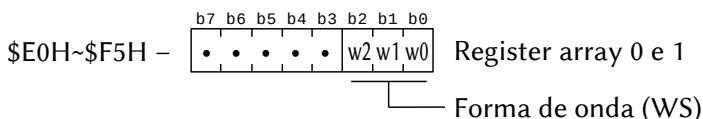


Até 4 canais de saída são disponíveis para o gerador FM. A saída estará habilitada quando o bit respectivo for 1.

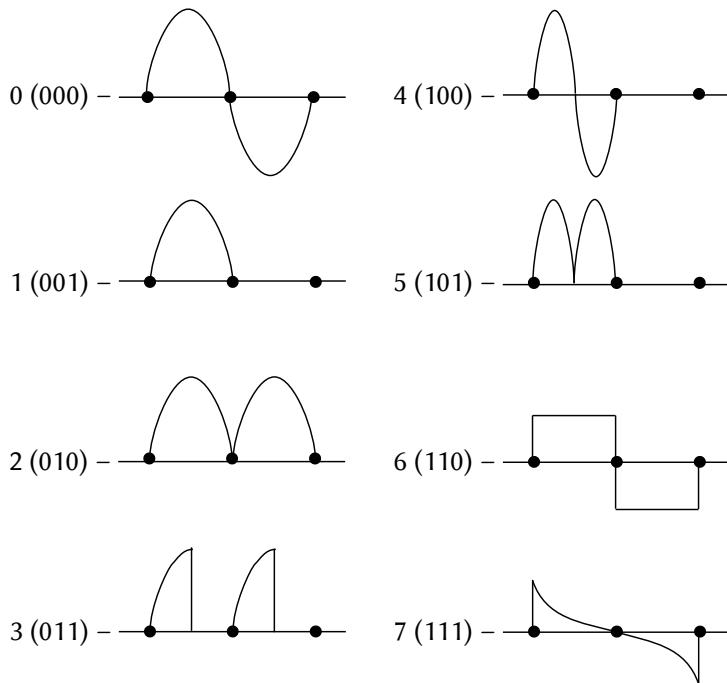
**Canais A e B:** a saída do sintetizador FM é mixada digitalmente com a “wave table synthesizer” do canal no qual o bit CH dos registradores \$68H a \$7FH da “wave table synthesizer” é colocado em “0”, e o sinal será enviado para o pino DO2. CHA é mixado com a saída esquerda da “wave table synthesizer” e CHB é mixado com a saída direita.

**Canais C e D:** A saída definida por CHC e CHD é emitida pelo pino DO0. Como FM sai do pino DO0 e PCM sai do pino DO1, efeitos sonoros podem ser aplicados a um determinado som conectando o chip YSS225 aos pinos DO0 e DO1. Obviamente, um conversor D/A também pode ser conectado.

### Seleção de forma de onda



Esse registrador seleciona a forma de onda a ser usada em cada slot. As formas de onda possíveis estão ilustradas abaixo. Quando em modo OPL2, somente WS0 (000) a WS3 (011) estarão disponíveis.



### 6.7.3 – Acesso ao OPL4

O OPL4 é acessado diretamente por portas de I/O, da mesma forma que o OPLL e o MSX-Audio. Não há rotinas padronizadas para acesso. As portas usadas pelo cartucho Moonsound são as seguintes:

0C4H – FM register array 0 (banco 1) e registrador de estado.

0C5H – FM (dados).

0C6H – FM register array 1 (banco 2).

0C7H – Espelho de C5H (o acesso por C5H é preferido).

07EH – Registradores PCM (wave).

07FH – Dados PCM (wave).

A forma de acesso é muito simples: basta selecionar o registrador através de C4H, C6H ou 7EH e depois escrever o dado através de C5H ou 7FH. Não é necessário se preocupar com pausas até o MSX turbo R; os acessos podem ser feitos sequencialmente sem problemas. O

acesso aos registradores do FM precisam de uma pausa de 56 ciclos e do PCM de 88 ciclos de relógio do OPL4, que é 33,8688 MHz. Os tempos são os seguintes:

Tempo	Ciclos T OPL4	Tempo (μs)	Ciclos T Z80	Ciclos T R800
FM	56	1,65 μs	6 ciclos	12 ciclos
PCM/MIX	88	2,60 μs	10 ciclos	19 ciclos

Entretanto, para habilitar o acesso ao PCM (wave), é necessário setar os bits b0 e b1 no registrador 5 do banco 2 do FM, como ilustrado abaixo:

```
OUT    0C6H,5
OUT    0C5H,00000011B
```

## 6.8 – COVOX

O Covox é um gerador de sons que usa a porta de impressora para reproduzir dados PCM com resolução de 8 bits. Mais detalhes de como funciona podem ser vistos na seção “4 – O PCM”.

A codificação do Covox é também em binário absoluto (complemento de dois), como no PCM e no SCC. Por usar um circuito extremamente simples, entretanto, não há taxas de amostragem (sampling rates) padronizados; a taxa deve ser determinada por temporização através de software.

### 6.8.1 – Acesso ao Covox

Para acessar o Covox, basta enviar os bytes de dados sequencialmente através da porta de I/O da impressora (91H). Não é necessário o uso de pausas entre bytes de dados consecutivos e não é necessário setar nenhum registrador adicional. A seguinte sequência pode ser repetida quantas vezes for necessário, com os dados PCM a partir de (HL):

```
LD    A, (HL)
INC  HL
OUT  (#91),A
```

A temporização, no caso do MSX2 em diante, pode ser obtida pelo VDP, setando a interrupção de linha, o que faz com que o VDP en-

vie sinais de interrupção na frequência de 15,75 KHz, o que permite obter uma boa qualidade sonora. Para o MSX1 é mais complicado, e a temporização deve ser obtida via software, contando os ciclos da CPU.

## ***Capítulo 7***

### **SISTEMAS DE ARMAZENAMENTO DE MASSA**

Grande capacidade de armazenamento de massa externo aliada a alta velocidade de acesso e grande confiabilidade são requisitos necessários a um grande número de aplicações. Esses requisitos são preenchidos por dispositivos de armazenamento de massa (disk-drive, hard disks, ZIP drive, CD-ROM, DVD-ROM, Cartões de Memória, etc). Esses periféricos são normalmente acionados por rotinas do BDOS (Basic Disk Operating System), atualmente conhecido como Kernel. No caso do MSX, o acesso direto a esses dispositivos não é recomendado, uma vez que cada fabricante tem liberdade para escolher qualquer tipo de controlador para o sistema de disco. Os acessos devem ser feitos através do BDOS ou da BIOS, através das rotinas PHYDIO e FORMAT.

Atualmente, há vários sistemas de disco disponíveis para o MSX: o MSXDOS, o MSXDOS2, o Nextor, o UZIX, o SymbOS e o WiOS, sendo que estes dois últimos usam interfaces gráficas.

O MSXDOS necessita de 64 Kbytes de RAM e pode acessar até seis drives simultâneos, designados por A: a E:, mas é muito simples. Embora possa ser conectado a HD's, o controle dos arquivos é sofrível pelo fato de não existirem subdiretórios.

Já o MSXDOS2 necessita de 128 Kbytes de memória mapeada e aceita até 8 drives simultâneos, de A: a H:, sendo que o drive H: é configurado como RAMDISK. Esse sistema tem subdiretórios, e pode ser facilmente configurado para o uso com HD's. O Nextor é uma evolução do MSXDOS2 e possui acesso nativo à FAT16, permitindo partições de até 2 ou 4 Gigabytes (A FAT12 só permite partições de até 32 Mbytes).

O UZIX é um sistema baseado no UNIX. Requer um mínimo de 256 Kbytes para funcionar bem, tem subdiretórios, é multitarefa e multiusuário e foi desenvolvido especialmente para ser usado com HD's, mas usa um sistema de arquivos diferente do MSXDOS e MSXDOS2, chamado "ext". Acessa partições de até 32 Mbytes.

O SymbOS é um sistema gráfico multitarefa e usa FAT como sistema de arquivos, mas é incompatível com programas MSXDOS. Necesita de um mínimo de 128 Kbytes, acessa diretamente até 1 Mbyte de

memória. Opcionalmente, pode também trabalhar com o V9990 para gráficos otimizados.

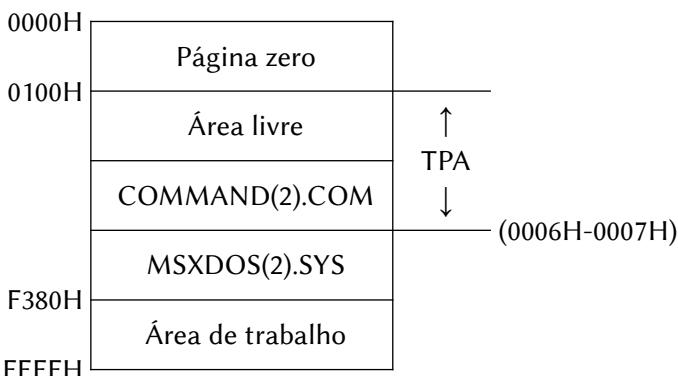
Já o WiOS é um sistema bastante específico que só roda no V9990. Necessita de 160 Kbytes de memória e é multitarefa, mas trabalha em cima do Kernel do MSXDOS2.

O padrão de formatação física dos dispositivos de disco do MSX é igual para todos os sistemas. Eles estão ilustrados abaixo.

ID da mídia	1DD 3½ F8H	2DD 3½ F9H	1DD 5¼ FCH	2DD 5¼ FDH	HD F0H
Número de lados	1	2	1	2	-
Trilhas por lado	80	80	40	40	-
Setores por trilha	9	9	9	9	63
Bytes por setor	512	512	512	512	512

## 7.1 – MSXDOS, MSXDOS2 e NEXTOR

O MSXDOS (1 ou 2) consiste nos seguintes módulos: interface de disco com BDOS (Kernel) em ROM e dos arquivos MSXDOS.SYS e COMMAND.COM (para o MSXDOS2, são MSXDOS2.SYS e COMMAND2.COM). O sistema de disco do MSX difere de outros sistemas pelo fato de que o DOS propriamente dito não se encontra no disco de sistema, mas sim na ROM da interface de disco, tanto que o Disk-BASIC não necessita de nenhum disco no drive para funcionar. Os arquivos MSXDOS.SYS e MSXDOS2.SYS servem simplesmente como uma espécie de boot para setar os parâmetros necessários para o funcionamento do COMMAND.COM ou COMMAND2.COM, que são os responsáveis pela execução dos comandos do MSXDOS. A ROM da interface de disco inclui rotinas para acionamento do drive, o DOS Kernel e o interpretador do Disk-BASIC, e é situada entre os endereços 4000H e 7FFFH (página 1) para o MSXDOS e MSXDOS2, embora este último possua 4 páginas (64K) que são intercambiadas exclusivamente na página física 1. Depois que o sistema foi carregado na memória, a ROM fica normalmente desligada e toda a RAM fica habilitada, conforme a ilustração abaixo.



A área compreendida entre 0000H e 00FFH é a página-zero (system scratch area) e é de extrema importância para o MSXDOS e para os programas aplicativos. Essa área será descrita com detalhes mais adiante. A área que começa em 0100H e termina no endereço indicado pelos bytes 0006H/0007H da página-zero chama-se TPA (Transient Program Area - Área para Programas Transitórios) e é nela que são carregados os programas que funcionam sob o DOS. O COMMAND.COM é situado na parte superior da TPA e o MSXDOS.SYS inicia no primeiro endereço após a TPA.

### 7.1.1 – O COMMAND.COM

O arquivo COMMAND.COM é o responsável pela execução dos comandos do MSXDOS. Esses comandos podem ser internos, externos ou batch.

Comandos internos são aqueles que residem no próprio COMMAND.COM. Ao serem chamados, são executados imediatamente.

No caso se comandos externos, o COMMAND.COM carrega a rotina do disco (que deve ter obrigatoriamente a extensão .COM) e a coloca na TPA a partir do endereço 0100H, sendo que a execução do comando é iniciada nesse mesmo endereço. Quando a execução do comando externo termina (através de uma instrução RET), o MSXDOS.SYS examina se o COMMAND.COM foi destruído (no caso de rotinas externas muito grandes) e, se necessário, recarrega o COMMAND.COM e lhe passa o controle.

Já os comandos batch (em lote) são uma série de comandos gravados em um arquivo (com a extensão .BAT) que o COMMAND.COM executa um a um, sequencialmente (para o COMMAND 2.41 pode haver desvio condicional). Os comandos presentes num arquivo batch podem ser tanto internos quanto externos, sendo possível até outro comando batch. Nesse caso, o comando batch chamado destrói o comando batch chamador.

### **7.1.2 – O MSXDOS.SYS**

O MSXDOS.SYS é o núcleo do MSXDOS. Ele controla o acesso e a comunicação com os dispositivos de disco. As funções do MSXDOS.SYS são executadas pelo BDOS (Basic Disk Operating System), presente na ROM da interface de disco, que constitui o que é chamado de DOS Kernel. O MSXDOS.SYS é apenas o intermediário entre as operações de I/O requeridas pelo COMMAND.COM ou comandos externos e o DOS Kernel.

### **7.1.3 – O DOS Kernel**

O DOS Kernel contém as rotinas básicas de I/O para acesso aos dispositivos de disco. Ele reside na ROM da interface de disco e executa as funções do BDOS do MSXDOS.SYS. Qualquer sistema que use acesso ao disco pode funcionar perfeitamente usando apenas o DOS Kernel. O DISK-BASIC executa suas operações chamando o DOS Kernel diretamente, não necessitando do disco de sistema.

### **7.1.4 – Estrutura dos arquivos no disco**

As informações sobre a estrutura de dados no disco e como são controladas são importantes para o desenvolvimento de programas que acessam o disco. Essa seção contém todas as informações necessárias para isso.

#### **7.1.4.1 – Setores**

Cada tipo de disco tem um determinado número de trilhas; assim, os disquetes de 5½" tem 40 ou 80 trilhas e os de 3½" tem 80 trilhas. No sistema MSX, cada trilha é dividida em 9 partes de 512 bytes cada, chamadas “setores”. O DOS Kernel considera cada setor como a unidade

de dados básica do disco. Os setores são endereçados por números, a partir de 0, até um máximo que depende da capacidade do disco.

#### **7.1.4.2 – Clusters (Aglomeradoss)**

Embora sejam consideradas unidades de dados básicas do disco, não é por setores que o DOS Kernel controla os dados no disco, mas sim por unidades chamadas “clusters”. Um cluster pode conter um ou mais setores. No caso de disquetes, cada cluster ocupa dois setores. Num HD formatado com FAT12, cada cluster ocupa 16 setores (8 Kbytes) para partições de 32 Mbytes. No caso de FAT16, cada cluster ocupa 64 setores (32 Kbytes) para partições de 2 Gbytes. O Nextor acessa clusters de 64 Kbytes, permitindo até 4 Gbytes por partição.

#### **7.1.4.3 – Divisão de dados no disco**

No MSXDOS, um disco é dividido em 4 áreas principais, mostradas na tabela abaixo. Os dados propriamente ditos são colocados na “área de dados”. O setor de boot é sempre o setor 0, mas os setores de início das outras áreas (FAT, diretório e área de dados) difere conforme o tipo de disco. Essas informações estão contidas no DPB.

Setor de boot: programa de inicialização do MSXDOS e informações.

FAT: controle físico e lógico da área de dados.

Diretório: informações sobre os arquivos na área de dados.

Área de dados: área para dados do usuário.

↑ Disco inteiro ↓	Setor de boot	Setor #0
	FAT	Os setores de início e o tamanho dessas áreas devem ser obtidos no DPB
	Diretório	
	Área de dados	Último setor

#### **7.1.4.4 – O setor de boot e o DPB**

A sigla DPB vem do inglês “Drive Parameter Block”, ou Bloco de Parâmetros do Drive. Para cada drive conectado, o MSXDOS aloca um DPB na RAM. As informações contidas no DPB são originalmente copia-

das do setor de boot do disco durante a inicialização, embora alguns dados sejam diferentes entre o setor de boot e o DPB.

Na tabela abaixo estão descritos os conteúdos do setor de boot e do DPB.

offset	Setor de boot
0BH/0CH	Tamanho de um setor (em bytes).
0DH	Tamanho de um cluster (em setores).
0EH/0FH	Número de setores reservados.
10H	Número de FAT's.
11H/12H	Número de entradas do diretório raiz.
13H/14H	Número de setores do disco.
15H	Identificação do tipo de disco.
16H/17H	Tamanho da FAT (em setores).
18H/19H	Número de setores por trilha.
1AH/1BH	Número de faces do disco.
1CH/1DH	Número de setores ocultos.
offset	DPB
+0	Número do drive (0=A; 1=B, etc).
+1	Identificação do tipo de disco.
+2/+3	Tamanho do setor em bytes.
+4	Máscara do diretório.
+5	Tamanho do diretório em setores.
+6	Máscara do cluster.
+7	Tamanho do cluster em setores.
+8/+9	Primeiro setor da FAT.
+10	Número de FAT's.
+11	Número de entradas do diretório raiz.
+12/+13	Primeiro setor da área de dados.
+14/+15	Total de clusters do disco + 1.
+16	Número de setores por FAT.
+17/+18	Primeiro setor da área do diretório.
+19/+10	Endereço da FAT na RAM.

Para acessar as informações do DPB, pode ser usada a função 1BH do BDOS, que, entre outros dados, traz o endereço do DPB na RAM.

#### 7.1.4.5 – O FIB (MSXDOS2)

A sigla FIB vem do inglês “File Info Block” (Bloco de Informações sobre o Arquivo). Ele só existe para o MSXDOS2 e é usado para operações mais complexas, como procurar diretórios de arquivos desconhecidos ou subdiretórios. É uma área de 64 bytes na RAM que contém informações sobre as entradas de diretórios ou de determinados arquivos ou subdiretório. Para obter as informações do FIB, devem ser usadas as funções 40H, 41H ou 42H do MSXDOS2.

offset	Informações do FIB.
+0	Sempre FFH.
+1/+13	Nome do arquivo em ASCII.
+14	Byte de atributos do arquivo.
+15/+16	Hora da última modificação do arquivo.
+17/+18	Data da última modificação do arquivo.
+19/+20	Cluster inicial do arquivo.
+21/+24	Tamanho do arquivo.
+15	Número do drive lógico.
+26/+63	Informações internas (não modificar).

O byte FFH no início serve para distinguir o FIB de uma string de caminho/nome de arquivo. Os dados do FIB são armazenados no mesmo formato dos dados do diretório. Eles estão detalhados na seção “DIRETÓRIO”, mais adiante.

#### 7.1.4.6 – A FAT (File allocation table)

A sigla FAT vem do inglês “File Allocation Table”, ou “Tabela de Alocação de Arquivos”. Ela é uma espécie de mapa do disco. No MSXDOS, o cluster é a unidade básica de dados no disco. Para arquivos grandes, são usados vários clusters a fim de armazená-los. Porém, se vários arquivos são criados e apagados, ficam clusters vazios entre os arquivos não apagados. Quando um arquivo maior é criado, ele é dividido em várias partes e estas são gravadas nos clusters disponíveis. É necessário, então, um meio para se saber quantos e quais clusters estão disponíveis e em quantos e em quais clusters está o arquivo desejado. Essa é a função da FAT.

Quando um cluster defeituoso é encontrado, a FAT também é usada para registrá-lo e impedir o acesso a ele. As informações sobre os clusters, inclusive os defeituosos, é necessária para o manuseio dos arquivos no disco. Sem essa informação, o disco fica inutilizado. Por isso é que existem duas FAT's, caso ocorra algum problema com uma, existe a outra.

Atualmente, existem dois tipos de FAT suportados pelo MSX: a FAT12 e a FAT16.

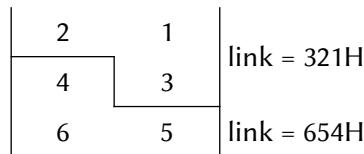
### FAT12

A estrutura da FAT12 está ilustrada logo abaixo. O primeiro byte é chamado de FAT ID e indica o tipo de disco (o mesmo valor contido no setor de boot e no DPB). Os próximos dois bytes contêm o “dummy”. A partir do quarto byte (endereço inicial + 3), a informação sobre os clusters (link) é gravada em um formato irregular de 12 bits por cluster. Cada grupo de 12 bits é chamado de “entrada da FAT”. O número de entrada da FAT é o número do cluster correspondente no disco.

Endereço inicial →	4 bits	4 bits	
	F	9	FAT ID (80 trilhas, 9 setores)
	F	F	Dummy
	F	F	Dummy
	0	3	
	4	0	1 <sup>a</sup> entrada FAT – link = 003H
	0	0	2 <sup>a</sup> entrada FAT – link = 004H
	F	F	
	6	F	3 <sup>a</sup> entrada FAT – link = FFFH (fim)
	0	0	4 <sup>a</sup> entrada FAT – link = 006H
	F	F	etc.

A informação “link” indica o próximo cluster do arquivo correspondente. O exemplo acima mostra um arquivo que ocupa dois clusters

(003H e 004H). Quando o valor “link” for FFFFH, significa que o arquivo terminou. Na prática, os números “link” não ficam necessariamente em ordem numérica. A ilustração abaixo mostra como os números “link” são organizados na FAT.



Como temos 12 bits, teoricamente a FAT12 poderia endereçar até 4096 clusters ( $2^12$ ). Entretanto, só pode endereçar um máximo de 4079 clusters. Isso porque clusters numerados de FF0H a FFFFH tem significado especial, conforme a tabela abaixo.

Link	Significado
000H	Cluster disponível (não usado).
002H a FEFH	Usado; indica o próximo cluster.
FF0H a FF6H	Clusters reservados.
FF7H	Cluster danificado.
FF8H a FFFFH	Usado; último cluster do arquivo.

A FAT12 é bastante eficiente para mapear dados em disquetes. Entretanto, ela limita o acesso a disco em 32 Mbytes. Acima disso, é necessário criar mais partições. Essa limitação ocorre porque, tanto no setor de boot quanto no DPB o número de setores do disco é especificado em dois bytes, totalizando um máximo de 65 536 setores no disco. Assim, como cada setor tem 512 bytes, pode-se fazer  $65\,536 * 512$ , o que dá 32 Mbytes. Para que a FAT possa endereçar esse total, cada cluster deve ter 8 Kbytes, o que propicia um certo desperdício de espaço no disco. Para poder acessar partições maiores, deve ser usada a FAT16.

## FAT16

Como o próprio nome diz, a FAT16 usa 16 bits para endereçar os clusters. A organização da FAT16 está ilustrada abaixo.

	4 bits	4 bits	
Endereço inicial →	F	0	FAT ID
	F	F	Dummy
	F	F	Dummy
	1	2	
	3	4	1 <sup>a</sup> entrada FAT – link = 1234H
	1	2	
	3	5	2 <sup>a</sup> entrada FAT – link = 1235H
	F	F	
	F	F	3 <sup>a</sup> ent. FAT – link = FFFFH (fim)
	0	0	etc.

Como na FAT12, também existem alguns números “link” com significado especial, conforme ilustrado na tabela abaixo.

Link	Significado
0000H	Cluster disponível (não usado).
0001H a FFEFH	Usado; indica o próximo cluster.
FFF0H a FFF6H	Clusters reservados.
FFF7H	Cluster danificado.
FFF8H a FFFFH	Usado; último cluster do arquivo.

A FAT16 no MSXDOS usa o mesmo esquema da FAT16 no PC; por isso, uma partição FAT16 pode ter até 2 Gbytes. Para partições desse tamanho, entretanto, os clusters são enormes (32 Kbytes), o que propicia um grande desperdício de espaço no disco.

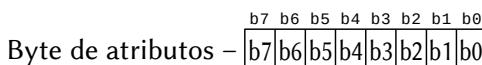
A FAT16 só existe em forma de patch para o MSXDOS2; não está disponível para o MSXDOS1. Para poder utilizá-la, entretanto, é necessário ter uma partição FAT12 para inicializar o sistema e carregar o patch, já que o DOS Kernel só trabalha nativamente com FAT12.

Por sua vez, o Nextor possui suporte nativo para partições FAT16, sendo ideal para trabalhar com armazenamento de massa, pois pode acessar até 4 Gbytes por partição. Entretanto, funciona apenas em interfaces IDE.

#### 7.1.4.7 – O diretório

A FAT, descrita acima, armazena a localização dos dados de um arquivo no disco, mas não contém nenhuma informação sobre o conteúdo do mesmo. Por isso, existe uma seção no disco chamada diretório, onde estão as informações sobre o arquivo. Cada entrada do diretório é composta por 32 bytes que contém o nome, atributos, hora e data da criação do arquivo, além do primeiro cluster e do tamanho do mesmo, conforme ilustrado abaixo.

Offset	Descrição
+0/+7	Nome do arquivo (até 8 caracteres).
+8/+10	Extensão (até 3 caracteres).
+11	Byte de atributos do arquivo.
+12/+21	Reservado (não utilizar).
+22/+23	Hora da criação do arquivo.
+24/+25	Data da criação do arquivo.
+26/+27	Primeiro cluster do arquivo.
+28/+31	Tamanho do arquivo em bytes.



**b0** – se esse bit for 1, o arquivo poderá ser lido mas não apagado ou modificado (somente MSXDOS2)

**b1** – se esse bit for 1, o nome do arquivo não aparecerá no comando DIR ou FILES, mas poderá ser acessado normalmente (MSXDOS1 e MSXDOS2)

**b2** – igual a b1, mas as funções do BDOS não podem apagar ou modificar o arquivo e este não poderá ser acessado pelo COMMAND2.COM. Significa que é um arquivo de sistema (somente MSXDOS2).

**b3** – se esse bit for 1, os 11 bytes do nome do arquivo conterão o nome do disco (volume name) e o restante do diretório será ignorado (somente MSXDOS2).

**b4** – se esse bit for 1, o arquivo é um subdiretório e não poderá ser lido nem escrito normalmente. Quando listado com o comando DIR, aparecerá a expressão “<DIR>” no lugar do tamanho do arquivo (MSXDOS1 e MSXDOS2, mas o MSXDOS1 não poderá acessar o subdiretório).

**b5** – se esse bit for 1, o arquivo não poderá ser fechado antes de ser escrito (somente MSXDOS2).

**b6** – reservado (sempre 0).

**b7** – se esse bit for 1, todos os outros serão ignorados e o FIB apontará para um caractere de dispositivo (ex. “.CON” - entrada de console). Somente MSXDOS2.

## Hora

[23º byte]								[22º byte]							
b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
h4 h3 h2 h1 h0								m5 m4 m3 m2 m1 m0							
Hora (0~23)						Minuto (0~59)						Segundo (0~29)			

Obs.: para obter o valor correto dos segundos, multiplique o valor do registrador por 2. Em FAT 32, os centésimos de segundos podem ser obtidos no byte anterior.

## Data

[25º byte]								[24º byte]							
b7 b6 b5 b4 b3 b2 b1 b0								b7 b6 b5 b4 b3 b2 b1 b0							
a6 a5 a4 a3 a2 a1 a0								m3 m2 m1 m0							
Ano (0~99)						Mês (1~12)						Dia (1~31)			

Obs: para obter o ano correto, some 1980 ao valor do registrador (1980 até 2079).

O primeiro setor do diretório pode ser obtido no DPB respectivo (ou no setor de boot). Quando um arquivo é criado, a entrada respectiva do diretório é colocada na parte livre mais próxima do início do diretório. Cada entrada do diretório é inicialmente preenchida com bytes 00H. Se um arquivo é criado e depois deletado, apenas o primeiro byte da entrada respectiva no diretório é modificado para E5H. Quando todas as entradas do diretório forem preenchidas, mais arquivos não podem ser criados mesmo que haja espaço disponível no disco. O número de entradas no diretório também pode ser obtido no DPB respectivo.

### **Subdiretórios (MSXDOS2)**

Somente o MSXDOS2 e o NEXTOR podem manipular subdiretórios. O subdiretório é um tipo especial de arquivo cuja estrutura é idêntica à do diretório. Por ser um arquivo, entretanto, não há área reservada para ele; fica na área de dados do disco. Seu funcionamento é extremamente simples: o cluster inicial do diretório aponta para o arquivo que é o subdiretório. O bit b4 do byte de atributos deve ser setado.

Um subdiretório não tem um tamanho fixo e portanto não tem limite de entradas. À medida que mais entradas são adicionadas, o subdiretório vai aumentando de tamanho conforme necessário.

Quando um subdiretório é criado, dois arquivos especiais, que ficam “dentro” dele, são criados simultaneamente: o “.” e o “..”. Esses arquivos servem para sair do subdiretório e voltar para o diretório raiz ou subdiretório anterior. Eles não podem ser apagados ou manipulados.

#### **7.1.5 – Acesso aos arquivos em disco**

Quando se fala em acesso a arquivos, deve-se ter em mente uma sigla: FCB. Essa sigla vem o inglês “File Control Block”, ou “Bloco de Controle de Arquivo”. Toda informação gravada no disco recebe o nome de arquivo. Cada arquivo recebe um nome, composto por até 8 caracteres mais uma extensão opcional de três (Ex. MSXDOS.SYS). O acesso direto ao arquivo usando o diretório e a FAT é muito complexo; por isso existe o FCB. Ele ocupa 37 bytes de memória e basta que o programador especifique o nome do arquivo e o drive para que se possa acessá-lo. O FCB pode estar localizado em qualquer parte da memória, mas normalmente o MSXDOS utiliza o endereço 005CH para armazená-lo. A estrutura do FCB está descrita abaixo.

Offset	Comentários
+0	Número do drive (0=default; 1=A; 2=B; etc).
+1/+11	Nome do arquivo e extensão.
+12/+13	Bloco atual.
+14/+15	Tamanho do registro aleatório em bytes.
+16/+19	Tamanho do arquivo em bytes.
+20/+21	Data (mesmo formato do diretório).
+22/+23	Hora (mesmo formato do diretório).
+24	ID do dispositivo.
+25	Localização do diretório.
+26/+27	Primeiro cluster do arquivo.
+28/+29	Último cluster acessado.
+30/+31	Localização relativa do cluster.
+32	Registro sequencial atual.
+33/+36	Número do registro aleatório.

- Número do drive (00H)

Indica o disk-drive no qual está o disco que contém o arquivo.

- Nome do arquivo (01H a 08H)

O nome do arquivo pode conter até 8 caracteres. Quando tiver menos, os bytes restantes serão preenchidos com espaços (20H).

- Extensão (09H a 0BH)

A extensão do nome do arquivo pode ter até 3 caracteres. Quando tiver menos, os bytes restantes serão preenchidos com espaços (20H). A extensão é opcional.

- Bloco atual (0CH a 0DH)

Indica o número do bloco atual para acesso sequencial (funções 14H e 15H do BDOS).

- Tamanho do registro aleatório (0EH a 0FH)

Especifica o tamanho em bytes da unidade de dados (registro) para leitura ou escrita aleatória (funções 14H, 15H, 21H, 27H e 28H do BDOS).

- Tamanho do arquivo (10H a 13H)

Contém o tamanho do arquivo em bytes.

- Data (14H a 15H)

Data do último acesso ao arquivo. O formato é igual ao do diretório.

- Hora (16H a 17H)

Hora do último acesso ao arquivo. O formato é igual ao do diretório.

- ID do dispositivo (18H)

Quando um periférico é aberto como um arquivo, o valor listado abaixo é especificado nesse byte. Para arquivos normais, o valor desse campo é de 40H + número do drive. Por exemplo, o byte ID do drive A: é 41H. Para futuras expansões, programas aplicativos não devem usar o byte ID.

#### byte ID Dispositivo

FFH	CON	(console ou teclado)
FEH	AUX	(auxiliar)
FDH	NUL	(nulo)
FCH	LST	(listar na impressora)
FBH	PRN	(impressora)

- Localização do diretório (19H)

Indica a posição de entrada no diretório do arquivo.

- Primeiro cluster do arquivo (1AH a 1BH)

Contém o número do primeiro cluster do arquivo no disco.

- Último cluster acessado (1CH a 1DH)

Contém o número do último cluster acessado.

- Localização relativa do cluster (1EH a 1FH)

Indica a localização relativa do último cluster acessado a partir do primeiro cluster do arquivo.

- Registro sequencial atual (20H)

Contém o número ao registro atual para acesso sequencial (funções 14H e 15H do BDOS).

- Número do registro aleatório (21H a 24H)

Contém o número registro aleatório a ser acessado. Especificando o valor de 1 a 63 para o tamanho do registro, todos os 4 bytes, de 21H a 24H são usados. Quando o tamanho do registro for maior que 63 apenas os

bytes de 21H a 23H têm significado (funções 14H, 15H, 21H, 22H, 27H e 18H do BDOS).

#### **7.1.5.1 – Abrindo um arquivo**

Antes de acessar um arquivo, é necessário abri-lo. “Abrir um arquivo” significa transformar uma informação incompleta contida do FCB (apenas nome do arquivo e número do drive) em todas as informações que o FCB pode conter.

Ao abrir um arquivo, o número do drive no FCB é convertido para drive real (1 a 6 para MSXDOS1 ou 1 a 8 para MSXDOS2) e os outros campos do FCB são preenchidos (função 0FH do BDOS).

#### **7.1.5.2 – Fechando um arquivo**

Quando um arquivo é aberto, a informação contida no diretório é transferida para o FCB. Durante o manuseio do arquivo, o conteúdo dos campos do FCB vão sendo modificados. Por isso, após ter completado o manuseio do arquivo, é necessário fechá-lo. A operação de fechar um arquivo faz com que a informação contida no FCB volte para o diretório atualizada, a fim de possibilitar acessos posteriores (função 10H do BDOS).

#### **7.1.5.3 – Acesso sequencial e aleatório**

No acesso aleatório, os registros que compõem o arquivo podem ser acessados livremente, sem qualquer padrão estabelecido, ao contrário do acesso sequencial, onde os registros são acessados um após o outro, impreterivelmente. O tamanho do registro pode ser qualquer um, desde que seja maior ou igual a um byte até o limite de 64 Kbytes. O registro pode ter, inclusive, o tamanho do arquivo inteiro (acesso sequencial extremo) ou de apenas um byte (acesso aleatório extremo). O valor padrão para o tamanho do registro é 128 bytes. Abaixo está ilustrado um arquivo com seus respectivos registros.

↑ Arquivo inteiro ↓	Registro #0	Tamanho de um registro
	Registro #1	
	Registro #2	
	Registro #3	
	:	
	Registro #n	

#### 7.1.5.4 – Headers (cabeçalhos)

Para que o sistema de disco possa reconhecer, carregar e executar (se for o caso) corretamente os arquivos ou programas armazenados no disco, esses normalmente contêm um header (cabeçalho).

O header varia conforme o tipo de arquivo. Os diversos tipos de header estão descritos abaixo.

#### Arquivos binários

Os arquivos binários contêm um header de 7 bytes cuja estrutura é a seguinte:

offset	Conteúdo
0	Tipo de arquivo (FEH = binário).
1–2	Endereço inicial dos dados na RAM.
3–4	Endereço final dos dados na RAM.
5–6	Endereço de execução (para arquivos executáveis).

Esse tipo de arquivo é usado pelo BASIC para manipular blocos de dados diretamente na RAM ou VRAM e também para salvar, carregar e executar programas assembly.

#### Arquivos de texto BASIC

Os programas BASIC são salvos no disco precedidos por um byte FFH. O formato dos dados após esse byte é idêntico ao texto tokenizado armazenado na RAM. O texto BASIC também pode ser salvo no formato ASCII.

## **Arquivos ASCII e texto**

Estes arquivos não têm header. O fim de linha normalmente é indicado pela combinação dos bytes 0DH+0AH (carriage return e line feed). O final do arquivo ASCII deve ser marcado com um byte 1AH (EOF – end of file). Arquivos .BAT (batch) do DOS são arquivos texto.

## **Arquivos ‘.COM’**

Os arquivos executáveis do CP/M e do MSXDOS (extensão ‘.COM’) não têm header e nenhum formato específico. Eles são carregados e executados sempre no endereço 0100H. Por causa da compatibilidade com o sistema de arquivo do CP/M, o tamanho dos arquivos CP/M deve ser múltiplo de 80H.

## **Outros arquivos**

Para outros tipos de arquivo, não há nenhum formato particular. São reconhecidos exclusivamente pela extensão de seu nome.

### **7.1.5.5 – Arquivos handle (MSXDOS2)**

Um arquivo handle (manipulador) nada mais é que um número que o usuário associa a um dispositivo ou arquivo comum. O valor de um arquivo handle pode variar de 0 a 63. Usando apenas o número handle como referência, pode-se manipular o arquivo ou dispositivo a ele associado. Esse tipo de arquivo só é suportado pelo MSXDOS2 através de funções acrescentadas ao BDOS, como 43H, 44H, 45H, 53H e outras.

A área de memória interna usada pelos arquivos handle é aloca- da em uma página lógica (16K) fora da área da TPA, não reduzindo, por tanto, o tamanho desta.

Os arquivos handle de 0 a 4 são pré-definidos, como descrito abaixo.

- 0 – Entrada standard (CON)
- 1 – Saída standard (CON)
- 2 – Entrada/saída standard de erro (CON)
- 3 – Entrada/saída auxiliar standard (AUX)
- 4 – Saída standard para impressora (PRN)

### 7.1.6 – As funções do BDOS

O BDOS consiste em um conjunto de rotinas que fazem as operações básicas de I/O para os dispositivos de disco. Essas rotinas permitem fácil acesso ao sistema de disco e residem na ROM da interface de disco. Também são conhecidas como DOS Kernel.

As funções do BDOS estão disponíveis tanto para o MSXDOS quanto para o Disk-BASIC, variando apenas o endereço de chamada:

MSXDOS:	0005H
Disk-BASIC:	F37DH (&HF37D)

Para executar as funções do BDOS, basta simplesmente fazer o seguinte:

1. Carregar o registrador C da CPU com o número da função desejada;
2. Carregar os registradores A, B, DE e HL (se necessário) com os valores adequados;
3. Fazer uma chamada (CALL) para o endereço do BDOS (0005H para o MSXDOS e F37DH para o Disk-BASIC).

O exemplo a seguir ilustra uma chamada à função 1FH do BDOS.

```
LD    A,000H ;carrega A com o valor 00H
LD    C,01FH ;carrega C com a função número 1FH
CALL  00005H ;executa a função
```

É importante ressaltar que as chamadas para o BDOS destroem o conteúdo dos registradores. Portanto, antes de chamar alguma função, o conteúdo dos registradores que não devem ser modificados deve ser salvo.

Existem 44 chamadas para o BDOS no caso do MSXDOS1 e 94 para o MSXDOS2 (que inclui todas as funções do MSXDOS1). As funções são numeradas de 00H a 70H, mas existem algumas que não estão implementadas: 1CH a 20H, 25H, 29H e 32H a 3FH. Uma chamada a essas funções apenas retorna o valor 0 no registrador A.

Sempre que se for acessar os dispositivos de disco, é aconselhável usar as funções do BDOS. O acesso direto é bantante complicado, já que cada fabricante pode usar o controlador que melhor lhe convier e os programas podem não funcionar em interfaces diferentes.

As funções do BDOS estão detalhadamente descritas no apêndice (Capítulo 4 – MSXDOS, item 4.3 – Chamadas para o BDOS)

### **7.1.6.1 – Leitura/escrita absoluta de setores**

O MSX acessa disco através de ‘setores lógicos’. Eles são definidos independentemente dos setores físicos do disco e são numerados de 0 até um máximo que depende da capacidade do disco:

40 trilhas, 1 face: 0 a 359

40 trilhas, 2 faces: 0 a 719

80 trilhas, 1 face: 0 a 719

80 trilhas, 2 faces: 0 a 1439

Partição de 32 Mb: 0 a 65535 (2 bytes)

Partição de 2 Gb: 0 a 4.294.967.296 (4 bytes, só FAT16)

As funções do BDOS descritas abaixo acessam diretamente os setores lógicos do disco.

#### **RDABS (2FH)**

Função: Leitura de setores lógicos do disco. Os setores lidos são colocados a partir da DTA.

Entrada: DE – Número do primeiro setor lógico a ler;  
H – Número de setores a ler;  
L – Número do drive (0=A; 1=B; etc.).

Retorno: A = 0 → Leitura bem-sucedida;  
A ≠ 0 → Código de erro.

Obs: Essa função lê os setores continuamente até atingir o total especificado no registrador H ou detectar algum erro.

#### **WRABS (30H)**

Função: Escrita de setores lógicos no disco. Os dados a serem escritos no disco serão lidos na RAM a partir da DTA.

Entrada: DE – Número do primeiro setor lógico a ser escrito;  
 H – Número de setores a escrever;  
 L – Número do drive (0=A:, 1=B:, etc.).

Retorno: A = 0 → Escrita bem-sucedida;  
 A ≠ 0 → Código de erro.

Para acessar partições maiores de 32 Mbytes é necessário ter instalado o driver de FAT16 ou o Nextor, que possui suporte nativo. Este último implementa duas funções para acesso direto:

#### RDDRV (73H)

Função: Ler os setores absolutos da unidade. Esta função é capaz de ler setores independentemente do sistema de arquivos visualizado na unidade (FAT12, FAT16 ou um sistema de arquivos desconhecido), e mesmo quando não há nenhum sistema de arquivos. Os setores lidos serão colocados a partir da DTA do disco atual.

Entrada: A – Número da unidade (0 = A:, 1=B:, etc.).  
 B – Número de setores a ler.  
 HL:DE – Número do setor.

Saída: A – Código de erro (se for 0, não houve erro).

#### WRDRV (74H)

Função: Gravar setores absolutos no disco. Esta função grava setores independentemente do sistema de arquivos na unidade (FAT12, FAT16 ou um sistema de arquivos desconhecido), e também quando não há nenhum sistema de arquivos. Os setores serão gravados a partir da DTA do disco atual.

Entrada: A – Número da unidade (0 = A:, 1=B:, etc.).  
 B – Número de setores a escrever.  
 HL:DE – Número do setor.

Saída: A – Código de erro (se for 0, não houve erro).

#### 7.1.6.2 – Acesso aos arquivos usando o FCB

Acessar os arquivos do disco usando acesso direto aos setores lógicos é um processo muito complicado. As funções do BDOS que acesam o disco usando o FCB tornam essas operações mais simples.

Existem três categorias de acesso a arquivos usando o FCB: acesso sequencial, acesso aleatório e acesso aleatório em blocos. Esse último tipo possui as seguintes facilidades: registros de qualquer tamanho podem ser especificados; o acesso pode ser feito em múltiplos registros e o tamanho do arquivo é controlado em bytes.

Uma informação importante é que algumas funções não funcionam corretamente quando o FCB estiver situado entre os endereços 4000H e 7FFFH (MSXDOS1 e MSXDOS2): função SFIRST (11H), função SNEXT (12H) e as funções de I/O para dispositivos (CON, PRN, NUL, AUX).

As seguintes funções permitem acesso aos arquivos usando o FCB:

FOPEN	(0FH)	Abrir arquivo (FCB).
FCLOSE	(10H)	Fechar arquivo (FCB).
SFIRST	(11H)	Procurar o primeiro arquivo. Esta função aceita caracteres coringa (* e ?).
SNEXT	(12H)	Procurar o próximo arquivo. Esta função aceita caracteres coringa (* e ?).
FDEL	(13H)	Apagar arquivos. Caracteres coringa (*) e (?) podem ser usados.
RDSEQ	(14H)	Leitura sequencial.
WRSEQ	(15H)	Escrita sequencial.
FMAKE	(16H)	Criar arquivos.
FREN	(17H)	Renomear arquivos. O caractere coringa "?" pode ser usado para renomear vários arquivos simultaneamente.
RDRND	(21H)	Leitura aleatória. O registro lido será colocado na área indicada pela DTA e tem o tamanho fixo de 128 bytes.
WRRND	(22H)	Escrita aleatória.
FSIZE	(23H)	Ler o tamanho do arquivo.
SETRND	(24H)	Setar campo do registro aleatório.
WRBLK	(26H)	Escrita aleatória em bloco. O número do registro aleatório é automaticamente incrementado depois da escrita, e seu tamanho pode variar de 1 até 65 535 bytes.

RDBLK	(27H)	Acesso aleatório em bloco.
WRZER	(28H)	Escrita aleatória com bytes 00H. Esta função é igual à 22H (WRRND), exceto pelo fato de preencher os registros restantes do arquivo com bytes 00H, se o registro especificado não for o último do arquivo.

As funções abaixo foram acrescentadas pelo MSXDOS2 e não estão disponíveis para a primeira versão do MSXDOS.

FFIRST	(40H)	Procura primeira entrada no diretório.
FNEXT	(41H)	Procura próxima entrada do diretório. Esta função só deve ser usada após a função 40H. Ela aceita os caracteres coringa "?" e "*" definidos em 40H.
FNEW	(42H)	Procura nova entrada
OPEN	(43H)	Abre arquivo handle. Se o bit "herdável" de A estiver setado, o arquivo handle deverá ser aberto por outro processo (ver função 60H).
CREATE	(44H)	Criar arquivo handle. O arquivo criado por esta função será automaticamente aberto (função 43H)
CLOSE	(45H)	Fechar arquivo handle.
ENSURE	(46H)	Proteger arquivo handle (o apontador do arquivo corrente não poderá ser modificado).
DUP	(47H)	Duplicar arquivo handle.
READ	(48H)	Ler de um arquivo handle. As quatro sequências de controle (Ctrl+P, Ctrl+N, Ctrl+S, Ctrl+C) são checadas.
WRITE	(49H)	Escrever por um arquivo handle. Se o fim de arquivo for encontrado, ele será estendido até o valor necessário.
SEEK	(4AH)	Mover apontador do arquivo handle.
IOCTL	(4BH)	Controle para dispositivos de I/O.
HTEST	(4CH)	Testar arquivo handle.
DELETE	(4DH)	Apagar arquivo ou subdiretório. Um subdiretório só poderá ser apagado se não contiver nenhum arquivo. Se um nome de dispositivo for especificado, não retornará erro, mas o dispositivo não será "apagado".
RENAME	(4EH)	Renomear arquivo ou subdiretório.
MOVE	(4FH)	Mover arquivo ou subdiretório. Um arquivo não poderá ser movido se o arquivo handle respectivo

		estiver aberto. O FIB do arquivo movido não será atualizado.
ATTR	(50H)	Setar ou ler atributos de um arquivo. Os atributos de um arquivo não podem ser modificados se o arquivo handle correspondente estiver aberto.
FTIME	(51H)	Ler ou definir data e hora em um arquivo.
HDELET	(52H)	Apagar arquivo handle. Se houver outro arquivo handle aberto para o mesmo arquivo, esse não poderá ser apagado.
HRENAM	(53H)	Renomear arquivo handle. O arquivo não poderá ser renomeado se houver outro arquivo handle aberto para o mesmo arquivo. Esta função é idêntica à função 4EH, exceto pelo fato do registrador HL não poder apontar para um FIB.
HMOVE	(54H)	Mover arquivo handle. O arquivo não poderá ser movido se houver outro arquivo handle aberto para o mesmo arquivo. Esta função é idêntica à função 4FH, exceto pelo fato do registrador HL não poder apontar para um FIB.
HATTR	(55H)	Ler ou setar atributos do arquivo handle. O byte de atributos não poderá ser modificado se houver outro arquivo handle aberto para o mesmo arquivo.
HFTIME	(56H)	Ler ou alterar hora e data do arquivo handle. Se houver outro arquivo handle aberto para o mesmo arquivo, a data e a hora não poderão ser modificadas. Esta função é idêntica à função 51H, exceto pelo fato de não haver apontador; somente o arquivo handle.
GETDTA	(57H)	Ler o endereço da DTA (Disk Transfer Area).
GETVFY	(58H)	Ler flag de verificação de escrita.
GETCD	(59H)	Ler diretório ou subdiretório corrente.
CHDIR	(5AH)	Trocar o subdiretório corrente.
PARSE	(5BH)	Analisa pathname (nome do caminho).
PFILE	(5CH)	Analizar nome do arquivo.
CHKCHR	(5DH)	Checa caractere. Caracteres de 16 bits também são checados.

WPATH	(5EH)	Ler string path completa, sem especificação de drive e caractere “\”. Para maior confiabilidade, chamar primeiro a função 40H ou 41H e depois chamar WPATH duas vezes, já que outras funções podem alterar os dados.
FLUSH	(5FH)	Descarregar buffers de disco.
FORK	(60H)	Ramificar arquivos em árvore.
JOIN	(61H)	Reunir arquivos em árvore. Esta função retorna para o arquivo handle original o arquivo handle que foi copiado pela função anterior.
TERM	(62H)	Finalizar com código de erro.

Estas rotinas (bem como todas as outras) estão detalhadamente descritas no apêndice (Capítulo 4, item 4.3 – Chamadas para o BDOS).

### 7.1.7 – Área de sistema para o MSXDOS

A área de sistema de disco, tanto para o MSXDOS1 quanto para o MSXDOS2, ocupa uma boa parte de memória logo abaixo da área de trabalho do sistema (que inicia em F380H). O MSXDOS1 ocupa mais memória nessa área porque copia a FAT do disco que está sendo utilizado e também do drive virtual B:. Por isso, ao pressionar a tecla CTRL durante o reset, desativando o drive B:, há um aumento de 1,5 Kbytes na memória disponível. Já o MSXDOS2 copia a FAT em outra área de memória, e a economia ao se desativar o drive B: é de apenas 21 bytes, referente ao DPB respectivo.

Assim, um MSX padrão sem interface de disco pode usar até o endereço F08FH. O espaço entre F08FH e F380H é usado...

Um MSX carregado com MSXDOS e dois drives pode ser usado até por volta de DB80H, com alguma variação para cima ou para baixo dependendo da interface. Se for pressionada CTRL no boot para liberar uma unidade de drive, o valor sobe para cerca de E160H.

Com o MSXDOS2, o endereço é por volta de E240H com dois drives, decrescendo apenas 21 bytes para cada unidade instalada.

Em qualquer caso, pode ser consultada a variável de sistema HI-MEM (FC4AH, 2) que armazena o endereço mais alto disponível.

As variáveis do sistema do MSXDOS e MSXDOS2 estão descritas com detalhes no apêndice, capítulo 7, itens 7.1 e 7.2 (Área de sistema para o MSXDOS e MSXDOS2)

### 7.1.8 – Rotinas da interface de disco

Existem algumas rotinas do BDOS que são chamadas diretamente da interface de disco. Essas rotinas possuem sua entrada na página 1, e por isso não é aconselhável chamá-las diretamente, pois sob o MSXDOS a página 1 contém RAM e sob o BASIC contém a ROM do interpretador. Portanto a ROM do DOS Kernel nunca estará ativa normalmente.

Assim, as rotinas da interface devem ser chamadas pela rotina CALSLT da BIOS, que está ativa normalmente tanto sob o MSXDOS quanto sob o BASIC. A sequência de chamada deve ser a seguinte:

```
CALSLT: EQU 0001CH ;endereço da rotina CALSLT
HPHYD: EQU OFFA7H ;end. hook da rotina PHYDIO
CALBAS: EQU 04022H ;endereço da rotina CALBAS
        LD IX,CALBAS ;IX <- end. da rotina CALBAS
        LD IY,HPHYD ;IY <- slot da interface
        CALL CALSLT ;executa a rotina CALBAS
```

Para usar a rotina CALSLT é necessário saber o slot onde a interface está instalada. Um ponto seguro para obter essa informação são os hooks dos comandos de disco. Eles contêm a identificação do slot da interface primária de disco em seu segundo byte. No caso, foi utilizado o hook da rotina PHYDIO da BIOS.

As rotinas da interface de disco estão detalhadamente descritas no apêndice, capítulo 8, item 8.6 (Rotinas da interface de disco). Todos os registradores são modificados pelas rotinas; portanto é necessário salvar na pilha os registradores que não devem ser modificados.

### 7.1.9 – A página-zero

A página-zero é a área de memória situada entre os endereços 0000H e 00FFH da RAM, ocupando 256 bytes. Essa área só é ativa sob o MSXDOS e é de extrema importância para os programas aplicativos. Algumas rotinas da BIOS estão disponíveis nessa área. Elas devem ser chamadas exatamente como se faz para a BIOS (instrução CALL ou RST), exceto a rotina WBOOT (0000H), que deve ser chamada com um JP 0000H. A página-zero é mapeada como descrito abaixo.

#### WBOOT (0000H, 3)

Warm boot. Ao se chamar essa rotina, promove-se uma partida a quente do MSXDOS (o MSXDOS é recarregado sem reiniciar o micro).

#### DRIVE (0004H, 1)

Esse byte armazena o drive default (00H=A:, 01H=B:, etc)

#### BDOS (0005H, 3)

Ponto de entrada das rotinas do BDOS.

#### RDSLT (000CH, 8)

Essa rotina lê um byte em qualquer slot. É exatamente igual à rotina RDSLT da BIOS.

#### WRSLT (0014H, 8)

Essa rotina escreve um byte em qualquer slot. É exatamente igual à rotina WRSLT da BIOS.

#### CALSLT (001CH, 8)

Chama uma rotina em qualquer slot. No presente caso, pode ser usada para chamar outras rotinas da BIOS. É exatamente igual à rotina CALSLT da BIOS.

#### ENASLT (0024H, 8)

Habilita uma página em qualquer slot. É exatamente igual à rotina RDSLT da BIOS.

#### CALLF (0030H, 8)

Chama uma rotina em qualquer slot, com parâmetros em linha. No presente caso, pode ser usada para chamar outras rotinas da BIOS. É exatamente igual à rotina CALLF da BIOS.

**INTPRT (0038H, 3)**

Chama a rotina do manipulador de interrupção. Essa entrada não deve ser utilizada pelo programador.

**CHSLTS (003BH, 33)**

Rotina usada pelo sistema para a troca de slots secundários. Essa entrada não deve ser utilizada pelo programador.

**FCBDOS (005CH, 24)**

Essa área contém o FCB usado pelo BDOS.

**DTA (0080H, 00FFH)**

Endereço inicial da DTA.

A área compreendida entre 0080H e 00FFH (DTA) é onde é colocada uma linha coletada pelo COMMAND.COM. Por exemplo, ao ser digitado um comando externo tipo “PROG ABC”, o COMMAND.COM procurará no disco o programa de nome PROG.COM. Se encontrar, o carregará a partir do endereço 0100H. O argumento “ABC” será carregado a partir do endereço 0080H, com a estrutura ilustrada na página seguinte.

0080H – byte 20H (espaço em branco)

0081H – byte 0DH (carriage return)

0082H – “A”

0083H – “B”

0084H – “C”

0085H – byte 0DH (carriage return)

0086H – byte 00H (fim do argumento)

Após carregar o argumento e o programa, a execução deste é iniciada no endereço 0100H. A área que vai de 0100H até o endereço mais alto disponível é conhecida como TPA (Transient Program Area).

### **7.1.10 – O setor de boot**

Em todos os disquetes (e outros dispositivos de disco) existe o “setor de boot”, que sempre é o setor 0 do disco. Toda vez que o micro for reiniciado, o DOS Kernel residente na ROM da interface de disco verifica se há algum disco conectado ao sistema. Em caso negativo, ativa o Disk BASIC; caso contrário, carrega o setor de boot no endereço C000H

(início da página 3 da RAM) e executa a rotina contida a partir do endereço C01EH. Abaixo está ilustrado como o setor de boot fica na memória.

C000H – Byte ID (55H para disquetes)

C001H~C002H – FEH, 90H (Instrução de partida do DOS, usada no boot “a quente” - WBOOT)

C003H~C00AH – Nome do fabricante ou identificação de formatação em ASCII. Pode ser modificado pelo programador.

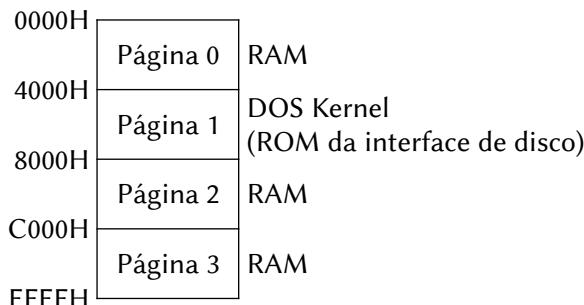
C00BH~C01DH – Dados do setor de boot. Esses dados estão detalhadamente descritos na seção 1.4.4 (O SETOR DE BOOT E O DPB).

C01EH~C0FFH – Rotina de inicialização. Está descrita detalhadamente logo adiante.

C100H~C1FFH – Área reservada. Não deve ser utilizada.

Apesar do setor ter 512 bytes, as instruções contidas no mesmo só podem ter até 256 bytes (C000H a C0FFH), pois logo após a carga do setor de boot o DOS Kernel preenche a área a partir de C100H com rotinas específicas. Se o disco não for um disco de sistema, a página-zero também será preenchida, excetuando a rotina WBOOT (0000H) e a entrada do BDOS (0005H). Nesse caso, deve ser usada a entrada do BDOS em F37DH.

O mapeamento da memória quando da execução do boot está ilustrado logo abaixo.



### 7.1.11 – A rotina de inicialização

Logo após setar todos os dados necessários, o DOS Kernel passa o controle à rotina contida a partir do endereço C01EH. A rotina padrão usada pelo MSXDOS1 original é a seguinte:

```

BDOS:    EQU   0F37DH
C01E      RET   NC
C01F      LD    (BOOT+1),DE
C023      LD    (0C04H,A)
C026      LD    (HL),056H
C028      INC   HL
C029      LD    (HL),0C0H
C02B  BOOT0: LD    SP,0F51FH
C02E      LD    DE,FCBDOS
C031      LD    C,000H
C033      CALL  BDOS
C036      INC   A
C037      JP    Z,BOOT2
C03A      LD    DE,00100H
C03D      LD    C,01AH
C03F      CALL  BDOS
C042      LD    HL,00001H
C045      LD    (0C0ADH),HL
C048      LD    HL,03F00H
C04B      LD    DE,FCBDOS
C04E      LD    C,027H
C050      CALL  BDOS
C053      JP    00100H
C056      LD    E,B
C057      RET   NZ
C058  BOOT1: CALL  00000H
C05B      LD    A,C
C05C      AND  0FEH
C05E      CP    002H
C060      JP    NZ,BOOT3
C063  BOOT2: LD    A,(0C0C4H)
C066      AND  A
C067      JP    Z,04022H
C06A  BOOT3: LD    DE,ERROR
C06D      LD    C,009H

```

```

C06F      CALL  BDOS
C072      LD    C,007H
C074      CALL  BDOS
C077      JR    BOOT0
C079  ERROR:  DEFB  'Boot error',00DH,00AH
              DEFB  'Press any key for retry'
              DEFB  00DH,00AH,024H
C09F  FCBDOS: DEFB  000H,'MSXDOS   SYS'
C0AB      END

```

O restante do setor de boot é preenchido com bytes 00H.

Antes de executar a rotina contida no boot, o DOS Kernel preenche com certos endereços os registradores DE e HL e seta um valor no registrador A.

A rotina de inicialização pode ser modificada pelo programador para adequá-la ao programa que quiser por no disco. Os detalhes a serem observados são os seguintes: deve haver uma instrução RET NC no início; os registradores A, DE e HL contêm um valor válido; o mapeamento quando da execução da rotina contém a ROM do DOS Kernel na página 1 e RAM nas páginas 0, 2 e 3; a área de memória reservada para a rotina de inicialização é de apenas 222 bytes (C01EH~C0FFH).

Os três bytes iniciais do setor de boot (EBH, FEH, 90H) não devem ser modificados pelo programador, apesar do sistema modificar o primeiro byte (EBH no disco) para 55H na memória (disquetes de 720K). Os dados do setor de boot devem estar setados (C00BH~C01DH).

A rotina padrão usada pelo MSXDOS2 original é a seguinte:

```

BDOS:    EQU   0F37DH
C01E      JR    BOOT0
C020      DEFB  'VOL_ID'
C026      DEFB  000H,015H,075H,005H,01BH
C02B      DEFB  000H,000H,000H,000H,000H
C030  BOOT0: RET   NC
C031      LD    (BOOT2+1),DE
C035      LD    (BOOT3+1),A
C038      LD    (HL),067H
C03A      INC   HL
C03B      LD    (HL),0C0H

```

```

C03D  BOOT1:   LD    SP,0F51FH
C040      LD    DE,FCBDOS
C043      LD    C,00FH
C045      CALL BDOS
C048      INC   A
C049      JR    Z,BOOT3
C04B      LD    DE,00100H
C04E      LD    C,01AH
C050      CALL BDOS
C053      LD    HL,00001H
C056      LD    (0C0B9H),HL
C059      LD    HL,03F00H
C05C      LD    DE,FCBDOS
C05F      LD    C,027H
C061      CALL BDOS
C064      JP    00100H
C067      LD    L,C
C068      RET   NZ
C069  BOOT2:   CALL 00000H
C06C      LD    A,C
C06D      AND   0FEH
C06F      SUB   002H
C071  BOOT3:   OR    000H
C073      JP    Z,04022H
C076      LD    DE,ERROR
C079      LD    C,009H
C07B      CALL BDOS
C07E      LD    C,007H
C080      CALL BDOS
C083      JR    BOOT1
C085  ERROR:   DEFB 'Boot error',00DH,00AH
                  DEFB 'Press any key for retry'
                  DEFB 00DH,00AH,024H
C0AB  FCBDOS:  DEFB 000H,'MSXDOS  SYS'
C0B6      END

```

### **7.1.12 – O NEXTOR**

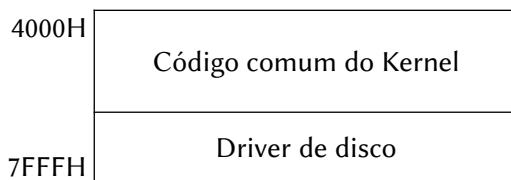
O Nextor é uma versão aprimorada do MSXDOS 2, baseado na versão 2.31, com a qual é 100% compatível.

### 7.1.12.1 – O Kernel MSXDOS 1

O Kernel MSXDOS original (versão 1) era gravado em uma ROM embutida nos controladores de disquete externos ou na ROM interna dos MSX com unidade de disquete. Tem 16K e é localizada na página 1 (endereços 4000H a 7FFFH).

O Kernel MSXDOS 1 é dividido em duas partes principais, o código comum do Kernel e o driver de disco. A parte do código comum do Kernel não é 100% independente do driver.

A ilustração abaixo mostra um diagrama com a estrutura de um Kernel MSXDOS 1.



Podem haver até quatro ROMs de Kernel MSXDOS ativas. Se mais de um estiver presente, aquele com o menor número de slot se torna o “mestre” (aquele cujo código comum do Kernel é realmente executado), e os outros são os “escravos” (apenas o código do driver é executado).

O MSXDOS vê os dispositivos de armazenamento como letras de unidade, enquanto o driver de disco apresenta uma ou mais unidades de driver. O mapeamento entre as duas entidades é fixo e um por um, portanto, por exemplo, a unidade A: é mapeada para a unidade de driver 0 do primeiro Kernel, a unidade B: é mapeada para a unidade de driver 1 e assim por diante.

### 7.1.12.2 – O Kernel do MSXDOS 2

O Kernel do MSXDOS 2 usa o espaço de endereço da página 1 do slot, como o Kernel do MSXDOS 1. No entanto, tem um tamanho de 64K. Este espaço é dividido em quatro bancos de 16K que são trocados durante a execução exclusivamente na página física 1 (4000H ~ 7FFFH). O conteúdo dos bancos é o seguinte:

- O banco 0 contém o código comum do Kernel e do driver de disco.
- Os bancos 1 e 2 contêm o código comum do Kernel.
- O banco 3 contém uma cópia do Kernel MSXDOS 1, com uma cópia do código do driver de disco (somente em máquinas MSX Turbo R).

A ilustração abaixo mostra um diagrama com a estrutura de um Kernel MSXDOS 2.

	Banco 0	Bancos 1 e 2	Banco 3
4000H	Página-zero	Página-zero	
40FFH	ID de banco (0)	ID de banco (1 ou 2)	
4100H	Código Kernel Banco 0	Código Kernel Bancos 1 e 2	Kernel do MSXDOS 1
7FD0H	Driver de disco		Driver de disco
7FFFH	Código de troca de banco	Código de troca de banco	Código de troca de banco

Existem três partes que são comuns a todos os bancos (o banco 3 contém apenas o código de troca do banco):

1. O código da página 0 tem 255 bytes e contém um ponto de entrada para a rotina de interrupção, uma rotina para chamar o código em outro banco e outro código de utilitário útil.
2. O ID do banco é apenas um byte com o número do banco (necessário para fazer chamadas inter-slot).
3. O código de mudança de banco é necessário para mudar o banco visível. O código exato colocado aqui depende do tipo de mapeador ROM usado (o mapeamento do cartucho DOS 2 original é ASCII16).

Ao inicializar no modo DOS 2, o banco 0 é trocado perma-nentemente e outros bancos são trocados apenas temporariamente quando o código do banco 0 precisa chamar uma rotina ou acessar dados em um

desses bancos. Ao inicializar no modo DOS 1, o banco 3 é ativado na inicialização.

Como foi o caso do Kernel MSXDOS 1, até quatro ROMs de Kernel MSXDOS podem estar ativas ao mesmo tempo, sendo um deles o "mestre" e os outros os "escravos". No entanto, desta vez o master não será o Kernel com o menor número de slot, mas o Kernel com o maior número de versão (o Kernel com o menor número de slot ainda é selecionado como o master no caso de dois ou mais Kernels terem a mesma versão número).

### 7.1.12.3 – O Kernel do Nextor

O Kernel do Nextor possui uma arquitetura baseada no Kernel do MSXDOS 2, com as seguintes mudanças:

- Há mais bancos de 16K.
- O código do driver de disco (“driver de dispositivo” na terminologia Nextor) não está mais embutido no final dos bancos de Kernel 0 e 3, mas ocupa um banco inteiro após o último banco de o código comum do Kernel. Se necessário, o driver pode ocupar mais de um banco.
- A estrutura do driver de dispositivo torna mais fácil adicionar comandos estendidos BASIC (comandos “CALL”), comandos BIOS estendidos e uma rotina de serviço de interrupção.
- O código da página 0 foi modificado para conter rotinas de utilitários extras. Essas rotinas podem ser usadas pelo código do driver.
- Há um byte de informação no endereço 4FFEh de todos os bancos, contendo o tamanho do código comum do Kernel em bancos de 16K (alternativamente, este valor pode ser visto como o número do banco do driver).
- O Kernel do MSXDOS 1 no banco 3 foi modificado para que possa realizar chamadas ao driver do dispositivo.
- Há um espaço não utilizado de 1K nos bancos 0 e 3 (visível nos endereços 7BD0H a 7FCFH). Este espaço não contém nenhum código de Kernel e pode ser usado para colocar qualquer código ou dado que seja

exigido pelo driver. Veja o item “O espaço livre no banco principal do Kernel” para mais detalhes.

- Existem cinco pontos de entrada nos bancos 0 e 3 do Kernel (começando nos endereços 7850H) que serão redirecionados para outros cinco pontos de entrada no banco de drivers. Dessa forma, o driver pode fornecer código que estará acessível por meio de chamada direta entre slots para o slot do Kernel. Veja os itens DRV\_DIRECT0 / 1/2/3/4 (4142h, 4145h, 4148h, 414Bh, 414Eh) para mais detalhes.

A ilustração abaixo mostra um diagrama com a estrutura de um Kernel Nextor (“K” é o contador de banco de código comum do Kernel).

	Banco 0-(K-1)	Banco K	Bancos (K+1) ... opc.
4000H	Página-zero	Página zero	Página zero
40FEH	K	K	K
40FFH	ID do banco	K (ID do banco)	ID do banco
4100H	Código Kernel do banco	Código do driver	Código do driver (adicional)
7BD0H	Espaço livre (1K) (bancos 0 e 3)		
7FD0H	Código de troca de banco	Código de troca de banco	Código de troca de banco
7FFFH			

Nextor usará a mesma regra do MSXDOS 2 para decidir qual Kernel será o mestre se mais de um Kernel for encontrado (o Kernel com o maior número de versão vencerá). No entanto, isso se aplica a outros Kernels Nextor apenas; Nextor sempre irá sobrescrever outros Kernels MSXDOS 1 ou 2 presentes no sistema, independente de seu número de versão.

#### 7.1.12.4 – Criação de um Kernel Nextor com driver embutido

Para criar uma ROM de Kernel Nextor completa são necessários até quatro componentes:

- O arquivo base do Kernel Nextor. Este arquivo contém o código comum do Kernel, ou seja, a parte “Banco 0–(K–1)” mostrada na ilustração acima. O código de troca de banco é para o mapeador ASCII16 (o mapeador original usado pelo Kernel MSXDOS 2).
- O arquivo do driver do dispositivo. Sua estrutura está **detalhada na seção 4**. O tamanho deve ter exatos 16 080 bytes (16K menos o tamanho do código da página 0 e do código de troca do banco). Se o driver estiver em mais de um banco, isso se aplica a todos os bancos.
- O arquivo de código de troca de banco (somente se o mapeador a ser usado pelo hardware de destino não for ASCII16). Arquivos de código de troca de banco compilados são fornecidos para mapeadores ASCII8 e ASCII16; para outros tipos de mapeadores devem ser desenvolvidos arquivos de código apropriados. Ver item “Regras para o código de troca de banco”. Mapeadores ROM com bancos de 8K são suportados apenas se for possível selecionar o banco visível na primeira metade da página 1 (4000H~FFFFH) escrevendo um único byte em uma porta mapeada de memória com um LD (xxxx),A. Este é o caso do ASCII8, por exemplo.
- Opcionalmente, o código que pode ser colocado no espaço não utilizado de 1K nos bancos 0 e 3 (Ver o item “O espaço livre no banco principal do Kernel”).

O procedimento para criar o arquivo ROM do Kernel Nextor completo consiste basicamente em anexar o código do driver ao arquivo base do Kernel e, em seguida, corrigir o arquivo resultante com o código de troca de banco apropriado. Isso pode ser feito manualmente ou usando o utilitário MKNEXROM. Ambas as opções são explicadas a seguir.

### **Criação manual**

Para criar manualmente um arquivo Nextor ROM completo, os seguintes passos devem ser seguidos (as posições de arquivo iniciam em zero):

1. Crie uma cópia do arquivo base do Kernel (NEXTOR.BASE.DAT).
2. Anexe o código da página 0 ao final do arquivo. Este código pode ser copiado dos primeiros 255 bytes do próprio arquivo base do Kernel.

3. Anexe um byte com o valor K (a contagem do banco de arquivos base do Kernel) no final do arquivo resultante (este será o ID do banco do driver). O valor de K pode ser lido da posição 254 do próprio arquivo base do Kernel.
4. Anexe o arquivo do driver (que deve ter exatamente 16 080 bytes) no final do arquivo obtido na etapa 3.
5. Anexe o código de troca de banco no final do arquivo resultante. Para um mapeador ASCII16 o código pode ser copiado dos últimos 48 bytes do próprio arquivo base do Kernel.
6. Se o código do driver não couber em um único banco, repita as etapas 2 a 5 para acrescentar bancos extras, aumentando o ID para cada banco acrescentado.
7. Se necessário, corrija o arquivo resultante para adicionar código personalizado ou dados no espaço livre de 1K nos bancos 0 e 3. Coloque o conteúdo do arquivo (até 1K de comprimento) duas vezes, nas posições 3BD0h e FBD0h no arquivo.
8. Se o tipo de mapeador não for ASCII16, faça o patch do código de comunicação dos bancos de código comum do Kernel (os últimos 48 bytes dos primeiros blocos “K” de 16K do arquivo resultante, onde “K” é o valor obtido na etapa 3) com código de troca de banco personalizado.
9. Se o tipo de mapeador do hardware de destino não for ASCII16, coloque o mesmo código de troca de banco personalizado usado nas etapas 5 e 8 na posição do arquivo 2012.
10. Somente se o mapeador de ROM usa bancos de 8K:
  - a. Escreva uma instrução LD (xxxxH), A na posição F7H do arquivo gerado, onde xxxx é a porta da memória mapeada que seleciona o banco 8K visível na primeira metade da página 1 (4000H~5FFFH).
  - b. Repita a etapa anterior para todas as partes de 16K do arquivo. Ou seja, você deve escrever a instrução LD (xxxxH), A nas posições do arquivo (4000H \* n) + F7H, onde n vai de zero até o total de bancos de 16K menos um.

O resultado é um arquivo ROM Nextor pronto para uso com o driver de dispositivo já incorporado.

### **Usando o utilitário MKNEXROM**

Em vez de realizar manualmente todas as etapas necessárias para construir uma ROM de Kernel Nextor completa, é mais conveniente usar o utilitário MKNEXROM. Esta ferramenta pode ser usada para criar um novo arquivo ROM do Kernel Nextor, permitindo também modificar um arquivo existente alterando o código do mapeador e/ou adicionando conteúdo extra nas áreas 1K livres presentes nos bancos 0 e 3.

O MKNEXROM vem como um arquivo executável de linha de comando para Windows, mas o código-fonte no padrão C também é fornecido.

A sintaxe de uso da ferramenta MKNEXROM é a seguinte:

```
MKNEXROM <basefile> <newfile> [/d:<driverfile>]
                           [/m:<mapperfile>] [/e:<extrafile>]
                           [/8:<8K bank selection port address>]
```

<basefile> pode ser:

- O arquivo base do Kernel Nextor, ou seja, o arquivo que contém apenas o código comum do Kernel.
- Um arquivo ROM de Kernel Nextor completo com o(s) banco(s) de driver já anexado(s), que deve estar de acordo com as regras e estrutura descritas na seção 4. O conteúdo deste arquivo deve ser o seguinte:
  1. 256 bytes “dummy”.
  2. A assinatura do driver
  3. A tabela de salto do driver
  4. O próprio código do driver

E, opcionalmente, se o driver gerar em mais de um banco de 16K, para cada bloco adicional de 16K:

5. 256 bytes fictícios.
6. O código ou dados adicionais do driver.
7. Espaço fictício de até 16K (não necessário para o último banco).

Especificar um arquivo de driver é obrigatório se um arquivo base do Kernel sem driver for especificado em <basefile>, e proibido se um arquivo ROM de Kernel completo for especificado.

<mapperfile> é o arquivo que contém o código de troca do banco. Se nenhum arquivo mapeador for especificado, o código do mapeador do próprio arquivo base será anexado ao código do driver.

<extrafile> é o arquivo que contém o código ou dados extras para o arquivo ROM resultante. Esses dados extras podem ter até 1K de tamanho e serão colocados na posição 0x3BD0H dos bancos 0 e 3, ficando visíveis para aplicativos por meio de chamadas entre slots padrão (como RDSLت ou CALSLT) e para o slot do Kernel, no endereço 0x7BD0H. Veja o item “O espaço livre no banco principal do Kernel para mais detalhes”.

/8 deve ser usado apenas se o mapeador ROM usar bancos de 8K. <Endereço da porta de seleção do banco 8K> é o endereço da porta mapeada na memória que seleciona o banco 8K visível na primeira metade da página 1 (4000H~5FFFH).

Como alternativa ao uso do parâmetro /8 ao usar mapeadores de ROM com bancos de 8K, MKNEXROM pode ser instruído a corrigir apropriadamente a ROM gerada adicionando um cabeçalho ao próprio arquivo de mapeamento. Este cabeçalho consiste em um byte FFH seguido pelo endereço da porta de seleção do banco no formato little-endian.

### **Regras para o código de troca de banco**

Se o tipo de mapeador do hardware de destino onde a ROM resultante será gravada não for ASCII16, um arquivo contendo o código de mapeamento próprio deve ser fornecido. Este código deve seguir as seguintes regras:

1. Deve ter no máximo 48 bytes.
2. Deve-se trocar na página 1 o banco de 16K ROM cujo número é passado no registro A (os bancos são numerados começando em zero). Presume-se que o slot ROM já esteja ativado na página 1.

3. Só pode corromper o par de registros AF. Todos os outros registros devem ser preservados.
4. Deve poder ser executado em qualquer endereço (não pode conter saltos absolutos).

Para fins de ilustração, este é um código-fonte do código de troca de banco válido para o mapeador ASCII8:

```
RLCA
LD    (6000H),A
INC   A
LD    (6800H),A
RET
```

Se um arquivo com o código anterior for passado para MKNEX-ROM como o arquivo mapeador a ser usado, é necessário adicionar um parâmetro /8:6000 à linha de comando para que o arquivo ROM gerado inclua o patch apropriado para mapeadores ROM de bancos de 8K. O arquivo pode ter um cabeçalho para tornar o parâmetro /8 desnecessário. Veja abaixo:

```
DB    0FFH
DW    6000H
RLCA
LD    (6000h),A
INC   A
LD    (6800h),A
RET
```

### **Estrutura de um driver Nextor**

Esta seção contém todos os detalhes necessários para desenvolver um driver de dispositivo para Nextor. O código-fonte de um driver “dummy” (um driver válido que não gerencia dispositivos) é fornecido e pode ser usado como referência para desenvolvimento de outros drivers.

#### **Drivers baseados em drive e baseados em dispositivo**

Ao desenvolver um driver de dispositivo Nextor, o desenvolvedor deve escolher entre dois tipos de driver: baseado em drive e baseado em dispositivo.

Drivers baseados em drive têm a mesma peculiaridade dos drivers MSXDOS: eles localizam um conjunto de unidades de driver e Nextor atribui uma letra de drive fixa para cada unidade; qualquer unidade necessária para o mapeamento de dispositivo ou partição deve ser executado pelo próprio driver. O conjunto de rotinas de acesso ao dispositivo de armazenamento localizado por esses drivers é o mesmo utilizado pelos drivers MSXDOS (DSKIO, DSKCHG, GETDPB, CHOICE, DSKFMT e MTOFF).

Os drivers baseados em dispositivo usam uma abordagem diferente. Eles não localizam unidades de driver, mas dispositivos de armazenamento diretamente. Além disso, eles expõem rotinas para acesso bruto aos dispositivos, e é o próprio Kernel Nextor que gerencia o mapeamento da unidade para o dispositivo e partição.

Em geral, é recomendado desenvolver drivers baseados em dispositivos, pois as rotinas de implementação são mais fáceis e o código do driver precisa apenas ler e escrever setores de dispositivos absolutos sem ter que se preocupar com partições; além disso, a ferramenta de particionamento de dispositivo do Nextor pode ser usada para criar partições em dispositivos controlados apenas por drivers baseados em dispositivo. O desenvolvimento de um driver baseado em drive pode, entretanto, ser uma boa opção para converter facilmente um driver MSXDOS existente para Nextor.

O Nextor irá realizar um mapeamento automático de unidade para dispositivo e partição na inicialização para as unidades atribuídas a drivers baseados em dispositivo. Este mapeamento pode ser modificado posteriormente usando o utilitário MAPDRV.

## Página 0 – Rotinas e dados

Esta seção explica as rotinas e dados que estão disponíveis na página 0 (endereços 4000H-40FFh) de todos os bancos Nextor, incluindo o(s) banco(s) de driver.

Ao criar um Kernel Nextor com driver embutido, o código da página 0 torna-se parte de todos os bancos de driver quando a ROM do Kernel Nextor completa é gerada.

## O cabeçalho do driver

O cabeçalho do driver contém algumas informações que ajudam o Nextor a identificá-lo e determinar seu tipo. Ele é colocado a partir de 4100H, conforme descrito abaixo:

### DRV\_SIGN (4100H)

Assinatura válida de driver. É usada pelo Kernel durante a inicialização para verificar se o banco contém efetivamente um driver válido. Consiste literalmente na string “NEXTOR\_DRIVER” (sem as aspas), terminada em zero e em maiúsculas.

### DRV\_FLAGS (410EH)

Byte de sinalizadores que contém informações sobre o driver:

bit 0: 0 → driver baseado em drive;  
1 → driver baseado em dispositivo.

bit 1: Reservado, deve ser zero.

bit 2: 1 se o driver implementa a rotina DRV\_CONFIG.

bits 3~7: Reservados, devem ser zero.

O DRV\_CONFIG é usado pelo Nextor a partir da versão 2.0.5.

### RESERVADO (410Fh)

Byte reservado, deve ser zero.

### DRV\_NAME (4110h)

String contendo o nome do driver. Deve consistir em 32 caracteres ASCII imprimíveis (códigos ASCII 32 a 126), justificada à esquerda e preenchida à direita com espaços.

## Rotinas para drivers

Esta seção explica as rotinas de driver que são necessárias para drivers. As entradas da tabela de salto para as rotinas usam de drivers baseados em dispositivos usam os mesmos endereços que as rotinas para drivers baseados em drive, já que um driver implementará apenas um dos dois conjuntos de rotinas.

Os drivers baseados em dispositivo não usam o conceito de unidades mapeadas em letras; em vez disso, eles acessam dispositivos de armazenamento diretamente, e é o próprio Kernel Nextor que gerencia a unidade para o dispositivo e atribuição de partição, tanto automaticamente e manualmente.

Drivers baseados em dispositivo podem controlar até sete dispositivos de armazenamento, numerados 1 a 7. Cada um tem, por sua vez, de uma a sete unidades lógicas, também numeradas de 1 a 7. Os dispositivos não podem ser fisicamente divididos em unidades lógicas, e o driver deve tratar o dispositivo como tendo uma única unidade lógica de índice 1.

"Dispositivos de bloco" são todos os dispositivos que podem ser lidos e gravados por meio do acesso a setores lógicos. Isso inclui disquetes, discos rígidos, pendrives, cartões multimídia, etc. Os dispositivos de bloco devem ser legíveis e opcionalmente graváveis por meio da rotina `DEV_RW`.

Na versão atual, o Nextor só trabalha com dispositivos em bloco e setores com 512 bytes de tamanho. O suporte para acesso direto ao sistema de arquivos está planejado para versões futuras.

As informações sobre cilindros, cabeçotes e setores por trilhas aplicam-se apenas ao disco rígido; para outros tipos de dispositivos, ou quando essa informação não estiver disponível, esses campos devem ser retornados com valor zero.

Os sinalizadores "somente leitura" devem ser setados apenas para dispositivos que, fisicamente, só podem ser lidos (por exemplo, um CD-ROM). Um dispositivo que pode ser protegido / habilitado para gravação não deve ser tratado como um dispositivo somente leitura.

Na versão atual do Nextor, não é possível formatar dispositivos controlados por drivers baseados em dispositivo. Este é um recurso planejado para versões futuras.

### **O espaço livre no banco principal do Kernel**

O Kernel Nextor possui um espaço não utilizado de 1K nos dois bancos principais (banco 0 quando rodando no modo normal, banco 3 quando rodando no modo MSXDOS 1) que pode ser preenchido com qualquer tipo de dado ou código útil para o driver.

O banco principal sempre fica disponível no slot do Kernel em circunstâncias normais; portanto, esta área pode ser acessada através dos mecanismos de acesso ao slot padrão (como chamada entre slots via CALSLT, inter-slot ler via RDSLTLT, etc) mesmo por software que não conhece o mecanismo de mapeamento do Nextor. Este espaço ocupa a área de memória de 7BD0H a 7FCFH.

Existem três casos em que pode ser necessário adicionar conteúdo personalizado a esta área:

- Quando uma string de escolha de formatação não nula (para dispositivos que não podem ser formatados) ou a string "Single side / Double side" (no caso de driver de disquete MSX legado) deve ser retornada pelo DRV\_CHOICE ou rotinas DRV\_FORMAT. Essas duas strings já são fornecidas pelo Kernel, outras strings devem ser colocadas no espaço de 1K.
- Quando os dados que devem ser lidos por RDSLTLT ou por um mecanismo equivalente (por exemplo, um identificador de implementação UNAPI).
- Quando um hook deve ser corrigido. Para isso, o hook deve ser configurado para fazer uma chamada interbancária para esta área, no endereço onde está o código. Os hooks de interrupção de extensão da BIOS não devem usar este mecanismo.

O código nesta área deve usar a rotina CALBNK se precisar chamar rotinas no banco de drivers, cujo número pode ser lido a partir do endereço K\_SIZE.

Qualquer código colocado nesta área deve ser idêntico nos bancos 0 e 3, para que tudo funcione corretamente. A ferramenta MKNEX-ROM corrigirá apropriadamente ambos os bancos se um arquivo de dados para esta área for fornecido.

### **Rotinas do Kernel Nextor**

Abaixo estão listadas todas as rotinas presentes no Kernel Nextor. Elas estão detalhadamente descritas no apêndice, capítulo 8, item 8.6.4 – Rotinas adicionadas pelo Nextor.

GSLOT1	(402DH) Retorna o slot do driver atual.
RDBANK	(403CH) Lê um byte em qualquer banco do Kernel.
CALLB0	(403FH) Alternar temporariamente o banco principal do Kernel e chamar a rotina cujo endereço está em CODE_ADD (F1D0H).
CALBNK	(4042H) Chamar rotina em outro banco do Kernel.
GWORK	(4045H) Obter o endereço da entrada SLTWRK de 8 bytes para o slot passado ou para o slot atual na página 1.
K_SIZE	(40FEH) Quantidade de bancos do Kernel Nextor.
CUR_BANK	(40FFH) Número do banco Kernel atual.
CHGBNK	(7FD0H) Faz com que o banco especificado seja visível na página 1 do Z80.
PROMPT	(41E8H) Exibe a mensagem “Insira o disco para a unidade X: e pressione uma tecla quando pronto”, aguardando a ação solicitada.
DRV_SIGN	(4100H) Assinatura de driver válida.
DRV_FLAGS	(410EH) Sinalizadores com informações sobre o driver.
RESERVADO	(410FH) Reservado, não usar.
DRV_NAME	(4110H) String contendo o nome do driver.
DRV_TIMI	(4130H) Entrada da rotina de interrupção do driver
DRV_VERSION	(4133H) Retornar a versão do driver.
DRV_INIT	(4136H) Rotina de inicialização do driver.
DRV_BASSTAT	(4139H) Manipulador de instruções estendidas BASIC (“CALLs”).
DRV_BASDEV	(413CH) Manipulador de dispositivos estendidos BASIC.
DRV_EXTBIO	(413FH) Manipulador da BIOS estendida.
DRV_DIRECT0/1/2/3/4	(4142H, 4145H, 4148H, 414BH, 414EH) Entradas para chamadas diretas ao driver.
DRV_CONFIG	(4151H) Permitir que o driver forneça informações sobre a configuração na inicialização.
RESERVADO	(4155H a 415FH) Reservado, não usar.
DRV_DSKIO	(4160H) Ler ou gravar setores do dispositivo de armazenamento de massa associado a uma unidade de drive.
DRV_DSKCHG	(4163H) Obter informações sobre o estado de alteração da mídia associada a uma determinada unidade de drive.

- DRV\_GETDPB (4166H) Obter um DPB (Drive Parameter Block) para a mídia associada a uma determinada unidade de driver.
- DRV\_CHOICE (4169H) Retorna uma string de escolha de formato para um disco.
- DRV\_FORMAT (416CH) Formata um disco e inicializa seu setor de boot, FAT e diretório raiz.
- DRV\_MTOFF (416FH) Parar o motor de todos os drives.
- DEV\_RW (4160H) Ler ou gravar setores absolutos de ou para um dispositivo.
- DEV\_INFO (4163H) Retornar informações sobre um dispositivo.
- DEV\_STATUS (4166H) Verificar a disponibilidade e alterar o estado de um dispositivo ou unidade lógica.
- LUN\_INFO (4169H) Obter informações para uma unidade lógica.

## 7.2 – O UZIX

O Uzix é um sistema operacional desenvolvido para o MSX que permite multitarefa. O Uzix, na verdade, é um Unix menos potente, podendo rodar, a rigor, qualquer aplicação para Unix desde que, depois de compilada, caiba em 32 Kbytes (Uzix 1.0) ou 48 Kbytes (Uzix 2.0). Esse sistema foi criado a partir do UZI (Unix Zilog Implementation), um sistema Unix criado para o Z80. Uzix significa "Unix Zilog Implementation for MSX", ou "Implementação Unix Zilog para MSX". O Uzix não necessita de nenhuma ROM específica para funcionar; carrega o Kernel do disco e usa as rotinas de acesso direto ao disco do BDOS. Na verdade, em sua segunda versão, faz acesso direto à maioria dos dispositivos de hardware, incluindo HD's. Isso melhora bastante a performance do sistema.

### 7.2.1 – Sistemas de arquivos no Uzix

Os arquivos são organizados no Uzix de forma bem diferente do MSXDOS. Para localizar e referenciar os arquivos, estes possuem três áreas:

#### **Nome**

É a identificação obrigatória do arquivo. No Uzix, um nome de arquivo pode conter no máximo 14 caracteres, que podem ser qualquer um (letras, números, ponto, barra, espaço, sinal de igual, etc.).

## **Conteúdo**

É o que compõe o arquivo propriamente dito (dados, texto, código executável, etc.)

## **Outros dados de identificação**

São dados com as atribuições do arquivo. Esses dados são bits de permissão de acesso, número de links, identificação do proprietário e do grupo, tamanho do arquivo, data de criação/modificação, etc. São armazenados numa estrutura chamada ‘inode’, separadamente do nome do arquivo, e aqui reside a maior diferença entre o Uzix e o MSXDOS.

### **7.2.1.1 – Tipos de arquivos**

O Uzix utiliza 3 tipos principais de arquivos:

- 1 – Arquivos ordinários (comuns);
- 2 – Arquivos diretórios;
- 3 – Arquivos especiais.

#### **Arquivos ordinários**

Constituem a maioria dos arquivos do sistema. Eles são usados para armazenar informações (dados de programas, textos, executáveis, etc.) e são caracterizados por não possuírem nenhum formato interno particular.

#### **Arquivos diretórios**

Para que se possa organizar os arquivos, existem os arquivos diretórios, ou simplesmente diretórios. No MSXDOS, o diretório é armazenado de forma bem diferente de outros arquivos, mas aqui o diretório é apenas um arquivo. Esses diretórios contém uma lista dos nomes de arquivos, que podem ser ordinários, especiais ou outros arquivos diretórios.

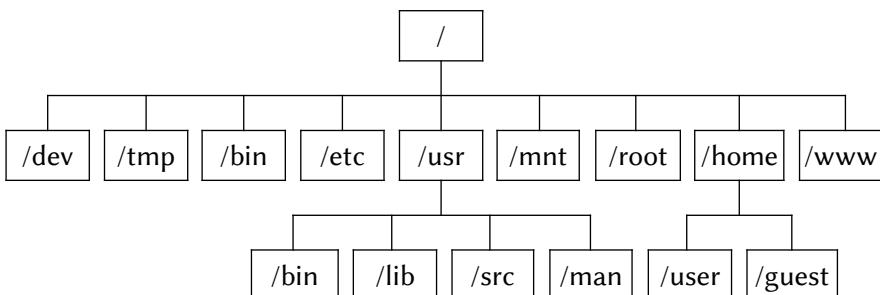
#### **Arquivos especiais**

São lidos e gravados como arquivos comuns, mas são utilizados para referenciar dispositivos lógicos do sistema, que por sua vez podem

ativar dispositivos de hardware, como impressoras, terminais, dispositivos de memória de massa, etc.

### 7.2.1.2 – Estrutura hierárquica

No Uzix, também temos o conceito de subdiretórios como no MSXDOS 2. Entretanto, aqui existe uma estrutura pré-definida de subdiretórios. Essa estrutura pode ser modificada pelo usuário, mas não é aconselhável fazê-lo porque ela é padrão no mundo Unix. Veja a ilustração abaixo.



Cada um desses subdiretórios tem um uso específico, mas não obrigatório. A descrição de cada um está abaixo.

/ – Diretório raiz

/dev – Contém os nomes arquivos especiais associados a dispositivos de hardware ou software.

/tmp – Usado por todo o sistema para a criação de arquivos temporários.

/bin – Contém as aplicações mais genéricas do sistema.

/etc – Arquivos usados para administrar o sistema.

/usr – Arquivos gerais do sistema. Esse subdiretório contém mais 4 subdiretórios:

/bin – Aplicações genéricas.

/lib – Bibliotecas.

/src – Códigos fonte.

/man – Manuais do sistema (arquivos texto).

- /mnt – Usado como ponto de conexão de um sistema de arquivo de outro dispositivo. Também usado para montagem (mount).
- /root – Diretório de trabalho do administrador do sistema.
- /home – Usado pelos usuários comuns como área de trabalho.
  - /user – Usuário “user”.
  - /guest – Usuário “guest”
- /www – Arquivos de internet.

### **7.2.1.3 – Permissões de acesso a arquivos**

O Uzix usa o mesmo sistema de permissão de acesso aos arquivos usados pelo Unix. Assim, o usuário pode definir, por exemplo, quem pode ler seus arquivos ou fazer alteração neles. Essa proteção pode ser aplicada a 3 classes de usuários:

- (u) – Usuário proprietário ou administrador do sistema
- (g) – Grupo, ou conjunto de usuários que possuem alguma característica em comum com o usuário proprietário
- (o) – Outros usuários do sistema.

As permissões de acesso têm três níveis:

- (r) – Leitura: permite listar o conteúdo do arquivo ou do diretório
- (w) – Escrita/gravação: permite alterar o conteúdo do arquivo ou criar/renomendar arquivos/diretórios.
- (x) – Execução: permite executar o arquivo ou entrar/manipular arquivos/diretórios.

O usuário “root” tem acesso ilimitado ao sistema; um usuário comum não pode impedir o acesso do usuário “root” aos seus arquivos.

As permissões de acesso são gravadas nos inodes e podem ser listadas pelo comando “ls” ou pelo seu alias padrão “dir”.

### **7.2.1.4 – Estrutura dos arquivos no disco**

Os arquivos Uzix são estruturados no disco em 4 grupos: setor de boot, superblock, inodes e blocos de dados.

## Setor de boot

O setor de boot é sempre o setor 0 do disco e sua função é a mesma do MSXDOS: dar partida na inicialização do sistema, carregando os arquivos necessários. Ele é executado da mesma forma que para o MSXDOS, carregando o setor 0 no endereço C000H e executando a rotina em C01EH. Os dados gravados de C000H até C01DH não são válidos para o Uzix.

### Superblock

Ocupa apenas um setor (setor 1) e contém informações sobre o disco. Essas informações são as seguintes:

- +0/+1 Assinatura. São os cinco primeiros dígitos do número PI armazenados como número inteiro (31415).
- +2/+3 Primeiro bloco lógico dos inodes.
- +4/+5 Total de blocos lógicos reservados aos inodes.
- +6/+7 Total de blocos lógicos reservados aos arquivos.
- +8/+9 Total de blocos lógicos do disco.
- +10/+11 Total de blocos reservados ao Kernel. O valor padrão para o Uzix é de 50 blocos.
- +12/+111 Apontadores para blocos Kernel. 50 apontadores de 2 bytes.
- +112/+113 Total de inodes fs do disco. O valor padrão é de 50 inodes.
- +114/+115 Total de inodes fs (sistema de arquivos) livres.
- +116/+215 Apontadores para inodes fs. 50 apontadores de 2 bytes.
- +216/+217 Hora da última modificação (formato MSXDOS).
- +218/+219 Data da última modificação (formato MSXDOS).
- +220 Flag de modificação do sistema de arquivos.
- +221 Flag somente leitura do sistema de arquivos.
- +222/+223 Usado para checar corrupção (montagem do inode).
- +224 Flag modificada (montagem do inode).
- +225/+226 Dispositivo referente ao inode (montagem do inode).
- +227/+228 Número do inode (montagem do inode).
- +229/+230 Contador de referência in-core (montagem do inode).
- +231 Flag somente leitura do sistema de arquivos.
- +222/+295 Cópia do inode do disco (conforme a estrutura abaixo).
- +296/+297 Dispositivo do sistema de arquivos.

Os 50 blocos reservados do superblock não fazem parte do fs (filesystem – sistema de arquivos). Eles contêm o Kernel do Uzix e são acessados pelo bootstrap secundário para inicializar o Uzix a partir do disco.

## Inodes

Os inodes vêm logo após o superblock. Cada inode ocupa 64 bytes, podendo ser definidos até 8 inodes por bloco lógico (cada bloco lógico equivale a um setor, ou 512 bytes). A função dos inodes é armazenar todas as informações sobre os arquivos (exceto o nome) e mapear o arquivo no disco através de apontadores diretos e indiretos. Eles estão organizados como se segue.

- +0/+1 Flag de modo. Nela estão todas as permissões de acesso e o tipo do arquivo.
- +2/+3 Total de apontadores para o arquivo.
- +4 Número do usuário do arquivo.
- +5 Número do grupo de acesso ao arquivo.
- +6/+9 Tamanho do arquivo.
- +10/+11 Hora do último acesso ao arquivo.
- +12/+13 Data do último acesso ao arquivo.
- +14/+15 Hora da última modificação do arquivo.
- +16/+17 Data da última modificação do arquivo.
- +18/+19 Hora da criação do arquivo.
- +20/+21 Data da criação do arquivo.
- +22/+57 Apontadores diretos (18 apontadores de 2 bytes).
- +58/+59 Apontador indireto de primeiro nível.
- +60/+61 Apontador indireto de segundo nível.
- +62/+63 0000H.

A data e a hora são armazenadas da mesma forma que para o MSXDOS. Tanto o usuário quanto o grupo e as respectivas senhas são armazenadas em um arquivo próprio. As senhas são criptografadas. Podem haver até 256 grupos cadastrados no sistema com até 256 usuários cada um.

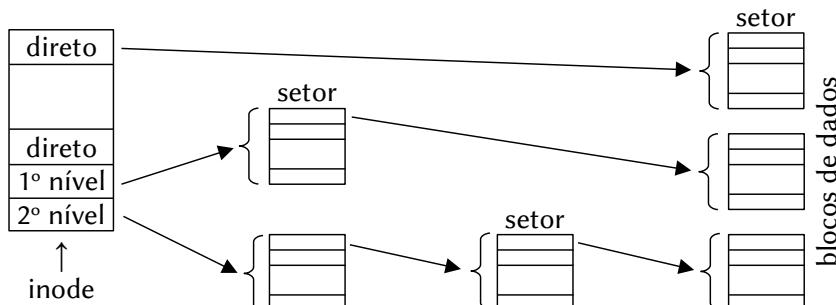
Cada apontador tem dois bytes de comprimento e aponta para um bloco lógico (que tem 512 bytes), logo a capacidade máxima endereçável pelo inode é 32 Mbytes por dispositivo ( $512 * 65\,536$ ). Como exis-

tem apenas 18 apontadores diretos em um inode, apenas arquivos de até 9 Kbytes são endereçados diretamente. Acima disso, é necessário usar apontadores indiretos.

Os apontadores indiretos funcionam da seguinte forma: o apontador indireto de primeiro nível aponta para um bloco lógico que contém apontadores para os blocos de dados. Como cada apontador possui 2 bytes, podem haver até 256 apontadores em um bloco lógico. Eles podem mapear arquivos de até 128 Kbytes ( $256 * 512$ ). Usando apontadores de primeiro nível, podemos acessar arquivos de até 135 Kbytes ( $128 + 9$ ).

Para arquivos de mais de 135 Kbytes, apontadores indiretos de segundo nível são usados. O apontador indireto de segundo nível aponta para um bloco lógico. Cada apontador deste bloco aponta para outro bloco lógico com apontadores, agora sim para o arquivo. Nesse caso, podem ser mapeados arquivos de até 32 Mbytes ( $128 \text{ Kbytes} * 256 \text{ apontadores}$ ), que é a capacidade máxima endereçável pelos inodes.

O esquema de apontadores está ilustrado logo. Na ilustração, é representado apenas um inode (64 bytes). Já os blocos por ele apontados correspondem a um bloco lógico (um setor do disco, 512 bytes).



É de se notar que o acesso vai ficando mais lento. Assim, para arquivos até 9 Kbytes, apenas uma operação com apontadores é necessária. Para arquivos de mais de 9 Kbytes até 135 Kbytes, são necessárias duas operações. Acima disso, até o limite de 32 Mbytes, três operações são requeridas.

## Arquivos diretórios

Os arquivos diretórios armazenam os nomes dos arquivos ou de outros diretórios. Eles são divididos em blocos de 16 bytes. Os dois primeiros bytes apontam para o inode respectivo e os outros 14 contém o nome do arquivo propriamente dito. Podem haver até 32 nomes em um bloco lógico. Os arquivos diretórios não podem ser abertos pelos comandos normais.

## Montagem

No sistema FAT do MSXDOS, ao carregar a mesma na memória, por si só, a FAT já constitui um mapa do disco, com todas as informações sobre os setores livres e ocupados. No caso do Uzix, essas informações estão espalhadas pelo disco, nos apontadores, e não num bloco único como no caso da FAT. Para que o sistema possa saber quais blocos lógicos estão livres e quais estão ocupados, é necessário um processo chamado de montagem. Nesse processo, o disco é analisado e todo seu espaço mapeado. O mapa resultante é carregado na RAM.

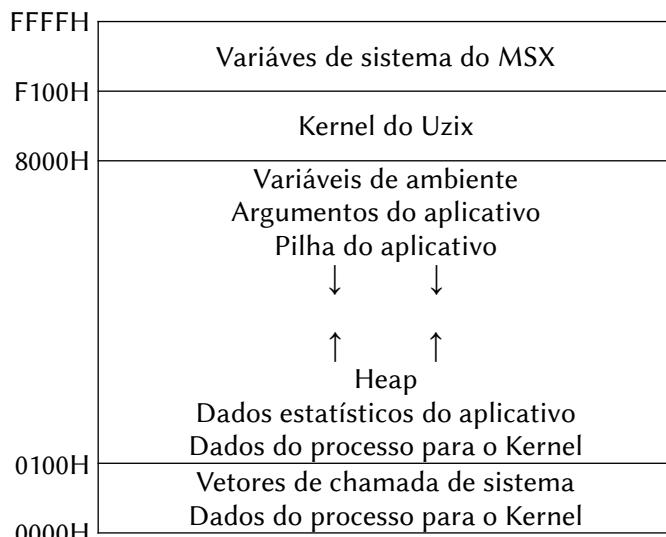
Todo dispositivo (de hardware ou software) deve sofrer o processo de montagem para que o sistema possa reconhecê-lo, muito embora isso seja processado de maneiras bem diversas de acordo com o dispositivo a ser montado. A unidade primária de disco é montada automaticamente durante a inicialização do sistema. Abaixo estão listados alguns dispositivos padrão que podem ser montados:

fd0~fd7	Drives de disquete.
null	Dispositivo nulo.
lpr	Impressora.
tty/tty0~tty2	Monitor.
console	Teclado.
mem/kmem	Memória.
sga0~sga(n)	Partições em disco rígido.
sge(n)	Partição em disco rígido onde está o UZIX.

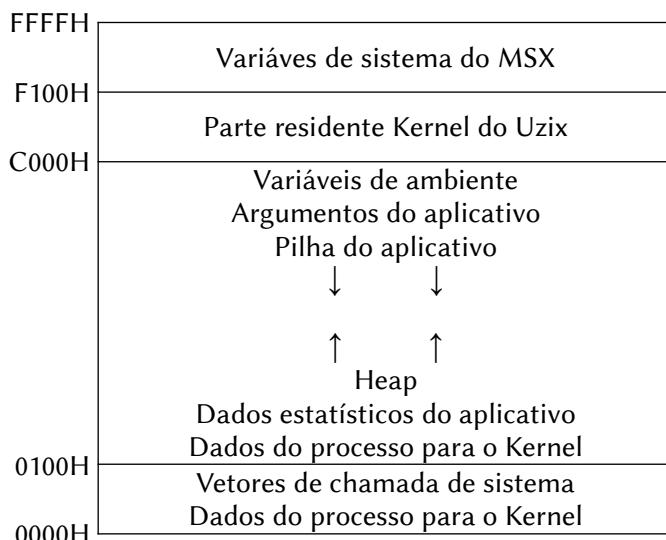
### 7.2.2 – Mapeamento de memória

O mapeamento de memória é a maior diferença entre o Uzix 1.0 e 2.0. Na verdade, o Uzix 2.0 superou em muito o 1.0 exigindo as mesmas características de hardware.

O mapeamento de memória para o Uzix 1.0 é o seguinte:



Abaixo está ilustrado o mapeamento de memória para o 2.0.



O Uzix 1.0 fica inteiramente residente na área alta de memória, a partir do endereço 8000H. Todo processo sempre ocupa 32 Kbytes de memória. Por isso, podem haver no máximo 127 processos concomitantes, se houver 4 Mbytes de memória mapeada.

Já o Uzix 2.0 tem um gerenciamento de memória mais eficiente. Embora seu Kernel ocupe mais memória que o 1.0, apenas uma parte fica residente na memória alta, na página 3, a partir do endereço C000H. Quando necessário, o Uzix chaveia as páginas de modo a acessar o restante do Kernel, executa a função e volta para o aplicativo. Cada processo pode ocupar 16 Kbytes, 32 Kbytes ou 48 Kbytes, dependendo de sua extensão. Por isso podem haver, no máximo, 252 processos concomitantes num MSX com 4 Mbytes de memória. O Kernel do Uzix 2.0 ocupa 64 Kbytes de memória no total.

### 7.2.3 – Desenvolvendo software para o Uzix

Os aplicativos para Uzix devem preferencialmente ser desenvolvidos em C, que é a linguagem em que o sistema foi escrito. Apenas alguns poucos cuidados devem ser tomados.

Se o aplicativo deve rodar sob o Uzix 1.0, o total de memória ocupada pelo mesmo (código, dados e pilha) deve ser menor que 32 Kbytes. Já para o Uzix 2.0, o limite é ampliado para 48 Kbytes. Após a compilação, é recomendável olhar endereço de Hbss no arquivo de mapa. Se estiver muito próximo do endereço mais alto disponível para o aplicativo (7FFFH para o Uzix 1.0 e BFFFH para o 2.0) é melhor reduzir o tamanho do código. Acontece que a pilha, as variáveis de ambiente e os argumentos do aplicativo são colocados no topo da memória, e se Hbss estiver muito próximo da pilha, poderá haver sobreposição e paralisação do sistema. Os valores máximos recomendáveis para Hbss são 7A00H para Uzix 1.0 e BA00H para Uzix 2.0.

Também é recomendável evitar variáveis locais muito grandes (tipo `char buffer[512]`), pois a pilha abaixará muito. É melhor declará-las como estáticas. Isso acaba gastando espaço na aplicação, mas evita que a pilha eventualmente se sobreponha aos dados dinâmicos, corrompendo-os.

Se forem usadas rotinas em código de máquina, deve ser observado o seguinte, sob pena de corromper e até paralisar o sistema:

- NUNCA devem ser usadas as instruções DI e EI do Z80;
- NUNCA deve ser feito acesso direto ao hardware e
- NUNCA devem ser acessados dados abaixo de 0100H ou acima da aplicação.

Uma biblioteca específica para desenvolvimento de software para o Uzix foi desenvolvida pelo autor do mesmo e pode ser encontrada na página oficial do Uzix (<http://uzix.sf.net>). Essa biblioteca é específica para o compilador Hitech-C.

#### **7.2.4 – Comandos de texto**

ADDUSER	Adiciona um usuário ao sistema.
ALIAS	Apresenta ou define um comando alias.
BANNER	Imprime uma mensagem em caracteres grandes.
BASENAME	Remove orientação de componentes de um diretório.
BOGOMIPS	Imprime a velocidade de processamento em BogoMips.
CAL	Apresenta um calendário.
CAT	Concatena arquivos e imprime na saída padrão.
CD	Troca diretórios.
CDIFF	Imprime a diferença entre dois arquivos com contexto.
CGREP	Procura uma string e imprime as linhas onde forem encontradas.
CHGRP	Troca o usuário proprietário do grupo para cada arquivo.
CHMOD	Troca as permissões de acesso aos arquivos.
CHOWN	Troca o usuário comum e o usuário proprietário do grupo para o arquivo especificado.
CHROOT	Troca o diretório raiz.
CKSUM	Apresenta o checksum e o tamanho do arquivo.
CLEAR	Limpa a tela.
CMP	Compara arquivos.
CRC	Apresenta o checksum dos dados do arquivo.
CP	Copia arquivos.
CPDIR	Copia diretórios.
DATE	Apresenta a data e a hora correntes do sistema.

DD	Copia arquivo convertendo o mesmo.
DF	Apresenta o espaço livre em disco em unidades de 512 bytes.
DHRY	Apresenta a velocidade de processamento em dhystones.
DIFF	Imprime a diferença entre dois arquivos
DIRNAME	Imprime o sufixo de um nome de arquivo.
DOSDEL	Apaga um arquivo em discos MSXDOS.
DOSDIR	Lista arquivos de um disco MSXDOS.
DOSREAD	Lê um arquivo de um disco MSXDOS.
DOSWRITE	Escreve um arquivo em um disco MSXDOS.
DU	Apresenta o espaço ocupado por diretórios e subdiretórios
ECHO	Apresenta uma linha de texto
ED	Executa um editor de texto padrão.
EXIT	Sai da sessão atual.
FALSE	Não faz nada; simplesmente retorna com estado de erro “1”.
FGREP	Procura uma string e imprime as linhas onde for encontrada.
FILE	Faz uma suposição sobre qual tipo o arquivo é.
FLD	Lê e concatena campos de um arquivo
FORTUNE	Imprime, aleatoriamente, um provérbio.
GREP	Procura uma string e imprime as linhas onde for encontrada.
HEAD	Imprime as primeiras linhas do arquivo.
HELP	Imprime alguns comandos com o respectivo formato.
INIT	Controle de inicialização de processos.
KILL	Termina processos do sistema.
LOGIN	Inicia uma sessão.
LN	Adiciona links entre arquivos.
LOGOUT	Encerra uma sessão.
LS	Lista o conteúdo de diretórios.
MAN	Apresenta o manual on-line.
MKDIR	Criar diretórios.
MKNOD	Cria arquivos especiais
MORE	Utilitário de paginação.
MOUNT	Monta o <dispositivo> no <caminho> especificado.
MV	Renomeia ou move arquivos.
PASSWD	Troca a senha do usuário.
PROMPT	Altera o prompt do Uzix.
PS	Imprime um relatório do estado do processo.
PWD	Imprime o caminho do diretório de trabalho atual.

QUIT	Encerra a sessão atual.
REBOOT	Reseta o computador.
RM	Remove arquivos.
RMDIR	Remove diretórios.
SASH	É um tipo de shell com comandos internos.
SET	Apresenta ou define variáveis de ambiente.
SLEEP	Faz o sistema “dormir” por <segundos> segundos.
SU	Conecta temporariamente como superusuário ou outro usuário.
SOURCE	Apresenta o “fonte” do arquivo.
SUM	Analiza a checksum e o contador de blocos do arquivo.
SYNC	Descarrega os buffers do sistema de arquivos.
TAIL	Imprime as últimas linhas de um arquivo.
TAR	Concatena/extrai arquivos para armazenagem.
TEE	Lê da entrada padrão e escreve em um arquivo.
TIME	Executa o comando e imprime a hora real, a hora do usuário e a hora do sistema (horas–minutos–segundos).
TOP	Lista os processos mais ativos.
TOUCH	Troca a hora e a data dos arquivos.
TR	Troca os caracteres de um arquivo (translitera).
TRACE	Modo trace?
TRUE	Não faz nada, somente retorna com estado de erro 0.
UMOUNT	Desmonta sistema de arquivos do dispositivo especificado.
UMASK	Remove máscaras.
UNALIAS	Remove um comando tipo alias.
UNAME	Imprime informações sobre o sistema.
UNIQ	Remove linhas duplicadas em arquivos ordenados.
WC	Imprime o número de bytes, palavras e linhas de um arquivo.
WHOAMI	Imprime o nome do usuário associado com o ID do usuário atual.
YES	Imprime “y” ou <string> repetidamente na saída padrão.

Todos estes comandos estão detalhadamente descritos no apêndice, capítulo 6, seção 6.1.2 – Descrição dos Comandos.

## 7.2.5 – Chamadas de sistema

Assim como o MSXDOS permite chamadas diretas ao BDOS/Kernel, o UZIX também tem suas próprias chamadas diretas, compatíveis com as funcionalidades do AT&T Unix Version 7. Devem ser tomados os mesmos cuidados descritos acima para rotinas ASM:

- NUNCA devem ser usadas as instruções DI e EI;
- NUNCA deve ser feito acesso direto ao hardware;
- NUNCA devem ser acessados dados abaixo de 0100H ou acima da aplicação.

Para fazer uma chamada de sistema é necessário empilhar os parâmetros na ordem inversa da declaração, depois o número da chamada e então executando um CALL 08H. É responsabilidade da aplicação desempilhar os parâmetros após o CALL. O valor de retorno, de 16 bits, é colocado no registrador DE. A única exceção é a chamada ‘lseek’, cujo valor de retorno é de 32 bits e é colocado em HL:DE (HL é a palavra mais significativa).

Estas chamadas dividem-se em diretas, indiretas e via GETSET. Elas estão detalhadamente descritas no apêndice, inclusive com a biblioteca necessária, capítulo 6, seção 6.4 – Chamadas de sistema.

### 7.2.5.1 – Chamadas diretas

ACCESS	(#00) – Determina o nível de acesso de um arquivo.
ALARM	(#01) – Agenda um sinal após um tempo especificado.
BRK	(#02) – Altera a alocação do núcleo.
CHDIR	(#03) – Altera diretório padrão.
CHMOD	(#04) – Altera atributos de um arquivo
CHOWN	(#05) – Troca usuário e grupo de um arquivo.
CLOSE	(#06) – Fecha um arquivo.
GETSET	(#07) – Implementa chamadas que leem ou alteram valores de variáveis de sistema.
DUP	(#08) – Duplica um descritor de arquivo aberto.
DUP2	(#09) – Duplica um descritor de arquivo aberto.
EXECVE	(#10) – Executa um arquivo.
EXIT	(#11) – Encerra um processo.
FORK	(#12) – Gera um novo processo.

FSTAT (#13) – Obtém informações sobre o arquivo.  
 GETFSYS (#14) – Obtém informações do sistema.  
 IOCTL (#15) – Controle de dispositivos.  
 KILL (#16) – Envia o sinal ‘sig’ para o processo especificado em ‘r0’.  
 LINK (#17) – Link para um arquivo.  
 MKNOD (#18) – Cria um diretório ou um arquivo especial.  
 MOUNT (#19) – Monta o sistema de arquivos.  
 OPEN (#20) – Abre um arquivo para leitura ou escrita.  
 PAUSE (#21) – Pausa o sistema.  
 PIPE (#22) – Cria um canal interprocesso.  
 READ (#23) – Lê de um arquivo.  
 SBRK (#24) – Altera a alocação do núcleo (ver BRK – #02).  
 LSEEK (#25) – Move o ponteiro de leitura / gravação.  
 SIGNAL (#26) – Pega ou ignora sinais.  
 STAT (#27) – Obtém o estado do arquivo  
 STIME (#28) – Configura data e hora do sistema.  
 SYNC (#29) – Atualiza o superblock.  
 TIME (#30) – Obtém a data e a hora.  
 TIMES (#31) – Retorna informações de tempo.  
 UMOUNT (#32) – Desmonta sistemas de arquivos.  
 UNLINK (#33) – Remove entrada de diretório.  
 UTIME (#34) – Define hora de um arquivo.  
 WAITPID (#35) – Espera que o processo mude de estado.  
 WRITE (#36) – Escreve em um arquivo  
 REBOOT (#37) – Reinicia o sistema  
 SYMLINK (#38) – Cria um novo nome para um arquivo.  
 CHROOT (#39) – Altera o diretório raiz.  
 MOD\_REG (#40)  
 MOD\_DEREG (#41)  
 MOD\_CALL (#42)  
 MOD\_SENDREPLY (#43)  
 MOD\_REPLY (#42)

### 7.2.5.2 – Chamada indireta

CREAT – Cria um novo arquivo

### 7.2.5.3 – Chamadas via GETSET

- GETPID – Obtém o ID do processo.
- GETPPID – Obtém o ID do processo chamador.
- GETUID – Retorna a identidade do usuário real do processo atual.
- SETUID – Define a identidade do usuário e do grupo.
- GETEUID – Retorna o ID do usuário efetivo do processo chamador.
- GETGID – Retorna o ID do usuário real do processo atual.
- SETGID – Define o ID do grupo.
- GETEGID – Retorna o ID do grupo efetivo do processo chamador.
- GETPRIO – Retorna a prioridade do processo atual.
- SETPRIO – Define a prioridade de um processo.
- UMASK – Define máscara de criação de um arquivo.
- SYSTRACE – Gerar e aplicar protocolos para as chamadas de sistema.

### 7.2.6 – Módulo TCP/IP

O módulo TCP/IP implementa um subconjunto de IPv4 e permite que o Uzix se comunique com outros sistemas que suportem este protocolo. A assinatura do módulo TCP/IP é 04950H, e ele provê as funções listadas abaixo.

Chamada FNC#	Protótipo C	
ipconnect	int ipconnect(char mode, ip struct t *ipstruct)	1
ipgetc	int ipgetc(uchar socknum)	2
ipputc	int ipputc(uchar socknum, uchar byte)	3
ipwrite	int ipwrite(uchar socknum, uchar *bytes, int len)	4
ipread	int ipread(uchar socknum, uchar *bytes, int len)	5
ipclose	int ipclose(uchar socknum)	6
iplisten	int iplisten(int aport, uchar protocol)	7
ipaccept	int ipaccept(ip struct t *ipstruct, int aport, uchar block)	8
ping	int ping(uchar *IP, unsigned long *unused, uint len)	9
setsockopttimeout	int setsockopttimeout(uchar socknum, uint timeout)	10
ipunlisten	int ipunlisten(int aport)	11
ipgetpingreply	icmpdata t *ipgetpingreply(void)	12
gettcpinfo	tcpinfo t *gettcpinfo(void)	13
getsockinfo	sockinfo t *getsockinfo(uchar socknum)	14

Os tipos de dados usados são:

```
// números de protocolo (protocolo para iplisten)
ICMP_PROTOCOL = 1
TCP_PROTOCOL = 6
UDP_PROTOCOL = 17

// modos de abertura
TCP_ACTIVE_OPEN = 255
TCP_PASSIVE_OPEN = 0

// protocolos (modo do ipconnect)
IPV4_TCP = 1
IPV4_UDP = 2
IPV4_ICMP = 3

// modos UDP
UDPMODE_ASC = 1
UDPMODE_CKSUM = 2

// códigos de erro
ECONTIMEOUT = 080H
ECONREFUSED = 081H
ENOPERM = 082H
ENOPORT = 083H
ENOROUTE = 084H
ENOSOCK = 085H
ENOTIMP = 086H
EPROT = 087H
EPORTINUSE = 088H

// estados permitidos para sockstatus em sockinfo_t
TCP_CLOSED = 000H
TCP_LISTEN = 001H
TCP_SYN_SENT = 042H
TCP_SYN_RECEIVED = 043H
TCP_ESTABLISHED = 0C4H
TCP_FIN_WAIT1 = 045H
TCP_FIN_WAIT2 = 046H
TCP_CLOSE_WAIT = 087H
TCP_CLOSING = 008H
```

```
TCP_LAST_ACK      = 009H
TCP_TIMEWAIT      = 00AH
UDP_LISTEN        = 091H
UDP_ESTABLISHED   = 094H

ip_struct_t = { uchar remote_ip[4],
                 uint remote_port,
                 uint local_port }

icmpdata_t = { uchar type,
                uchar icmpcode,
                unsigned long unused,
                uchar data[28], /* pad para 64 bytes */
                uint len;
                uchar sourceIP[4],
                uchar ttl }

tcpinfo_t = { uchar IP[4],
               uchar dns1ip[4],
               uchar dns2ip[4],
               char datalink[5],
               char domainname[DOMSIZE=128],
               int used_sockets,
               int avail_sockets,
               int used_buffers,
               int avail_buffers,
               int IP_chksum_errors }

sockinfo_t = { int localport,
               int remoteport,
               uchar remote_ip[4],
               char socketstatus, /* bit 7: permissao
                                   de escrita
                                   bit 6: estado de
                                   listen
                                   bits 3-0: estado
                                   */
               char sockettype,    /* TCP=1, UDP=2 */
               char sockerr,       /* codigo de erro */
               int pid }
```

### 7.2.7– Códigos de erro

As chamadas de sistema do Uzix retornam um valor maior que 0 em caso de sucesso e menor que 0 em caso de erro. O código de erro é colocado na variável global (definida no stub dos programas Uzix) **erro-no**. Abaixo estão relacionados os possíveis códigos de erro.

EPERM	1 Operation not permitted
ENOENT	2 No such file or directory
ESRCH	3 No such process
EINTR	4 Interrupted system call
EIO	5 I/O error
ENXIO	6 No such device or address
E2BIG	7 Arg list too long
ENOEXEC	8 Exec format error
EBADF	9 Bad file number
ECHILD	10 No child processes
EAGAIN	11 Try again
ENOMEM	12 Out of memory
EACCES	13 Permission denied
EFAULT	14 Bad address
ENOTBLK	15 Block device required
EBUSY	16 Device or resource busy
EXIST	17 File exists
EXDEV	18 Cross-device link
ENODEV	19 No such device
ENOTDIR	20 Not a directory
EISDIR	21 Is a directory
EINVAL	22 Invalid argument
ENFILE	23 File table overflow
EMFILE	24 Too many open files
ENOTTY	25 Not a typewriter
ETXTBSY	26 Text file busy
EFBIG	27 File too large
ENOSPC	28 No space left on device
ESPIPE	29 Illegal seek
EROFS	30 Read-only file system
EMLINK	31 Too many links

EPIPE	32	Broken pipe
EDOM	33	Math argument out of domain of func
ERANGE	34	Math result not representable
EDEADLK	35	Resource deadlock would occur
ENAMETOOLONG	36	File name too long
ENOLCK	37	No record locks available
EINVFNC	38	Function not implemented
ENOTEMPTY	39	Directory not empty
ELOOP	40	Too many symbolic links encountered
ESHELL	41	It's a shell script
ENOSYS		EINVFNC

### 7.3 – SYMBOS

O nome SymbOS vem de SYmbiosis Multitasking Based Operating System e é um sistema operacional multitarefa para sistemas baseados no Z80. É baseado em um microkernel, que fornece multitarefa preemptiva e orientada por prioridade e gerencia até 1024 KB de RAM. Tem uma interface gráfica (GUI) semelhante ao Microsoft Windows, suporta discos rígidos com capacidade de até 128 GB usando FAT32. Está disponível para máquinas Amstrad CPC, Amstrad PCW, CPC-TREX, C-ONE, Enterprise 64/128 e, é claro, para todos os modelos MSX2 ou superior com suporte a V9990.

O gerenciamento de memória divide a RAM em pequenos blocos de 256 bytes, que podem ser atribuídos dinamicamente. Os aplicativos estão sempre sendo executados em um banco secundário de 64K, onde nenhum espaço de memória é ocupado pelo sistema operacional ou pela memória de vídeo, sendo possível reservar até 63K simultaneamente. O gerenciamento de memória permite ao SymbOS acessar até 1 Mbyte de RAM.

A comunicação entre as diferentes tarefas e o sistema operacional geralmente é feita por meio de mensagens. Isso é necessário dentro de um ambiente multitarefa para evitar problemas de organização com a pilha, variáveis globais e recursos compartilhados do sistema. O kernel do SymbOS suporta IPC síncrono e assíncrono.

### 7.3.1 – Biblioteca do sistema SymStudio

Devido ao ambiente multitarefa, a maioria das funções do sistema só pode ser acessada através de mensagens de processo. Para facilitar o acesso às funções do sistema, existem bibliotecas especiais incluídos no SymStudio. Nesta documentação para cada função que você encontrará uma referência à biblioteca, se a função puder ser acessada por ela.

Elas podem ser usadas sem o SymStudio, mas este oferece a vantagem de montar só as partes, fazendo com que o código fique mais curto.

### 7.3.2 – Configuração do sistema

Todas as configurações do usuário no SymbOS são armazenadas no arquivo “SYMBOS.INI”. Este capítulo descreve o conteúdo da configuração do sistema e suas funções, permitindo acessar e manipular as configurações. O usuário fará isso geralmente com o painel de controle. As informações a seguir serão úteis apenas para escrever seu código próprio ou se precisar de algumas variáveis para sua aplicação.

O arquivo SYMBOS.INI é dividido em 5 partes:

- Cabeçalho, que contém o identificador e o comprimento das três partes seguintes;
- Parte da área do núcleo, que contém dados carregados no primeiro banco de RAM;
- Parte da área de dados, que contém dados adicionais geralmente carregados em um banco secundário;
- Parte da área de transferência (atualmente vazia);
- Fonte.

Existe uma função do System Manager, que permite alterar a configuração e descobrir seus endereços de memória. Esta função é a SYSINF (8103H) e está detalhadamente descrita no apêndice, capítulo 5, item 5.5.4 – Funções do Gerenciador de Sistema.

### 7.3.3 – Kernel

O kernel é o coração do SymbOS e controla os recursos principais do sistema. Ele é dividido nos módulos “multitarefa”, “memória”,

“banco” e “mensagem”. É também a interface entre os aplicativos e todas as partes do sistema operacional.

### 7.3.3.1 – Acesso ao Kernel

O Kernel SymbOS acessa as mesmas entradas RST que a BIOS e o Kernel do MSXDOS, porém com funções diferentes. Elas são as seguintes:

- RST 08H (MSGSLP) – Verifica se há uma nova mensagem de outro processo.
- RST 10H (MSG SND) – Envia uma mensagem para outro processo.
- RST 18H (MSGGET) – Verifica se há uma nova mensagem de outro processo.
- RST 20H (BNKSCL) – Chama uma rotina, que é colocada no primeiro banco de RAM.
- RST 28H (BNKFCL) – Chama uma rotina, que é colocada no primeiro banco de RAM. É mais rápida do que RST 20H (BNKSCL).
- RST 30H (MTSOFT) – Libera o tempo de CPU para o sistema.
- RST 38H (MTHARD) – Entrada do manipulador de interrupção do Z80, que é chamada 50, 60 ou 300 vezes por segundo. Não deve ser chamada pelo usuário.

Estas rotinas estão detalhadamente no apêndice, capítulo 5, seção 5.1.1 – Kernel Restarts.

### Comandos e Respostas do Kernel (Gestão multitarefa)

Os comandos do Kernel são acionados por meio de uma mensagem, que deve ser enviada com RST #10 (MSG SND) para o processo do kernel. O processo do kernel sempre tem o ID 1. A lista completa dos comandos está detalhada no apêndice, capítulo 5, item 5.1.2 – Comandos do Kernel.

As respostas do Kernel vêm na forma de mensagens, que devem ser recebida com RST 18H (MSG SND) ou RST 08H (MSG SLIP) do processo do Kernel, que sempre tem o ID 1. A lista completa das mensagens está detalhada no apêndice, capítulo 5, item 5.1.3 – Respostas do Kernel.

## Gerenciamento de Memória / Bancos de Memória

A maioria das funções de banco e de memória do kernel devem ser chamadas com RST #20 (BNKSCL) ou RST #28 (BNKFCL), dependendo da função. Muitas funções só podem ser chamadas uma vez ao mesmo tempo, então elas são protegidas com um mecanismo de sinalização. O processo de chamada será alterado para o modo inativo, enquanto a função estiver trabalhando para outro processo. O SymbOS pode manipular até 16 bancos de 64K, o que permite um total de 1 Mbyte de RAM.

A lista completa das funções relacionadas à memória está disponível no apêndice, capítulo 5, seção 5.1.4 – Funções do Kernel (Gerenciamento de Memória)

### Mapa de Memória

O diagrama a seguir mostra de que forma os diferentes bancos e blocos de memória são usados no SymbOS.

	Banco 0	Banco 1	Banco n
FFFFH	Dados e gerenciador de sistema	Livre	Livre
C000H		Livre	Livre
BFFFH	Buffers Subrotinas Ger. Disp. Ger. Tela	Livre	Livre
8000H		Livre	Livre
7FFFH	Gerenciador da Área de Trabalho	Livre	Livre
4000H		Livre	Livre
3FFFH	Ger. área trab., sistema e arquivo – LL	Gerenciador de arquivo – HL	Livre
0000H	Kernel/jumpers	Kernel jumps	Kernel jumps

## Mapa de memória do aplicativo

A memória dentro de um banco de RAM de aplicativo (1 – n) é usada da seguinte maneira:

1. 0000–03FF Kernel jumps, multitarefa do Kernel e rotinas de gerenciamento de bancos.
2. 0400–FFFF Código e dados internos do aplicativo.
3. 0400–3FFF Dados do aplicativo usados pelo gerenc. de tela.  
4000–7FFF Um objeto deve estar dentro de um bloco de 16K.  
8000–BFFF  
C000–FFFF
4. Dados de “transferência” do aplicativo C000–FFFF, usados pelo gerenciador de área de trabalho, buffer de mensagens e pilha.

## Configurações de memória

O diagrama a seguir mostra como a memória é configurada durante a atividade de um dos módulos do SymbOS.

	área de trabalho-Manager (C1)	ScreenManager (C4-7)	FileManager-HL (C4)
FFFFH	Banco n Bloco 3 Transfer RAM	Banco 0 Bloco 3	Banco 0 Bloco 3
C000H		Banco 0 Bloco 2 ScreenManager	Banco 0 Bloco 2
BFFFH	Banco 0 Bloco 2		
8000H			
7FFFH	Banco 0 Bloco 1 área de trabalho-Manager	Banco n Bloco m RAM de dados	Banco 1 Bloco 0 FileManager-HL
4000H		Banco 0 Bloco 0	
3FFFH	Banco 0 Bloco 0		Banco 0 Bloco 0
0000H		Banco 0 Bloco 0	

	FileManager-LL (**)	Application (C2)
FFFFH	Banco 0 Bloco 3	Banco n Bloco 3 Trnf, Code, Data
C000H		
BFFFH	Slot x,y Disk-ROM	Banco n Bloco 2 Code, Data
8000H		
7FFFH	Banco n Bloco m	Banco n Bloco 1 Code, Data
4000H	RAM de dados	
3FFFH	Banco 0 Bloco 0	Banco n Bloco 0 Code, Data
0000H	FileManager-LL	

### 7.3.4 – Rede (Network Daemon)

O daemon de rede SymbOS fornece todos os serviços para acesso total à rede, a partir da versão 3.0.

O aplicativo Network Daemon habilita o SymbOS para rede baseada em TCP/IP completo e acesso à Internet, incluindo várias conexões para vários aplicativos ao mesmo tempo. Ambos os protocolos TCP e UDP dentro da camada de transporte são suportados, bem como serviços como DHCP e DNS. O Network Daemon é executado como um serviço em segundo plano e pode ser usado por qualquer aplicativo por meio de sua API de rede.

O hardware suportado inclui DenYoNet (MSX), GR8NET (MSX) e M4Board (CPC) para acesso via ethernet ou wifi.

Uma versão especial de host local do Network Daemon possibilita a execução de todos os aplicativos de rede, mesmo em máquinas sem hardware de rede adicional.

A descrição das mensagens de acesso à rede estão detalhadamente descritas no apêndice, capítulo 5, seção 5.12 – Daemon de Rede.

### 7.3.5 – Gerenciador de Tela

Atualmente, o gerenciador de tela contém apenas uma função para acesso direto ao vídeo. Esta função também pode ser usada por aplicativos. Ela deve ser chamada através de RST #20 (BNKSCL).

#### TXTLEN (815DH) – Screen\_TextLength

**Descrição:** Retorna a largura e a altura de uma linha de texto em pontos, caso seja impresso na tela. O comprimento do texto (número de caracteres) pode ser definido em IY. Se o texto terminar em 0 ou 13, deve ser usado -1 para o comprimento. Observe, que esta função sempre usa a fonte do sistema para calcular a largura e a altura.

**Como chamar:**   rst 20H : dw 815DH

**Entrada:**   HL – Endereço de texto.

    A – Banco de memória do texto (1~15).

    IY – Número máximo de caracteres (comprimento).

**Saída:**   DE – Largura do texto em pontos.

    A – Altura do texto em pontos.

**Registradores:**   F, BC, HL, IX.

### 7.3.6 – Protetor de Tela

O gerenciador de área de trabalho do SymbOS suporta o manuseio de aplicativos de proteção de tela. São programas especiais, que são carregados automaticamente durante a inicialização ou ao entrar no modo de suspensão. Se o sistema estiver ocioso por um período de tempo (sem atividade de mouse e teclado), o gerenciador de área de trabalho envia uma mensagem para o aplicativo de proteção de tela para executá-lo.

Este capítulo descreve a implementação de um aplicativo de proteção de tela e as mensagens usadas para comunicação entre o sistema e o protetor de tela.

Um aplicativo de proteção de tela pode ser executado de três modos diferentes:

- **Modo de operação:** O gerenciador de área de trabalho carrega e iniciaiza o protetor de tela. O sistema permanece suspenso até o gerenciador da área de trabalho enviar mensagem de “start”.

- Modo de configuração: O painel de controle carrega o protetor de tela temporariamente e envia mensagens “start” para teste ou mensagens “config” para propósitos de configuração.
- Modo de demonstração: O protetor de tela é carregado manualmente pelo usuário. Neste caso ele inicia imediatamente. Quando uma tecla for pressionada ou o mouse for movido, o protetor de tela é encerrado.

Para atender a essas solicitações, o protetor de tela deve executar a seguinte sequência de operações:

- 0 – O aplicativo de proteção de tela é iniciado;
- 1 – Aguardar a mensagem inicial;
- 2 – Se a mensagem não puder ser recebida:
  - Iniciar a fase de animação;
  - Aguardar a tecla ou o mouse;
  - Ir para 5.
- 3 – Checar a mensagem recebida:
  - a – Se a mensagem de comando “start” (consulte MSC\_SAV\_INIT) foi recebida
    - Receber dados de configuração;
    - Se os dados não forem válidos, use os dados de configuração padrão;
    - Inicializar o protetor de tela.
  - b – Se a mensagem de comando “config” for recebida (consulte MSC\_SAV\_CONFIG):
    - Abrir a janela de configuração;
    - Deixar o usuário fazer as modificações na configuração;
    - Enviar mensagem de resposta “config” (ver MSR\_SAV\_CONFIG) com os dados de configuração atualizados para o processo que enviou o MSC\_SAV\_INIT.
  - c – Se a mensagem de comando “start” (consulte MSC\_SAV\_START) for recebida:
    - Iniciar a fase de animação;
    - Aguarde a resposta do teclado ou do mouse.
  - d – Se a mensagem “quit” (0) foi recebida:
    - Vá para 5.
- 4 – Aguarde a próxima mensagem enquanto estiver no modo de suspensão e, em seguida, vá para 3.
- 5 – Sair do aplicativo.

Os comandos e respostas para acesso ao protetor de tela estão detalhadamente descritos no apêndice, capítulo 5, item 5.9 – Aplicativos do Protetor de Tela.

### 7.3.7 – Terminal de Texto (SymShell)

O aplicativo de sistema SymShell fornece um ambiente de programa com interface de usuário baseada em texto. Aplicativos, que estão trabalhando dentro do ambiente SymShell, podem usar rotinas de entrada e saída de terminal para enviar e receber dados de texto de/para o console padrão. Pode também ser redirecionado para outras origens e destinos.

Os comandos do SymShell são acionados por meio de mensagens, que devem ser enviadas com RST#10 (MSGSEND) para o processo SymShell. O SymShell passará seu ID de processo e a resolução de tela de texto para o aplicativo por meio da linha de comando.

Uma linha de comando tem o seguinte formato:

```
[drive:/path/filename.com] (parâmetros) %spPPXXYYVV
```

A string de 256 bytes é sempre colocada logo após o último byte da área de código de um aplicativo. A lista de parâmetros do usuário deve vir entre parênteses. Logo após, é necessário colocar um parâmetro adicional que deve ser lido pelo aplicativo. Ele inicia com "%sp" e tem a seguinte estrutura (todos os números devem estar no formato ASCII e representam 2 dígitos decimais para cada valor):

PP – Número do processo SymShell. A aplicação deve saber o número do processo, para o qual tem que enviar os comandos de entrada e saída de texto.

XX – Largura da janela do terminal de texto em caracteres.

YY – Altura da janela do terminal de texto em caracteres. Essas duas informações podem ser usadas para reformatação de saída de texto.

VV – Versão do SymShell; o primeiro dígito é a versão maior, o segundo é a versão menor

Exemplo:

```
c:\symbol\symshell\quizgame %level:9 %sp07 602021
```

A aplicação "quizgame.com" será executada no caminho "c:\symbol\symshell". O usuário passou o parâmetro "%level:9" para ela. A versão do SymShell é 2.1, ele é executado com o ID de processo 7 e a janela do terminal de texto tem um tamanho de 60x20 caracteres.

É importante ressaltar que, se não houver o parâmetro '%sp', o aplicativo provavelmente não foi iniciado dentro do SymShell e deve sair sozinho, pois não pode usar os recursos do terminal de texto.

Os comandos aceitos pelo SymShell são os seguintes:

ATTRIB	Apresenta atributos atuais do arquivo.
CD	Apresenta ou troca o subdiretório corrente.
CLS	Limpa a tela.
COPY	Copia arquivos. Aceita o caractere coringa '*'.
DATE	Apresenta ou altera a data do sistema.
DEL	Deleta um ou mais arquivos. Aceita o caractere coringa '*'.
DIR	Apresenta os nomes dos arquivos do disco.
HELP	Apresenta o arquivo de ajuda.
MKDIR	Cria um subdiretório.
MOVE	Move arquivos para outra parte do disco.
RMDIR	Remove um subdiretório vazio.
REN	Renomeia o arquivo <nome antigo> com <nome novo>.
TIME	Apresenta ou altera a hora do sistema.

### **Aplicativos padrão**

DIMON.COM – Monitor de disco de linha de comando.

NETSTAT.COM – Exibe as conexões de rede ativas com o estado.

TELNET.COM – Cliente telnet.

UNZIP.COM – Extrai arquivos .ZIP ou .GZ.

WGET.COM – Aplicativo para download de arquivos via HTTP.

Todos os comandos estão detalhadamente descritos no apêndice, capítulo 5, seção 5.7.1 – Comandos do terminal de texto.

### 7.3.8 – Gerenciador de sistema

O gerenciador do sistema é responsável por iniciar e parar aplicativos e executar os trabalhos gerais do sistema. Presta diversos serviços de diálogo e é proprietário do gerenciador de arquivos, que só pode ser acessado através do processo do gerenciador do sistema.

Os comandos do gerenciador de sistema são acionados por meio de uma mensagem, que deve ser enviada com RST #10 (MSG SND) para o processo do gerenciador de sistema que sempre tem o ID 3.

Todos os seus comandos estão detalhadamente descritos no capítulo 5, seção 5.5 – Gerenciador de sistema.

### 7.3.9 – Aplicativos

Em geral os aplicativos são programas executáveis, que podem ser iniciados no ambiente multitarefa do SymbOS, que podem ser de dois tipos.

O primeiro tipo são os aplicativos baseados em GUI com a extensão “\*.EXE” e usam o gerenciador de área de trabalho como interface para o do usuário. Os segundos tipo são aplicativos baseados em terminal de texto (\*.COM), que requerem o SymShell como uma interface para o usuário.

A estrutura de ambos os tipos é idêntica, e a única diferença é que o shell dos aplicativos adicionalmente podem usar o terminal SymShell para entrada de texto e saída do resultado.

#### 7.3.9.1 – Tipos de área de memória

Antes de começarmos com a descrição da estrutura dos aplicativos, é preciso conhecer os três tipos de memória existentes no SymbOS.

Como o barramento de endereços da CPU Z80 tem 16 bits, a CPU não pode endereçar mais de 64 K (65 536 bytes) ao mesmo tempo. Para poder trabalhar com mais de 64K é necessário o uso de ‘bancos’ o que significa que partes da memória serão visíveis dentro do espaço de endereço de 64K do Z80, enquanto outras partes serão ocultadas. Isso também é conhecido como mapeamento de memória.

Inicialmente o SymbOS foi desenvolvido para os computadores Amstrad CPC. O CPC-6128 já tinham possibilidades de troca de banco bastante poderosas em comparação com outros computadores de 8 bits dos anos 80. No entanto, os métodos de comutação bancária do CPC ainda apresentam algumas limitações. Enquanto as possibilidades de outras máquinas como MSX2, Amstrad PCW ou Enterprise 128 permitem serviços bancários de 4x16 Kbytes muito flexíveis, o CPC é limitado a algumas configurações de banco especializadas. O SymbOS é baseado nestas limitações. Isso torna possível executar o microkernel SymbOS em muitas plataformas como CPC, MSX2+/TurboR, PCW, Enterprise e até mesmo o Sam Coupé

A RAM é dividida em partes de 64K, que são chamadas de “banhos”. Cada banco é dividido em quatro partes de 16K chamadas “blocos”. O CPC é limitado a quatro diferentes modos de comutação de banco. Três deles são usados pelo SymbOS, que podem também ser configurados em todas as outras plataformas suportadas.

O SymbOS consiste em vários módulos chamados "gerentes". A maioria das partes do sistema operacional são colocados no primeiro banco de 64K. As aplicações são sempre colocadas dentro de um banco de RAM secundário de 64K, que será ativado enquanto o aplicativo está em execução.

Para poder acessar a memória dos aplicativos, o sistema deve mudar partes de sua RAM para os 64K visíveis. Como as possibilidades de mudança de banco original são limitadas, nem todos os gerentes podem acessar todas as partes da memória do aplicativo:

**a.) Área de código:** A área de código pode ser colocada em qualquer lugar da memória. Por causa dessa flexibilidade, é mais fácil para o sistema operacional alocar memória desse tipo. Aqui deve ser colocado tudo que não precisa estar em outras áreas.

O kernel é capaz de acessar todas as partes da memória. O mesmo vale para o gerenciador do sistema, que usa apenas o kernel para acessar a memória do aplicativo. Isso significa que todas as funções e

dados de um aplicativo, que estão em contato com o kernel ou com o gerenciador de sistema, podem ser colocados em qualquer lugar da memória.

Esse tipo de memória é chamado de “área de código”, mas pode conter qualquer tipo de dados. Em geral, a área de código de um aplicativo contém todas as suas rotinas e dados adicionais, que não precisam ser colocados em uma das outras duas memória.

**b.) Área de dados:** A área de dados precisa ser colocada dentro de um bloco de 16K. Todos os textos e gráficos, que são plotados na tela, devem ser colocados aqui.

Por motivos de desempenho, uma única linha de texto ou um único gráfico deve ser colocado dentro de um bloco de 16K, para que o gerenciador de tela pode acessá-lo em uma peça. Esta áres contém todos os dados do aplicativo que são acessados pelo gerenciador de tela.

O gerenciador de tela também é capaz de acessar todas as partes da memória, mas apenas um bloco de 16K de cada vez. Sua função é desenhar textos, gráficos, linhas e caixas na tela. Ele faz todos os trabalhos de baixo nível para o gerenciador de área de trabalho.

**c.) Área de transferência:** A área de transferência precisa ser colocada no último bloco de 16K (#C000–#FFFF), já que, por causa de sua posição na memória e das limitações do banco original, o “gerenciador da área de trabalho” só pode acessar os últimos 16K de cada banco de RAM. Todos os dados do aplicativo que precisam ser acessados diretamente pelo gerenciador devem ser colocados nesta área. São os registros de dados de janela e de dados de controle. Também a pilha e o buffer de mensagem deve ser colocado nesta área, pois a função RST #28 (BNKF-CL) só pode ser usada com tal posição de pilha. Este tipo de memória é chamado de “área de transferência”, pois é a única parte, cujos dados podem ser transferidos para o gerenciador da área de trabalho e ao módulo de mensagens do kernel com o uso rápido de comutação de banco.

Observe que o aplicativo em si não precisa se preocupar com as posições das áreas de memória. Isso é feito pelo gerenciador do sistema, quando aloca memória durante o carregamento do aplicativo e pelo kernel, quando o aplicativo deseja alocar memória adicional de um dos três tipos.

### 7.3.9.2 – Estrutura dos aplicativos

Cada aplicação é subdividida em quatro partes. As três primeiras são as principais de acordo com os três tipos de memória diferentes. A quarta contém a tabela de realocação e não será mantida na memória após o processo de inicialização.

A parte da área de código do aplicativo sempre começa com um cabeçalho de 256 bytes, que contém todas as informações necessárias que o gerenciador de sistema precisa saber para carregar e inicializar o aplicativo. A estrutura de dados do cabeçalho é descrito abaixo.

Após a inicialização do aplicativo, o gerenciador de sistema escreve algumas informações úteis no cabeçalho como o número do banco de RAM e o ID do processo. A parte da área de transferência de um aplicativo deve sempre começar com a pilha, cujo comprimento precisa ser escrito no cabeçalho do aplicativo. Isso permite que o gerenciador informe kernel onde iniciar o processo principal do aplicativo.

As três partes de um aplicativo devem ser colocadas imediatamente umas após as outras. O gerenciador de sistema irá dividi-los e colocar cada um deles em áreas apropriadas da memória.

A quarta parte contém a tabela realocadora, que é uma lista de apontadores (palavras) para os endereços de referência para permitir que o código do aplicativo possa ser realocado. A tabela é temporária, sendo removida após o processo de realocação.

#### **Cabeçalho do aplicativo**

A estrutura do cabeçalho do aplicativo está ilustrada abaixo.

000 02	Comprimento da área de código.
002 02	Comprimento da área de dados.
004 02	Comprimento da área de transferência.
006 02	Origem original do código montador.
008 02	Número de entradas na tabela de realocadores.
010 02	Comprimento da pilha em bytes.
012 02	*Reservado* (devem ser 0).
014 01	*Reservado* (deve ser 0)
015 24	Nome do aplicativo. A string deve terminar com 0. '0' (terminador).
049 01	Flags (+1=ícone de 16 cores incluído).
041 02	16 cores de deslocamento de arquivo de ícone.
043 05	*Reservado* (devem ser 0).
048 08	"SymExe10" (Identificação de arquivo executável SymbOS)
056 02	Tamanho em bytes da área de código reservada (adicional).
058 02	Tamanho em bytes da área de dados reservada (adicional).
060 02	Tamanho em bytes da área de transf. reservada (adicional).
062 26	*Reservado* (deve ser 0).
088 02	Versão mínima do sistema operacional (secundária/principal).
090 19	Ícone do aplicativo (versão pequena), 8x8 pixels, formato gráfico SymbOS
109 147	Ícone do aplicativo (versão grande), 24x24 pixel, formato gráfico SymbOS

As palavras 0, 2 e 4 contêm o comprimento das partes da memória estática. Se for necessária mais memória no banco de RAM em que o aplicativo está sendo executado, ela deve ser alocada durante o processo de carregamento. Em vez de adicionar buffer no próprio código do programa, o que desperdiçaria memória no disco, podem ser alocadas áreas adicionais de código, dados e transferência do com as palavras 56, 58 e 60.

Não devem ser usados apontadores para acessar a área adicional (estendida) uma vez que o realocador pensaria que você quer acessar a memória em outra área, pois o ponteiro está fora da parte estática. É necessário primeiro calcular o endereço dentro do código.

Depois que o aplicativo for inicializado, algumas partes do cabeçalho são substituídos por novos dados, conforme abaixo.

- 006 02 Endereço da área de dados.
- 008 02 Endereço da área de transferência.
- 010 04 IDs de subprocessos ou temporizadores adicionais; 4 IDs de processo/temporizador podem ser registrados aqui.
- 014 01 Número do banco RAM (1~15) onde o aplicativo está localizado
- 048 40 Áreas de memória adicionais; até 8 áreas de 5 bytes podem ser registradas aqui, com a estrutura descrita abaixo:
  - 00 01 Número do banco RAM (1~15; se 0, a entrada será ignorada).
  - 01 02 Endereço.
  - 03 02 Comprimento.
- 088 01 ID do aplicativo.
- 089 01 ID do processo principal.

O ID do processo do aplicativo é necessário para enviar mensagens para outros processos e para quando o aplicativo precisar encerrar a si próprio. O número do banco de RAM é necessário para todo tipo de chamadas de sistema onde um ponteiro de memória é necessário. Os bytes em 10 e 48 devem ser usados para registrar áreas de memória adicionais alocadas, IDs de subprocessos e temporizadores.

### **Caminho e parâmetros anexados**

Quando o gerenciador de sistema carrega um aplicativo, ele anexa 256 bytes logo após o final da área de código, contendo o comando completo linha, com a qual o aplicativo foi iniciado. A string anexada é semelhante à seguinte:

```
B:\symbos\apps\symsee.exe A:\pics\mycat.scr -full
```

Assim, o aplicativo pode saber onde está e carregar seu arquivo de configuração do mesmo local. Pode também ler quaisquer parâmetros adicionados ao comando.

### **Uso de registro da CPU**

Todo processo pode usar apenas o conjunto de registradores primários (AF,BC,DE,HL,IX,IY). O conjunto de registradores secundários é usado pelo gerenciador de multitarefa, e o registrador I é usado para ar-

mazenar a configuração de banco atual. Tais registradores não devem ser usados pelo aplicativo.

### **7.3.10 – Menus pulldown**

Podem ser definidos até 8 níveis de submenu. O “Window Data Record” aponta para o nível de menu mais alto. São as entradas que você vê na barra de menus de uma janela. Essas entradas geralmente apontam para seus submenus, que também contêm entradas clicáveis ou que apontam para um novo submenu adicional.

### **7.3.11 – Gráficos**

#### **7.3.11.1 – Gráficos padrão**

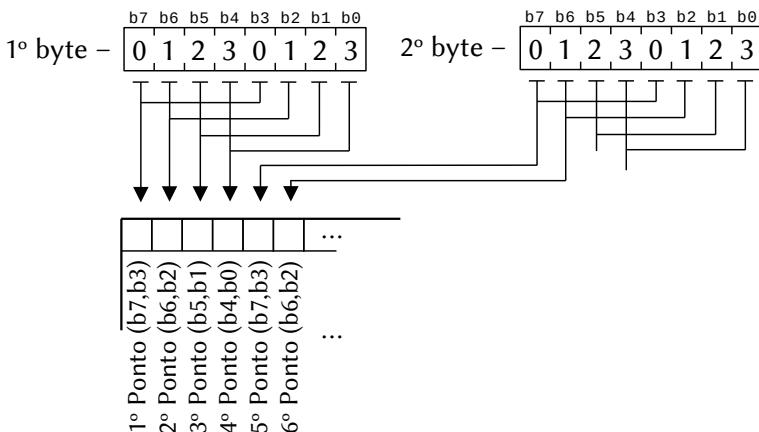
Um gráfico padrão SymbOS tem 4 cores com um tamanho máximo de 255x255 pontos. Cada objeto gráfico começa com um cabeçalho de 3 bytes:

- 00 1B [bit0~6] Largura do gráfico em bytes.  
[bit7] Tipo de codificação (0=CPC, 1=MSX).
- 01 1B Largura do gráfico em pontos.
- 02 1B Altura do gráfico em pontos.

Logo após o cabeçalho, a quantidade de [largura\_em\_bytes] \* [altura\_em\_pontos] bytes contém os dados gráficos. Cada gráfico é armazenado linha por linha como um sprite.

Os pontos devem ser codificados no formato CPC (Modo 1). Gráficos em um sistema MSX serão automaticamente convertidos para o formato MSX, quando forem exibidos primeira vez. O bit 7 do byte 0 do cabeçalho contém o formato de codificação atual. Note que não é permitido armazenar um gráfico original no formato MSX, uma vez que o sistema CPC não é capaz de lidar com esses gráficos.

A seguir está uma descrição do formato de codificação CPC. Cada byte contém 4 pontos:



Apenas aplicativos, que precisam modificar um gráfico após ele ter sido exibido na primeira vez devem se preocupar com o tipo de codificação e com o formato MSX.

### 7.3.11.2 – Gráficos com cabeçalho estendido

Como a largura de um gráfico é limitada a 255 pontos, não seria possível armazenar uma tela completa (como 320 x 200 no modo CPC 1) em um único bloco. A tela precisa ser dividida em duas partes (por exemplo, 2 x 160 x 200), o que a torna muito difícil escrever rotinas de modificação gráfica.

Os gráficos estendidos não têm essa limitação e também permitem mais de 4 cores. Eles só podem ser usados para o ID de controle 10, “graphic\_extended”. Um gráfico pode ser armazenado em uma única peça com largura de até 1020 pontos. O controle “graphic\_extended” então é capaz de exibir uma parte de um grande armazenamento linear gráfico.

A estrutura do cabeçalho estendido é a seguinte:

- 00 1B Largura do gráfico completo em bytes (deve ser um valor par!)
- 01 1B Largura da área gráfica, que deve ser exibida, em pixel
- 02 1B Altura da área gráfica, que deve ser exibida, em pixel
- 03 1W Endereço dos dados gráficos, incluindo o deslocamento da área
- 05 1W Endereço do byte de informação de codificação (este byte deve ser SEMPRE colocado imediatamente antes dos dados gráficos).
- 07 1W Tamanho do gráfico completo

## ?? 1B Informação de codificação

bit0-1: Codificação de cor (0 → CPC, 1 → MSX)

bit2-3: Profundidade de cor (0 → 4 cores, 1 → 16 cores)

Apenas os dois valores iniciais a seguir são permitidos:

0 → 4 cores, formato CPC; um sistema MSX irá converter o gráfico para o formato MSX, quando for mostrado pela primeira vez.

5 → 16 cores, formato MSX; um sistema CPC e PCW irá processar o gráfico completo para 4 cores (formato CPC), quando for exibido pela primeira vez

## ??+1 x dados gráficos

O cabeçalho gráfico não precisa ser armazenado imediatamente antes dos dados do gráfico, ele só precisa estar localizado na mesma área de 16K como do gráfico. Este tipo de gráfico pode ser usado:

- Se o gráfico for maior que 255 pixels;
- Se for exibir apenas uma parte do gráfico;
- Se o cabeçalho não puder ser armazenado imediatamente antes dos dados do gráfico;
- Se forem usados gráficos de 16 cores.

Em qualquer outro caso devem ser usados gráficos padrão, pois eles são mais rápidos. Abaixo está ilustrado um exemplo para um gráfico de 320 x 200:

```
encoding_type: db 0 ;= formato CPC de 4 cores
                ;** deve ser colocado diretamente em **
                ;** frente dos dados gráficos! **

graphic_data: db x,x,x,x,x,x,...x
                ; ↑ linha 1, inclui 80 bytes = 320 pixels
db x,x,x,x,x,x,...x
                ; ↑ linha 2
                [...]
db x,x,x,x,x,x,...x ;linha 200

graphic_header_for_area_1:
                db 80
                ;80 * 4 = total de 320 pixels
```

```

        db 160
        ;esta área tem apenas 160 pixels de largura
        db 200 ;altura é 200 pixels
        dw graphic_data
        dw encoding_type
        dw 80*200

graphic_header_for_area_2:
        db 80 ;o mesmo que acima...
        db 160
        db 200
        dw graphic_data+40      ;...mas esta área começa
                                ; no byte offset 40
        dw encoding_type
        dw 80*200

```

O gráfico em si ("graphic\_data") é armazenado em um único bloco na memória (sem cabeçalho). Então temos dois cabeçalhos ("graphic\_header\_for\_area\_1" e "graphic\_header\_for\_area\_2") que estão apontando para duas áreas diferentes do gráfico completo. Como pode ser visto, ainda são necessários dois controles para exibir o gráfico, mas os dados em si não precisam ser divididos.

### 7.3.12 – Fontes

Uma fonte define a aparência dos caracteres usados para imprimir textos no SymbOS. Uma fonte começa com um cabeçalho simples de 2 bytes, seguido pela máscara de bits dos caracteres. Seu formato é o seguinte:

- 00 1B Altura de cada caractere em pontos (1~15, padrão: 8).
- 00 1B Primeiro caractere da fonte (0~255).
- 00 1B Largura do primeiro caractere em pontos.
- 01 1B Máscara de bits da 1<sup>a</sup> linha do primeiro caractere.
- 02 1B Máscara de bits da 2<sup>a</sup> linha do primeiro caractere.
- ⋮
- 15 1B Máscara de bits da 15<sup>a</sup> linha do primeiro caractere.
- 16 1B Largura do segundo caractere em pontos.
- 17 1B Máscara de bits da 1<sup>a</sup> linha do segundo caractere.
- ⋮

### 7.3.13 – Gerenciador de dispositivos

O gerenciador de dispositivos controla todas as partes do hardware. Em geral os aplicativos não se comunicam diretamente com o gerenciador de dispositivos, pois outras partes do sistema operacional estão entre o gerenciador de dispositivos e o aplicativo, como o gerenciador de sistema e o gerenciador de área de trabalho.

Portanto, existem apenas algumas funções do gerenciador de dispositivos, que podem ser chamadas e usadas diretamente. Todas as funções do gerenciador de dispositivos estão detalhadamente descritas no apêndice, capítulo 5, seção 5.6.6 – Funções do Gerenciador de Dispositivos.

### 7.3.14 – Gerenciador de arquivos

O gerenciador de arquivos é controlado pelo gerenciador do sistema, que é o único que tem permissão para chamar as funções do gerenciador de arquivos. Se um aplicativo quiser usar o gerenciador de arquivos, ele precisa enviar uma mensagem para o processo do gerenciador do sistema, que chamará a função de gerenciador de arquivos e devolverá uma mensagem com o resultado para o aplicativo chamador.

O processo do gerenciador do sistema sempre tem o ID 3.

Observe que no SymbOS todos os textos devem terminar com byte 0. Assim, os caminhos e nomes de arquivos usados no gerenciador de arquivos também devem terminar da mesma forma.

### 7.3.15 – Gerenciador de diretório

As funções de gerenciamento de diretório permitem mostrar e editar o conteúdo de um diretório. Muitas funções requerem um caminho com ou sem um nome de arquivo. Estes caminhos devem ter um tamanho máximo de 255 bytes. Eles são construídos assim:

```
[Drive:] [\] [dir1\] [dir2\] [...] [filename or filemask]
      A       B       C           D
```

[A] Se a unidade for especificada, não será usado o dispositivo ativo, mas um especificado. Se o dispositivo não estiver incluído no caminho, o sistema pesquisa no dispositivo atual.

[B] Se o próprio caminho começar com um "\", o sistema começa a trabalhar a partir do diretório raiz. Se não houver "\" no início, o sistema inicia pesquisando no diretório atual.

[C] Você pode entrar em quantos subdiretórios quiser. Você também pode mudar para o diretório pai ("..") ou o atual ("."). Na sequência exemplo a seguir, você permanecerá em seu diretório atual:

```
new_path = "subdir\..\..\subdir\.."
```

[D] Se você deseja especificar um diretório, o caminho deve terminar com um \. Se você quiser para especificar um nome de arquivo ou uma máscara de arquivo, você o anexa atrás do último \.

Uma máscara de arquivo pode conter ? (qualquer caractere neste local) e \* (qualquer caractere até o final da parte do nome do arquivo atual).

Em vez da barra invertida ("\", Microsoft), também pode ser usada a barra normal ("/", Unix, Linux).

Observe que a unidade e o caminho devem sempre ser especificados, pois o aplicativo está sendo executado em um ambiente multitarefa, e outros aplicativos podem alterar a unidade e caminho padrão. Devido aos recursos de memória limitados, o SymbOS não suporta instâncias de unidade/caminho adicionais para cada processo.

## 7.4 – WiOS

WiOS é um sistema operacional gráfico multitarefa para computadores MSX com V9990 VDP feito por Michael Stellmann. O WiOS é uma coleção de rotinas que suportam a programação de aplicações em ambiente gráfico. Permite um número compartilhado de até 252 drivers ou tarefas (que é um sinônimo para “aplicação”), gerencia até 3072 segmentos de memória e até 256 janelas.

Para poder funcionar, o WiOS necessita de:

- Mouse para mover o cursor.
- Placa gráfica baseada em V9990 na porta 60H.
- Pelo menos 160k de memória livre.
- MSX-DOS 2 ou superior.

Entretanto, para um desempenho confortável, deve ser usado um MSX turbo R com 512 Kbytes de memória e um disco rígido.

### **7.4.1 – Aplicativos para o WiOS**

A linguagem ASCII-C v1.2 é aconselhável, devido ao fato de que as chamadas de função devem ter a mesma estrutura. Consulte o arquivo READ.ME no “Application Creation Toolkit” para obter uma lista de embalagem completa

Não há necessidade de ter ASCII-C no mesmo diretório que o WiOS. Para encontrar os arquivos do compilador, você pode especificar o diretório no ambiente PATH ou coloque a unidade e o caminho completos antes dos arquivos .COM no arquivo de lote de compilação.

Para encontrar as bibliotecas C padrão fora do diretório atual, inclua-as com “#include <headfile.h>” e configure a variável de ambiente INCLUDE com a unidade e o diretório dos arquivos de cabeçalho.

Se você nunca trabalhou com WiOS (e com certeza nunca trabalhou com WiOS) é importante ler esta seção completamente para entender melhor a estrutura básica de como o WiOS funciona.

#### **7.4.1.1 – Estrutura do Programa**

A programação de aplicativos para WiOS é diferente da maneira usual de estruturação de programas. A principal diferença é o ambiente multitarefa. Há dois tipos principais:

**Multitarefa preemptiva** – Cada aplicativo recebe uma certa quantidade de tempo. O controle é dado e retirado do sistema. O grande problema é preservar o estado do que um aplicativo fez quando o controle foi tomado.

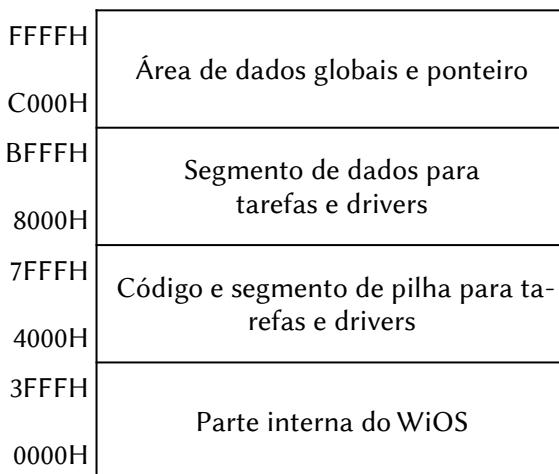
**Multitarefa Cooperativa** – Com este tipo, a velocidade do sistema depende da cooperação das tarefas. Cada tarefa tem o controle to-

tal do tempo de execução. Para oferecer a possibilidade de outras tarefas serem capazes de fazer algo em paralelo, cada tarefa deve deixar o controle em algum ponto da execução.

O WiOS usa multitarefa cooperativa, pois é mais fácil, rápido e com maior economia de memória.

### 7.4.2 – Estrutura de Memória

Para executar vários programas ao mesmo tempo, mais de 64K de memória deve ser acessível. Além disso, deve haver uma área fixa que esteja sempre presente para garantir que algumas funções, como o polling, possam ser chamadas de qualquer lugar. A estrutura de memória do WiOS é a seguinte:



#### 7.4.2.1 – Área de Dados Globais (GDA)

Aqui fica o manipulador de interrupção e a tabela fixa de pontos de entrada. Sempre que a parte interna do WiOS for alterada, os endereços corretos serão colocados no GDA. Este endereçamento indireto permite que outros programas accessem as funções mesmo que sua posição mude. A lista de ponteiros pode ser expandida no futuro, mas as entradas existentes não podem ser alteradas. O WiOS também usa esta área para sua própria pilha interna.

#### **7.4.2.2 – Segmento de Dados**

É aqui que a tarefa pode alternar livremente os segmentos de memória para armazenar dados. Não é aconselhável colocar código aqui, pois o comportamento do sistema será imprevisível

#### **7.4.2.3 – Código e Segmento de Pilha**

Cada tarefa tem 16K para código. Para maior rapidez, o ponteiro da pilha também é mantido neste segmento. Portanto, sempre que a rotina de pesquisa tiver que mudar para outro segmento para retornar a próxima tarefa, a respectiva pilha já está lá; apenas a posição da pilha deve ser corrigida.

#### **7.4.2.4 – Parte interna do WiOS**

Aqui estão as funções que devem ser acessíveis a partir de qualquer ponto e que processa o chaveamento no segmento de código e de dados. Os endereços de funções e variáveis podem variar em diferentes versões de WiOS.

#### **7.4.3 – Drivers**

Uma página de 16K é pequena, por isso apenas as partes necessárias são colocadas na página 0, como a rotina de pesquisa, variáveis globais, gerenciamento de memória e tarefas. Assim, o WiOS é modular, dividindo as funções em internas, que são acessadas diretamente, e externas que são acessadas indiretamente por meio de funções internas.

As funções externas são mapeadas na página 1. Como consequência, o WiOS comuta os segmentos de 16K para acessá-las. Isso possibilita um número quase ilimitado de funções externas, que são chamadas de *drivers*.

#### **Diferença entre tarefas e drivers**

Os drivers têm duas diferenças principais das tarefas. Eles não podem fazer polling, então as rotinas nas tarefas bloqueiam qualquer outra atividade desde quando executadas e não podem estar presentes na tela com janelas (o que decorre do fato de não poderem fazer polling).

Mas os drivers podem chamar outros drivers (como sub-rotinas) As funções externas são chamadas usando números, que devem ser definidos no próprio driver. Cada driver tem um ponto de entrada fixo para onde o chamador de função interno envia os argumentos da tarefa ou driver. Como cada driver pode usar os mesmos números de função que outros drivers eles são separados por alças (handles).

#### 7.4.3.1 – Alças (handles)

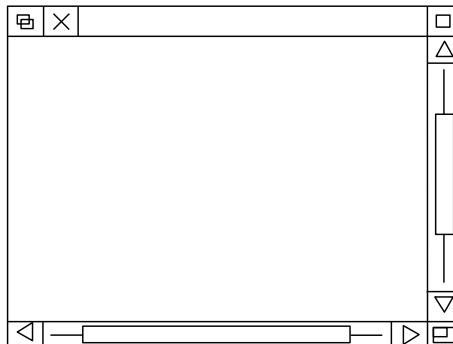
O gerenciador de memória do WiOS precisa salvar o usuário de um segmento para evitar tarefas/drivers liberem segmentos de outras tarefas/drivers. Para garantir isso, cada tarefa deve ter seu próprio código de identificação. Esta identificação é chamada de alça (handle) como identificadores de arquivo no DOS 2. Sempre que uma tarefa é carregada, o WiOS procura a próxima alça livre e a aplica à tarefa.

As alças de tarefa são pesquisadas do início ao fim, as alças do driver vice-versa. O número máximo de alças é 252 e a quantidade total de tarefas e drivers possíveis depende quantas tarefas ou drivers já estão instalados. O WiOS tem alguns drivers padrão cujas alças são armazenadas no GDA.

Cada tarefa/driver tem sua própria identificação, independente do identificador atribuído pelo WiOS. Esta identificação é uma string terminada em 0, cujo comprimento e conteúdo podem ser escolhidos livremente pelo programador. Um identificador de tarefa ou driver pode ser pesquisado por seu nome. Um driver que não tenha entrada no GDA, pode ser procurado por nome, e o WiOS retorna o identificador ou um código. A alça pode então ser armazenada em uma variável e o novo driver poderá ser acessado como todos os outros. As janelas, ícones padrão e fontes também são acessados por essas alças

#### 7.4.4 – Interface da Janela

As janelas são quase totalmente controladas pelo WiOS. Assim os aplicativos não precisam se preocupar com as atividades do usuário fora da área de trabalho. Uma janela do WiOS é parecida com a ilustração abaixo.



Os símbolos que fazem parte da janela são chamados de ícones de janela. Existem 7 ícones e cada um pode ser definido separadamente para cada janela usando o ‘icon-flags’. Começando no canto superior esquerdo no sentido horário são os seguintes:

- Ícone Voltar
- Ícone Fechar
- Barra de título
- Alternar tamanho da janela
- Barra de rolagem vertical
- Redimensionar
- Barra de rolagem horizontal

#### 7.4.4.1 – Parâmetros

Para criar uma janela, é preciso configurar um heap de parâmetros com os sinalizadores de ícone e com os dados a seguir:

- Coordenada absoluta do canto superior esquerdo da tela;
- Tamanho da área de trabalho visível (a partir de agora chamada ‘work-area size’);
- Tamanho da área de trabalho total (tamanho virtual);
- Deslocamentos de rolagem;
- Tamanho mínimo da área de trabalho (ou seja, quão pequena a janela pode ser feita pelo usuário);
- Tamanho máximo da área de trabalho;
- Sinalizadores de área de janela;
- Sinalizadores de ícones;
- Sinalizadores de área de trabalho.

A área de trabalho é o local onde a tarefa exibe suas próprias informações. O tamanho da área de trabalho nunca é alterado pelo WiOS. Por exemplo, ao adicionar uma barra de título, esta não ocupará o espaço da sua área de trabalho, mas aumentará a altura total da janela. O WiOS não verifica se os ícones se encaixam na janela. Se a janela for muito pequena, alguns ícones sobrepõem-se a outros.

O WiOS faz uma verificação dos ícones da janela mas a tarefa tem o controle total sobre se deve ou não aceitar o que o usuário faz. O WiOS envia o heap de parâmetro para a tarefa antes de fazer qualquer ação na tela, cabendo à tarefa executar o comando para redesenhar a nova janela.

#### **7.4.4.2 – Gerenciamento de Janelas**

O WiOS contém apenas uma lista de parâmetros para cada janela na memória. Sua identificação é semelhante à das tarefas e drivers, mas eles têm apenas alças, sem nomes. As janelas e seu conteúdo não são armazenados como bitmap, mas apenas como gráficos de tela.

Como partes de uma janela podem ser ocultadas por outras janelas, não é fácil desenhar apenas as partes visíveis. Quando o WiOS envia uma lista de fragmentos na tela, você pode redesenhar (este recurso não foi testado) a janela completa nas coordenadas que o WiOS oferece (em uma página oculta), permitindo ao WiOS copiar partes desta janela para a tela.

Ressalte-se que o WiOS é otimizado para funcionalidade, não para velocidade nem para eficiência.

#### **7.4.4.3 – Tipos de Janelas**

##### **Janelas-pai (parent windows)**

São janelas convencionais. Se a tarefa abrir uma única janela, esta receberá o estado “pai”.

##### **Janelas-filho (pane windows)**

São janelas que estão ligadas a uma janela pai. Eles sempre são desenhadas diretamente sobre a janela pai. Se a janela pai for para trás,

todas as janelas-filho também irão. Elas serão fechadas se a janela pai for fechada.

#### **7.4.5 – Eventos (Comunicação entre WiOS e Tarefas)**

Imagine a seguinte situação: A tarefa faz um cálculo muito complexo que dura alguns minutos. Agora o usuário fecha uma janela que está na frente da janela da tarefa de cálculo. Embora a tarefa esteja ocupada, o sistema deve ser capaz de redesenhar sua janela sob demanda.

Assim, o WiOS dispõe de um manipulador de eventos que envia informações para a tarefa logo após o polling. Um número chamado de ‘Event’ é retornado para a tarefa. Ela tem que lidar com isso, então cada tarefa deve chamar seu próprio manipulador de eventos e agir sobre o evento antes de continuar com seu cálculo.

Para garantir uma atualização imediata da tela após uma operação de movimentação de janela pelo usuário, os eventos têm uma prioridade diferente. Isso significa que a tarefa que precisa redesenhar uma janela deve ser chamada antes de sua vez na sequência normal e de outras tarefas que estão apenas esperando para serem devolvidas sem a necessidade de redesenhar.

Isso requer uma diferenciação entre tarefas que devem ser chamadas para atualizar a tela e outras tarefas. No WiOS existem três prioridades de eventos. São eles (prioridade mais alta primeiro):

1 – Abrir;

Redesenhar;

Rolagem;

Perto;

Mensagem do usuário.

2 – Clique do mouse;

O ponteiro deixou a janela;

Ponteiro entra na janela;

O ponteiro está sobre a área de trabalho;

Fim de uma operação de arrasto.

3 – Evento nulo.

O evento com a prioridade mais baixa (evento nulo) só é enviado se não houver mais nada a fazer e sinaliza que a tarefa pode continuar seu trabalho. Tarefas que recebem um evento de alta prioridade devem pesquisar novamente imediatamente após este evento ter sido tratado, e esperar o evento nulo para continuar seu trabalho.

Nem toda tarefa precisa fazer cálculos complexos o tempo todo. Se uma tarefa apenas espera alguma ação do usuário, não é necessário ser chamada o tempo todo se nada acontecer. Dessa forma, há o recurso de mascarar eventos.

A tarefa pode mascarar eventos para não ser chamada o tempo todo. O caso mais simples é o ‘null-event’. Se uma tarefa espera por algo, ela pode dizer à rotina de pesquisa para ativá-la somente quando um clique do mouse é feito ou uma janela precisa ser redesenhada.

#### **7.4.6 – Entrada do usuário**

O sistema verifica as teclas e o mouse o tempo todo através de uma rotina conectada à interrupção. Portanto, as teclas pressionadas enquanto a interrupção estiver desabilitada não são colocadas no buffer, assim como o movimento e cliques do mouse, uma vez que o movimento deste é feito via interrupção.

Assim, o clique do mouse é lido durante uma varredura direta do mesmo. Para saber se o mouse foi clicado, o WiOS armazena um cronômetro no buffer quando um clique é feito. A tarefa deve usar este cronômetro para saber se o mouse foi liberado e pressionado novamente durante o polling. Se o mouse ainda estiver pressionado e o cronômetro no buffer é o mesmo de antes do polling, o mouse está em espera. Se o cronômetro for diferente, o mouse foi pressionado novamente.

Para o teclado, o WiOS usa um buffer com vários ponteiros para cada tarefa. Sempre que uma tarefa chama a função para esvaziar ou ler o buffer do teclado, apenas seu ponteiro é redefinido ou incrementado, então uma tarefa pode limpar seu buffer enquanto a próxima tarefa obterá as últimas 100 teclas pressionadas. O buffer tem 256 bytes e é atualizado em polling-time e sempre que a função ‘getkey’ do WiOS é chamada. Se uma tarefa começar a ler o buffer após cerca de 300 teclas pressionadas, se recebe apenas as últimas  $300 - 256 = 44$  teclas.

Se uma tarefa ler as teclas diretamente por meio da chamada ‘getkey’, somente essa tarefa obterá seus valores. Se uma tarefa espera por um pressionamento de tecla, esvazie o buffer e faça um loop com a rotina ‘getkey’.

#### **7.4.7 – Arrastar e Soltar**

A estrutura multitarefa do WiOS permite que as tarefas chamem rotinas de outras tarefas. Isso torna desnecessário que os programadores codifiquem rotinas comuns repetidas vezes. Um bom exemplo é o menu do disco. Existe apenas uma tarefa que faz todo o tratamento de arquivos como deletar, mover, copiar e até mesmo abrir arquivos. Este menu é uma tarefa básica padrão do WiOS.

Normalmente, todas as interações entre tarefas são feitas pelos eventos. Há um evento chamado ‘End-Of-Drag-Operation’. Se o usuário arrasta arquivos do gerenciador de arquivos para a janela de uma tarefa ou para seu ícone na barra de ferramentas, a tarefa receberá este evento com o ID de que há nomes de arquivos chegando e onde eles estão armazenados. Assim, podem ser escritas tarefas do WiOS sem o menu do disco.

#### **7.4.8 – Gerenciamento de Arquivos**

Um ponto fraco do WiOS que nenhuma verificação é feita ao serem abertos em várias instâncias de uma tarefa ou em tarefas diferentes. Assim, se forem feitas alterações em um arquivo este deve ser salvo pelo aplicativo. Entretanto, erros de disco são planejados para serem tratados pelo próprio WiOS.

#### **7.4.9 – Linguagem de Programação**

É fortemente recomendável que os aplicativos para o WiOS sejam escritos especificamente em ASCII-C 1.2, e não em BDS, Hi-Tech ou qualquer outro C.

C consiste em dois programas para compilar o código-fonte, um para montagem e outro para vincular um arquivo COM executável. O programa ‘CF.COM’ pega o arquivo C e verifica todos os erros de sintaxe. Sua saída é um arquivo com a extensão ‘TCO – o chamado arquivo

de código T. Esta etapa é muito importante porque a próxima etapa requer esse arquivo de código T.

O ‘Gerador de Código’ pega o arquivo de código T e gera código de montagem puro. O resultado é colocado em um arquivo MAC. É texto ASCII simples e este é o ponto em que você ainda pode otimizar manualmente ou usar o otimizador XelaSoft para otimizar o código compilado. Para execuções de teste, é suficiente deixar o código como está.

O gerador de código produz um arquivo montável pelo M80, que cria um arquivo REL (relocável), mas que não pode ser executado. Ele deve ser vinculado primeiro.

O último passo é o vinculador (linker) L80. Ele pega todos os arquivos REL que fornecidos na linha de comando e calcula todos os endereços. Nesta etapa, devem ser fornecidos todos os arquivos que contêm as funções e variáveis para possibilitar a geração de um arquivo COM executável.

C é estritamente limitado ao espaço de endereçamento de 16 bits, sem mapeamento de memória embutido. A última ação para impedir que os programas fiquem muito grandes é o processo de vinculação. Existe a versão L80-2, mas use a versão 1 para vincular programas C: seu programa pode ser maior do que na versão 2!

Todo arquivo COM é colocado a partir de 0100H e executado a partir desse mesmo endereço. Entretanto, para criar executáveis do WiOS, o vinculador deve ser informado de que a tarefa deve iniciar em 4000H. Isso pode ser feito com a opção ‘/P:4000’ como o primeiro argumento na linha de comando L80. Será obtido um arquivo com cerca de 16k de zeros em seu início. Este arquivo deve ser deixado COMO ESTÁ. Com o ‘Application-Creation-Toolkit’ vem o utilitário MAKDRV.COM, com o qual é possível fazer aplicações grandes reais. Ele elimina os 16K de zeros iniciais e preenche algumas informações importantes no cabeçalho do arquivo.

É essencial, entretanto, que aplicações grandes devem ser divididas em blocos de até 16K cada, o que pode dificultar seu desenvolvimento.

Quanto ao vinculador, todas as funções externas que você deseja chamar de uma tarefa devem estar vinculadas no executável. Não é possível vincular o WiOS completo à sua tarefa, pois O próprio WiOS é maior que 64KB. O que precisa ser vinculado é arquivo GDA que contém os endereços globais que permitem que as tarefas acessem funções e variáveis do WiOS. Este arquivo contém apenas etiquetas que apontam para uma tabela de endereços específica na página 3 (acima de C000H).

Sempre inclua o arquivo ‘GDA.H’ se você quiser ter acesso ao WiOS.

#### **7.4.10 – Chamando funções WiOS**

Em C, as funções são normalmente chamadas usando uma sintaxe como função(); onde o código gerado para esta função sempre chama o endereço ‘function’ diretamente.

Para chamar em um endereço para onde uma variável aponta, a sintaxe é um pouco diferente. Desde que haja endereçamento indireto, o rótulo não apontará para a função em si, mas para um endereço no GDA que contém o endereço da função. Este rótulo é nomeado com um sublinhado inicial: ‘\_function’.

Em C, o conteúdo de um ponteiro é acessado com um asterisco antes do nome do rótulo. Para chamar o endereço para onde um ponteiro aponta, o rótulo e o asterisco devem estar entre colchetes.

Para chamar o endereço para onde a etiqueta aponta, escreva:

```
(* _função) () ;
```

Poucas funções são chamadas diretamente via GDA. Muitas partes do WiOS estão na página 1, de 4000H a 7FFFh, onde está a tarefa de chamada. Neste caso, as tarefas devem chamar uma função que mapela no driver, chama, restaura o segmento e retorna à tarefa. Esta função é chamada ‘caldrv’ (call driver).

A entrada GDA de referência é denominada ‘\_caldrv’. As funções do WiOS devem sempre ser chamadas com:

```
(* _caldrv) (...) ;
```

onde ‘(...’ representa os argumentos a serem enviados para a função ‘caldrv’.

O WiOS precisa saber qual driver você deseja chamar. Normalmente, você pode pesquisar o driver por nome e receber o identificador. Mas existem alguns drivers padrão que são sempre carregados quando o WiOS é iniciado. Suas respectivas alças também podem ser encontradas no GDA.

Os drivers padrão são os seguintes:

- \_hextdrv      Driver externo (funções WiOS internas que não se encaixavam na parte interna).
- \_hfsdrv      Driver do sistema de arquivos (usado para todas as operações de arquivo).
- \_hgiodrv      Graphic I/O driver (carrega fontes e ícones).
- \_hgrpdrv      Driver gráfico (manipula a saída gráfica para a tela).
- \_hmemdrv      Driver de memória (mapeamento de memória).
- \_hstddrv      Driver padrão (funções padrão).
- \_htaskdrv      Driver de tarefas (faça qualquer coisa com tarefas).
- \_hwindrv      Driver de janela.

O primeiro argumento para a função ‘caldrv’ é o identificador do driver. Então é só escrever

```
(* _caldrv) (* _hwindrv) ;
```

para acessar o driver de janela.

Cada driver pode ter mais de uma função, então precisamos dizer à tarefa qual função que queremos chamar. Essas funções são numeradas de 1 a 32 767. Os drivers padrão (ou seja, todos os 8 drivers mencionados anteriormente) são numerados linearmente, começando com 1.

Como a função 4 no driver da janela faz algo completamente diferente da função 4 no driver gráfico, deve ser tomado cuidado para não acessar o driver errado.

Para facilitar as coisas, os números de função de todos os drivers podem ser acessados via texto. Cada driver tem seu próprio arquivo de cabeçalho onde todas as funções são definidas. Para obter a os dados

atuais de uma janela no endereço 5000H, o código-fonte se parece com isso:

```
(*_caldrv) (*_hwindrv, GET_WIN_STATE, 0x5000);
```

onde ‘hwindrv’ significa ‘handle of window driver’.

Para acessar as funções do driver por nomes, devem ser incluídos os arquivos de cabeçalho, por exemplo, ‘WINFNC.H’ se você precisar das funções do driver de janela.

Cada função de driver retorna um inteiro sem sinal de 16 bits. Mas nem todas as funções retornam um valor válido (porque não precisam devolver nada). Valores únicos devem ser retornados diretamente da função. Para mais de um valor, deve ser retornado um ponteiro para o endereço onde os valores de retorno são armazenados.

#### 7.4.10.1 – Chamadas Diretas

O WiOS possui chamadas que não podem ser acessadas pelos drivers. Elas devem ser chamados diretamente. As variáveis são declaradas em ‘GDA.H’. A lista mostra o uso dos comandos no código C.

```
(*_poll)((sem assinatura)máscara)
```

Chama a rotina poll com uma máscara de eventos (os eventos cujos bits estão definidos na máscara não são enviados para a tarefa). Ele retorna o endereço do bloco de eventos. Este bloco é definido em ‘DEFSTR.H’ e possui a estrutura descrita no capítulo “Estrutura do Bloco de Eventos”.

```
(*_caldrv)( (char)d_handle , (D_FNC)func , (uint)arg1 , (uint)arg2 , ... )
```

Chama o driver-caller, uma rotina que chama o driver com o identificador ‘d\_handle’, e envia o número da função como primeiro parâmetro e depois os argumentos.

```
(*_cal_seg)( (uint)addr , (T_SEG)seg , (char *)block )
```

Somente para tarefas. Muda para o segmento multipartes ‘seg’, chama o endereço que é apontado por ‘addr’ e envia ‘block’ como o endereço de ‘adrblk’. O ponteiro de pilha do segmento antigo é salvo e o do novo segmento é restaurado. O endereço da rotina para salvar a pilha, restaurar o segmento do chamador

e restaurar a pilha antiga é colocada na pilha da função chamada, então o retorno da função multipartes é feito com uma instrução ‘RET’ ou com fim de função normal em C. Esta função sempre retorna um valor de 16 bits (em HL) para a função chamada. Funções multipartes podem ser aninhadas desde que a pilha não ultrapasse os 16K da tarefa ou (o que é mais provável) o código da tarefa é sobreescrito. Nenhuma verificação é feita para isto.

(\* \_dcal\_seg)( (uint)address , (T\_SEG)segment , (char \*)block )

Somente para drivers. Muda para o segmento multipartes ‘seg’, chama o endereço que é apontado por ‘addr’ e envia ‘block’ como o endereço de ‘adrblk’. O ponteiro da pilha não é alterado, pois os drivers usam a pilha WiOS. O endereço da rotina para restaurar o segmento do chamador é colocado na pilha da função chamada, então o retorno de uma função multipartes é feito com uma instrução ‘RET’ ou com fim de função normal em C. Esta função sempre retorna um valor de 16 bits em HL para a função chamadora. As funções multipartes podem ser aninhadas desde que a pilha não ultrapasse o limite superior do GDA, caso contrário o sistema ficará instável.

#### **7.4.11 – Referência do Evento**

Os eventos a seguir são definidos como eventos padrão e os manipuladores de eventos devem ser capazes de lidar com eles. A lista a seguir fornece o nome, número e os argumentos necessários. É também a estrutura para a tarefa que recebe os dados. Ao contrário das chamadas de sub-rotina, esses eventos nunca retornam um valor, pois o programa que recebe os eventos não é chamado imediatamente. Um evento é apenas um bloco de dados que é colocado em uma pilha, informando à tarefa de destino, após o ‘polling’, que deveria fazer algo. Devido a este fato, você deve se certificar que os dados estejam corretos. Tenha em mente que normalmente não é o seu programa que recebe os dados que você envia, e as tarefas de outras pessoas podem não ter uma rotina de tratamento de erros tão abrangente como suas tarefas têm. Além disso, os eventos são enviados um a um após cada ‘polling’. Portanto, se houver 50 eventos na pilha, isso demandará um certo tempo para limpá-la.

Os nomes de eventos com um asterisco (\*) são enviados pelo WiOS e NÃO devem ser enviados por nenhuma tarefa. Eles devem ser tratados como ‘apenas receptor’.

#### **7.4.11.1 – Estrutura do Bloco de Eventos**

Cada evento contém um bloco de 16 bytes para armazenamento de dados:

Descrição	Nome	Tipo
-----------	------	------

Identificador da tarefa de destino.	task	T_TASK
Número do evento.	event	D_FNC
Para armazenamento de dados.	array[6]	uint
Identif. da tarefa de origem que enviou este evento.	sender	char

Esta estrutura é definida como ‘EBSTR’ em ‘DEFSTR.H’.

#### **7.4.12 – Especificação de troca de dados**

Esses eventos são chamados de “eventos de usuário”. Eles são enviados pelo evento E\_USERMSG e, embora os dados não tenham efeito no próprio WiOS, devem ser usadas as definições abaixo para transferência de dados padrão para outras tarefas para que todo programador que leia esta documentação possa escrever aplicativos que possam lidar com esse tipo de dados.

#### **7.4.12.1 – Execução da troca de dados**

Para enviar dados para outras tarefas, é necessário um procedimento especial. Primeiro, um ‘end-of-drag-event’ (‘E\_EDRAG’) com o tipo de dados básico deve ser enviado para a tarefa de destino. Embora seja possível que a tarefa de destino tenha mascarado esse evento, é altamente recomendável não fazer isso. Normalmente, a tarefa de destino recebe este evento e tem que responder. Assim que a tarefa de origem (remetente) recebe a mensagem ‘SEND’, ela pode enviar à tarefa de destino a mensagem ‘DATA’ com informações detalhadas sobre onde estão os dados e que tipo e tamanho esses dados possuem.

Se uma tarefa não receber nenhum dado, ela também deve responder ao ‘end-of-drag-event’ com a mensagem ‘NOSEND’ para que qualquer tarefa de origem que queira enviar dados saiba que esta tarefa não receberá dados.

Entre essas etapas, é necessário que essas tarefas sejam sondadas. E, como se já não fosse difícil o suficiente, cada tarefa também deve estar preparada para que o próximo evento após a sondagem não seja a resposta, mas também possa ser uma solicitação de abertura ou redese-nho.

O fluxo do programa fica assim:

Tarefa de origem	Tarefa de destino
• enviar evento ‘E_EDRAG’ com tipo de dados • polling	
	• responder com msg ‘SEND’ ou ‘NOSEND’ • polling
• verificar por eventos ‘padrão’ (se sim, ‘polling’ novamente) • verificar por msg ‘SEND’ ou ‘NOSEND’	
Essas etapas são necessárias apenas se a mensagem ‘SEND’ foi enviada	
• Se ‘SEND’, enviar msg ‘DATA’ com detalhes • polling	
	• verificar por eventos ‘padrão’ (se sim, ‘polling’ novamente) • gerenciar recepção de dados

A partir deste fluxograma, pode ser visto que esse processo não é muito difícil de lidar. Ao visualizar cada tarefa como separada, o remetente deve enviar o evento ‘E\_EDRAG’ e sondar até a mensagem ‘SEND’ ou ‘NOSEND’ chegar, então enviar os dados (se a tarefa de destino desejar) e sondar novamente.

Para a tarefa de destino, é ainda mais simples: avalia o evento ‘E\_EDRAG’ e envia a mensagem ‘SEND’ ou ‘NOSEND’, depois faz o pol-

ling. Se quiser receber os dados, pesquisa até que a mensagem ‘DATA’ chegue, senão está finalizado. Quando a tarefa não precisar mais dese tipo da dado, basta enviar o evento ‘NOSEND’.

A partir disso, é óbvio que responder ao evento ‘E\_EDRAG’ é es-sencial, ou então a tarefa de origem ‘trava’ em um loop infinito até que um evento ‘E\_SHUTDOWN’ chegue.

#### **7.4.12.2 – Definição do tipo de dados**

Este é um ponto fraco. A definição dos tipos de dados não é per-fecta nem os tipos de dados foram criados a partir de sua necessidade em aplicações reais – ainda não houve nenhuma aplicação para WiOS.

#### **7.4.12.3 – Dados enviados com o evento ‘E\_EDRAG’**

Os nomes dos tipos de dados entre colchetes são definidos em ‘DTDEF.H’ e devem ser enviados em array[0], e o especificador em array[1].

Tipo de dados: 0 (‘TEXT’)

Especificador: 0 (‘STRING’)

Descrição: Uma ou mais strings de texto sem função especial.

Tipo de dados: 0 (‘TEXT’)

Especificador: 1 (‘FILENAME’)

Descrição: Uma ou mais strings de texto com nomes de arquivos a serem manipulados/editados/abertos.

Tipo de dados: 1 (GRÁFICO)

Especificador: 0~255 bits por pixel.

Descrição: Uma imagem de bitmap retangular (a resolução é defi-nida nos próprios dados).

#### **7.4.12.4 – Dados enviados com o evento ‘E\_USERMSG’**

Sempre que enviar mensagens de usuário, array[0] mantém o código da mensagem. Ou seja, neste caso, código ‘DATA’, que é definido em ‘DTDEF.H’.

Dado: **STRING**

Argumentos: array[1] endereço do bloco de dados

array[2] segmento do bloco de dados

Bloco de dados

offset tam.	Faixa	Descrição
-------------	-------	-----------

+0	2	0~65 535 Número de strings.
----	---	-----------------------------

+2	2	1~3071 Número de segmentos usados para dados.
----	---	---

+4,6,...	2	0~3071 N° de segmentos com strings.
----------	---	-------------------------------------

Depois dos números do segmento:

2	0	Strings terminadas em nulo.
---	---	-----------------------------

1~255	Número de campos.
-------	-------------------

- se as strings forem terminadas em nulo, elas virão imediatamente após o ‘0’.
- se as strings estiverem em campos de comprimento fixo, elas vêm após o comprimento do campo – 2 bytes de cada campo. Após os dados de comprimento vem o campo formatado de dados da string.

Nota: Se as strings excederem 16K, cabe ao programa trocar os segmentos.

Dado: **FILENAME**

array[1] endereço do bloco de dados

array[2] segmento do bloco de dados

Bloco de dados

offset tam.	Faixa	Descrição
-------------	-------	-----------

+0	2	0~65 535 Número de nomes de arquivos
----	---	--------------------------------------

Depois do número de nomes de arquivos:

Strings terminadas em nulo
----------------------------

Nota A estrutura de dados do nome do arquivo é igual à das strings terminadas em nulo, com a diferença de que a tarefa de destino não precisa inserir dados em algum lugar, mas sim abrir os arquivos. Também cabe à tarefa de destino analisar as extensões de nome de arquivo ou os arquivos e verificar se eles são válidos para a tarefa.

Dado:	<b>GRÁFICO</b>
	array[1] Endereço do bloco de dados.
	array[2] Segmento do bloco de dados.
	Bloco de dados
	offset tam. Faixa Descrição
	+0 2 Não utilizado.
	+2 2 1~3071 Número de segmentos usados para a imagem.
	+4,6,... 2 0~3071 Número de segmentos com dados de imagem.
	Depois dos números do segmento:
	2 0~65 535 Largura da imagem.
	2 0~65 535 Altura da imagem.
	Dados de imagem mapeados de 15 bits (consulte VDP-spec).
Nota	Se uma imagem exceder 16k, cabe ao programa trocar os segmentos.

#### 7.4.13 – Referência GDA

Esta seção descreve a lista de variáveis GDA necessárias. A área GDA começa em C030H. Não há marcação de final, pois a GDA pode ser expandida no futuro. Variáveis reservadas não devem ser alteradas.

**IMPORTANTE:** Todas as variáveis GDA são ponteiros para o endereço da variável ou campo. As definições de tipo estão em ‘DEF.H’.

Tipo	Nome	Offset	tam.	Descrição
T_SEG	* _tot_seg	+0	2	Número total de segmentos.
T_SEG	* _free_seg	+2	2	Número atual de segmentos livres.
T_SEG	* _tmp_seg	+4	2	Número do segmento temporário.
<não usado>		+6	2	
<uso interno>		+8	2	
T_TASK	* _p1_task	+10	2	Identif. da tarefa atual na página 1.
T_SEG	* _p1_seg	+12	2	Segmento atual na página 1.
char	* _p1rseg	+14	2	Nº do segmento atual na página 1.
char	* _p1_slt	+16	2	Slot do segmento atual na página 1.
<não usado>		+18	2	
T_SEG	* _p2_seg	+20	2	Nº do segmento atual na página 2.

char	* _p2rseg	+22	2	número do segmento atual do mapeador na página 2
char	* _p2_slt	+24	2	Slot do segmento atual na página 2.
<uso interno>		+26	2	
uint	(*_caldrv)(.)	+28	2	Endereço de entrada de chamada da função WiOS.
T_TASK	* _hmemdrv	+30	2	Identificador do driver de memória.
T_TASK	* _hstddrv	+32	2	Identificador do driver padrão.
T_TASK	* _htaskdrv	+34	2	Identificador do driver de tarefa.
T_TASK	* _hfsdrv	+36	2	Identif. driver do sistema de arquivos.
T_TASK	* _hgiodrv	+38	2	Identificador do driver gráfico de E/S.
T_TASK	* _hgrpdrv	+40	2	Identificador do driver gráfico.
T_TASK	* _hwindrv	+42	2	Identificador do driver da janela.
T_TASK	* _hextdrv	+44	2	Identificador do driver externo.
uint	* _wiosver	+46	2	Versão atual da parte interna do WiOS.
VOID	(*_wiosend)()	+48	2	Chamada de desligamento do WiOS (somente na versão Alpha).
uint	(*_poll)()	+50	2	Endereço de entrada de “polling”.
<uso interno>		+52 até +106		
VOID	(*_dump)()	+108	2	Entrada para fazer um dump de memória no V9958 (argumento é endereço).
struct	* _eventblk	+110	2	Endereço do bloco de eventos EBSTR que contém os dados após o “polling”.
struct				Endereço do bloco de informações da janela.
WIBSTR	* _wibblk	+112	2	Endereço do bloco de dados da janela.
struct				Identificador da fonte atual
WINSTR	* _winblk	+114	2	Identif. da fonte padrão ('system')
uint	* _act_font	+116	2	
uint	* _std_font	+118	2	
<uso interno>		+120	2	
char	* _busyflag	+122	2	Estado do VDP (gravação direta)
uint	* _adrbblk	+124	2	Endereço de chave de segmento global (bloco de endereço independente).
char	* _mb	+126	2	Estado atual dos botões do mouse: bit 0 – estado do botão esquerdo (1=pressionado). bit 1 – estado do botão direito (1=pressionado).

char	* _cltype	+128	2	Tipo de último clique do mouse: 0 – clique único esquerdo (1 = clicado). 1 – duplo clique esquerdo (1=clicado). 2 – clique único com o botão direito (1=clicado). 3 – clique duplo com o botão direito (1=clicado).
uint	* _cltime	+130	2	Tempo de clique (matriz de 2): offset descrição +0 Hora do último clique com o botão esquerdo +2 Hora do último clique com o botão direito.
int	* _coord	+132	2	Coordenadas atuais do mouse (matriz de 2).
uint	(* _cal_seg)()	+134	2	Ponto de entrada para as tarefas chamarem funções multipartes
<uso interno>		+136	2	
uint	(* _dcal_seg)()	+138	2	Ponto de entrada para os drivers chamarem funções multipartes
<uso interno>		+140	2	
<uso interno>		+142	2	

Esta lista pode ser expandida no futuro.

#### 7.4.14 – Referência do Menu

Os menus e ícones do WiOS devem ser manipulados diretamente pelas tarefas. No entanto, não existe um sistema automático de detecção de cliques do mouse em áreas definidas, mas o WiOS oferece funções úteis para facilitar o uso dos menus.

Existem duas funções principais para isso: ‘DRW\_MENU’ para desenhar o menu na tela e ‘CHK\_MENU’ para verificar se as coordenadas específicas estão em um dos os pontos definidos ou não.

A função CHK\_MENU retorna o número do item correspondentes às coordenadas, que são contados a partir do início da lista de menus. Cada texto, botão e caixa é um item. Se o menu for composto por uma lista com 5 entradas e duas opções (nesta ordem), as últimas entra-

das retornarão os números de item 5 e 6 (a contagem inicia em 0). Um tipo especial é o barra de rolagem, que contém 5 itens na seguinte ordem:

seta para cima/esquerda  
página para cima/esquerda  
controle deslizante  
página para baixo/direita  
seta para baixo/direita

Para entender melhor como os itens são avaliados, crie um menu com diferentes tipos em uma lista e imprima os valores de retorno da função CHK\_MENU (com as coordenadas do mouse e do argumento) na tela.

#### **7.4.14.1 – Tipos de menu**

As funções do menu oferecem 5 tipos diferentes de menus:

##### **Opção**

Strings de texto com uma caixa de opção à esquerda. As opções são um bloco de um ou mais itens onde cada item pode ser habilitado ou desabilitado separadamente.

##### **Botão de rádio**

Strings de texto com um botão de rádio à esquerda. Os botões de opção são um bloco de um ou mais itens em que apenas um item pode ser ativado. Se outro item neste bloco for selecionado, todos os outros itens serão desabilitados

##### **Lista**

São strings de texto simples. Normalmente, as listas são usadas para menus suspensos ou pop-up.

##### **Barra de rolagem**

Exibição gráfica de uma determinada escala definida pelo usuário com a possibilidade de selecionar uma determinada área na escala dada.

## **Caixa**

Área retangular. As caixas não são visíveis na tela. Eles podem ser usados para verificar qualquer área retangular, por exemplo, para menus que não podem ser manipulados com os tipos de menu acima.

Os dados completos do menu de uma janela são normalmente colocados em um bloco de ‘big data’. Tal bloco usa a ideia básica de uma linguagem macro, mas em um nível muito baixo, ou seja, word-, byte- ou bitmap. A ideia por trás disso é ter uma estrutura onde cada item tenha seu próprio número. A verificação de coordenadas deve ser chamada apenas uma vez para obter o número do item desejado, não importa se você tem apenas um menu suspenso ou uma estrutura complexa com muitas opções, botões de rádio e barras de rolagem.

### **7.4.14.2 – Recursos do Menu**

Os itens de opção/botão de rádio podem ser impressos em ambas as direções, de cima para baixo e da esquerda para direita. Independente da direção, o verificador de coordenadas validará as coordenadas começando da esquerda – no ícone – até a extremidade direita do texto de cada item separado, ou seja, se você marcar uma área à direita do texto, você não receberá o número do item.

Ao escrever itens no destino da esquerda para a direita, o WiOS calculará a largura da string mais longa da opção / botão de rádio e usa isso como a distância padrão entre cada item. Embora a distância seja igual, os itens começam com o ícone e terminam com a direita final do texto de cada item.

Itens desativados serão impressos em cinza (em vez de preto). Sempre que um item é selecionado, cabe à tarefa definir (e para botões de opção, redefinir todos os outros) os bits habilitados para ícones.

Os itens de listas só podem ser impressos de cima para baixo. O verificador de coordenadas calcula a largura da string mais longa e, ao contrário da opção / botão de rádio, usará essa largura para cada item no bloco de lista.

Para itens desabilitados veja ‘opção / botão de rádio’.

Os itens podem ter um ícone de seta na borda direita para indicar a existência de um submenu, que deve ser aberto como uma nova janela pela tarefa.

As barras de rolagem podem ser ajustadas na direção horizontal e vertical. Barras de rolagem desabilitadas não terão nenhuma diferença visível no gráfico (a tarefa deve indicar que separadamente na tela).

A escala do tamanho virtual (semelhante às janelas), o tamanho da tela em pixels e o tamanho da área representada pelo controle deslizante (ver ‘work-area’) podem ser escolhidos livremente.

O manuseio do slider, a rolagem e a página para cima/para baixo devem ser processados pela tarefa (o step-size não é pré-definido). Se o usuário clicar no controle deslizante e arrastá-lo, cabe à tarefa rastrear o mouse e mostrar o controle deslizante na tela.

A caixa pode ser livremente definida com um par de coordenadas (superior esquerdo / inferior direito).

#### **7.4.14.3 – Convenções gerais sobre menus**

Se os itens se sobreponerem, o verificador retornará o número do item da primeira área válida das coordenadas. Se eles se sobreponerem, retornará, não conforme desenhado na tela, o número da área mais baixa (por exemplo, se duas caixas se cruzarem, o número da caixa desenhada primeiro será devolvido embora apenas a segunda caixa esteja visível, pois ela substituiu a primeira caixa).

Cada item pode ser desativado separadamente. Itens desativados ainda são exibidos na tela, mas não são verificados por ‘CHK\_MENU’. Os números dos itens que o seguem permanecem os mesmos, então, por exemplo, se o primeiro item de uma caixa de listagem for desabilitado (o que tem o número ‘0’), os que os seguem também são, mas o segundo item ainda tem o número ‘1’.

Estilos de borda podem ser escolhidos para cada bloco de item, exceto para as barras de rolagem. Apenas dois estilos de borda são válidos: sem borda e borda de caixa preenchida com a cor padrão de fundo (que não pode ser alterada). A distância da borda em pixels é dobrada na

direção x porque na resolução 512x212, um pixel é duas vezes mais alto que largo.

#### 7.4.14.4 – Estrutura do Bloco de Menu

O bloco de menu é um grande campo de dados onde um ou mais tipos de menu podem ser definidos.

Os seguintes símbolos são usados na definição da estrutura:

##### prefixo

b Byte (1 byte).

w Palavra (2 bytes).

t Texto (comprimento de byte variável).

# Definição de valores válidos para 'D' (ver sufixo).

##### sufixo

N Qualquer número.

D Número de uma lista predefinida.

S Cadeia de caracteres (string).

0 Deve ser este valor.

Estilos de borda válidos:

0 Sem borda desenhada.

1 Caixa pintada com cor de fundo.

##### Definição

bN Número de blocos de itens

tipo

bD

#0 → Opção

Uma vez para cada bloco de opções:

wN Font-handle.

wN Deslocamento x.

wN Deslocamento y.

w0 Largura do bloco de opções  
(preenchido pelo WiOS).

w0 Altura do bloco de opções  
(preenchido pelo WiOS).

bD Estilo de borda.

bN	Distância da borda em pixels.
bD	Direção.
#0	De cima para baixo.
#1	Da esquerda para a direita.

Para cada entrada:

bD	Sinalizadores de entrada (bitmap)		
	bit	(=0)	(=1)
0	Fim da lista	Entrada válida	
	1	Desabilitado	Habilitado
			(selecionável)
	2	Não selec.	Selecionada
	3-7	Não utilizados	(deve ser reiniciado)

continua em Entry Flags (até que o bit 0 seja resetado).

#1 → Rádio

Uma vez para cada bloco de rádio:

wN      Font-handle.

wN Deslocamento x.

wN Deslocamento y.  
w0 Largura do bloco de opções

w0 (preenchido pelo WiOS).  
Altura do bloco de opções

(preenchido pelo)

bD Estilo de borda.  
LN Distâncias de borda.

BN Distance  
LD  $\tilde{D}$

bD Dileção.  
#0 Da cima para baixo

#0 De cima para baixo.  
#1 Da esquerda para a direita.

Para cada entrada:

#### bD Sinalizadores de entrada (bitmap)

manzaderos de entrada (bitmap)

seleções válidas para entradas de estado	0	fim da lista	entrada válida
	1	desabilitado	habilitado (selecionável)
	2	não selecionada	selecionada
	3~7	Não utilizados (deve ser reiniciado)	

**tS** String terminada em nulo

Continua em Entry Flags (até que o bit 0 seja resetado).

#2 → Lista

Uma vez para cada bloco de lista

wN	font-handle
wN	x-offset
wN	deslocamento y
w0	largura do bloco de opções (preenchido por WiOS)
w0	altura do bloco de opções (preenchido por WiOS)
bD	estilo de borda
bN	distância da borda em pixels

Para cada entrada:

bD	Sinalizadores de entrada (bitmap)	
bit	(=0)	(=1)
0	Fim da lista	Entrada válida
1	Desabilitado	Habilitado (selecionável)
2	Sem sub-lista	Sub-lista(seta à direita)
3~7	Não utilizados	(deve ser reiniciado)
tS	String terminada em nulo	

Continua em Entry Flags (até que o bit 0 seja resetado).

#3 → Barra de rolagem

bD	Sinalizadores de entrada (bitmap)	
bit	(=0)	(=1)

Estado de entrada:	0	inválido	válido
	wN	Deslocamento x	
	wN	Deslocamento y	
	wN	Tamanho visível do campo em pixels	
	wN	Tamanho virtual do campo em pixels	
	wN	Tamanho real do campo em pixels	
	wN	Deslocamento de rolagem da barra de rolagem em pixels	
	bD	Direção	
	#0	De cima para baixo	
	#1	Da esquerda para a direita	

	#4 → Caixa		
	bD	Sinalizadores de entrada (bitmap)	
	bit	(=0)	(=1)
Estado de entrada:	0	inválido	válido
	wN	x-offset	
	wN	deslocamento y	
	wN	largura	
	wN	altura	
	bD	estilo de borda	
	bN	distância da borda em pixels	

## 7.4.15 – Dicas de programação

Não é apenas importante saber quais funções existem, mas também sua ordem e quando elas devem ser usadas. Você pode ver como situações especiais podem ser gerenciadas nesta seção.

### 7.4.15.1 – Janelas

Para colocar janelas na tela, há duas etapas principais necessárias. Um passo trata dos elementos ‘virtuais’ para uma janela, (cálculo e armazenamento da janela na memória).

O segundo passo é mudar a janela fisicamente, ou seja, desenhar e mover a janela na área de trabalho visível.

#### Abrindo Janelas

Em primeiro lugar, você só pode desenhar janelas na rotina de manipulação de eventos depois do ‘polling’. Não há diferença se o usuário, sua tarefa ou outra tarefa deseja redesenhar uma janela.

#### Usuário

Após o polling, a tarefa recebe um open-window-event sempre que uma janela é arrastada, redimensionada, etc pelo usuário. A tarefa deve chamar a função ‘OPEN\_WIN’ que adiciona o ‘redraw-event’. A tarefa deve então redesenhar a janela na tela.

## Sua tarefa

A criação de uma nova janela pode ser feita com a função ‘CREATE\_WIN’ do driver de janela. Isso adicionará sua janela à lista de janelas e preencherá algumas variáveis. Para desenhar a janela na tela, você deve providenciar para receber um evento de redesenho após o ‘polling’. Este evento é enviado pela função ‘OPEN\_WIN’. Para obter os dados completos da nova janela, chame ‘GET\_WIN\_STATE’. Esses dados podem ser enviados para a função ‘OPEN\_WIN’, que adiciona o ‘redraw-event’.

Mover uma janela existente é feito conforme descrito nas etapas anteriores sem a função ‘CREATE\_WIN’.

## Outras Tarefas

Forçar outras tarefas a mover suas janelas pode ser feito enviando para elas o comando ‘open-window-event’, que é a mesma forma que o WiOS informa as tarefas depois que o usuário moveu uma janela. A tarefa correspondente irá então chamar a função ‘OPEN\_WIN’ que envia o evento de redesenho para si mesmo e, após o polling, redesenha sua janela.

Com relação a esses fatos, uma tarefa usando a janela deve ter as seguintes rotinas:

- window-creation-routine para as janelas ‘padrão’ (chamada apenas no início);
- window-redraw-routine (lidar com o redesenho das janelas na tela);
- window-open-routine (verifica os parâmetros da janela e chama ‘OPEN\_WIN’);
- event-handling-routine (chama window-open- ou window-redraw-routine)

## Solicitações de rolagem

Sempre que um evento ‘E\_SCROLL’ é enviado para outra tarefa, deve-se ter o cuidado de enviar um evento ‘E\_WOPEN’ imediatamente depois. Nenhum ‘polling’ deve ser feito entre o envio dos dois eventos.

#### 7.4.15.2 – Entrada do usuário

Um ponto fraco no WiOS é que não há rotina embutida que permite ao usuário editar um ou mais caracteres. Isso precisa ser simulado pelo próprio programa.

Obtendo a próxima tecla pressionada:

- Chame o driver externo com a função ‘GETKEY’.

Aguarde um pressionamento de tecla atual:

- Chamar ‘RESKEY’;
- Loop até que ‘GETKEY’ retorne diferente de 0.

Somente códigos ASCII serão retornados. Teclas de controle não podem ser verificadas se não houver código ASCII separado para a tecla com e sem a tecla de controle (por exemplo, SHIFT+‘A’ retorna um código diferente de ‘A’, mas SHIFT+CURSOR retorna o mesmo código que apenas CURSOR).

#### 7.4.15.3 – Usando o Segmento Temporário

Sempre que houver necessidade de armazenar 16K ou menos temporariamente, por exemplo, para fins de cálculo ou para transferências de dados, pode ser usado o segmento temporário do sistema. Seu número está no GDA (\_tmp\_seg’).

Este segmento também é usado pelo sistema como uma página temporária (por exemplo, sempre que os dados do disco são carregados) e pode ser usado em outras tarefas. Portanto, ao usar esta página, certifique-se de não carregar do disco ou ‘poll’, pois os dados serão corrompidos.

O segmento temporário está sempre no slot principal.

#### 7.4.15.4 – Ponteiro do Mouse

O ponteiro do mouse é movido em interrupção, então a cada 1/60 de segundo, uma função lê o mouse coordena e move o ponteiro na tela. Os dados do ponteiro são escritos na VRAM e, portanto, é necessário configurar o endereço de gravação e enviar as coordenadas para a

VRAM. Assim, não pode haver outro programa que grava dados lineares na VRAM através de P#0 ou configura um comando VDP.

Isso é tratado com a variável ‘busyflag’, que pode ser encontrada no GDA como ‘\_busyflag’. Sempre que este sinalizador for colocado em um valor diferente de 0, a atualização do ponteiro na tela não será processada. Se for necessário executar uma gravação direta na VRAM via P#0 ou configurar um comando gráfico, ‘busyflag’ deve ser escrito com um valor diferente de 0, e, ao terminar, deve ser recolocado em ‘0’. Ao escrever muitos dados lineares, também pode ser configurada uma caixa gráfica e enviar os dados através do P#2 para evitar períodos longos onde o ponteiro não se move.

#### **7.4.15.5 – Enviando dados para outras partes do programa usando o bloco de endereço**

Às vezes, é necessário enviar a outros programas mais do que apenas um ou dois argumentos. Isso funciona de forma transparente com a função ‘caldrv’, mas há casos em que os dados devem ser armazenados em um local que não seja afetado pela comutação de segmento pelo programador. Um desses casos é a chamada de tarefas multipartes. Há um lugar onde uma tarefa pode armazenar dados: esta área é apontada pela variável GDA ‘\_adrblk’ e tem apenas 16 bytes que podem ser definidos livremente. Isso não é muito, mas é o suficiente para a maioria dos casos.

Outro uso para o bloco de endereço é eliminar um ponto fraco da linguagem C, que é o fato de que apenas um parâmetro pode ser retornado. Às vezes, é necessário retornar mais de um valor. Esses valores podem ser armazenados nesta área.

Não é um local para armazenar dados permanentemente, pois outras tarefas também podem usá-lo. Assim, de forma semelhante ao segmento temporário, armazene apenas dados muito imediatos neste endereço.

#### **7.4.15.6 – Pilhas**

Cada tarefa tem seu próprio ponteiro de pilha dentro do segmento real. Sempre que uma tarefa é carregada, esta pilha é definida para o topo da página de 16K (ou seja, para 8000H) Sempre que forem

chamadas funções WiOS, os dados são armazenados na pilha. Não há limitação ou verificação interna se a pilha sobrescreve áreas do código. Se você tiver problemas com aplicativos grandes (mais de 15K), tente mover algumas funções para outro lugar.

#### **7.4.15.7 – Encerramento do Programa**

Para aplicativos WiOS, é importante nunca encerrar uma tarefa retornando da função principal que é chamado após o carregamento. Além disso, você nunca pode usar a rotina ‘exit()’, CTRL+STOP ou qualquer outra rotina de aborto do DOS, uma vez que as páginas de memória são alteradas e o driver de interrupção do WiOS ainda está ativo. A tarefa deve ser fechada com a função ‘KILLTASK’ do driver de tarefa. Se não funcionar, deve se usada a tecla ‘ESC’ que é verificada durante o ‘polling’. Esta rotina desliga o WiOS, restaura os segmentos de memória originais, conecta o driver de interrupção e retorna ao DOS.

#### **7.4.16 – Compilação**

A criação de uma tarefa WiOS requer várias etapas, que diferem ligeiramente para tarefas simples e múltipartes.

Primeiro, é necessário um arquivo de lote de compilação (vamos chamá-lo de COMP.BAT) que deve conter os seguintes comandos:

[driver\path\]cf %1	analisa e criar arquivo .TCO
[driver\path\]fpc -u %1	verifica os parâmetros da função com funções neste arquivo
[driver\path\]cg %2 %1	gera arquivo .MAC a partir do arquivo .TCO
[driver\path\]m80 =%1/m	gera arquivo .REL a partir do arquivo .MAC
del %1.tco+%1.mac	Exclui arquivos desnecessários

A opção ‘%2’ em ‘cg’ normalmente deve ser ‘-r2000’ para ter espaço suficiente para funções. Isto pode ser definido permanentemente com esse valor para não haver necessidade de digitá-lo para cada compilação. Para arquivos C que requerem uma proporção especial de ‘pool:symbol:hash’ para o analisador e configuração de memória

(`-rnnnn`) para o código gerador, é aconselhável criar arquivos de lote separados para eles.

Para usuários avançados de C, é preferível ter dois arquivos de lote de compilação. Se o código gerador precisar de um valor diferente de '`-rnnnn`', não é necessário criar o arquivo `.TCO` novamente.

Aqui estão ilustrados dois arquivos de lote:

#### COMP.BAT

```
cf -r7:5:2 %1
fpc -u %1
comp2 %1 %2
```

#### COMP2.BAT

```
cg -r2000 %2 %1
m80 =%1/m
del %1.tco+%1.mac
```

O primeiro arquivo em lote é chamado com 'COMP filename' (sem extensão), cria apenas o arquivo `.TCO` e verifica os parâmetros da função. Se nenhum erro ocorrer, ele chama COMP2 com os argumentos da linha de comando.

No segundo arquivo de lote, o gerador e montador de código é chamado, e os arquivos não usados são deletados.

A vantagem de usar dois arquivos é poder dar continuidade ao processo de compilação na geração de código, uma vez que a análise não é necessária se o gerador de código tiver um erro de memória.

Além disso, você pode substituir a opção '`-r2000`' escrevendo um segundo argumento na linha de comando, como:

```
comp2 task1 -r1500
```

O gerador de código recebe a linha '`comp2 -r2000 -r1500 task1`' e usa as opções posteriores em vez de opções anteriores.

#### 7.4.16.1 – Aplicativos de parte única (código de até 16k)

Eles consistem (normalmente) em vários arquivos C. Para vincular os arquivos .REL, é necessário um comando do tipo:

```
c:\c\180 /p:3ff6,taskhead,task1a,task1b,task1c,task1d,  
...,gda,c:clibs,c:crun/s,c:cend,task/n/e
```

Isso cria um único arquivo .COM com cerca de 16K de zeros e o código começando em 3FF6H, e a aplicação real (sem o cabeçalho do arquivo) inicia em 4000H.

Então ‘MAKDRV’ deve ser chamado para cortar os 16K anteriores e preencher os valores corretos no cabeçalho como:

```
makdrv task.com task.tsk
```

Isso cria o arquivo executável do WiOS0 ‘TASK.TSK’. O tamanho do arquivo deve ser menor que 16K. Se o arquivo .COM tiver mais de 16K de código (ou seja, for maior que 32K), o MAKDRV não prosseguirá.

Depois que a tarefa é criada, o arquivo .COM não é mais necessário e pode ser excluído. É melhor fazer esse processo em lote para facilitar. Assim, vamos supor que o arquivo de tarefa deve ser chamado TASK.TSK, o código fonte principal TASK.C e o arquivo de cabeçalho padrão TASKHEAD.MAC. Em seguida, um arquivo chamado LT.BAT (que significa Link Task) deve ser escrito:

**LT.BAT**

```
c:\c\180 /p:3ff6,%1head,%1,%1,%3,%4,%5,%6,%7,%8,%9,  
                  gda,c:clib/s,c:crun/s,c:cend, %1/n/e  
makdrv %1.com %1.tsk  
del %1.com
```

Se TASK.C for o único arquivo necessário para a tarefa, basta escrever:

**LT TASK**

para vincular TASKHEAD e TASK, criar o arquivo .TSK e remover o arquivo .COM. Se houver outra parte chamada ‘TASKB.C’ que tem que ser vinculada, basta adicionar ‘TASKB’ no final, assim como até 9 outras

partes. Se houver mais peças, é aconselhável escrever um arquivo de lote extra para o processo de vinculação.

A criação da tarefa é então feita com dois comandos:

```
COMP TASK
LT TASK
```

Se nenhum erro ocorreu, basta iniciar o WiOS (W.COM) e executar o programa.

#### **7.4.16.2 – Aplicativos de várias partes (mais de 16k de código)**

A primeira parte (com a rotina init) é criada exatamente da mesma maneira que uma única parte aplicativo até que o arquivo .COM seja concluído. Cada parte (normalmente) também consiste em vários arquivos C.

Todas as partes de código a seguir são compiladas com as mesmas etapas mencionadas acima – completamente independentes das outras partes – mas diferem na forma como estão ligados. Elas não tem o arquivo de cabeçalho padrão para tarefas, pois é necessário apenas no início do arquivo .TSK.

A linha de link deles se parece com isso:

```
c:\c\180 p:3ffe,etskhead,task2a,task2b.task2c,task2d,
...,gda, c:clib/s, c:crun/s,c:cend, task2/n/e
```

Para um aplicativo de várias partes, você deve ter dois ou mais arquivos .COM, que nunca devem ser excluídos, embora usem muito espaço em disco, pois eles são sempre necessários para criar o arquivo .TSK. Isso é feito por:

```
makdrv task.com task2.com task3.com ... task.tsk
```

Podem ser escritas quantas partes forem necessárias nesta linha e o nome do arquivo .TSK no final, respeitando o tamanho limite da linha de comando.

Para este tipo de tarefa, também é mais fácil ter alguns arquivos em lote para compilação. Para esse propósito, vamos supor que o arquivo de tarefa deve ser chamado TASK.TSK, o código fonte principal

TASK.C e TASKB.C (para a parte 1) e TASK2.C e TASK2B.C (para a parte 2), o arquivo de cabeçalho padrão TASKHEAD.MAC e o arquivo de cabeçalho estendido ETSKHEAD.MAC. Em seguida, um arquivo chamado LT1.BAT para vincular a primeira parte deve ser escrito:

#### LT1.BAT

```
c:\c\180 p:3ff6,%1head,%1,%2,%3,%4.%5,%6,%7,%8,%9,
      gda,c:clib/s, c:crun/s,c:end, %1/n/e
```

Este arquivo de lote é o mesmo que LT.BAT sem a chamada de MAKDRV e sem excluir o .COM, que é necessário para a criação posterior da tarefa.

LT TASK TASKB

liga TASKHEAD, TASK e TASKB, supondo que você já compilou TASK e TASKB com o arquivo de lote COMP.

Para criar as outras partes, você precisa de outro arquivo de lote porque o arquivo de cabeçalho é diferente. Então crie um arquivo chamado LT2.BAT que é usado para vincular todas as outras partes:

#### LT2.BAT

```
c:\c\180 p:3ffe,e%1head,%2,%3,%4,%5,%6,%7,%8,%9,gda,
      c:clib/s,c:crun/s,c:cend, %2/n/e
```

Para vincular os arquivos de amostra TASK2, TASK2B e ETSKHEAD, digite:

LT2 TSK TASK2 TASK2B

Agora, se você tiver os arquivos .COM, pode ser útil escrever um novo arquivo em lote para criar a tarefa. Este arquivo, chamado MAKETASK.BAT pode ser como abaixo:

#### MAKETASK.BAT

```
makdrv task.com task2.com task.tsk
```

### 7.3.17 – Especificações da Tarefa

Tarefas multipartes são compilados e vinculados separadamente, e a ‘interface’ entre elas é uma tabela de endereços no início da página (por volta de 4000H) e as funções WiOS que oferecem a possibilidade de transportar dados e saltar e retornar de e para os segmentos multipartes.

A primeira parte ou uma aplicação multipartes não tem uma tabela de endereços padrão. Se outras partes quiserem chamar uma função na primeira parte, deve ser configurada uma tabela de endereços nesta parte e o respectivo endereço deve ser colocado na função ‘cal\_seg’. A tabela pode estar em qualquer lugar entre 4008H a 7FFFH e pode ser incluída no arquivo de cabeçalho padrão como é feito no cabeçalho estendido.

#### 7.4.17.1 – Cabeçalhos

O cabeçalho padrão de uma tarefa deve ter uma lista de informações sobre a tarefa. Toda tarefa deve ter este cabeçalho. Para aplicativos multipartes, o cabeçalho fica na primeira parte. Cabeçalhos estendidos são apenas para partes estendidas, ou seja, parte 2 e superior de uma tarefa multipartes.

Os cabeçalhos são arquivos ‘.MAC’ para garantir a ordem correta das posições de variáveis e ponteiros. Todos os nomes com um ‘@’ são declarados públicos e podem ser acessados como uma variável normal no código C. Rótulos que não são apontados em uma instrução ‘ds’ ou ‘dw’ em outra linha podem ser deixados fora.

Os arquivos de cabeçalho devem ser montados com:

M80 = nome do arquivo/m

para criar um arquivo .REL vinculável.

Você pode definir o número de segmentos necessários no arquivo de cabeçalho, por isso é garantido que, quando sua tarefa for executada, ela terá segmentos suficientes, ou então o WiOS não a carregará. O número de segmentos necessários devem ser pelo menos tão grandes quanto o número de partes do programa porque cada parte precisa de um segmento. Mas esse número também pode ser maior se uma tarefa

precisar de mais segmentos para armazenar dados permanentemente. Você pode adicionar o número de segmentos ao número escrito em ‘PPARTS’ e escrever o resultado na constante ‘NEEDSEGS’.

Lembre-se de que esse é apenas o número mínimo de segmentos necessários para executar a tarefa. Mais segmentos podem ser alocados temporariamente durante a execução se houver necessidade.

### **Arquivos de cabeçalho**

Os seguintes arquivos de cabeçalho são essenciais para escrever tarefas em C:

```
#include <stdio>
#include "def.h"
#include "gda.h"
```

E um ou mais dos seguintes são opcionais:

#include "eventfnc"	Eventos.
#include "defstr.h"	Estruturas.
#include "winfnc"	Consulte o capítulo sobre referência de funções (Drivers WiOS).
#include "taskfnc.h"	
#include "stdfnc.h"	
#include "memfnc.h"	
#include "grpfn.h"	
#include "giofnc.h"	
#include "fsfnc.h"	
#include "extfnc.h"	

### **Estrutura de Cabeçalho Padrão**

Normalmente, só é preciso alterar essas duas linhas (e o nome da tarefa).

NEEDSEGS	equ 1	Número mínimo de segmentos necessários para esta tarefa (incluindo segmentos de dados).
PPARTS	equ 1	Partes do programa (para tarefas de várias partes).

### Cabeçalho externo:

WiOS_ver:	dw 0	Versão do WiOS necessária para executar o driver.
segs:	dw NEEDSEGS	Segmentos necessários para carregar o driver.
p_parts:	dw PPARTS	Nº de partes do programa no arquivo.
t_main:	dw taskinit@	Endereço de início da tarefa.
lastadr:	dw @endx@-4000h	Informação de tamanho para o loader, preenchido por L80.

### Cabeçalho interno (4000H)

tsknam@:	dw taskname	Ponteiro para o nome da tarefa.
p_stable:	dw segtable@	Ponteiro para nºs de segmentos.
stack:	dw taskmain@	Ponteiro de pilha interno da tarefa (deve apontar para a função taskmain no arquivo).
usedsegs:	ds 2	Número total de segmentos usados (preenchidos pelo WiOS)

### Área das variáveis (4008H)

segtable:	ds NEEDSEGS*2	Espaço para numeração de segmentos durante alocação (preenchida pelo WiOS).
taskname:	dw 0	Versão da tarefa.
	db "Test-Task"	Nome da tarefa.
	db 0	Término do nome da tarefa.
	public tsknam@	
	public segtable@	
	extrn taskmain@	
	extrn taskinit@	
	extrn @endx@	
end		

### Estrutura de cabeçalho externo estendida

lastadr:	dw @endx@-4000	Informação de comprimento para loader, preenchido por L80.
----------	----------------	--

### Estrutura de cabeçalho interno estendida (4000H)

mp@seg@:	dw 0	Segmento da 1 <sup>a</sup> parte com todas as task-infos (preenchido pelo WiOS)
mp@seg@:	dw 0	Segmento desta parte (preenchido pelo WiOS)
stack:	dw 0	Ponteiro de Pilha Interno da tarefa (preenchido por WiOS)

### Área das variáveis para cabeçalho estendido (4006H)

```

func_1:    dw test1@  Endereço da rotina 'test1' (4006H)
func_2:    dw test2@  Endereço da rotina 'test2' (4008H)
func_3:    dw test3@  Endereço da rotina 'test3' (400AH)

.
.
.

extrn test@

public mp@seg@,my@seg@
extrn @endx@

end

```

#### 7.4.17.2 – Diretrizes de pré-processador

Para obter o melhor desempenho com o compilador C, as seguintes linhas devem estar no topo de cada arquivo C:

#pragma optimize time	Parece não ter efeito no código.
#pragma regalo	Se possível, habilita alocação de registro.
#pragma nonrec	Não armazena todas as variáveis para cada função.

Para uma função recursiva, você pode escrever a mesma linha e substituir ‘nonrec’ por ‘recursive’ imediatamente antes da função e adicionar a linha com ‘nonrec’ novamente depois dela.

Embora possa ser ótimo teoricamente, muitas vezes é melhor desabilitar a alocação de registradores para que o gerador de código não tente otimizar o código. O programa roda um pouco mais lento e usa

alguns bytes a mais, mas o processo de compilação será até duas vezes mais rápido sem otimização.

Estas linhas são mais prováveis para o programador real:

```
#pragma optimize time
#pragma noregalo
#pragma nonrec
```

#### **7.4.17.3 – Variáveis Externas**

Existem variáveis definidas pelo programador (nome da tarefa) e WiOS (números do segmento) no arquivo de cabeçalho de cada tarefa. Para ter acesso a elas, eles devem ser declarados no arquivo C:

```
extern char *tsknam;
extern uint segtable(0);
```

O número entre colchetes pode ser definido livremente, já que o C não faz qualquer verificação. Significa apenas que a variável ‘segtable’ é uma variável indexada de 16 bits.

Com a variável ‘segtable’ você sabe onde está sua tarefa, todas as outras partes e os dados dos segmentos. Para obter o número do primeiro segmento sem código, basta usar o número de partes de sua tarefa como o índice. Por exemplo, se você tiver uma tarefa multipartes com 2 partes, o índice 0 e 1 é usado para os segmentos de código e o índice 2 e superior contém os números dos segmentos de dados.

Ao escrever o nome da tarefa, lembre-se de adicionar 2 ao ponteiro, pois os dois primeiros bytes do nome da tarefa contém o número da versão da tarefa.

#### **7.4.17.4 – Inicialização da Tarefa**

Sempre que uma tarefa é carregada, o WiOS chama o endereço apontado por ‘stack’ na primeira parte da tarefa (consulte a seção sobre a estrutura de cabeçalho padrão). Isso pode ser usado para tarefas inicialização, por exemplo, para verificar se outra instância desta tarefa já está em execução ou para verificar a presença de drivers necessários e, se necessário, carregá-los.

É importante ressaltar que essa rotina não é o início real da tarefa. Ela não pode chamar a rotina de “polling” nem algum evento deve ser verificado ou as janelas devem ser abertas. Normalmente a sequência de inicialização é apenas para verificar se a tarefa encontra tudo o que é necessário para ser executado.

A tarefa começa “oficialmente” no endereço apontado por ‘t\_main’. O identificador da tarefa é passado como um argumento do tipo ‘char’.

#### **7.4.17.5 – Rotina Principal**

Esta é a função onde a tarefa começa. A função nunca deve retornar e, para parar a tarefa, deve ser usada a função ‘KILLTASK’ do driver de tarefa. Este endereço é apontado por ‘t\_main’ no arquivo de cabeçalho padrão.

```
VOID taskmain()
{
    printf("task has started!!!\n";
    ...create_and_open_windows...
    ...poll...
    ...do_something...
    ...format_harddisk...
    ...print_virus_message...
}
```

Esta é a função que deve definir qualquer coisa que possa ser vista na tela, manipular eventos do usuário, fazer qualquer coisa, como a função ‘main()’ do C ‘normal’.

#### **7.4.17.6 – Criação de Janelas**

Antes de qualquer coisa, é necessário configurar os dados de janela adequados para que o WiOS possa fazer todo o trabalho com o usuário.

Para isso, e também para algumas funções posteriores na seção de redesenho e abertura, a variável ‘windat’ deve apontar para o próprio bloco de janela da tarefa que é usado para criar e alterar as janelas, então em algum lugar no arquivo C, deve haver uma linha como:

```
struct WINSTR windat;
```

Segue um exemplo de uma configuração adequada de uma janela para o WiOS:

```
VOID creatwin()
{
int whandle,i;
windat.x=50;
windat.y=60;
windat.nx=100;           /* work-area size */
windat.ny=75;
windat.vx=100*2;         /* virtual size */
windat.vy=75*2;
windat.scrx=100/2;        /* scroll offset (middle) */
windat.scry=75/2;
windat.minx=0;            /* no minimum limit */
windat.miny=0;
windat.maxx=0;            /*max size is virtual or scr*/
windat.maxy=0;            /* size */
windat.behind=-1;          /* open window on top */
windat.winflag=0;          /* see referring section */
windat.iconflag=127 /* allow all icons */
windat.workflag=32+8+4+2; /* allow all clicks */
windat.parent=-1;          /* window has no parent */
windat.statflag=0;          /* . . . */

for (i=0; i<25; i++) windat.dummy[i]=0
                           /* reset dummies */
whandle=windat.handle=(_caldrv)
                           (*_hwindrv,CREATE_WIN,&windat);
printf("created window has handle %d\n",
                           windat.handle;
(*_caldrv) (*_hwindrv,GET_WIN_STATE,&windat);
                           /* get win data */
(*_caldrv) (*_hwindrv,OPEN_WIN,&windat;
                           /* send data for opening the window */
}
```

Não deve haver problemas para entender o código. Uma coisa importante é que ‘whandle’ deve ser armazenado em uma variável global porque é o único ponto onde a tarefa pode pegar o identificador da janela criada.

#### 7.4.17.7 – “Polling” e Tratamento de Eventos

Aqui está um exemplo do que uma rotina de manipulação de eventos deve ser capaz de lidar. É usado o tipo mais simples de loop principal:

```
VOID taskmain()
{
    struct EBSTR *block;
    .
    .
    .
    while (1)
    {
        /* poll and return sdress of block */
        /* allow all events */
        block=(struct EBSTR *) (*_poll) (0);
        /* process the event */
        procevent(block);
    }
}
```

ou as duas últimas linhas de código (para ‘cracks’ C), então ‘block’ não é necessário:

```
procevent(struct EBSTR *) (*_poll) (0);
```

Para a descrição do bloco de eventos, consulte a seção sobre referência de eventos. A função ‘procevent’ deve então manipular todos os eventos necessários. Veja o exemplo abaixo.

```
VOID procevent(block)
struct EBSTR *block;
{
char c; printf("processing event %d\n",block->event);
switch(block->event)
{
    case E NULL:
        printf("receiving the NULL-event...");
        break;

    case E REDRAW:
        printf("REDRAW event for window %d
               received\n", block->array[0]);
}
```

```
    redraw(block);
    break;

case E_SCROLL:
    printf("SCROLL event for window %d
           received\n", block->array[0]);
    wscroll(block);
    break;

case E_WOPEN:
    printf("OPEN event for window %d
           received\n", block->array[0]);
    wopen(block);
    break;

case E_WCLOSE:
    printf("CLOSE event for window %d
           received\n", block->array[0]);
    (* caldrv)
    (* hwindrv,CLOSE WIN,block->array[0]);
break;

case E_PNTOUT:
    printf("pointer has left window %d\n",
           block->array[0]);
break;

case E_PNTIN:
    printf("pointer has entered window %d\n",
           block->array[0]);
break;

case E_MCLICK:
    printf("CLICK whandle=%d coord=%d,%d
           click-type:%d time=%d\n",
           block->array[0],block->array[1],
           block->array[2],
           block->array[3],block->array[4]); break;

case E_WKAREA:
    printf
        ("pointer is over work-area of window %d\n",
```

```

        block->array[0]);
    break;

    case E_EDRAG:
        nosend();
        break;

    default:
        printf ("Event Handler was called with
                an unknown event\n");
        break;
    }
}
}

```

O exemplo requer a existência das funções ‘redraw’ , ‘wscroll’ e ‘wopen’. A função ‘redraw’ redesenha a janela fisicamente na tela. O exemplo a seguir simplesmente desenharia uma caixa branca na área de trabalho, assumindo que temos uma rotina de desenho de caixa com os argumentos (x,y,nx,ny,color).

```

VOID redraw(block)
struct EBSTR *block;
{
struct WIBSTR *wib;
    wib=(struct WIBSTR *) (*_caldrv) (*_hwindrv,GETWIB,
                                         block->array[0]);
    _box( wib->sx + wib->offx , wib->sy + wib->offy,
          wib->windat->nx , wib->windat->ny , 32767);
    if (wib->elements) (*_caldrv)
                                (*_hwindrv,COPYWIN,wib);
}

```

Depois que GETWIB foi chamado, a moldura da janela está desenhada nas coordenadas retornadas com o membros ‘sx’ e ‘sy’ e os dados da janela são copiados para a área global (ou seja, fora da página 1 e 2) apontado pelo parâmetro ‘windat’. Se a janela estiver visível ela é copiado da área oculta para a tela.

A rotina ‘wscroll’ simplesmente tem que determinar se a rolagem realizada pelo usuário (ou solicitado por outra tarefa) é válido e confirme atualizando o deslocamento de rolagem da janela, faz ‘poll’ novamente para receber o evento ‘E\_WOPEN’ e retornar à tarefa normal

para ‘polling’ (para que ‘E\_REDRAW’ possa ser enviado). Como essa rotina também tem que lidar com o evento ‘open’, pode ser usada uma função comum para o scroll e para o evento ‘open’. Esta função é chamada ‘do\_wopen’:

```
VOID do_open(block)
struct EBSTR *block
{
    windat.behind=block->array[1]; /*setup window*/
    windat.x=block->array[2];     /*according*/
    windat.y=block->array[3];     /*to the block*/
    windat.nx=block->array[4];
    windat.ny=block->array[4];
                    /* send data for opening the window
                     and creation of REDRAW-events */
    (*_caldrv) (*_hwndrv,OPEN_WIN,&windat);
}
```

A função ‘OPEN\_WIN’ altera a janela na lista de memória e envia o evento de redesenho para a tarefa, reorganiza as janelas filho e todos as outras janelas que podem estar visíveis, envia solicitações de janelas abertas, etc.

Agora para a função:

```
VOID wscroll(block)
struct EBSTR *block;
{
    windat.handle=block->array[0];
                    /* get window data */
    (*_caldrv) (*_hwndrv,GET_WIN_STATE,&windat);
    windat.scrx=block->array[1];
    windat.scry=block->array[2];
                    /* do one more possing to receive
                     the window-open request and send
                     the data to do_wopen           */
    do wopen((struct EBSTR *)(*_pol1)(0));
}
```

Embora pareça um pouco precário porque não há verificação de eventos feita após a rotina de pesquisa, é correto. Cada solicitação de rolagem de janela do WiOS é seguida por um evento ‘E\_WOPEN’. Se uma tarefa envia uma solicitação de rolagem para outra janela, também deve enviar uma solicitação de janela aberta imediatamente depois.

A manipulação de eventos está quase completa agora:

```
VOID wopen(block)
struct EBSTR *block;
{
    windat.handle=block->array[0];
    (*_caldrv) (*_hwindrv, GET_WIN_STATE, &windat);
                                /* get win data */
    do_wopen(block);
}
```

Esta função obtém uma cópia dos dados da janela no próprio bloco de janela da tarefa para poder alterar o tamanho sem precisar configurar os dados de toda a janela.

#### **7.4.18 – Segurança**

Uma vez que uma tarefa foi criada com ‘MAKDRV’, ela não deve ser alterada por meio de um editor de disco. Para proteger tarefas contra alterações, elas contêm uma checksum que é verificada por WiOS antes de executar a tarefa. Se a soma de verificação estiver incorreta, a tarefa não será iniciada. Ainda não há janela de informações para isso. A checksum não é uma proteção contra crackers hard-core, nem é garantia de que a tarefa seja inalterada, mas oferece alguma segurança.

#### **7.4.19 – Iniciando tarefas com a versão Alfa**

Sempre que o WiOS é carregado, duas tarefas são carregadas e iniciadas. Eles devem ser nomeados

TASK.TSK e  
TSK2.TSK

Se esses arquivos não estiverem presentes, o WiOS não iniciará corretamente. Para Alpha-Testing, um arquivo contém o código normal, e outro o código para enviar dados ou arquivos, assim como o gerenciador de arquivos do WiOS fará no futuro. Se nenhum dado precisar ser

enviado, pode ser criada uma tarefa fictícia que não faz nada ou inicia a mesma tarefa duas vezes apenas copiando TASK.TSK para TSK2.TSK.

#### 7.4.20 – Descrição dos termos usados

**Adress block:** Área de 16 bytes que permanece intocada pela comutação de segmentos. Pode ser usada para armazenar dados imediatos que devem ser enviados para outras partes do programa. A variável GDA ‘\_adrblk’ aponta para seu endereço.

**Aplicativo:** Tarefa.

**GDA (Global Data Area):** área onde ficam as variáveis globais e endereços de função que devem poder ser acessados por outros programas.

**Tarefa:** Aplicativo.

**WIB (Window Information Block):** Estrutura de dados definida como WIBSTR.

**Window Block:** Estrutura completa de uma janela que é definida como WINSTR. Este bloco deve ser configurado para usar as funções da janela.

### 7.5 – ACESSO DIRETO AO CONTROLADOR DE DISCO

Muito embora não seja recomendado, é possível o acesso direto ao disco, sobrepujando ao sistema operacional e ao BDOS/Kernel. Nesse caso, é necessário saber qual tipo de interface está instalada, pois diferentes interfaces usam diferentes meios para o acesso. Nessa seção está descrito somente o acesso ao FDC (floppy disk control), de forma bastante resumida.

O FDC é acessado escrevendo e lendo dados em seus registradores internos. Esses registradores são os seguintes:

- 1 – Registrador de estado.
- 2 – Registrador de comando.
- 3 – Registrador de trilha.
- 4 – Registrador de setor.
- 5 – Registrador de dados.
- 6 – Registrador de drive, lado do disquete e motor do drive.

## 7 – Registrador de IRQ ocupado e requisição de dados.

Alguns FDC's têm diferenças entre esses registradores. Isso será descrito detalhadamente mais adiante.

### 7.5.1 – Comandos do FDC

Existem 4 categorias de comandos que podem ser executados pelos controladores de disco, conforme descrito abaixo:

Tipo	Comando	b7	b6	b5	b4	b3	b2	b1	b0
I	Restore	0	0	0	0	h	V	r1	r0
I	Seek	0	0	0	1	h	V	r1	r0
I	Step	0	0	1	T	h	V	r1	r0
I	Step-In	0	1	0	T	h	V	r1	r0
I	Step-Out	0	1	1	T	h	V	r1	r0
II	Read Sector	1	0	0	m	S	E	C	0
II	Write Sector	1	0	1	m	S	E	C	a0
III	Read Adress	1	1	0	0	0	E	0	0
III	Read Track	1	1	1	0	0	E	0	0
III	Write Track	1	1	1	1	0	E	0	0
IV	Force Interrupt	1	1	0	1	i3	i2	i1	i0

As flags mostradas na tabela estão descritas abaixo.

- r1,r0 Taxa de passo do motor (0=6ms, 1=12ms, 2=20ms, 3=30ms)
- V Flag de verificação do número da trilha (0=não, 1=verifica dest)
- h Flag de cabeçote (1=posiciona cabeçote na trilha 0)
- T Flag de atualização de trilha (1=atualiza registrador de trilha)
- a0 Marca de endereço de dados (0=FB, 1=F8 [DAM deletada])
- C Flag de comparação de lado (1=habilita comparação de lado)
- E Atraso de 15ms (1=ativa atraso de 15ms)
- S Flag de comparação de lado (0=compara para lado 0, 1=compara para lado 1)
- m Flag de múltiplos registros (0=registro simples, 1=múltiplos registros)

i3-i0 Termina sem interrupção (INTRQ)

i3 = 1 → interrupção imediata, requer reinicialização

i2 = 1 → pulso de índice

i1 = 1 → transição pronto para não pronto

i0 = 1 → transição não pronto para pronto

### 7.5.1.1 – Comandos tipo I

Os comandos tipo I são usados para mover o cabeçote do drive. O passo do motor normalmente é setado para 6 ms ( $r1 = r0 = 0$ ) para drives de disquetes de  $3\frac{1}{2}$ ". Uma verificação opcional da posição do cabeçote pode ser feita setando o bit 2 ( $V=1$ ) da palavra de comando.

Quando  $V=1$ , ao completar a busca, o número da trilha do campo ID do primeiro setor encontrado é lido e comparado com o conteúdo do registrador de trilha. Se os dois forem iguais e o CRC do campo ID estiver correto, será gerada uma INTRQ sem erros. Caso contrário, o bit Seek Error do registrador de estado será setado.

Quando  $V=0$ , ao completar uma busca, o número da trilha não será verificado. Esse modo deve ser habilitado para disquetes não formattados. O comando termina quando o último pulso for enviado ao motor de passo. É necessária uma pausa antes de ler ou gravar para que o cabeçote estabilize sobre a trilha.

Quando a busca for completada, uma requisição de interrupção é gerada e o bit “busy” do registrador de estado é zerado 0. Quando a CPU ler o registrador de estado, o sinal de interrupção é zerado.

#### **Comando ‘Restore’ (busca trilha 0)**

Esse comando posiciona o cabeçote do drive na trilha 0. O registrador de trilha será levado a 0 e uma interrupção será gerada quando a trilha 0 for atingida.

#### **Comando ‘Seek’**

Esse comando posiciona o cabeçote na trilha indicada pelo registrador de dados. O FDC atualizará o registrador de trilha e enviará pulsos ao motor de passo até que o cabeçote atinja a posição desejada. Uma interrupção será gerada ao final do comando.

## **Comandos ‘Step-In’, ‘Step-Out’ e ‘Step’**

Esses comandos enviam um pulso ao motor de passo. O comando Step-in movimenta o cabeçote em direção à trilha 0, Step-out em direção à última trilha e Step movimenta para a mesma direção do comando anterior. O registrador de trilha será atualizado somente se o bit “T” estiver setado na palavra de comando. Uma interrupção será gerada ao final do comando.

### **7.5.1.2 – Comandos tipo II**

Os comandos tipo II são usados para ler e escrever setores no disco. Antes de executar um comando tipo II, o registrador de setor deve ser carregado com o setor desejado. Ao receber um comando tipo II, o bit “busy” do registrador de estado é setado. Se o campo ID do setor com a trilha e setor corretos não for encontrado, a flag “setor não encontrado” do registrador de estado será setada e uma interrupção será gerada.

A flag ‘m’ indica múltiplos setores. Se for 0, será acessado um único setor; se for 1, múltiplos setores são acessados. Nesse caso, o registrador de setor vai sendo atualizado e uma verificação de endereço pode ocorrer a cada setor lido. O FDC vai acessando os setores em ordem crescente até que o registrador de setor exceda o número de setores da trilha ou até que uma interrupção forçada seja solicitada (comando Force Interrupt).

A flag C é usada para habilitar a comparação de lado do disco. Se for 0, não haverá comparação. Se for 1, o bit LSB do campo ID do disco é lido e comparado com o conteúdo da flag S.

## **Comando ‘Read Sector’**

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de estado é setado, e quando o campo ID é encontrado e contiver a trilha, setor, lado e CRC corretos, o campo de dados é disponibilizado à CPU. Uma DRQ é gerada sempre que o registrador de dados contiver um dado válido. Nesse caso, a CPU deve ler o dado imediatamente. O bit “lost data” do registrador de estado será setado se a CPU não leu o dado em tempo, mas a leitura continuará até o fim do setor ser atingido. Ao final da operação de leitura, o tipo “data address mark” en-

contrado no campo de dados será gravado no registrador de estado (bit 5).

### **Comando ‘Write Sector’**

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de estado é setado, e quando o campo ID é encontrado e contiver a trilha, setor, lado e CRC corretos, uma DRQ é gerada. O FDC conta 22 bytes (em dupla densidade) do CRC e a saída “write gate” é ativada se a DRQ for respondida. Se a DRQ não for respondida, o comando é encerrado e o bit “lost data” do registrador de estado é setado. Se a DRQ for respondida, 12 bytes 00H (em dupla densidade) serão escritos no disco. Então a DAM (data adress mark) é determinada pelo campo ‘a0’ do comando. Após isso, o FDC escreverá o campo de dados e gerará DRQ’s para a CPU. Se a DRQ não for respondida em tempo para escrita contínua, o bit “lost data” do registrador de estado será setado e um byte 00H será escrito no disco. O comando continuará até que o último byte do setor seja atingido. Após o último byte de dados ser escrito, um CRC de dois bytes é computado internamente e escrito no disco, seguido por um byte FFH.

#### **7.5.1.3 – Comandos tipo III**

Os comandos tipo III são usados para acessar os cabeçalhos (headers) das trilhas e setores do disco.

### **Comando ‘Read Adress’**

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de estado é setado. O próximo campo ID encontrado é lido e os seis bytes de dados do campo ID são montados e transferidos para o registrador de dados. Uma DRQ é gerada para cada byte lido. Os seis bytes ID são:

1 – Endereço de trilha	4 – Tamanho do setor
2 – Número do lado	5 – CRC1
3 – Endereço de setor	6 – CRC2

Embora os bytes CRC sejam transferidos para a CPU, o FDC checa a validade dos mesmos e o bit “CRC error” do registrador de estado será setado se houver erro de CRC. O endereço de trilha do campo ID é escrito no registrador de setor para possibilitar uma comparação pelo

usuário. Ao final do comando, uma interrupção é gerada e o bit “busy” do registrador de estado é zerado.

### **Comando ‘Read Track’**

Ao receber esse comando, o cabeçote é posicionado e o bit “busy” do registrador de estado é setado. A leitura é iniciada imediatamente ao primeiro pulso de indexação encontrado e continua até o pulso de indexação seguinte. Todos os gaps, headers e bytes de dados são montados e transferidos para o registrador de dados. Uma DRQ é gerada para cada byte transferido. A acumulação de bytes é sincronizada para cada marca de endereço encontrada. Uma interrupção é gerada quando o comando for completado. O ID da marca de endereço, campo ID, ID dos bytes CRC, DAM, dados e bytes de dados do CRC para cada setor devem estar corretos. Os bytes gap podem ser lidos incorretamente durante a pausa na escrita por causa da sincronização.

### **Comando ‘Write Track’ (formatação de trilha)**

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de estado é setado. A escrita é iniciada imediatamente ao primeiro pulso de indexação encontrado e continua até o próximo pulso de indexação, quando então a interrupção é ativada. A requisição de dados é ativada imediatamente ao receber o comando, mas a escrita não será iniciada antes do primeiro byte de dados ser escrito no registrador de dados. Se este não for carregado na temporização do pulso de indexação, a operação é terminada com dispositivo não ocupado, o bit “lost data” do registrador de estado é setado e a interrupção é ativada. Se um byte não estiver presente no registrador de dados quando necessário, será assumido um byte 00H. Essa sequência é repetida de uma marca de indexação a outra.

Normalmente, qualquer padrão de dados que for carregado no registrador de dados é escrito no disco com um ciclo padrão normal. Entretanto, se o FDC detectar um padrão de dados de F5H até FEH no registrador de dados, será interpretado como marca de endereço, sem geração de ciclos ou CRC. O gerador CRC é inicializado quando um byte F5H está para ser transferido (em MFM). Um byte F7H gerará dois bytes CRC. Como consequência, os bytes F5H a FEH não podem fazer parte

dos gaps, campos de dados ou campos ID. Na formatação das trilhas, os setores podem conter 128, 256, 512 ou 1024 bytes.

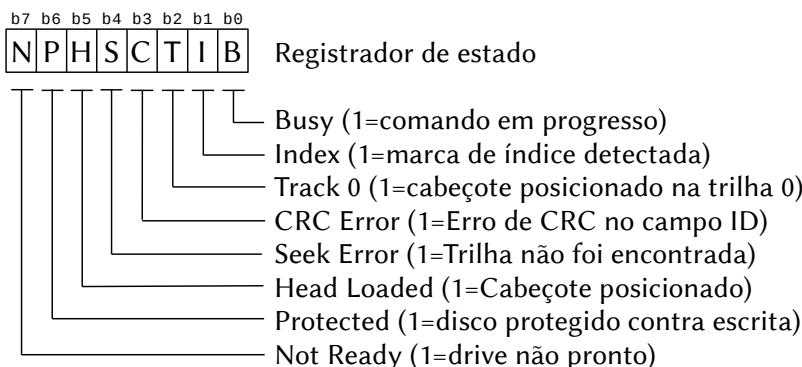
#### 7.5.1.4 – Comando tipo IV

O comando tipo IV (force interrupt) é geralmente usado para encerrar o acesso a múltiplos setores. Esse comando pode ser carregado no registrador de comando a qualquer tempo. Se houver um comando em execução (bit “busy” = 1), o comando será encerrado e o bit “busy” do registrador de estado será zerado.

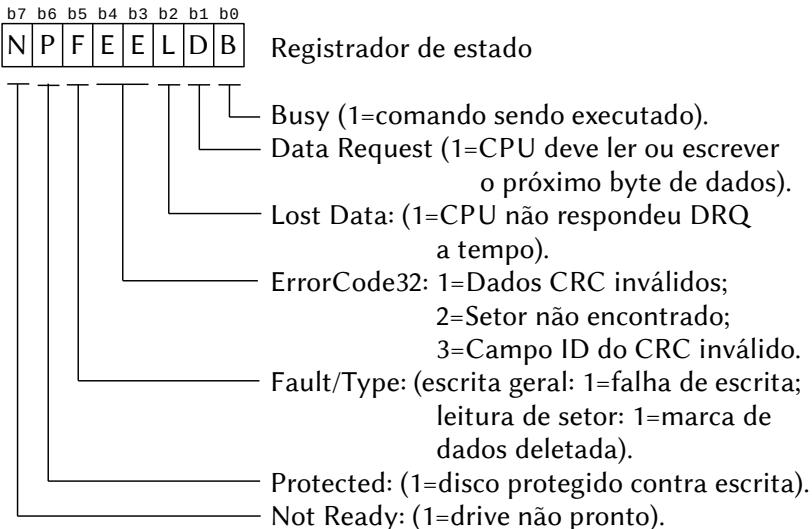
#### 7.5.2 – O registrador de estado

Ao receber qualquer comando, à exceção do comando Force Interrupt, o bit “busy” é setado e os outros bits do registrador de estado são atualizados ou limpos para o novo comando. O usuário tem a opção de ler o registrador de estado através do programa ou usar a linha DRQ juntamente com o DMA ou interrupção. Quando o registrador de dados é lido, o bit DRQ no registrador de estado e a linha DRQ são automaticamente zeradas. Uma escrita no registrador de dados também causa o mesmo efeito. O bit “busy” deve ser sempre monitorado para que o usuário saiba quando um comando está sendo executado. Ao usar a INTRQ, a checagem do bit “busy” não é recomendada porque a leitura desse bit resetará a linha INTRQ.

#### Registrador de estado para comandos tipo I



## Registrador de estado para comandos tipo II e III



Os bits 1~6 são zerados quando atualizados. Os códigos de erro (ErrorCode32) não são válidos para escrita ou leitura de trilhas.

## Registrador de estado para comandos tipo IV

Se um comando “Force Interrupt” for recebido durante a execução de outro comando, o bit 0 (busy) do registrador de estado é zerado e os bits restantes permanecem inalterados. Entretanto, se um comando “Force Interrupt” for recebido quando nenhum outro comando estiver em execução, o bit 0 (busy) do registrador de estado é zerado e os outros bits serão atualizados ou limpos. Nesse caso, o registrador de estado se comporta como nos comandos tipo I.

### 7.5.3 – Funções e cuidados adicionais

O FDC não possui internamente seleção para número de drive, lado, densidade de disco e controle liga/desliga motor dos drives. Essas funções devem ser adicionadas por circuitos externos, que devem ser controlados separadamente.

Como o FDC tem somente um registrador de trilha que deve ser usado para todos os drives, a posição da trilha deve ser salva na memória e o registrador de trilha deve ser atualizado a cada troca de drive.

### 7.5.4 – Formatação

Para que o disco possa ser utilizado, é necessário um processo chamado formatação. Na formatação, o disco é dividido logicamente em trilhas e setores. A tabela abaixo mostra o padrão de dados e sua interpretação pelo FDC no sistema MFM.

00~F4	Escreve 00 até F4
F5	Escreve A1, preset CRC
F6	Escreve C2
F7	Gera 2 bytes CRC
F8~FF	Escreve F8 até FF

### Exemplo de formatação

O exemplo a seguir mostra a sequência de dados que deve ser enviada para o comando “Write Track” para formatar um disco com 256 bytes por setor (o MSX usa setores de 512 bytes por padrão). Os valores à esquerda são contadores de repetição para escrita (em decimal) para os valores da direita.

Primeiro, o cabeçalho de trilha (Track Header) deve ser escrito, seguido pelo ID de setor e campos de dados dos setores (para cada setor). Finalmente, bytes 4EH devem ser escritos até o comando ser completado.

#### Track Header (cabecalho de trilha)

80	*	4EH
12	*	00H
03	*	F6H (escreve C2)
01	*	FCH (marca de índice)
50	*	4EH Campo ID do setor
12	*	00H
03	*	F5H (escreve A1, preset CRC)
01	*	FEH (marca ID de endereço)
01	*	número da trilha
01	*	número do lado

```

01 * número do setor
01 * 01 (tamanho do setor - 256 bytes)
01 * F7H (escreve 2 bytes CRC)
22 * 4EH

Campo de dados do setor
12 * 00H
03 * F5H (escreve A1, preset CRC)
01 * FBH (marca da dados de endereço)
256 * dados do setor
1 * F7H (escreve 2 bytes CRC)
54 * 4EH

Fim de trilha (preencher bytes não usados)
... * 4EH

```

### 7.5.5 – Endereços de acesso ao FDC

Nessa seção estão descritos diversos endereços de acesso para interfaces de drive, baseadas tanto em memória como em I/O. Para interfaces acessadas por memória, o slot onde a mesma estiver instalada deve estar habilitado. Para interfaces acessadas por I/O, não é necessário esse cuidado. Esse tipo de acesso foi utilizado somente em algumas interfaces brasileiras. O acesso padrão para o MSX é o por memória.

#### Endereços para acesso por memória (Padrão)

7FF8H	R	registrador de estado
7FF8H	W	Registrador de comando
7FF9H	R/W	Registrador de trilha
7FFAH	R/W	Registrador de setor
7FFBH	R/W	Registrador de dados
7FFCH	R?/W	Lado (bit 0) [Motor aqui?]
7FFDH	R?/W	Drive (bit 0) [Motor aqui?]
7FFEH	-	Não usado
7FFFH	R	Requisição de dados (bit 7) e busy (bit 6)

Nota: O MSXDOS/BarbarianLoader seleciona a memória nos endereços 8000H~BFFFH; nesse caso, devem ser usados os endereços BFFxH em vez de 7FFxH.

### **Endereços para acesso por memória (Alternativo)**

Esse mapeamento é utilizado somente pelo modelo SV738 (X'Press) da SpectraVideo, pelo BDOS da Technoahead e pelo BDOS arábico. Nesse último caso, os endereços usados são 7F80H~7F87H, e nos dois primeiros casos são 7FB8H~7FBFH.

7FB8H/7F80H	R	registrador de estado
7FB8H/7F80H	W	Registrador de comando
7FB9H/7F81H	R/W	Registrador de trilha
7FBAH/7F82H	R/W	Registrador de setor
7FBBH/7F83H	R/W	Registrador de dados
7FBCH/7F84H	R	bit 7 = IRQ/Não ocupado bit 6 = Requisição de dados
7FBCH/7F84H	W	bits 0/1 = Seleciona drive bit 2 = lado bit 3 = lado

Os endereços 7FBDH~7FBFH e 7F85H~7F87H não são usados.

### **Endereços para acesso por portas de I/O**

D0H	R	registrador de estado
D0H	W	Registrador de comando
D1H	R/W	Registrador de trilha
D2H	R/W	Registrador de setor
D3H	R/W	Registrador de dados
D4H	W	Drive (bit 1), Lado (bit 4), Motor (bit ??)
D4H	R	IRQ/Não ocupado (bit 7), Requisição de dados (bit 6)

Os endereços de D5H a D7H não são usados. Esse tipo de acesso é usado por todas as interfaces brasileiras, exceto pela ACVS/CIEL que usa o acesso por memória padrão.

A leitura pela porta D4H somente é suportada pela versão 3.0 ou superior. Para versões anteriores são usados os bits 0 e 1 do registrador de estado, que têm o mesmo significado. A versão 2.7 e superiores usam acesso misto, por portas de I/O e por memória padrão.

## ***Capítulo 8***

### **DISPOSITIVOS ADICIONAIS**

Este capítulo descreve alguns dispositivos que não foram descritos nos capítulos anteriores.

#### **8.1 – O RELÓGIO E A SRAM**

Nos micros MSX2 e superiores, é usado um chip específico para as funções de relógio do sistema, o RP-5C01. Ele é chamado de CLOCK-IC. Como é alimentado por baterias, está sempre ativo, mesmo com o micro desligado. O relógio dispõe de uma pequena SRAM que é usada para armazenar algumas funções que o MSX realiza automaticamente ao ser ligado.

##### **8.1.1 – Funções do Clock-IC**

###### **Relógio**

- Ler e atualizar os dados do ano, mês, dia do mês, dia da semana, horas, minutos e segundos;
- Apresentação da hora em 12 ou 24 horas;
- Meses de 30 e 31 dias são reconhecidos. O mês de fevereiro (28 dias) e os anos bissextos também são reconhecidos.

###### **Alarme**

- Quando ativo, o relógio gera um sinal na hora escolhida;
- O alarme é setado como “XXdia, XXhoras, XXminutos”.

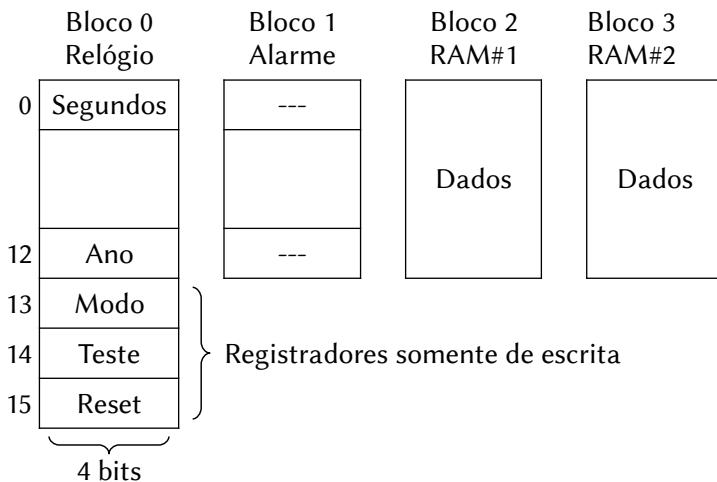
###### **Memória**

- Ajuste de tela (set adjust);
- Valores iniciais de SCREEN, WIDTH e COLOR;
- Volume e tom do beep;
- Cor da tela inicial;
- Código do país;
- Senha, prompt do BASIC ou título da tela inicial.

### 8.1.2 – Estrutura e registradores do Clock-IC

O CLOCK-IC possui quatro blocos de memória sendo que cada um consiste em 13 registradores de 4 bits cada, endereçados de 0 a 12. Possui também mais três registradores de 4 bits, para a seleção de blocos e controle das funções, sendo acessados pelos endereços 13 a 15.

Os registradores dos blocos (#0 a #12) e o registrador de modo (#13) podem ser lidos ou escritos. Os registradores de teste (#14) e de reset (#15) só podem ser escritos. Eles estão ilustrados abaixo.



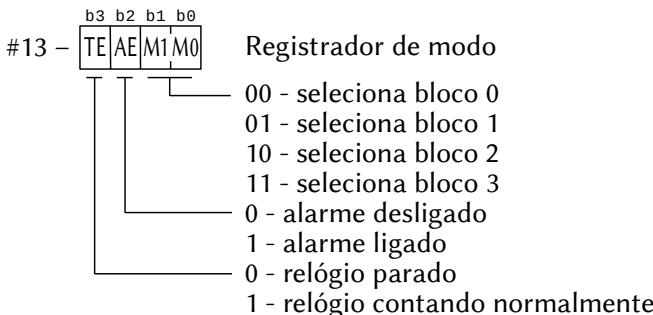
#### 8.1.2.1 – O registrador de modo (#13)

O registrador de modo tem três funções.

A primeira é a seleção de blocos. Os quatro blocos de 13 registradores de 4 bits cada (endereçados de #0 a #12) são selecionados pelos dois bits mais baixos do registrador de modo. Os registradores de #13 a #15 são acessados independentemente do bloco selecionado.

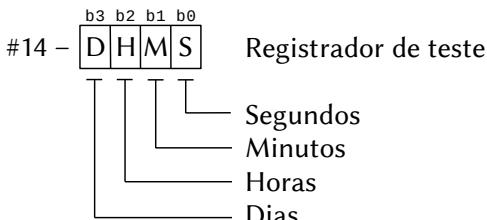
A segunda função é ligar ou desligar a saída de alarme. O bit 2 do registrador de modo é usado para isso. Porém o MSX2 standard não suporta a função de alarme, sendo que a alteração desse bit não causa efeito algum.

A terceira função é a parada do relógio. Escrevendo 0 no bit 3 do registrador de modo, a contagem de segundos é interrompida e a função de relógio paralisada. Setando o bit 3 em 1, a contagem é retomada.



### 8.1.2.2 – O registrador de teste (#14)

O registrador de teste (#14) é usado para incrementar rapidamente e confirmar a data e a hora do relógio. Setando em 1 cada bit desse registrador, pulsos de 16 384 Hz são inseridos diretamente nos registradores de dia, hora, minuto e segundo.



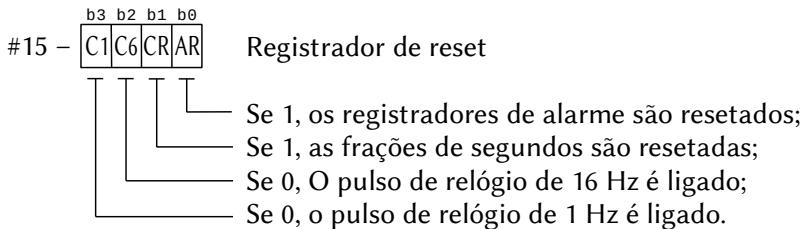
### 8.1.2.3 – O registrador de reset (#15)

Esse registrador tem três funções.

Para resetar o alarme, basta colocar o bit 0 em 1 e todos os registradores de alarme serão resetados.

O bit 1, quando colocado em 1, causa o reset das frações do contador de segundos. Essa função é útil para acertar os segundos corretamente.

Colocando o bit 2 em 0, o pulso do relógio de 16 Hz é ativado e colocando o bit 3 em 0, é ativado o pulso de 1 Hz.



#### 8.1.2.4 – Acertando o relógio e o alarme

O bloco 0 de memória é usado para o relógio. Para acertar a data e a hora, deve-se selecionar esse bloco e escrever os dados nos registradores corretos.

Já o bloco 1 é usado para o alarme. Nesse caso, só podem ser definidos os dias, horas e minutos.

No relógio, o ano é representado por 2 dígitos apenas (registradores #11 e #12). Para obter o ano correto deve-se somar 80 ou 1980 a esse valor. Por exemplo, se esses registradores forem 0, o ano correto será 1980.

O dia da semana é representado por um valor que varia de 0 a 6 no registrador #6. Veja a ilustração abaixo. Os bits indicados com “•” devem ser sempre 0.

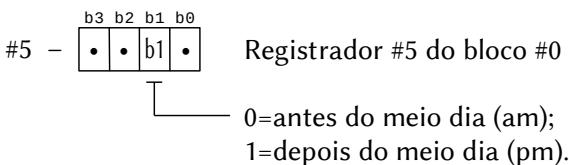
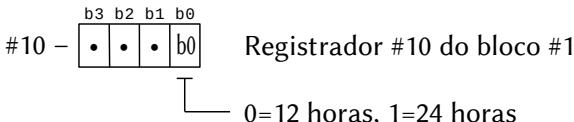
Bloco #0 – Relógio

	Bits →	3	2	1	0
0	Seg. 1º dígito	x	x	x	x
1	Seg. 2º dígito	•	x	x	x
2	Min. 1º dígito	x	x	x	x
3	Min. 2º dígito	•	x	x	x
4	Hora 1º dígito	x	x	x	x
5	Hora 2º dígito	•	•	x	x
6	Dia da semana	•	x	x	x
7	Dia 1º dígito	x	x	x	x
8	Dia 2º dígito	•	•	x	x
9	Mês 1º dígito	x	x	x	x
10	Mês 2º dígito	•	•	x	x
11	Ano 1º dígito	x	x	x	x
12	Ano 2º dígito	•	v	x	x

Bloco #1 – Alarme

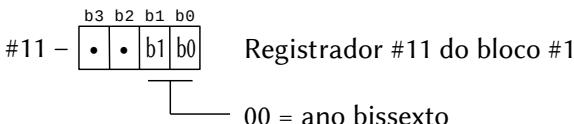
	Bits →	3	2	1	0
0	---	•	•	•	•
1	---	•	•	•	•
2	Min. 1º dígito	x	x	x	x
3	Min. 2º dígito	•	x	x	x
4	Hora 1º dígito	x	x	x	x
5	Hora 2º dígito	•	•	x	x
6	Dia da semana	•	x	x	x
7	Dia 1º dígito	x	x	x	x
8	Dia 2º dígito	•	•	x	x
9	---	•	•	•	•
10	12/24 horas	•	•	•	x
11	Ano bissexto	•	•	x	x
12	---	•	•	•	•

Dois modos podem ser selecionados para a contagem de horas: 12 horas ou 24 horas. No modo 24 horas, quando for 1 hora da tarde, o relógio indicará 13:00 horas e no modo 12 horas indicará 1:00 pm. O registrador #10 do bloco 1 é usado para essa função.



A flag am/pm no registrador #5 do bloco 0 só pode ser usada no caso de seleção de 12 horas pelo registrador #10 do bloco 1.

O registrador #11 do bloco 1 é um contador de 4 (0 a 3) incrementado a cada ano. Quando os dois bits mais baixos desse registrador forem 0, o ano será considerado bissexto e serão contados 29 dias para o mês de fevereiro. A referência para esse contador é o ano de 1980, que foi bissexto.



### 8.1.2.5 – Conteúdo da SRAM adicional

Os blocos 2 e 3 de SRAM do CLOCK-IC não têm função para o relógio. No MSX, eles são usados para armazenar alguns dados que o micro reconhece quando ligado para executar automaticamente algumas funções baseadas nesses dados.

Conteúdo do bloco #2			
	bit 3	bit 2	bit 1
0			ID
1			Ajuste horizontal (-8 ~ +7)
2			Ajuste vertical (-8 ~ +7)
3	•	•	Entrelaçado Screen
4			Largura inicial da tela (WIDTH) – low
5			Largura inicial da tela (WIDTH) – high
6			Código da cor inicial da frente
7			Código da cor inicial do fundo
8			Código da cor inicial da borda
9	Vel. CAS	Impressora	Click teclas Tec. Função
10			Tom do beep Volume do beep
11	•	•	Cor da tela abertura
12			Código nativo

O bloco 3 pode ter três funções diferentes, dependendo do conteúdo da posição ID (registrar #0 do bloco 3). Se o ID for igual a 0, o micro apresentará um título de até 6 caracteres na tela inicial. Se for igual a 1, o bloco 3 armazenará uma senha de até 6 caracteres que deverá ser digitada ao ligar o micro para que este possa ser acessado. Se o ID for igual a 2, será armazenado um novo prompt para o BASIC, no lugar do “Ok”, que também poderá ter até 6 caracteres.

A organização do bloco 3 para essas funções está ilustrada abaixo.

ID = 0 → Apresenta um título na tela inicial:

Bloco #3			
	bit 3	bit 2	bit 1
0			0 → Título tela inicial
1			1º caractere (low)
2			1º caractere (high)
:			:
11			6º caractere (low)
12			6º caractere (high)

ID = 1 → Armazena a senha (password)

Bloco #3

	bit 3	bit 2	bit 1	bit 0
0				1 → Senha
1				Usage ID=1
2				Usage ID=1
3				Usage ID=1
4	Senha			
5	Senha			A senha é armazenada
6	Senha			de forma comprimida
7	Senha			em 4 x 4 bits
8				Key cartridge flag
9				Key cartridge value
10				Key cartridge value
11				Key cartridge value
12				Key cartridge value

ID = 2 → armazena um novo prompt para o BASIC

Bloco #3

	bit 3	bit 2	bit 1	bit 0
0				2 → Prompt BASIC
1				1º caractere (low)
2				1º caractere (high)
:				:
11				6º caractere (low)
12				6º caractere (high)

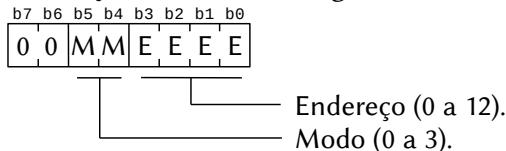
### 8.1.3 – Acesso ao Clock-IC

O acesso ao relógio e à memória mantida a bateria é feito através de duas rotinas da BIOS da Sub-ROM, sendo necessário o uso de chamada inter-slot para acessá-las. Duas rotinas estão disponíveis:

REDCLK (01F5H/SubROM)

Função: Lê um nibble de dados da memória do relógio (Clock-IC).

Entrada: C – Endereço da SRAM do relógio, conforme abaixo:



Saída: A – Nibble lido (4 bits mais baixos).

Registradores: AF.

WRTCLK (01F9H/SubROM)

Função: Escreve um nibble de dados na memória do relógio.

Entrada: C – Endereço da SRAM do relógio (igual a REDCLK).

A – Nibble a ser escrito (4 bits mais baixos).

Saída: Nenhuma.

Registradores: F.

Saída: Nenhuma.

A rotina abaixo veio como exemplo no MSX2 Technical Handbook.

```

; -----
; Define prompt do BASIC
; -----
;
WRTCLK: EQU 01F9H
EXTROM: EQU 015FH

ORG 0B000H

; Program start           ; Note: Define mensagem de
                           ;       prompt do BASIC.

START:
    LD   C,00110000B      ; Endereço do dado
    LD   A,2                ; ID := modo prompt
    CALL WRTRAM            ; Escreve na RAM do relógio
    LD   B,6                ; Contador
    LD   HL,STRING          ; Dados do prompt

L01:
    LD   A,(HL)            ; Lê string de dados
    AND  0FH                ; A := 4 bits altos
    INC   C                 ; Incrementa endereço

```

```

CALL WRTRAM           ; Escreve dados na RAM relógio
LD A, (HL)
RRCA
RRCA
RRCA
RRCA
AND 0FH
INC C                ; Incrementa endereço
CALL WRTRAM           ; Escreve 4 bits baixos
INC HL
DJNZ L01
RET

;--- Escreve dados na RAM do relógio ---

WRTRAM:
PUSH HL
PUSH BC
LD IX,WRTCLK
CALL EXTROM           ; Chamada inter-slot
POP BC
POP HL
RET

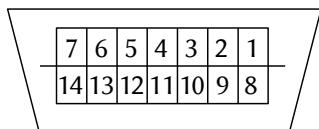
;--- String de dados ---

STRING:
DB 'Ready?'
END

```

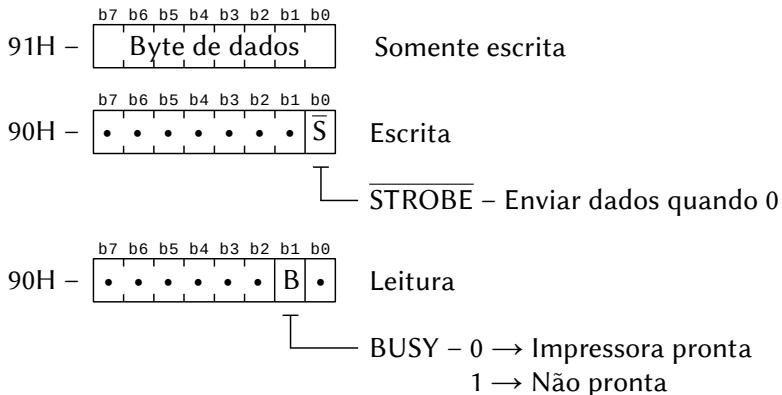
## 8.2 – INTERFACE DE IMPRESSORA

Essa seção descreve como acessar a impressora pela BIOS e diretamente através de portas de I/O. A interface de impressora é suportada pela BIOS, pelo BASIC e pelo DOS. O MSX usa porta paralela para acesso à impressora. O padrão adotado é o Centronics. O conector padrão também é definido (Amphenol 14 contatos com conector fêmea no micro).



- 1 – STROBE
- 2 ~ 9 – DADOS (b0~b7)
- 11 – BUSY
- 14 – GND

As portas de I/O usadas são as seguintes:



Os dados a serem enviados para a impressora dependem se esta foi especialmente desenvolvida para o padrão MSX ou não.

Numa impressora padrão MSX podem ser impressos todos os caracteres que saem no vídeo. Os caracteres gráficos especiais de código 01H a 1FH também podem ser impressos enviando o cabeçalho gráfico 01H seguido do código do caractere + 40H.

A mudança de linha numa impressora padrão MSX é feita enviando os caracteres de controle 0DH e 0AH.

O MSX tem uma função para transformar o código TAB (09H) para o número adequado de espaços em impressoras que não dispõem da função TAB. Isso é feito através de uma flag na área de variáveis de sistema:

RAWPRT (F41FH,1) – Substitui TAB por espaços quando for 0; caso contrário, não substitui.

### 8.2.1 – Acesso à impressora

A impressora pode ser acessada tanto diretamente quanto através de rotinas da BIOS. O acesso deve ser feito preferencialmente através das rotinas da BIOS para prevenir problemas de incompatibilidade e sincronização. As rotinas da BIOS dedicadas à impressora são as seguintes:

**LPTOUT (00A5H/Main)**

Função: Envia um caractere para a impressora.

Entrada: Código ASCII do caractere a ser enviado.

Saída: Se falhar, CY retorna setada.

Registradores: F.

**LPTSTT (00A8H/Main)**

Função: Testa o estado da impressora.

Entrada: Nenhuma.

Saída: A = 0 (e flag Z = 1) → Impressora não está pronta.

255 (e flag Z = 0) → Impressora pronta.

Registradores: AF.

**OUTDLP (014DH/Main)**

Função: Saída formatada para a impressora. Difere de LPTOUT nos seguintes pontos:

- 1 – Se o caractere enviado for um TAB (09H) serão enviados espaços até atingir um múltiplo de 8;
- 2 – Para impressoras não-MSX, hiraganas são convertidos para katakanas e caracteres gráficos são convertidos para caracteres de 1 byte;
- 3 – Se houver falha, ocorrerá um erro de I/O.

Entrada: A – Caractere a ser enviado.

Saída: Nenhuma.

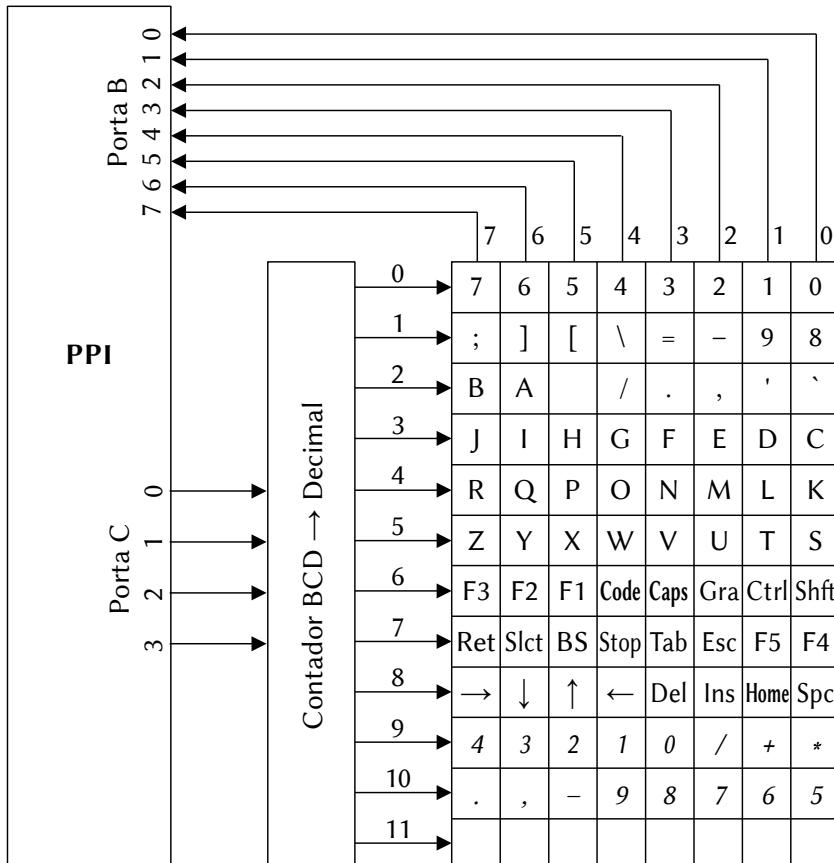
Registradores: F.

### 8.3 – INTERFACE DE TECLADO

A interface de teclado é controlada pelas portas B e C da PPI. Os 4 bits mais baixos da porta C enviam um valor de 0 a 10 correspondente à linha da matriz de teclado a ser lida e a porta B da PPI lê o estado

das teclas. Um bit 0 lido indica tecla pressionada. Como existem 11 linhas e 8 bits em cada linha, o teclado pode ter, no máximo, 88 teclas ( $11 * 8$ ).

Abaixo está ilustrada a matriz de teclado internacional.



As linhas '9' e '10' correspondem ao teclado numérico independente, quando o micro possuir um. Já a linha '11' é usada em apenas alguns modelos japoneses para acessar os programas que vem na ROM interna.

A matriz de teclado é diferente para cada país para adequar-se aos caracteres da língua local de cada um. Várias delas estão detalhadas no Apêndice, capítulo 1, seção 1.2 – Matrizes de teclado.

### **8.3.1 – Acesso ao teclado**

O acesso ao teclado pode ser feito tanto diretamente, acessando as portas B e C da PPI (A9H para a porta B e AAH para a porta C) quanto através da rotina SNSMAT (0141H/Main) da BIOS. Por se tratar de periférico lento, o acesso pela BIOS é preferível ao acesso direto. A rotina SNSMAT está descrita abaixo.

**SNSMAT      (0141H/Main)**

Função: Lê o valor de uma linha da matriz de teclado.

Entrada: A – Linha a ser lida.

Saída: A – Valor lido (o bit correspondente a uma tecla pressionada será 0).

Registradores: AF, C.

Outras rotinas relacionadas à interface de teclado são as seguintes:

**CHSNS      (009CH/Main)** - Checa o estado do buffer de teclado.

**CHGET      (009FH/Main)** - Entrada de um caractere pelo teclado.

**KILBUF      (0156H/Main)** - Limpa o buffer de teclado.

**CNVCHR      (00ABH/Main)** - Converte caractere gráfico.

**PINLIN      (00AEH/Main)** - Entrada de linha pelo teclado.

**INLIN      (00B1H/Main)** - Entrada de linha pelo teclado com prompt.

**INIFNK      (003EH/Main)** - Inicializa conteúdo das teclas de função.

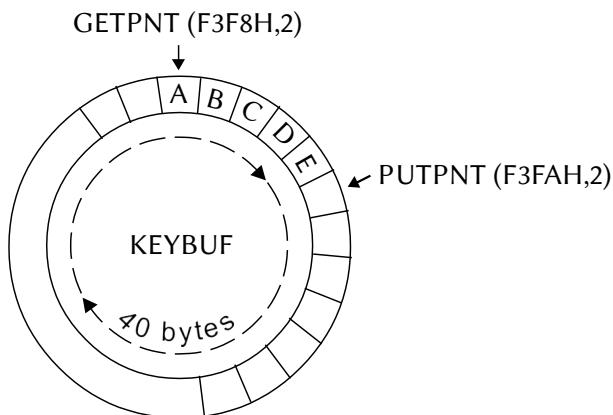
**BREAKX      (00B7H/Main)** - Detecta teclas CTRL+STOP.

A descrição detalhada dessas rotinas pode ser vista no apêndice, capítulo 8, seção 8.1 – Rotinas da MainROM.

### **8.3.2 – Varredura de teclado**

O MSX varre automaticamente toda a matriz de teclado a cada três ciclos de interrupção, ou seja, 20 vezes por segundo em uma máquina PAL (60 Hz), desde que as interrupções estejam habilitadas. Quando encontra uma tecla pressionada, ela é armazenada em um buffer circular

de 40 bytes. Esse buffer é designado KEYBUF (FBF0H~FC17H) e funciona conforme ilustrado abaixo.

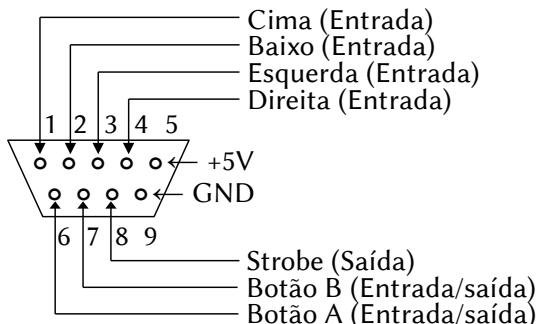


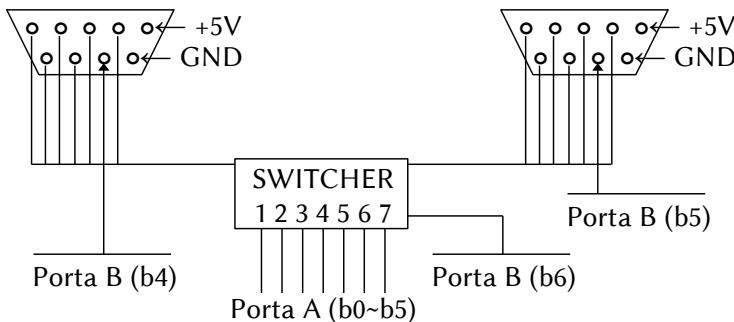
GETPNT aponta para o próximo caractere a ser obtido pela rotina CHGET e PUTPNT aponta para a próxima posição livre no buffer, a ser preenchida com o valor da próxima tecla pressionada.

#### 8.4 – INTERFACE UNIVERSAL DE I/O

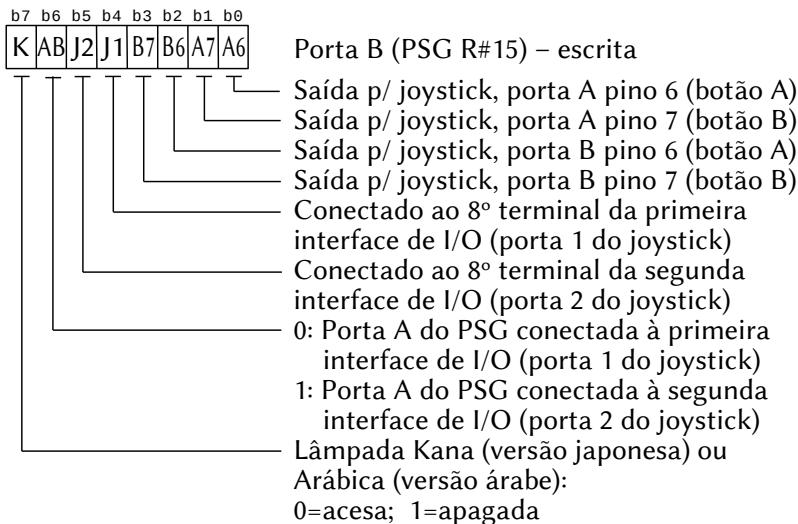
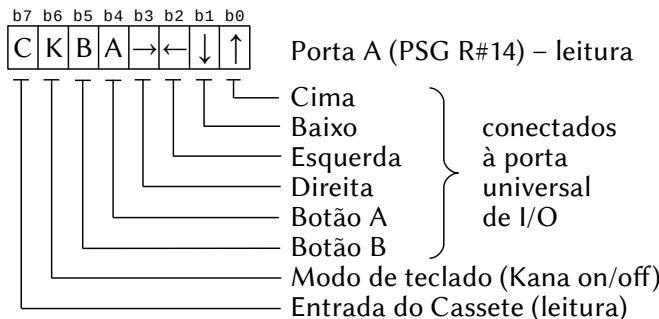
Como descrito no capítulo 5, o PSG tem duas portas de I/O para uso geral. Essas portas são conectadas à interface universal de I/O (portas do joystick). Vários dispositivos, além do joystick, podem ser conectados a essa porta, como mouse ou paddles. Para facilitar o acesso, existem algumas rotinas na BIOS que dão suporte a essas portas.

Essas interfaces são conectadas como ilustrado abaixo.





As duas portas do PSG são usadas como descrito abaixo:



Os bits b0~b3 da porta B são ligados aos pinos 6 e 7 de cada conector de joystick via buffers TTL de coletor aberto, a fim de poder enviar sinais para a porta. Os bits de pulso (bit 4 – J1 e bit 5 – J2) enviam um curto pulso através do pino 8. Se o bit 4 estiver em 0, nada é enviado. Se estiver em 1, o pulso é enviado. Os bits 0 e 1 indicam qual pino deve ter seu sinal elevado. O bit 0 corresponde ao pino 6 e o bit 1 ao pino 7. Ambos podem enviar dados ao mesmo tempo. De acordo com a tabela abaixo, observa-se as diversas possibilidades de enviar sinal pela porta de joystick.

R#15 (Porta B)	Pino 6	Pino 7	Pino 8
b7 b6 b5 b4 b3 b2 b1 b0	Botão A	Botão B	Strobe
x x 0 0 1 1 1 1	0	0	Vcc
x x 0 1 1 1 1 1	0	0	Pulso
x x 0 0 1 1 1 0	Vcc	0	Vcc
x x 0 1 1 1 1 0	Vcc	0	Pulso
x x 0 0 1 1 0 1	0	Vcc	Vcc
x x 0 1 1 1 0 1	0	Vcc	Pulso
x x 0 0 1 1 0 0	Vcc	Vcc	Vcc
x x 0 1 1 1 0 0	Vcc	Vcc	Pulso

O valor xx001111 é sempre devolvido ao registrador 15 após cada operação. Portanto, para se manter algum pino ligado (6 ou 7), é necessário criar um “loop” que ajuste constantemente o registrador 15 para o valor desejado, de acordo com a tabela acima. O pino 8, ao contrário dos pinos 6 e 7, mantém o nível de tensão alto.

O acesso à interface universal de I/O deve ser feito preferencialmente pelas rotinas da BIOS descritas abaixo:

GTSTCK (005DH/Main) - Lê estado do joystick.

GTTRIG (00D8H/Main) - Lê estado dos botões de disparo.

GTPDL (00DEH/Main) - Lê informação do paddle.

GTPAD (00DBH/Main) - Acessa vários dispositivos de I/O.

A descrição detalhada dessas rotinas pode ser vista no Apêndice, capítulo 8 – Rotinas da BIOS.

## **Capítulo 9**

### **O MSX TURBO R**

Nos modelos MSX turbo R, foi introduzida uma CPU de 16 bits, totalmente compatível com o Z80 a nível de instruções. A CPU R800 é construída em um chip LSI com encapsulamento QFP de 100 terminais. O clock interno do R800 é de 7,16 MHz. Ele também dispõe de 2 canais DMA e saída para multitarefa, mas essas opções não foram utilizadas.

O conjunto de instruções do R800 engloba todas as instruções do Z80 e acrescenta mais algumas, como multiplicação direta de 8 e 16 bits e oficializa algumas instruções “secretas” do Z80: o tratamento dos registradores de índice IX e IY como dois registradores de 8 bits cada, denominados de .ixl, .ixh, .iyL e .iyh.

Como o R800 é totalmente compatível com o Z80 a nível de instruções, é possível fazer um programa que funcione no MSX2 e colocar uma pequena rotina que detecta se o programa está residente em um MSX turbo R, e, nesse caso, ativar o R800 para acelerar, em média, 7 vezes a execução do programa. Alguns cuidados, entretanto, devem ser tomados. No caso de acesso direto por portas de I/O, mesmo a alguns componentes internos, como o OPLL, o modo R800 deve ser desligado, pois haverá dessincronização devido à maior velocidade do R800. No caso de acesso ao VDP não há problema, pois outro chip específico do MSX turbo R (o S1990) acerta o timing por hardware quando necessário. No caso da memória mapeada também não há problema. Qualquer outro acesso direto, entretanto, deve ser feito no modo Z80, para prevenir problemas de sincronização. No caso de acesso através da BIOS, BDOS ou BASIC não há nenhum problema, pois a BIOS compensa as diferenças de timing quando necessário.

#### **9.1 – ORGANIZAÇÃO DE SLOTS E PÁGINAS**

No MSX turbo R a organização de slots e páginas foi padronizada, sendo que a RAM é conectada diretamente ao R800 permitindo um tempo de acesso menor. Isso também simplifica o desenvolvimento de software específico. Os slots primários 0 e 3 são reservados para o sistema e os slots 1 e 2 são slots externos para o usuário. Os slots 0 e 3 são expandidos e sua organização é a seguinte:

	Slot 0-0	Slot 0-1	Slot 0-2	Slot 0-3
0000H	MainROM	-	-	-
3FFFH	MainROM	-	MSX Music	-
7FFFH	-	-	-	-
BFFFH	-	-	-	-
FFFFH	-	-	-	-

	Slot 3-0	Slot 3-1	Slot 3-2	Slot 3-3
0000H	Main RAM (Mínimo de 256K mapeados)	SubROM	-	Kanji Driver e Programas internos
3FFFH	-	Extended BASIC	DOS Kernel	-
7FFFH	-	-	-	-
BFFFH	-	-	-	-
FFFFH	-	-	-	-

Observe que o segmento correspondente ao DOS Kernel possui 4 segmentos de 16 Kbytes que são trocados exclusivamente na página 1. Os primeiros três segmentos são para o MSXDOS2 e o último para o MSXDOS1.

## 9.2 – WAIT STATES

Os wait states (ciclos de espera) no MSX turbo R são gerados em algumas condições especiais.

Quando um slot externo é acessado, são gerados 3 wait states. Isso é necessário para permitir que todo hardware desenvolvido antes do turbo R funcione corretamente, já que a maior velocidade do modo R800 poderia inviabilizar tais periféricos.

Quando a ROM interna é acessada, são gerados 2 wait states, devido à relativa lentidão dos chips de ROM.

Quando a DRAM interna é acessada, é gerado 1 wait state. Por isso, o acesso é mais rápido na DRAM que na ROM.

### 9.3 – MODOS DE OPERAÇÃO

Como o MSX turbo R tem 2 CPU's, existem alguns modos de operação envolvendo essas CPU's. Elas podem ser trocadas livremente durante o processamento, mas não podem ser ativadas simultaneamente. Duas combinações específicas entre o DOS e as CPU's são recomendadas: Z80/ROM e R800/DRAM, mas nada impede que o R800 funcione no modo ROM e o Z80 no modo DRAM. Quando o sistema inicializa, verifica o boot do disco para entrar no modo correto. Se não houver disco, o sistema entrará automaticamente no modo R800 DRAM com o Kernel MSXDOS2, a menos que a tecla “1” seja pressionada durante o reset, o que força o sistema a entrar no modo Z80.

Observe que há dois modos de operação do R800: o ROM e o DRAM. No modo ROM, toda a memória mapeada fica livre para uso. Já no modo DRAM, o sistema transfere para as quatro últimas páginas da memória mapeada o conteúdo de Main ROM (32 K), da Sub ROM (16 K) e da primeira parte do Kanji Driver. A vantagem disso é que as rotinas da BIOS passam a ser processadas mais rapidamente, já que a ROM é bem mais lenta em relação à DRAM. Em vista disso, há uma perda de 64 Kbytes de RAM disponível. Entretanto, se o programa que estiver sendo executado fizer muitos acessos à BIOS, a perda de memória em troca do ganho de velocidade pode ser vantajosa. Isso deve ser decidido durante o desenvolvimento do software. Os 64 Kbytes reservados no modo DRAM sempre ficam nas páginas lógicas mais altas da memória mapeada e não podem ser escritos, a despeito de serem RAM. Um modelo de rotina que pode ser incluída nos programas para que estes utilizem a velocidade do R800 está ilustrada abaixo.

```
RDSLت: EQU 0000CH
CALS廖: EQU 0001CH
CHGCPU: EQU 00180H
SLTROM: EQU 0FCC1H
;
;--- VERIFICA VERSAO ---
;
LD A, (SLTROM)
```

```

LD HL,0002DH
CALL RDSLTD
CP A,2
JR C,NAOTUR
;
;--- PREPARA TROCA DE MODO ---
;--- (ESCOLHER APENAS UMA DAS OPCOES) ---
;
;MODO Z80
LD A,11001110B
AND 002H
XOR 082H
;
;MODO R800 ROM
LD A,01000100B
AND 002H
XOR 081H
;
;MODO R800 DRAM
LD A,11001101B
AND 002H
XOR 082H
;
;--- TROCA DE MODO ---
;
LD IY,(SLTROM-1)
LD IX,CHGCPU
CALL CALSLT
;
NAOTUR:
END

```

A rotina apresentada faz um teste para verificar se está rodando em um MSX turbo R ou não. Se não estiver, pula para a label NAOTUR (termina), mas se estiver chama a rotina CHGCPU da BIOS, que troca os processadores de acordo com o valor passado no registrador A. Essa rotina funciona tanto sob o DOS com sob o BASIC, em qualquer endereço.

### 9.3.1 – Comparação de velocidade

A tabela abaixo mostra o ganho de velocidade quando se usa o R800 no lugar do Z80.

Instruções		Z80(us)	R800(us)	Ganho
LD	r,s	1.40	0.14	x 10.0
LD	r,(HL)	2.23	0.42	x 5.3
LD	r,(IX+n)	5.87	0.70	x 8.4
PUSH	qq	3.35	0.56	x 6.0
LDIR	(BC<>0)	6.43	0.98	x 6.6
ADD	A,r	1.40	0.14	x 10.0
INC	r	1.40	0.14	x 10.0
ADD	HL,ss	3.35	0.14	x 24.0
INC	ss	1.96	0.14	x 14.0
JP		3.07	0.42	x 7.3
JR		3.63	0.42	x 8.7
DJNZ	(B<>0)	3.91	0.42	x 9.3
CALL		5.03	0.84	x 6.0
RET		3.07	0.56	x 5.5
MULUB	A,r	160	1.96	x 81.6
MULUW	HL,rr	361	5.03	x 71.7

O ganho de velocidade em relação ao Z80 é muito grande, atingindo uma média de 7 vezes. As instruções MULUB e MULUW (multiplicação de operandos de 8 e 16 bits, respectivamente) são exclusivas do R800, não existindo no Z80. Para a obtenção do tempo em microsegundos, foram usadas rotinas otimizadas para o Z80.

### 9.3.2 – Instruções específicas do R800

O R800 oficializou algumas instruções secretas do Z80 e acrescentou duas de multiplicação. São as seguintes:

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
LD u,u'	u ← u'	• • • • • •	11 011 101 01 u u'	DD --	08	10	02	02
LD v,v'	v ← v'	• • • • • •	11 111 101 01 v v'	FD --	08	10	02	02

LD u,n	u ← n	• • • • •	11 011 101 00 u 110 ← n →	DD -- --	11	13	03	03
LD v,n	u ← n	• • • • •	11 111 101 00 v 110 ← n →	FD -- --	11	13	03	03
ADD p	A ← A + p	↑ ↓ v ↓ 0 ↑	11 011 101 10 000 r	DD --	08	10	02	02
ADD q	A ← A + q	↑ ↓ v ↓ 0 ↑	11 111 101 10 000 r	FD --	08	10	02	02
ADC p	A ← A+p+CY	↑ ↓ v ↓ 0 ↑	11 011 101 10 001 r	DD --	08	10	02	02
ADC q	A ← A+q+CY	↑ ↓ v ↓ 0 ↑	11 111 101 10 001 r	FD --	08	10	02	02
SUB p	A ← A - p	↑ ↓ v ↓ 1 ↑	11 011 101 10 010 r	DD --	08	10	02	02
SUB p	A ← A - q	↑ ↓ v ↓ 1 ↑	11 111 101 10 010 r	FD --	08	10	02	02
SBC p	A ← A-p-CY	↑ ↓ v ↓ 1 ↑	11 011 101 10 011 r	DD --	08	10	02	02
SBC p	A ← A-q-CY	↑ ↓ v ↓ 1 ↑	11 111 101 10 011 r	FD --	08	10	02	02
AND p	A ← A ∧ p	0 ↑ p ↓ 0 1	11 011 101 10 100 p	DD --	08	10	02	02
AND q	A ← A ∧ q	0 ↑ p ↓ 0 1	11 111 101 10 100 p	FD --	08	10	02	02
OR p	A ← A ∨ p	0 ↑ p ↓ 0 1	11 011 101 10 110 p	DD --	08	10	02	02
OR q	A ← A ∨ q	0 ↑ p ↓ 0 1	11 111 101 10 110 p	FD --	08	10	02	02
XOR p	A ← A ⊕ p	0 ↑ p ↓ 0 1	11 011 101 10 110 p	DD --	08	10	02	02
XOR q	A ← A ⊕ q	0 ↑ p ↓ 0 1	11 111 101 10 110 p	FD --	08	10	02	02
CP A,p	A - p	↑ ↓ v ↓ 1 ↑	11 011 101 10 111 p	DD --	08	10	02	02
CP A,q	A - q	↑ ↓ v ↓ 1 ↑	11 111 101 10 111 p	FD --	08	10	02	02
MULUB r	HL ← A * r	↑ ↓ 0 0 • •	11 101 101 11 r 001	ED --	--	--	14	14
MULUW HL,tt	DE:HL ← HL * tt	↑ ↓ 0 0 • •	11 101 101 11 tt0 011	ED --	--	--	36	36

Convenção dos registradores:

	00	01	10	11	100	101
p	•	•	•	•	IXH	IXL
q	•	•	•	•	IYH	IYL
rr	BC	DE	IY	SP	•	•
tt	BC	•	•	SP	•	•

**TZ** – Ciclos T Z80  
**Z1** – Z80 + M1  
**TR** – Ciclos T R800  
**RW** – R800 + Wait

As duas últimas instruções ('mulub' e 'muluw') não existem no Z80, sendo exclusivas do R800.

## 9.4 – A MSX-MIDI

A partir do segundo modelo MSX turbo R, a MSX-MIDI foi padronizada. MIDI quer dizer “Musical Instruments Digital Interface”, ou seja, interface digital para instrumentos musicais. Com ela é possível controlar instrumentos musicais que tenham entrada MIDI.

### 9.4.1 – Acesso à MSX-MIDI

A MSX-MIDI é controlada diretamente por portas de I/O. As portas reservadas são E8H a EFH quando a MIDI for interna e mais três se a MIDI for externa: E0H a E2H. Elas estão descritas abaixo:

E0H - Transmissão / recepção de dados (interface externa).

E1H - Porta de controle (interface externa).

E2H - Porta de seleção.

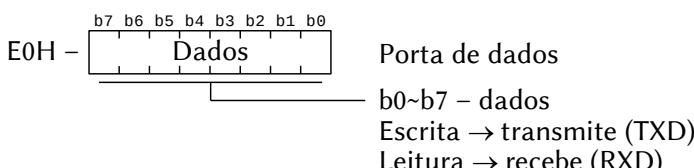
E8H - Transmissão / recepção de dados E9H - Porta de controle.

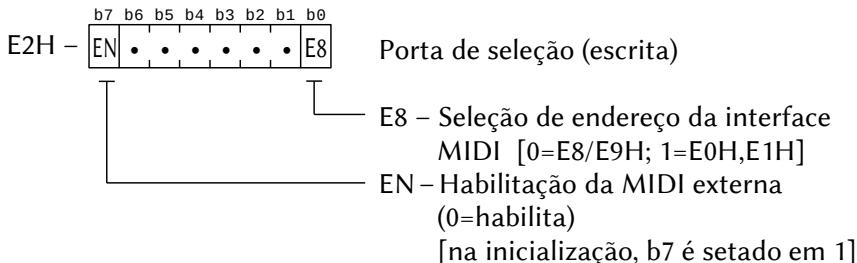
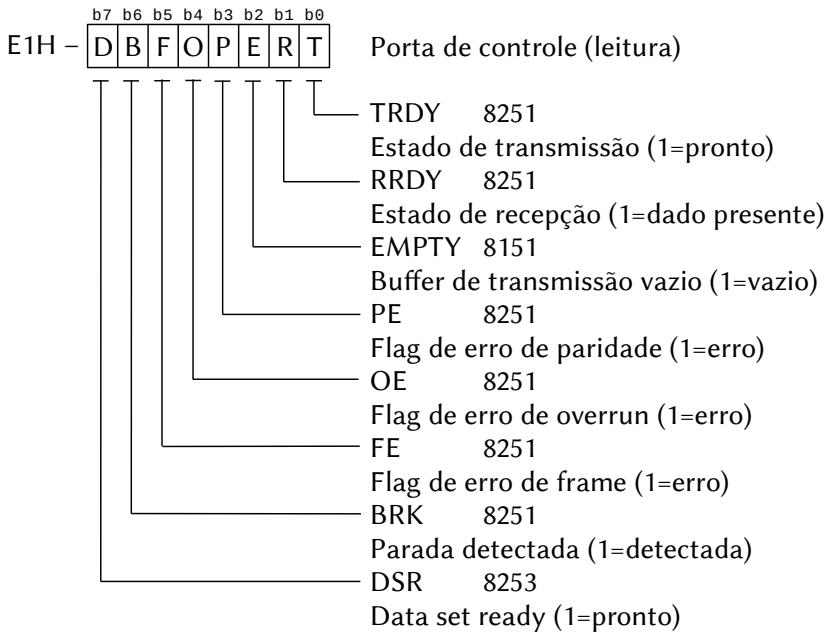
EAH - Latch dos sinais (escrita somente) EBH - Espelho de EAH.

ECH - Contador 0 EDH - Contador 1 EEH - Contador 2.

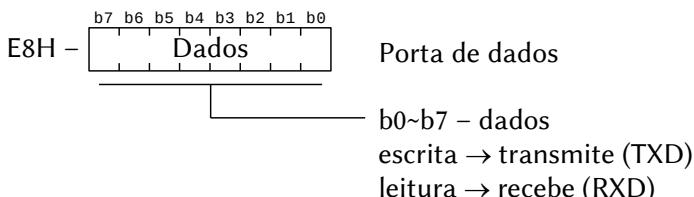
EFH - Controle dos contadores (escrita somente).

### 9.4.2 – Descrição das portas da MIDI externa



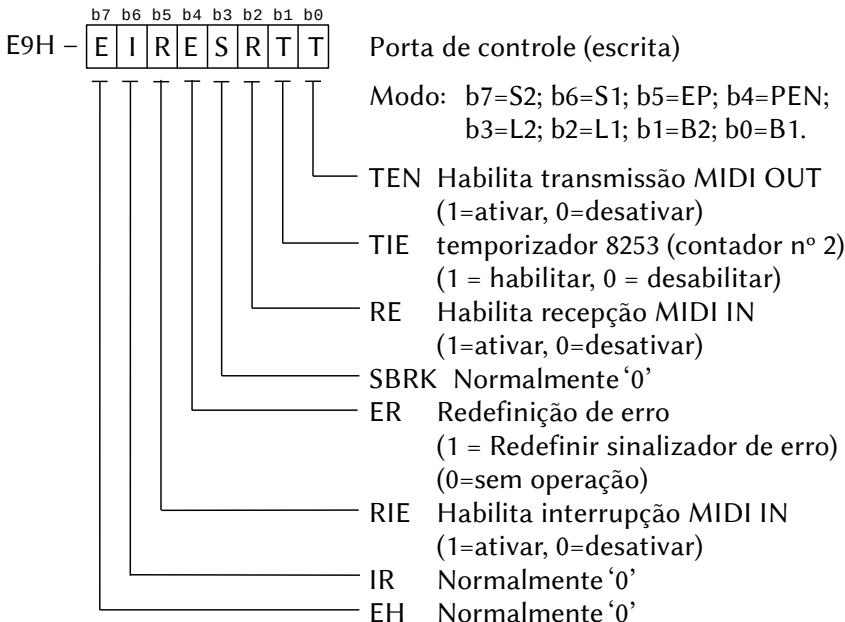


#### 9.4.3 – Descrição das portas da MIDI interna

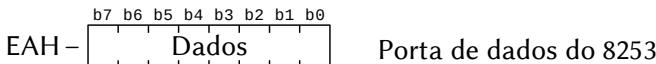




\* Organização idêntica à porta E1H para leitura. Para escrita, a organização é a seguinte:

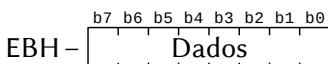


Quando um dado for escrito no modo comando, é necessário uma espera de 16 ciclos T (3,58 MHz) para o resultado e para escrever uma sequência de comandos.

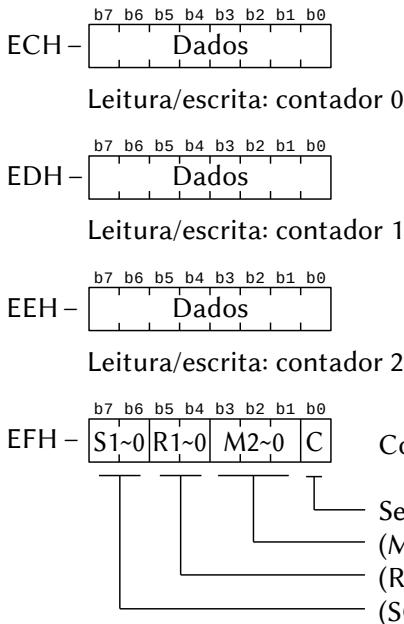


Escrita: 8253 OUT2 – latch dos sinais do terminal.

Leitura: sem efeito.



Essa porta é uma imagem de EAH.



#### 9.4.4 – MIDI interna e MIDI externa

A MSX-MIDI pode já vir internamente ao MSX turbo R como também pode ser implementada através de cartucho, mas somente para o MSX turbo R em diante. Se a MSX-MIDI for interna, o bit 0 do endereço 002EH da ROM estará setado em 1.

A diferença entre a MIDI interna ou externa pode ser obtida no endereço 4018H, conforme mostrado abaixo:

Endereço	Interna	Externa
4018H	41H (A)	??H (?)
4019H	50H (P)	??H (?)
401AH	52H (R)	??H (?)
401BH	4CH (L)	??H (?)
401CH	4FH (O)	4DH (M)
401DH	50H (P)	49H (I)
401EH	4CH (L)	44H (D)
401FH	4CH (L)	49H (I)

A interface MIDI também altera alguns hooks, e estes são diferentes conforme a MIDI seja interna ou externa.

Se a MIDI for interna, os hooks redirecionados serão:

Endereço	Novo nome	Nome antigo	Nova função
FF75H	HMDIN	HOKNO	MIDI IN
FF93H	HMDTM	HFRQI	Timer do 8253

No caso de MIDI externa, os hooks acima não podem ser usados; nesse caso, pode ser usado o hook HKEYI (FD9AH).

## 9.5 – TEMPORIZAÇÃO PARA O V9958

Embora o V9958 seja muito lento para o R800, não há nenhum problema de temporização no acesso direto ao mesmo porque o MSX-Engine S1990 gera pausas de 8µs por hardware entre acessos consecutivos ao VDP. Porém, esse é um tempo relativamente longo para o R800, correspondendo a 57 ciclos T. É possível evitar que o S1990 gere pausas para o R800 quando este acessa o VDP. Acontece que a pausa só é gerada a partir do segundo acesso, se este for feito antes do contador retornar a 0. Basta, então, temporizar por software, fazendo com que o R800 execute algumas operações entre acessos consecutivos ao VDP. As operações executadas devem tomar um mínimo de 57 ciclos T, o que faz com que o contador retorne a 0 antes do segundo acesso e evita a geração de pausas.

## 9.6 – A SRAM INTERNA

O MSX turbo R tem internamente uma pequena SRAM mantida a bateria, além da do relógio. O modelo FS-A1ST tem 16 Kbytes de SRAM e o modelo FS-A1GT tem 32 Kbytes.

Essa SRAM é dividida em segmentos de 8 Kbytes, que podem ser acessados exclusivamente no slot 3-3, o mesmo onde está o Kanji-Driver e os softwares gravados na ROM. Aliás, essa mesma ROM é mapeada em 192 segmentos de 8 Kbytes, num total de 1,5 Mbytes. A SRAM é mapeada com os números de segmento de 128 a 131.

O procedimento para desabilitar a ROM e habilitar a SRAM nesse slot é muito simples: basta escrever o número do segmento da SRAM num dos endereços de chaveamento, que são os seguintes:

- 6000H – habilita segmento em 0000H~1FFFH.
- 6400H – habilita segmento em 2000H~3FFFH.
- 6800H – habilita segmento em 4000H~5FFFH.
- 6C00H – habilita segmento em 6000H~7FFFH.
- 7000H – habilita segmento em 8000H~9FFFH.
- 7400H – habilita segmento em A000H~BFFFH.
- 7800H – habilita segmento em C000H~DFFFH.
- 7C00H – habilita segmento em E000H~FFFFH.

A SRAM interna é usada pelos softwares da ROM para salvar configurações dos mesmos, mas pode ser usada para muitos outros propósitos. Entretanto, é necessário um certo cuidado ao manipular dados no segmento 6000H~7FFFH porque este contém os endereços de chaveamento e a SRAM poderia ser desabilitada ou sofrer alteração de segmento ou endereços.

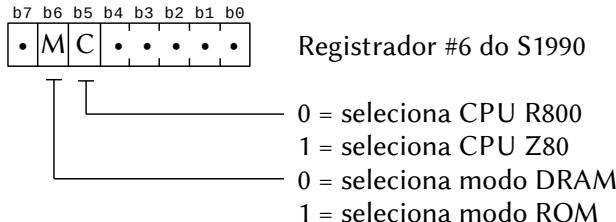
A SRAM interna não é compatível com a SRAM dos cartuchos PAC ou FM-PAC.

## 9.7 – O MSX ENGINE S1990

Nos modelos MSX turbo R, foi introduzido um chip para controlar suas diversas funções, o S1990. Esse chip contém muitos registradores internos, que são acessados pelas portas de I/O E4H (seleção de registradores) e E5H (dados). Entretanto, apenas os registradores 6, 14 e 15 têm funções conhecidas até o momento.

Os registradores 14 e 15 são somente leitura. O S1990 armazena o último byte escrito na memória no registrador 15 e o byte antecedente no registrador 14.

O registrador 6 é usado para controlar a CPU ativa e o modo ROM ou DRAM. Ele é organizado conforme ilustração abaixo:



Com esse registrador, é possível selecionar o modo Z80 DRAM. Para isso, primeiro, deve ser selecionado o modo Z80 ROM através da rotina CHGCPU (0180H) e depois deve ser escrito o valor 40H (64) no registrador 6.

# Apêndice

# 1 - CARACTERES E TECLADO

## 1.1 - TABELAS DE CARACTERES

### 1.1.1 - Tabela Japonesa

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_0
1	CTRL_F	CTRL_D	INS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	/	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ	Ⓖ	Ⓗ	Ⓘ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓝ	Ⓞ	Ⓝ
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	}	~	DEL	
8	▲	▼	◆	●	■	❖	❖	❖	❖	❖	❖	❖	❖	❖	❖	❖
9	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	ぞ	
A	。」	「	、	・	フ	ア	イ	ウ	エ	オ	カ	キ	ク	コ	サ	シ
B	一	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	
C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	
D	ミ	ム	メ	モ	ヤ	ユ	ラ	リ	ル	レ	ロ	ワ	ン	”	。	
E	た	ち	つ	て	ど	な	に	め	ね	の	は	ひ	ふ	へ	ほ	
F	み	む	め	も	や	ゆ	よ	ら	り	る	れ	る	わ	ん	■	

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	月	火	水	木	金	土	日	年	円	時	分	秒	百	千	万	
5	π	±	π	↔	ト	+	।	—	□	□	□	×	×	大	中	小

### 1.1.2 – Tabela Internacional

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BUFP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_0
1	CTRL_P	CTRL_Q	INS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ	Ⓖ	Ⓗ	Ⓘ	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ
5	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ
6	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ	ⓕ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	
7	ⓟ	ⓡ	ⓢ	ⓣ	ⓤ	ⓨ	ⓤ	ⓨ	ⓤ	ⓨ	ⓤ	ⓨ	ⓤ	ⓨ	ⓤ	⠀DEL⠀
8	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
9	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
A	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
B	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
C	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
D	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
E	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ
F	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		◎	●	◆	♦	♣	•	■	○	▢	♂	♀	♪	♫	*	
5	+	—	×	÷	×	○	•	-	□	□	▢	▢	▢	▢	▢	+

### 1.1.3 – Tabela Brasileira 1.0 (Expert 1.0)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_O
1	CTRL_P	CTRL_Q	INS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	À	Á	È	É	Ò	Ó	Ì	Í	Ã	Ñ	È	Í	Ò	Ó	Ã	Ñ
5	Ð	Þ	Ŕ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ	Ŗ
6	ç	â	ê	ë	î	î	ô	ô	ô	ô	â	ê	ë	î	î	ô
7	þ	ð	r	s	t	u	v	w	x	y	z	ç	í	ð	~	DEL
8	~	ó	é	á	ã	à	è	é	ë	î	í	ñ	â	ê	ë	â
9	É	æ	Æ	Ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø
A	á	í	ó	ú	ñ	ä	ö	ü	ë	ñ	ñ	ñ	ñ	ñ	ñ	ñ
B	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß	ß
C	—	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
D	◀	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
E	α	β	Γ	Π	Σ	σ	μ	γ	Φ	Θ	Ω	δ	ω	φ	ε	η
F	≡	±	≥	≤	ʃ	J	÷	≈	°	*	-	∫	^n	2	I	█

### 1.1.4 – Tabela Brasileira 1.1 (Expert 1.1 e Hotbit 1.2)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BUEEP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_0
1	CTRL_P	CTRL_Q	INS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	À	Á	Ê	Ó	Í	Ô	Ã	É	Ã	Ã	Ã	Ã	Ã	Ã	Ã	Ã
5	Ð	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ	Ñ
6	~	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	}	^	_	DEL
8	ç	é	á	à	â	ê	ó	ô	õ	ã	æ	ø	å	æ	ø	å
9	É	æ	Æ	Ø	ò	ô	û	ÿ	ö	ü	¢	£	¥	¤	ƒ	
A	Á	í	ó	ú	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ
B	Ã	ß	í	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö	ö
C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
D	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀	◀
E	α	β	Γ	Π	Σ	σ	μ	γ	Φ	Θ	Ω	δ	♀	♂	♪	♫
F	≡	±	≥	≤	ƒ	J	÷	≈	°	•	-	∜n	²	█	█	

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		◎	●	♥	♦	♣	·	■	○	⊗	♂	♀	♪	♫	*	
5	+	—	—	—	—	—	—	—	—	—	—	—	—	—	—	+

**Obs.:** o caractere na posição 9E (Cz) era “Pt” na primeira versão do Hotbit.

### **1.1.5 – Tabela Cirílica (Russa)**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	◎	●	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	*
5	+	-	T	T	T	+	I	-	R	I	L	-	X	/	+	

### 1.1.6 – Tabela Coreana

### 1.1.7 – Tabela Arábica (AX-170)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_0
1	CTRL_P	CTRL_R	IMS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	»	?
4	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ	Ⓖ	Ⓗ	Ⓘ	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ
5	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ
6	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ	ⓕ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ
7	ⓟ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯
8	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
9	*	۱	۲	۳	۴	۵	۶	۷	۸	۹	:	؛	۰	۱	۲	۳
A	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ	Ⓖ	Ⓗ	Ⓘ	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ
B	ܒ	ܓ	ܔ	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏
C	܂	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏	ܐ	ܑ
D	܂	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏	ܐ	ܑ
E	܂	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏	ܐ	ܑ
F	܂	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏	ܐ	ܑ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4																
5	܂	܃	܄	܅	܆	܇	܈	܉	܊	܋	܌	܍	܎	܏	ܐ	ܑ

### 1.1.8 – Tabela Arábica (AX-500)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL_B	CTRL_C	CTRL_D	CTRL_E	CTRL_F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL_N	CTRL_0
1	CTRL_F	CTRL_D	IMS	CTRL_S	CTRL_T	CTRL_U	CTRL_V	CTRL_W	SEL	CTRL_Y	CTRL_Z	ESC	→	←	↑	↓
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	»	?
4	Ø	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	]	^	_	
6	ا	ب	ت	ت	ت	ت	ت	ت	ت	ت	ت	ل	م	ن	و	
7	پ	ر	س	ت	ع	و	و	و	و	و	و	و	و	و	و	DEL
8	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
9	*	۱	۲	۳	۴	۵	۶	۷	۸	۹	:	؛	۰	=	۲	۱
A	Ø	ا	ب	ت	ت	ت	ت	ت	ت	ت	ت	ل	م	ن	و	
B	ø	ش	ش	ش	ش	ش	ش	ش	ش	ش	ش	[	]	^	_	
C	خ	خ	خ	خ	خ	خ	خ	خ	خ	خ	خ	ل	ل	ل	ل	
D	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	
E	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	
F	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	ـ	

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4																
5	۳	۴	۵	۶	۷	۸	۹	۰	۱	۲	۳	۴	۵	۶	۷	۸

## 1.2 – MATRIZES DE TECLADO

### 1.2.1 – Matriz Japonesa

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	;	:	] }	[ {	\	= +	- _	9 ( 8 *
Col. 2	b B	a A	_	/ ?	. >	, <	' ~	' "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	かな	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num .	Num ,	Num -	Num 9	Num 8	Num 7	Num 6	Num 5
Col. 11					実行		取消	

**Obs.1:** A coluna 11 é usada apenas pelos Panasonic modelos FS-A1WX, FS-A1WSX e turbo R, para acesso ao software interno em ROM. “実行” significa selecionar e “取消” significa cancelar.

**Obs.2:** A posição “かな” é a tecla “KANA” e corresponde à tecla CODE da versão internacional.

### 1.2.1.1 – Matriz Japonesa com a tecla かな/KANA travada

JIS	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	や ゃ	お も	え ん	う る	あ ん	ふ ん	め ん	わ を
Col. 1	れ れ	。 「	、	ー	へ エ	ほ エ	よ よ	ゆ ゆ
Col. 2	こ こ	ち ち	ろ ろ	め め	る る。	ね ね	む む	け け
Col. 3	ま ま	に に	く く	き き	は は	い い	し し	そ そ
Col. 4	す す	た た	せ せ	ら ら	み み	も も	り り	の の
Col. 5	つ つ	ん ん	さ さ	て て	ひ ひ	な な	か か	と と

ANSI	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	に に	な な	お も	え ん	う る	い い	あ ん	の の
Col. 1	も も	ろ 「	れ れ	る る	り り	ら ら	ね ね	ぬ ぬ
Col. 2	と と	さ さ	ん ん	。 を	わ わ	、 よ よ	。 」	ー ー
Col. 3	み み	ふ ふ	ま ま	そ そ	せ せ	く く	す す	つ つ
Col. 4	け け	か か	ほ ほ	へ へ	や や	ゆ ゆ	め め	む む
Col. 5	た た	は は	ち ち	き き	て て	ひ ひ	こ こ	し し

GRAPH	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	土 土	金 金	木 木	水 水	火 火	月 月	日 日	万 万
Col. 1	♣ ♣	。 。		円 円		ー ー	千 千	百 百
Col. 2	丶 丶		◆ ◆	♦ ♦	大 大	小 小	・ ・	♥ ♥
Col. 3			時 時	十 十	十 十	「 「	ト ト	レ レ
Col. 4	〒 〒		π π			分 分	中 中	
Col. 5		年 年	×		丶 丶		丶 丶	秒 秒

### 1.2.2 – Matriz do PX-7

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	;	:	] }	[ {	\	= +	- _	9 (
Col. 2	b B	a A	, „ „ ^	/ ?	. >	, <	' ~	` "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	かな	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9						SupI	Video	Comp

SupI → Superimpose

Video → Video

Comp → Computer

**Obs.:** O PX-7 não possui teclado numérico separado.

### 1.2.3 – Matriz Internacional

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	; :	] }	[ {	\	= +	- _	9 (	8 *
Col. 2	b B	a A		/ ?	. >	, <	' ~	` "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	CODE	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num .	Num ,	Num -	Num9	Num8	Num7	Num6	Num5

### 1.2.4 – Matriz Brasileira (Expert 1.1 e Hotbit)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 "	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	ç Ç	.. '	' `	\ ^	= +	- _	9 (	8 *
Col. 2	b B	a A	< >	/ ?	. :	, ;	[ ]	~ ^
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	CODE	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num .	Num ,	Num -	Num9	Num8	Num7	Num6	Num5

**Obs.:** O Expert 1.0 utiliza a matriz internacional.

### 1.2.5 – Matriz Argentina / Espanhola

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	ñ Ñ	] }	[ {	\	= +	- _	9 (	8 *
Col. 2	b B	a A		/ ?	. >	, <	; :	' "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	CODE	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num.	Num,	Num-	Num9	Num8	Num7	Num6	Num5

**Obs.:** Apenas as colunas 1 e 2 diferem da internacional.

### 1.2.6 – Matriz do Reino Unido (Inglaterra)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	; :	] }	[ {	\	= +	- _	9 (	8 *
Col. 2	b B	a A	£	/ ?	. >	, <	' ~	' "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	CODE	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num.	Num,	Num-	Num9	Num8	Num7	Num6	Num5

**Obs.:** Apenas a coluna 2 difere da internacional.

### 1.2.7 – Matriz Cirílica (Russa)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	& 6	% 5	¤ 4	# 3	" 2	! 1	+ ;	) 9
Col. 1	v V	* :	h H	- ^	= _	\$ 0	( 8	' 7
Col. 2	i I	f F	? /	< ,	@	b B	> .	\
Col. 3	o O	[ }	r R	p P	a A	u U	w W	s S
Col. 4	k K	j J	z Z	] }	t T	x X	d D	l L
Col. 5	q Q	n N	~	c C	m M	g G	e E	y Y
Col. 6	F3	F2	F1	РУС	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num.	Num,	Num-	Num9	Num8	Num7	Num6	Num5

#### 1.2.7.1 – Matriz Cirílica com tecla РУС/CODE travada

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	& 6	% 5	¤ 4	# 3	" 2	! 1	+ ;	) 9
Col. 1	ж Ж	* :	х Х	ъ ъ	= _	\$ 0	( 8	' 7
Col. 2	и И	ф Ф	? /	< ,	ю Ю	б Б	> .	э Э
Col. 3	о О	ш Ш	р Р	п П	а А	у У	в В	с С
Col. 4	к К	й Й	з З	щ щ	т Т	ь ь	д Д	л Л
Col. 5	я Я	н Н	ч Ч	ц Ц	м М	г Г	е Е	ы ы

### 1.2.8 – Matriz Coreana

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 ' 6 & 5 % 4 \$ 3 # 2 " 1 ! 0 0							
Col. 1	;	+	[ { @ ` ￦   ^ ~ - = 9 ) 8 (					
Col. 2	b B a A _		/ ? . > , < ] }					
Col. 3	j J i I h H g G f F e E d D c C							
Col. 4	r R q Q p P o O n N m M l L k K							
Col. 5	z Z y Y x X w W v V u U t T s S							
Col. 6	F3 F2 F1 한글		CAPS	GRAPH	CTRL	SHIFT		
Col. 7	RET SLCT BS STOP TAB ESC F5 F4							
Col. 8	→ ↓ ↑ ← DEL INS HOME SPACE							
Col. 9	Num4 Num3 Num2 Num1 Num0 Num/ Num+ Num*							
Col. 10	Num. Num, Num- Num9 Num8 Num7 Num6 Num5							

#### 1.2.8.1 – Matriz coreana com a tecla 한글/CODE travada

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0								
Col. 1								
Col. 2	ㅌ ㅁ							
Col. 3	ㅓ ㅏ ㅗ ㅜ ㅎ ㅕ ㅛ ㅚ ㅚ							
Col. 4	ㄱ ㄲ ㅂ ㅃ ㅋ ㄲ ㆁ ㆁ							
Col. 5	ㅋ ㅌ ㅍ ㅌ ㅌ ㅍ ㅌ ㅌ ㅌ							

### 1.2.9 – Matriz Arábica

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	7 &	6 ^	5 %	4 \$	3 #	2 @	1 !	0 )
Col. 1	؛ :	] }	[ {	\	= +	- _	9 (	8 *
Col. 2	b B	a A		/ ?	. >	, <	' ~	` "
Col. 3	j J	i I	h H	g G	f F	e E	d D	c C
Col. 4	r R	q Q	p P	o O	n N	m M	l L	k K
Col. 5	z Z	y Y	x X	w W	v V	u U	t T	s S
Col. 6	F3	F2	F1	CODE	CAPS	GRAPH	CTRL	SHIFT
Col. 7	RET	SLCT	BS	STOP	TAB	ESC	F5	F4
Col. 8	→	↓	↑	←	DEL	INS	HOME	SPACE
Col. 9	Num4	Num3	Num2	Num1	Num0	Num/	Num+	Num*
Col. 10	Num .	Num ,	Num -	Num 9	Num 8	Num 7	Num 6	Num 5

#### 1.2.9.1 – Matriz arábica com modo árabe ativado

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Col. 0	٧	٦	٥	٤	٣	٢	١	.
Col. 1	ك		ج				هـ	ـ
Col. 2	لـ لـ	ـ شـ		ـ ؟ـ		ـ ،ـ	ـ ؟ـ	ـ "
Col. 3	ـ تـ	ـ هـ	ـ أـ	ـ لـ	ـ بـ	ـ ثـ	ـ ذـ يـ	ـ ذـ دـ
Col. 4	ـ قـ	ـ ضـ	ـ .ـ	ـ حـ	ـ خـ	ـ لـ	ـ وـ	ـ مـ
Col. 5	ـ طـ	ـ غـ	ـ ءـ	ـ صـ	ـ زـ	ـ عـ	ـ فـ	ـ إـ شـ

## 1.3 – LAYOUTS DE TECLADO

### **1.3.1 – Layout Internacional**

ESC	!	@	#	\$	%	^	&	*	(	)	-	=		BS
TAB	Q	W	E	R	T	Y	U	I	O	P	{	}	[	] RETURN
CTRL	A	S	D	F	G	H	J	K	L	:	"	,	~	←
SHIFT	Z	X	C	V	B	N	M	<	>	?	'	..	^	SHIFT
	CAPS	GRAPH	SPACE						CODE					

### **1.3.2 – Layout Japonês (JIS)**

### 1.3.3 – Layout Japonês (ANSI)

ESC	!あ 1あ	"い 2い	#う 3う	\$え 4え	%お 5お	&な 6な	'に 7に	(ぬ 8ぬ	)ね 9ね	=の 0の	-ら -	~へり り	¥ ¥	BS
TAB	Qか Q	Wき か	Eく き	Rけ く	Tこ け	Yは こ	Uひ は	Iふ ひ	Oへ ふ	Pほ へ	@れ ほ	{「 {「	ろ ろ	RETURN
CTRL	Aさ A	Sし S	Dす D	Fせ F	Gそ G	Hま H	Jみ J	Kむ J	Lめ み	; ム	+も も	* :	- 。	}`。 }`。
SHIFT	Zた Z	Xち X	Cつ C	Vて つ	Bと V	Nや て	や B	Mゆ や	ゆ N	<よ M	>よ ゆ	?よ ,	。わ .よ	।を 。わ
	CAPS	GRAPH	SPACE									かな		

### 1.3.4 – Layout Brasileiro 1.0 (Expert 1.0)

ESC	!	@	#	\$	%	^	&	*	(	)	-	=		BS
TAB	Q	W	E	R	T	Y	U	I	O	P	{	}		RETURN
CTRL	A	S	D	F	G	H	J	K	L	:	"	,	~	←
SHIFT	Z	X	C	V	B	N	M	<	>	?	/	'	^	SHIFT
CAPS	L GRA	SPACE									R GRA			

### 1.3.5 – Layout Brasileiro 1.1 (Hotbit / Expert 1.1)

ESC	!	@	#	\$	%	"	&	*	(	)	-	=	^	BS
TAB	Q	W	E	R	T	Y	U	I	O	P	,	'	..	RETURN
CTRL	A	S	D	F	G	H	J	K	L	Ç	^	[	]	←
SHIFT	Z	X	C	V	B	N	M	;	:	?	/	>	<	SHIFT
CAPS		SPACE									GRAPH	CODE		

**Obs.:** Para o Expert, as teclas GRAPH e CODE são renomeadas para L GRA e R GRA, na mesma posição da versão 1.0.

### 1.3.6 – Layout do Reino Unido (Inglês)

ESC	!	@	#	\$	%	^	&	*	(	)	-	=		BS
TAB	Q	W	E	R	T	Y	U	I	O	P	{	}		RETURN
CTRL	A	S	D	F	G	H	J	K	L	:	"	,	£	←
SHIFT	Z	X	C	V	B	N	M	<	>	?	/	'	^	SHIFT
CAPS	GRAPH	SPACE									CODE			

### 1.3.7 – Layout Argentino / Espanhol

ESC	!	@	#	\$	%	^	&	*	(	)	-	=		BS
TAB	Q	W	E	R	T	Y	U	I	O	P	{	}	[	] RETURN
CTRL	A	S	D	F	G	H	J	K	L	Ñ	"	:	;	◀
SHIFT	Z	X	C	V	B	N	M	<	>	?	/	'	^	SHIFT
	CAPS	GRAPH	SPACE									CODE		

### 1.3.8 – Layout Russo (Cirílico)

ESC	;	1	2	3	4	5	6	7	8	9	0	-	Ћ	^	BS
TAB	Й	Ц	У	К	Е	Н	Г	Ш	Щ	З	Х	:	*	RETURN	
CTRL	Ф	Ы	В	А	П	Р	О	Л	Д	Ж	Э	.	>	◀	
SHIFT	Я	Ч	С	М	И	Т	Ь	Х	Б	Ю	Ѡ	,	/	SHIFT	
	CAPS	GRAPH	SPACE									РУС			

### 1.3.9 – Layout Coreano (CPC-400)

ESC	!	"	#	\$	%	&	'	(	)	0	=	~		₩	BS
TAB	Q	ㅃ	W	ㅉ	E	ㄸ	R	ㄱ	ㅋ	U	ㅓ	I	ㅗ	ㅐ	RETURN
CTRL	A	ㅁ	S	ㄴ	D	ㅇ	F	ㄹ	ㅎ	H	ㅓ	J	ㅓ	ㅣ	◀
SHIFT	Z	ㅋ	X	ㅌ	C	ㅊ	V	ㅍ	ㅂ	N	ㅏ	M	ㅡ	?	SHIFT
	CAPS	GRAPH	SPACE									한글			

### 1.3.10 – Layout Arábico (AX-170)

ESC	!	@	#	\$	%	^	&	*	(	)	-	=		BS
TAB	Q	W	E	R	T	Y	U	I	O	P	[	{	}	RETURN
CTRL	A	S	D	F	G	H	J	K	L	:	"	"	~	◀
SHIFT	Z	X	C	V	B	N	M	<	>	?	!	❖	SHIFT	
	CAPS	GRAPH	SPACE								CODE			

### **1.3.11 – Layout Francês (ML-F80)**

ESC	1 &	2 é	3 “	4 ’	5 (	6 §	7 è	8 !	9 ç	0 à	º )	- -	> <	BS
TAB	A	Z	E	R	T	Y	U	I	O	P	..	*	\$	RETURN
CTRL	Q	S	D	F	G	H	J	K	L	M	%	£	#	←
SHIFT	W	X	C	V	B	N	M	?	,	.	/	+	=	SHIFT
	CAPS	GRAPH	SPACE							CODE				

### **1.3.12 – Layout Germânico (HB-F700D)**

## 1.4 – CÓDIGOS DE CONTROLE

Atalho	DEC	HEX	Função
Ctrl+A	001	01H	Determina caractere gráfico
Ctrl+B	002	02H	Desvia cursor p/ início da palavra anterior
Ctrl+C	003	03H	Encerra a condição de entrada
Ctrl+D	004	04H	
Ctrl+E	005	05H	Cancela caract. do cursor até fim da linha
Ctrl+F	006	06H	Desvia cursor p/ início da palavra seguinte
Ctrl+G	007	07H	Aciona o beep
Ctrl+H	008	08H	Apaga letra anterior ao cursor (BS)
Ctrl+I	009	09H	Move cursor p/ pos. Tab. seguinte (TAB)
Ctrl+J	010	0AH	Muda de linha (Linefeed)
Ctrl+K	011	0BH	Volta cursor para a posição 1,1 (HOME)
Ctrl+L	012	0CH	Limpa a tela e volta cursor p/ a pos. 1,1
Ctrl+M	013	0DH	Retorno do Carro (RETURN)
Ctrl+N	014	0EH	Move o cursor para o fim da linha
Ctrl+O	015	0FH	
Ctrl+P	016	10H	
Ctrl+Q	017	11H	
Ctrl+R	018	12H	Liga/desliga modo de inserção (INS)
Ctrl+S	019	13H	
Ctrl+T	020	14H	
Ctrl+U	021	15H	Apaga toda a linha na qual está o cursor
Ctrl+V	022	16H	
Ctrl+W	023	17H	
Ctrl+X	024	18H	(SELECT)
Ctrl+Y	025	19H	
Ctrl+Z	026	1AH	(EOF) – End of File → Fim de Arquivo
Ctrl+[	027	1BH	(ESC) – Escape
Ctrl+\	028	1CH	Move o cursor para a direita
Ctrl+]	029	1DH	Move o cursor para a esquerda
Ctrl+^	030	1EH	Move o cursor para cima
Ctrl+_	031	1FH	Move o cursor para baixo
Delete	127	7FH	Apaga a letra que está sob o cursor (DEL)

## 2 – MAPA DAS PORTAS DE I/O

00H~01H	Porta MIDI (output) do Music Module (não usar ao mesmo tempo com o Sony Sensor Kid Cartridge)
00H~01H	Sony Sensor Kid Cartridge (não usar c/ Music Module)
02H~03H	FAC MIDI Interface (espelhado em 00H~07H)
04H~05H	Music Module MIDI (input)
00H~07H	MD Telcom modem
08H~09H	Sem uso conhecido
0AH	DAC do Music Module
08H~0EH	Sem uso conhecido
0FH	MegaRAM Zemina
10H~11H	Emulação do PSG para MegaflashROM em FPGA
12H~13H	Sem uso conhecido
14H~17H	Cartucho YM2608 OPNA
18H~19H	Leitor de código de barras Philips NMS 1170/20 1AH~1FH Sem uso conhecido
20H~28H	Modem Philips NMS1251 (config. 30H~38H via jumper) Modem Miniware M4000 (config. 30H~38H via jumper)
21H~27H	Sunrise MP3 player
27H~2FH	Interface serial Philips NMS 1210/1211/1212 (configurável em 37H~3FH via jumper)
28H~29H	DenYoNet ethernet interface
2AH~2BH	Cartucho PlaySoniq (setting registers)
30H~38H	Modem Philips NMS1251 (config. 20H~28H via jumper) Modem Miniware M4000 (config. 20H~28H via jumper) Interface SCSI Green-Mak Interface para CD-ROM Philips NMS 0210
37H~3FH	Interface serial Philips NMS 1210/1211/1212 (configurável em 27H~2FH via jumper)
3CH	Registrador de controle do Musical Memory Mapper
3FH	Acesso ao reg. do SN76489 do Musical Memory Mapper
40H~4FH	Acesso às portas de I/O comutáveis. 40H (R/W) ID do dispositivo 41H~4FH (R/W) acesso ao dispositivo
48H~49H	Cartucho Franky (SN76489 e VDP )
50H~5DH	Sem uso conhecido
5EH~5FH	Interface GR8NET (Ethernet)

60H~6FH	VDP V9990
60H	(R/W) Acesso à VRAM
61H	(R/W) Acesso à paleta de cores
62H	(R/W) Acesso aos comandos de hardware
63H	(R/W) Acesso aos registradores
64H	(W) Seleção de registradores
65H	(R) Porta de status
66H	(W) Flag de interrupção
67H	(W) Controle do sistema
68H	(W) Endereço da Kanji-ROM (low) – 1
69H	(R/W) Endereço da Kanji-ROM (high) e dados – 1
6AH	(W) Endereço da Kanji-ROM (low) – 2
6BH	(R/W) Endereço da Kanji-ROM (high) e dados – 2
6CH~6FH	– Não usadas
70H~73H	Cartucho MIDI Saurus
74H~76H	Sem uso conhecido
77H	Super Game 90
78H~7BH	Sem uso conhecido
7CH~7DH	MSX-MUSIC (YM2413)
7CH	(W) Seleciona registradores
7DH	(W) Porta de dados
7EH~7FH	Cartucho Moonsound (OPL4) – Síntese PCM
7EH	Registradores PCM (wave)
7FH	Dados PCM (wave)
80H~87H	Interface serial RS232C padrão
80H	(R/W) USART 8251 – Registrador de dados
81H	(R/W) USART 8251 – Reg. de status e comando
82H	(R/W) USART 8251 – Status / comunicação
83H	(R/W) Máscara de interrupção
84H	(R/W) 8253 – Contador 1
85H	(R/W) 8253 – Contador 2
86H	(R/W) 8253 – Contador 3
87H	(W) Comando dos contadores
88H~8BH	Acesso ao V9938 externo
8CH~8DH	MSX Modem
8EH~8FH	Megaram
8EH	Seleção de páginas
8FH	Megaram-Disk

90H~91H	Impressora
90H	(R) Status
91H	(W) Dados
92H~93H	Sem uso conhecido
94H	Direção a porta de impressora (não padronizado)
95H~97H	Sem uso conhecido
98H~9BH	VDP TMS9918/V9938/V9958
98H	(R/W) Lê/escreve dados na VRAM
99H	(R/W) Lê registrador de estado Escreve no registrador de controle
9AH	(W) Escreve nos registradores de paleta
9BH	(W) Escreve no reg. especificado indiretamente
9CH~9FH	Sem uso conhecido
A0H~A2H	PSG AY-3-8910
A0H	(W) Porta de endereço
A1H	(W) Porta de escrita de dados
A2H	(R) Porta de leitura de dados
A3H	Sem uso conhecido
A4H~A5H	PCM (Turbo R)
A4H	(R/W) Porta de dados
A5H	(R/W) Porta de comando
A6H	Sem uso conhecido
A7H	Controla luzes do painel no MSX turbo R
	bit 1 = LED Pause
	bit 7 = LED turbo
A8H~ABH	PPI 8255
	A8H (R/W) Porta A da PPI (seleção de slot)
	A9H (R/W) Porta B da PPI (leitura de teclado)
	AAH (R/W) Porta C da PPI (linha teclado / click teclas)
	ABH (W) Porta de comando da PPI
ACH~AFH	MSX-Engine (1chipMSX control)
B0H~B3H	Expansão de memória (especificação SONY 8255)
	B0H Linhas de endereço A0~A7
	B1H Linhas endereço A8~A10, A13~A15, controle, R/W
	B2H Linhas de endereço A11~A12 e dados D0~D7
B4H~B5H	IC do relógio (RP-5C01)
	B4H Endereço dos registradores
	B5H Leitura/escrita de dados
B6H~B7H	Leitor de cartão?
B8H~BBH	Controle de caneta ótica (especificação SANYO)

BCH~BFH	Controle VHD (especificação JVC 8255)
C0H~C1H	MSX-Audio Y8950 C0H (R/W) Seleciona regs e lê reg. de status C1H (R/W) Escreve ou lê reg. especificado
C0H~C3H	Portas alternativas para Moonsound/OPL4
C4H~C7H	Cartucho Moonsound (OPL4) – Síntese FM C4H FM register array 0 (banco 1) e reg. de status C5H FM (dados) C6H FM register array 1 (banco 2) C7H Espelho de C5H (o acesso por C5H é preferido)
C8H~CCH	Interface serial assíncrona
CDH~CFH	Sem uso conhecido
D0H~D7H	Reservadas para interface de disco
D8H~D9H	Kanji-ROM Jis 1 D8H (W) Linhas de endereço A0~A5 D9H (R/W) Linhas de endereço A6~A11 e dados D0~D7
DAH~DBH	Kanji-ROM Jis 2 DAH (W) Linhas de endereço A0~A5 DBH (R/W) Linhas de endereço A6~A11 e dados D0~D7
DCH~DDH	Cartucho Playsoniq (suporte ao Sega Gamepad)
DEH~DFH	Sem uso conhecido
E0H~E2H	MSX-MIDI externa E0H Transmissão / recepção de dados E1H Porta de controle E2H Porta de seleção
E3H	Sem uso conhecido
E4H~E7H	Acesso ao S1990 (MSX turbo R) E4H Registradores E5H Dados E6H Contador de 16 bits (LSB) e reset do contador E7H Contador de 16 bits (MSB)
E8H~EFH	MSX-MIDI E8H Transmissão / recepção de dados E9H Porta de controle EAH Latch dos sinais (escrita somente) EBH Espelho de EAH ECH Contador 0 EDH Contador 1 EEH Contador 2 EFH Controle dos contadores (escrita somente)

F0H~F2H	Sem uso conhecido
F3H	Modo screen atual (Só MSX2+) bit 0 = M3 bit 1 = M4 bit 2 = M5 bit 3 = M2 bit 4 = M1 bit 5 – TP bit 6 = YUV bit 7 = YAE
F4H	Estado do RESET para o MSX2+ e MSX turbo R bit 5 – Flag para indicar que o sistema já está inicializado bit 7 – 0 = Hard reset 1 = soft reset Obs.: em alguns MSX2+, deve-se inverter o dado lido para obter o valor correto.
F5H	Controle do sistema (setando o bit em 1 habilita): b0 – Kanji-ROM                       b4 – MSX-Interface b1 – Reservado Kanji               b5 – Serial RS232C b2 – MSX-Audio                       b6 – Caneta ótica b3 – Superimpose                       b7 – IC do relógio
F6H	Barramento I/O de cores (Color Bus)
F7H	Controle AV (setando o bit em 1 habilita): b0 – Audio direito e esquerdo simultâneos b1 – Audio L (esquerdo) somente b2 – Seleciona entrada de vídeo (RGB21) b3 – Flag indicativa se há entrada de vídeo ou não b4 – Controle AV (RGB21) b5 – Controle Ym (RGB21) b6 – Inverso do bit 4 do reg. #9 do VDP b7 – Inverso do bit 5 do reg. #9 do VDP
F8H	Acesso ao registrador de controle PAL A/V
F8H~FBH	Acesso aos 8 bits MSB do registrador de 16 bits do cartucho Playsoniq (são as mesmas portas usadas em alguns MSX que estão desabilitadas por padrão)
FCH~FFH	Memória Mapeada FCH (R/W) Página física 0 (0000H~3FFFH) FDH (R/W) Página física 1 (4000H~7FFFH) FEH (R/W) Página física 2 (8000H~BFFFH) FFH (R/W) Página física 3 (C000H~FFFFH)

## 3 – MSX-BASIC

### 3.1 – FORMATO

NOME DA INSTRUÇÃO (tipo da instrução, versão do BASIC)

Formato: Formatos válidos para a instrução.

Função: Forma de operação da instrução.

Há cinco tipos de instruções, a saber: declarações, comandos, funções, variáveis de sistema e operadores lógicos.

A versão do BASIC assinala a versão para a qual a instrução está implementada. Valores separados por “-” indicam que há diferenças de sintaxe ou comportamento para versões diferentes.

1~4	Versão do MSX-BASIC	M	MSX-MUSIC BASIC
K	Necessário Kanji-ROM	D	Disk-BASIC 1.0
D2	Disk-BASIC 2.0		

#### 3.1.1 – Abreviações de Instruções

REM	'
PRINT	?
CALL	_

#### 3.1.2 – Códigos de Operação Lógica

PSET	TPSET	Usa a cor especificada (padrão)
PRESET	TPRESET	Faz “NOT (cor especificada)”
XOR	TXOR	Faz “(cor destino) XOR (cor especificada)”
OR	TOR	Faz “(cor destino) OR (cor especificada)”
AND	TAND	Faz “(cor destino) AND (cor especificada)”

Obs.: quando a operação vier precedida por “T”, nenhuma operação será feita quando a cor for transparente.

### 3.1.3 – Notações de Código

&B	Precede uma constante na forma binária
&O	Precede uma constante na forma octal
&H	Precede uma constante na forma hexadecimal
%	Assinala variável como inteira
!	Assinala variável como precisão simples
#	Assinala variável como precisão dupla
\$	Assinala variável como alfanumérica
-	Operador matemático para subtração
+	Operador matemático para adição
/	Operador matemático para divisão
*	Operador matemático para multiplicação
^	Operador matemático para potenciação
=	Denota igualdade e atribui valores
<>	Denota diferença

### 3.1.4 – Notações de Formato

<exprA>	variável, constante ou expressão string ou numérica.
<exprN>	variável, constante ou expressão numérica.
<expr\$>	variável, constante ou expressão string.
<n>	é um número definido. Quando entre parênteses pode ser uma expressão ou variável numérica.
[ ]	delimita parâmetro opcional.
	significa que apenas um dos itens pode ser utilizado.
{ }	delimita opção.
X	variável qualquer.
X %	variável inteira qualquer.
X !	variável de precisão simples qualquer.
X #	variável de precisão dupla qualquer.
X \$	variável alfanumérica qualquer.

Caracteres entre parênteses após múltiplos formatos para uma instrução indicam a versão do BASIC na qual aquele formato da instrução está disponível.

## 3.2 – DESCRIÇÃO DOS COMANDOS

**ABS** (função, 1)

Formato:  $X = \text{ABS}(<\text{exprN}>)$

Função: Retorna em X o valor absoluto (módulo) de  $<\text{exprN}>$ .

**AND** (operador lógico, 1)

Formato:  $<\text{exprA1}> \text{ AND } <\text{exprA2}>$

Função: Efetua operação lógica AND entre  $<\text{exprA1}>$  e  $<\text{exprA2}>$ .

$$0 \text{ and } 0 \rightarrow 0 \quad 1 \text{ and } 0 \rightarrow 0$$

$$0 \text{ and } 1 \rightarrow 0 \quad 1 \text{ and } 1 \rightarrow 1$$

**ASC** (função, 1)

Formato:  $X = \text{ASC}(<\text{expr\$}>)$

Função: Retorna em X o código ASCII do primeiro caractere de  $<\text{expr\$}>$ .

**ATN** (função, 1)

Formato:  $X = \text{ATN}(<\text{exprN}>)$

Função: Retorna em X o valor do arcotangente de  $\text{exprN}$  ( $\text{exprN}$  deve ser expresso em radianos).

**AUTO** (comando, 1)

Formato:  $\text{AUTO} [\text{numlinha}, [\text{incremento}]]$

Função: Gera automaticamente números de linha, iniciando com  $[\text{numlinha}]$  e incrementado com o valor de  $[\text{incremento}]$ .

**BASE** (variável de sistema, 1-2-3)

Formato:  $X = \text{BASE}(<n>)$

$\text{BASE}(<n>) = <\text{exprN}>$

Função: Retorna em X ou define os endereços de início das tabelas na VRAM para cada modo de tela.  $<n>$  é um número inteiro que segue a seguinte tabela:

	MODOS DE TELA												Tabela de
	0	1	2	3	4	5	6	7	8	10	11	12	
VALOR BASE	0	5	10	15	20	25	30	35	40	50	55	60	nomes dos padrões
	-	6	11	16	21	26	31	36	41	51	56	61	cores
	2	7	12	17	22	27	32	37	42	52	57	62	geradora de padrões
	-	8	13	18	23	28	33	38	43	53	58	63	atributos dos sprites
	-	9	14	19	24	29	34	39	44	54	59	64	geradora dos sprites

**BEEP** (declaração, 1)

Formato: BEEP

Função: Gera um beep.

**BIN\$** (função, 1)Formato: X\$ = BIN\$(*<exprN>*)Função: Converte o valor de *<exprN>* em uma string de códigos binários e retorna o valor obtido em X\$.**BLOAD** (comando, 1-D)

Formato: BLOAD “&lt;nome do arquivo&gt;”[,R[,&lt;offset&gt;]]

BLOAD “&lt;nome do arquivo&gt;”[,{R | S}][,&lt;offset&gt;]] (D)

Função: Carrega um bloco binário na RAM ou, se especificado [S], na VRAM (somente Disk Basic). Se especificado [,R], executa programa em código de máquina.

**BSAVE** (comando, 1-D)

Formato: BSAVE “&lt;nome arquivo&gt;”,&lt;endini&gt;,&lt;endfim&gt;[,&lt;endexec&gt;]

BSAVE “&lt;nome arquivo&gt;”,&lt;endini&gt;,&lt;endfim&gt;

[,&lt;endexec&gt;,[S]]

Função: Salva em disco ou fita um bloco binário. Se especificado [S] salva um bloco da VRAM (opção disponível somente com Disk Basic).

**CALL** (declaração, 1-2-3-4-D-M-Nextor)

Formato: CALL &lt;comando estendido&gt; [(&lt;argumento&gt;[,&lt;argumento&gt;...])] ]

Função: Executa comandos estendidos através de cartuchos de ROM ou rotinas carregadas na RAM. Veja a seção “DESCRIÇÃO DE COMANDOS ESTENDIDOS”.

**CDBL** (função, 1)Formato: X# = CDBL(*<exprN>*)Função: Converte o valor de *<exprN>* em um valor de dupla precisão e retorna o valor obtido na variável X#.**CHR\$** (função, 1)Formato: X\$ = CHR\$(*<exprN>*)Função: Retorna em X\$ o caractere cujo código ASCII é expressado em *<exprN>*.

**CINT** (função, 1)

Formato: X% = CINT(<exprN>)

Função: Converte o valor de <exprN> em um valor inteiro e retorna o valor obtido na variável X%.

**CIRCLE** (declaração, 1-2)

Formato: CIRCLE { (X,Y) | STEP(X,Y)},<raio>[,<cor>[,<ângulo inicial> [, <ângulo final>[,<proporção>]]]]

Função: Desenha uma circunferência com ponto central em (X,Y). Se for especificado STEP, as coordenadas serão calculadas a partir da atual. <ângulo inicial> e <ângulo final> devem ser especificados em radianos. <proporção> é a relação para elipse, sendo <1> circunferência perfeita.

**CLEAR** (declaração, 1)

Formato: CLEAR [<tamanho área string>[,limite superior memória>]]

Função: Inicializa as variáveis do BASIC e seta o tamanho da área para string e o limite superior de memória usado pelo BASIC.

**CLOAD** (comando, 1)

Formato: CLOAD [<nome do arquivo>]

Função: Carrega um programa BASIC de fita cassete.

**CLOAD?** (comando, 1)

Formato: CLOAD? [<nome do arquivo>]

Função: Compara um programa BASIC na fita cassete com o da memória.

**CLOSE** (comando, 1-D)

Formato: CLOSE [[#]<nº arquivo>,[#]<nº arquivo>...]]

Função: Fecha os arquivos especificados. Se não for especificado nenhum arquivo, fecha todos os arquivos abertos.

**CLS** (declaração, 1)

Formato: CLS

Função: Limpa a tela.

**CMD** (comando, 1)

Formato: Sem formato definido.

Função: Reservado para implementação de novos comandos.

**COLOR** (declaração, 1-2)

Formato: COLOR [<cor frente>[,<cor fundo>[,<cor borda>]]] (1-2)

Função: Especifica as cores da tela.

**COLOR =** (declaração, 2)

Formato: COLOR = (<nº paleta>,<nível vermelho>,<nível verde>,<nível azul>)

Função: Especifica as cores da paleta. O nível pode variar de 0 a 7 para cada cor.

**COLOR – NEW** (declaração, 2)

Formato: COLOR [= NEW]

Função: Inicializa a paleta de cores para o padrão no reset.

**COLOR = RESTORE** (declaração, 2)

Formato: COLOR = RESTORE

Função: Copia o conteúdo da paleta de cores armazenada na VRAM para os registradores de paleta do VDP.

**COLOR SPRITE** (declaração, 1-2)

Formato: COLOR SPRITE (<nº do plano do sprite>)=<cor>

Função: Especifica a cor dos sprites.

**COLOR SPRITES\$** (declaração, 2)

Formato: COLOR SPRITES\$ (<nº do plano do sprite>)=<expr\$>

Função: Especifica a cor de cada linha dos sprites.

<expr\$> = CHR\$(cor 1ª linha) + CHR\$(cor 2ª linha) ...

**CONT** (comando, 1)

Formato: CONT

Função: Continua a execução de um programa que foi interrompido.

**COPY** (declaração, 1-2-D)

Formato: COPY <nome arquivo1> [TO <nome arquivo 2>] (1-D)

Função: Copia o conteúdo de <nome do arquivo 1> para <nome do arquivo 2>.

Formato: COPY (X1,X2)-(Y1,Y2) [,<página fonte>] TO (X3,Y3)  
[,<página destino>[,<operação lógica>]] (2)

Função: Copia uma área retangular da tela para outra.

Formato: COPY (X1,X2)-(Y1,Y2) [,<página fonte>] TO  
                   {<variável matriz | <nome do arquivo>} (2-D)

Função: Copia o conteúdo de uma área retangular da tela para  
                   uma variável matriz ou para um arquivo em disco.

Formato: COPY {<variável matriz> | <nome do arquivo>}  
                   [,<direção>] TO [,<direção>] TO (X3,Y3)  
                   [,<página destino> [,<operação lógica>]] (2-D)

Função: Copia o conteúdo de uma variável matriz ou de um arquivo  
                   em disco para uma área retangular na tela.

Formato: COPY <nome do arquivo> TO <variável matriz> (2-D)

Função: Copia o conteúdo de um arquivo para uma variável matriz.

Formato: COPY <variável matriz> TO <nome do arquivo> (2-D)

Função: Copia o conteúdo de uma variável matriz para um arquivo.

### **COPY SCREEN** (declaração, 2, opcional)

Formato: COPY SCREEN [<modo>]

Função: Escreve os dados do Color Bus na VRAM.

<modo> pode ser:

- 0 – digitaliza na página de vídeo atual.
- 1 – digitaliza duas páginas, a primeira na página anterior  
                   à ativa e a segunda na página ativa (entrelaçado)

Obs.: requer um digitalizador ou superimposer.

### **COS** (função, 1)

Formato: X = COS (<exprN>)

Função: Retorna em X o valor do cosseno de <exprN>  
                   (exprN deve ser expresso em radianos).

### **CSAVE** (comando, 1)

Formato: CSAVE <nome do arquivo> [,<baud rate>]

Função: Salva um programa BASIC na fita cassete.

### **CSNG** (função, 1)

Formato: X! = CSNG(<exprN>)

Função: Converte o valor de <exprN> em um valor de precisão  
                   simples e retorna o valor obtido em X!.

### **CSRLIN** (variável de sistema, 1)

Formato: X = CSRLIN

Função: Contém a posição vertical do cursor.

## CVD (função, D)

Formato: X# = CVD (<string de 8 bytes>)

**Função:** Converte a string em um valor de dupla precisão e armazena o valor obtido em X#.

CVI (função, D)

Formato: X% = CVI (<string de 2 bytes>)

**Função:** Converte a string em um valor inteiro e armazena o valor obtido em X%.

## **CVS** (função, D)

Formato: X! = CVS (<string de 4 bytes>)

**Função:** Converte a string em um valor de precisão simples e armazena o valor obtido em X!.

**DATA** (declaração, 1)

Formato: DATA <constante>[,<constante> ...]

Função: Armazena uma lista de dados para o comando READ.

## **DEF FN** (declaração, 1)

Formato: DEF FN <nome> [(<argumento>[,<argumento>...])] =  
<expressão definidora de função de usuário>

Funcão: Define uma função do usuário.

**DEFDBL** (declaração, 1)

Formato: DEFDBL <faixa de caracteres>[,<faixa de caracteres>...]

**Função:** Declara as variáveis especificadas como dupla precisão.

**DEFINT** (declaração, 1)

Formato: DEFINT <faixa de caracteres>[,<faixa de caracteres>...]

Função: Declara as variáveis especificadas como inteiros.

## **DEFSNG** (declaração, 1)

Formato: DEFSNG <faixa de caracteres>[,<faixa de caracteres>...]

**Função:** Declara as variáveis especificadas como precisão simples.

**DEFSTR** (declaração, 1)

Formato: DEFSTR <faixa de caracteres>[,<faixa de caracteres>...]

Função: Declara as variáveis especificadas como strings.

**DEFUSR** (declaração, 1)

Formato: DEFUSR[<número>] = <endereço>

Função: Define um endereço inicial para execução de programa assembly a ser chamado pela função USR. <número> pode variar de 0 a 9.

**DELETE** (comando, 1)

Formato: DELETE {<linha inicial> – <linha final>} | <linha> | – <linha final>}

Função: Apaga as linhas especificadas do texto BASIC.

**DIM** (declaração, 1)

Formato: DIM <variável> (<índice máximo>[,<índice máximo>...])

Função: Define uma variável matriz e aloca espaço na memória.

**DRAW** (macro declaração, 1)

Formato: DRAW <expr\$>

Função: Desenha uma linha de acordo com <expr\$>. Os comandos válidos para <expr\$> são os seguintes:

Un	para cima	Dn	para baixo
Ln	para esquerda	Rn	para direita
En	cima e direita	Fn	baixo e direita
Gn	baixo e esq.	Hn	cima e esq.
B	mov. sem desenho	N	volta origem
Mx,y	vai p/ X,Y	An	gira n*90 graus
Sn -	escala n/4	Cn	cor n

Xsérie – Executa macro em série.

Ex. A\$="C15U10" → DRAW "XA\$"

=<variável> – coloca um parâmetro como inteiro após o comando. Ex. A\$=A\$="C15U10", S=50,  
→ DRAW "XA\$;R=S;D=S"

**DSKF** (função, D)

Formato: X = DSKF(<nº drive>)

Função: Retorna o espaço livre no drive especificado em clusters. Se o Nextor estiver instalado, retornará o espaço em Kbytes.

**EOF** (função, 1-D)

Formato: X – EOF(<nº do arquivo>)

Função: Retorna -1 caso o fim de arquivo seja detectado.

**ERASE** (declaração, 1)

Formato: ERASE <variável matriz>[,<variável matriz>...]

Função: Deleta as variáveis matriz especificadas.

**EQV** (operador lógico, 1)

Formato: <exprA1> EQV <exprA2>

Função: Efetua operação lógica EQV entre <exprA1> e <exprA2>. O resultado será 1 se os dois bits forem iguais e zero se forem diferentes.

0 eqv 0 → 1	1 eqv 0 → 0
-------------	-------------

0 eqv 1 → 0	1 eqv 1 → 1
-------------	-------------

**ERL** (variável de sistema, 1)

Formato: X – ERL

Função: Contém o número de linha onde o último erro ocorreu.

**ERR** (variável de sistema, 1)

Formato: X – ERR

Função: Contém o código de erro do último erro ocorrido.

**ERROR** (declaração, 1)

Formato: ERROR <código de erro>

Função: Coloca o programa na condição de erro.

**EXP** (função, 1)

Formato: X – EXP (<exprN>)

Função: Retorna em X o valor da potenciação natural de <exprN>.

**FIELD** (declaração, D)

Formato: FIELD [#]<nº arq>,<tamanho do campo> AS <nome var. string>[,<tamanho do campo> AS <nome var. string>...]

Função: Atribui a <var. string> para acesso aleatório ao disco.

**FILES** (comando, D)

Formato: FILES [<nome do arquivo>]

Função: Apresenta os nomes de arquivos do disco de acordo com <nome do arquivo>. Se <nome do arquivo> for omitido, apresenta os nomes de todos os arquivos presentes no disco.

**FIX** (função, 1)

Formato: X – FIX(&lt;exprN&gt;)

Função: Retorna em X a parte inteira de &lt;exprN&gt;, sem arredondar.

**FOR** (declaração, 1)Formato: FOR <nome variável> = <valor inicial> TO <valor final>  
[STEP <incremento>]

Função: Repete a execução do trecho entre o FOR e o NEXT.

**FRE** (função, 1)

Formato: FRE (0 | "")

Função: Retorna o tamanho da memória restante para o texto  
BASIC (0) ou para as variáveis string ("").**GET** (declaração, D)

Formato: GET [#]&lt;nº arq&gt;[,&lt;nº registro&gt;]

Função: Lê um registro de um arquivo de acesso aleatório.

**GET DATE** (declaração, 2)

Formato: GET DATE &lt;variável string&gt; [,A]

Função: Retorna uma string com a data atual na &lt;variável string&gt;. Se „A” for especificado, retorna a data do alarme.

**GET TIME** (declaração, 2)

Formato: GET TIME &lt;variável string&gt; [,A]

Função: Retorna um string com a hora atual na &lt;variável string&gt;. Se „A” for especificado, retorna a hora do alarme.

**GOSUB** (declaração, 1)

Formato: GOSUB &lt;nº linha&gt;

Função: Chama um sub-rotina que inicia na linha &lt;nº linha&gt;.

**GOTO** (declaração, 1)

Formato: GOTO &lt;nº linha&gt;

Função: Salta para a linha &lt;nº linha&gt;.

**HEX\$** (função, 1)

Formato: X\$ = HEX\$(&lt;exprN&gt;)

Função: Converte o valor de <exprN> em uma string hexadecimal  
e retorna o valor obtido em X\$.

**IF** (declaração, 1)

Formato: IF <condição> THEN {<comando> | <nº linha>} [ELSE {<comando> | <nº linha>}]  
 IF <condição> GOTO <nº linha> [ELSE <nº linha>]

Função: Executa comandos de acordo com a <condição>.

**IMP** (operador lógico, 1)

Formato: <exprA1> IMP <exprA2>

Função: Efetua operação lógica IMP entre <exprA1> e <exprA2>. O resultado será 0 quando o primeiro bit for verdadeiro e implica que o segundo é falso. Caso contrário será 1.

0 imp 0 → 1	1 imp 0 → 0
0 imp 1 → 1	1 imp 1 → 1

**INKEY\$** (função, 1)

Formato: X\$ = INKEY\$

Função: Retorna em X\$ um caractere quando a tecla está sendo pressionada; caso contrário, retorna uma string nula.

**INP** (função, 1)

Formato: X = INP(<número da porta>)

Função: Lê uma porta de I/O do Z80 e retorna seu valor em X.

**INPUT** (declaração, 1)

Formato: INPUT ["<prompt>";] <nome variável>[,<nome variável>...]

Função: Lê uma entrada de dados pelo teclado e armazena o(s) valor(es) obtido(s) na(s) variável(is) respectiva(s).

**INPUT#** (declaração, 1)

Formato: INPUT #<nº arq>, <nome variável>[,<nome variável>...]

Função: Lê dados do arquivo especificado e armazena o(s) valor(es) obtido(s) na(s) variável(is) respectiva(s).

**INPUT\$** (função, 1)

Formato: X\$ = INPUT\$ (<nº caracteres>,[#]<nº arq>)

Função: Lê o número especificado de caracteres do teclado ou de um arquivo e armazena o valor obtido em X\$.

**INSTR** (função, 1)

Formato: X = INSTR ([<exprN>,<expr\$1>,<expr\$2>])

**Função:** Procura a ocorrência de <expr\$2> em <expr\$1> a partir da posição <exprN> e retorna o valor obtido em X. Se <expr\$1> não for localizado, retorna 0.

#### **INT** (função, 1)

**Formato:** X = INT (<exprN>)

**Função:** Retorna em X a parte inteira de <exprN>, arredondando.

#### **INTERVAL** (declaração, 1)

**Formato:** INTERVAL {ON | OFF | STOP}

**Função:** Ativa, desativa ou suspende interrupção por intervalo de tempo.

#### **IPL** (comando, 1)

**Formato:** Sem formato definido.

**Função:** Reservado para implementação de novos comandos.

#### **KEY** (comando/declaração, 1)

**Formato:** KEY <número de tecla>,<expr\$>

**Função:** Redefine o conteúdo da tecla de função especificada.

**Formato:** KEY (<número de tecla>) {ON | OFF | STOP}

**Função:** Ativa, desativa ou suspende interrupção de tecla de função.

**Formato:** KEY {ON | OFF}

**Função:** Liga ou desliga a apresentação do conteúdo das teclas de função na última linha da tela.

#### **KEY LIST** (comando, 1)

**Formato:** KEY LIST

**Função:** Lista o conteúdo das teclas de função.

#### **KILL** (comando, D)

**Formato:** KILL <expr\$>

**Função:** Apaga arquivos no disco. <expr\$> deve conter um nome de arquivo válido.

#### **LEFT\$** (função, 1)

**Formato:** X\$ = LEFT\$ (<expr\$>,<exprN>)

**Função:** Retorna em X\$ os <exprN> caracteres esquerdos de <expr\$>.

**LEN** (função, 1)

Formato: X = LEN(<expr\$>)

Função: Retorna em X o número de caracteres de <expr\$>.

**LET** (declaração, 1)

Formato: [LET] <nome variável> = <exprA>

Função: Armazena na variável o valor de <exprA>.

**LFILES** (comando, 1)

Formato: LFILES [<nome do arquivo>]

Função: Lista os nomes dos arquivos do disco na impressora de acordo com <nome do arquivo>. Se <nome do arquivo> for omitido, lista os nomes de todos os arquivos presentes no disco.

**LINE** (declaração, 1-2)

Formato: LINE [{(X1,Y1) | STEP(X1,Y1)}] – {(X2,Y2) | STEP(X2,Y2)}  
[,<cor>[{B | BF} [, <operação lógica>]]]

Função: Desenha uma linha, um retângulo vazio (,B) ou um retângulo pintado (,BF). O subcomando STEP, quando especificado, define o deslocamento.

**LINE INPUT** (declaração, 1)

Formato: LINE INPUT [“<prompt>”;]<variável string>

Função: Lê uma sequência de caracteres do teclado e armazena o valor lido na <variável string>.

**LINE INPUT #** (declaração, 1-D)

Formato: LINE INPUT #<nº arq>,<variável string>

Função: Lê uma sequência de caracteres de um arquivo e armazena o valor lido na <variável string>.

**LIST** (comando, 1)

Formato: LIST [[<linha inicial>] – [<linha final>]]

Função: Lista na tela o programa BASIC que está na memória.

**LLIST** (comando, 1)

Formato: LLIST [[<linha inicial>] – [<linha final>]]

Função: Lista na impressora o programa BASIC que está na memória.

**LOAD** (comando, 1-D)

Formato: LOAD “<nome do arquivo>” [,R]

Função: Carrega um programa na memória. O parâmetro [,R] faz com o programa seja executado após o carregamento.

**LOC** (função, D)

Formato: X = LOC (<nº arq>)

Função: Retorna em X o número do último registro acessado do arquivo.

**LOCATE** (declaração, 1-2)

Formato: LOCATE [<coord. X>[,<coord. Y>[,<tipo cursor>]]]

Função: Posiciona o cursor nas telas de texto. Se <tipo cursor> for 0 (valor padrão) o cursor não será exibido quando o computador estiver ocupado; se for qualquer outro valor, o cursor sempre será exibido.

**LOF** (função, D)

Formato: X = LOF (<nº arq>)

Função: Retorna em X o tamanho do arquivo especificado.

**LOG** (função, 1)

Formato: X = LOG (<exprN>)

Função: Retorna em X o logaritmo natural de <exprN>.

**LPOS** (variável de sistema, 1)

Formato: X = LPOS

Função: Armazena a localização horizontal da cabeça da impressora.

**LPRINT** (declaração, 1)

Formato: LPRINT [<exprA>[{: | ,}<exprA>...]]

Função: Envia para a impressora os caracteres correspondentes às expressões <exprA>. “ ; ” imprime o próximo caractere logo em seguida, “ , ” imprime o caractere na próxima parada de tabulação.

**LPRINT USING** (declaração, 1)

Formato: LPRINT USING <“formato”>;<exprA>[{: | ,}<exprA>...]

LPRINT USING <“formato expr\$”>

**Função:** Envia para a impressora os caracteres correspondentes às expressões <exprN> ou <expr\$>, formatando. “;” imprime o próximo caractere logo em seguida, “,” imprime o caractere na próxima parada de tabulação. Os caracteres usados para formatar a saída são os seguintes:

→ Formatação numérica:

- # Espaço para um dígito
- . Inclui ponto decimal
- + Indica + ou -; usado antes ou depois do número
- Indica -; usado depois do número
- \$\$ Coloca \$ à esquerda do número
- \*\* Substitui espaços à esquerda por asteriscos
- \*\*\$ Coloca um \$ à esquerda precedido por asteriscos
- ^^^ Apresenta o número em Notação científica

→ Formatação alfanumérica:

- \ \ Espaço para caracteres
- ! Espaço para um caractere
- & Espaçamento variável
- \_ Próximo caractere é impresso normalmente
- outro Imprime caractere

### **LSET** (declaração, D)

Formato: LSET <variável string> = <expr\$>

Função: Armazena o conteúdo de <expr\$> à esquerda na variável string definida pela declaração FIELD.

### **MAXFILES** (declaração, 1-D)

Formato: MAXFILES = <número de arquivos>

Função: Define o número máximo de arquivos que podem ser abertos ao mesmo tempo.

### **MERGE** (comando, 1-D)

Formato: MERGE <nome do arquivo>

Função: Intercala o programa na memória com um programa salvo no formato ASCII em disco ou fita.

### **MID\$** (função/declaração, 1)

Formato: X\$ = MID\$ (<expr\$>,<exprN1>[,exprN2])

Função: Retorna, em X\$, <exprN2> caracteres a partir do caractere <exprN1> de <expr\$>.

**Formato:** MID\$ (<variável string>,<exprN1>[,<exprN2>]) = <expr\$>

**Função:** Define <expr\$> usando <exprN2> caracteres a partir da posição <exprN1> da <variável string>.

### **MKD\$ (função, D)**

**Formato:** X\$ = MKD\$ (<valor de dupla precisão>)

**Função:** Converte um valor de dupla precisão em uma string de 8 bytes e a armazena em X\$.

### **MKI\$ (função, D)**

**Formato:** X\$ = MKI\$ (<valor inteiro>)

**Função:** Converte um valor inteiro em uma string de 2 bytes e a armazena em X\$.

### **MKS\$ (função, D)**

**Formato:** X\$ = MKS\$ (<valor de precisão simples>)

**Função:** Converte um valor de precisão simples em uma string de 4 bytes e a armazena em X\$.

### **MOTOR (comando, 1)**

**Formato:** MOTOR [{ON | OFF}]

**Função:** Liga ou desliga o motor do cassete.

### **NAME (comando, D)**

**Formato:** NAME <nome do arquivo1> AS <nome do arquivo2>

**Função:** Renomeia o arquivo <nome do arquivo 1> com <nome do arquivo 2>.

### **NEW (comando, 1)**

**Formato:** NEW

**Função:** Deleta o programa da memória e limpa as variáveis.

### **NEXT (declaração, 1)**

**Formato:** NEXT [<nome da variável>[,<nome da variável>...]]

**Função:** Indica o fim do laço FOR.

### **NOT (operador lógico, 1)**

**Formato:** NOT (<exprA>)

**Função:** Efetua a negação de <exprA>.

not 0 → 1

not 1 → 0

**OCT\$** (função, 1)

Formato: X\$ = OCT\$ (<exprN>)

Função: Converte o valor de <exprN> em uma string octal e retorna o valor obtido em X\$.

**ON ERROR GOTO** (declaração, 1)

Formato: ON ERROR GOTO <número de linha>

Função: Define a linha inicial da rotina para manipulação de erro.

**ON GOSUB** (declaração, 1)

Formato: ON <exprN> GOSUB <nº linha>[,<nº linha>...]

Função: Executa a sub-rotina em <nº linha> de acordo com <exprN>.

**ON GOTO** (declaração, 1)

Formato: ON <exprN> GOTO <nº linha>[,<nº linha>...]

Função: Salta para a linha <nº linha> de acordo com <exprN>.

**ON INTERVAL GOSUB** (declaração, 1)

Formato: ON INTERVAL = <tempo> GOSUB <nº linha>

Função: Define o intervalo e o número da linha para interrupção de tempo. <tempo> é definido em unidades de 1/60 segundos em uma máquina MSX padrão.

**ON KEY GOSUB** (declaração, 1)

Formato: ON KEY GOSUB <nº linha>[,<nº linha>...]

Função: Define os números de linha para interrupção de teclas de função.

**ON SPRITE GOSUB** (declaração, 1)

Formato: ON SPRITE GOSUB <nº linha>

Função: Define o número de linha para interrupção por colisão de sprites.

**ON STOP GOSUB** (declaração, 1)

Formato: ON STOP GOSUB <nº linha>

Função: Define o número de linha para interrupção pelo pressionamento das teclas CTRL+STOP.

**ON STRIG GOSUB** (declaração, 1)

Formato: ON STRIG GOSUB <nº linha>[,<nº linha>...]

Função: Define os números de linha para interrupção pelo pressionamento dos botões de disparo do joystick.

**OPEN** (declaração, 1-D)

Formato: OPEN <nome do arquivo> [FOR {INPUT | OUTPUT}] AS  
#<nº arq> [LEN=<tamanho do registro>]

Função: Abrir um arquivo em fita ou disco.

**OR** (operador lógico, 1)

Formato: <exprA1> OR <exprA2>

Função: Efetua operação lógica OR entre <exprA1> e <exprA2>.

0 or 0 → 0                            1 or 0 → 0

0 or 1 → 0                            1 or 1 → 1

**OUT** (declaração, 1)

Formato: OUT <nº da porta>,<exprN>

Função: Escreve o valor de <exprN> em uma porta de I/O do Z80.

**PAD** (função, 1-2)

Formato: X = PAD (<exprN>)

Função: Examina o estado do mouse, trackball, caneta ótica ou tablete digitalizador e retorna o valor obtido em X.

<exprN> pode ser:

0 – checa touch pad na porta 1 (255 se conectado)

1 – retorna a coordenada X (horizontal).

2 – retorna a coordenada Y (vertical).

3 – retorna o estado de tecla (255 se pressionada).

4 – checa touch pad na porta 2 (255 se conectado).

5 – retorna a coordenada X (horizontal).

6 – retorna a coordenada Y (vertical).

7 – retorna o estado de tecla (255 se pressionada).

8 – checa caneta ótica (255 se conectada ou tocando a tela).

9 – retorna a coordenada X (horizontal).

10 – retorna a coordenada Y (vertical).

11 – retorna o estado de chave (255 se pressionada).

- 12 – checa mouse na porta 1 (255 se conectado).
- 13 – retorna offset da coordenada X (horizontal).
- 14 – retorna offset da coordenada Y (vertical).
- 15 – sempre 0.
- 16 – checa mouse na porta 2 (255 se conectado).
- 17 – retorna offset da coordenada X (horizontal).
- 18 – retorna offset da coordenada Y (vertical).
- 19 – sempre 0.
- 20 – checa 2<sup>a</sup> caneta ótica (255 se conectada ou tocando a tela). \*obs
- 21 – retorna a coordenada X (horizontal). \*obs
- 22 – retorna a coordenada Y (vertical). \*obs
- 23 – retorna o estado de tecla (255 se pressionada). \*obs

**Obs.:** Os valores 20 a 23 requerem o uso da instrução CALL ADJUST antes. Disponível apenas para MSX2 fabricados pela Daewoo.

#### **PAINT** (declaração, 1-2)

Formato: PAINT {(X,Y) | STEP(X,Y)} [,<cor>[,<cor da borda>]]

Função: Preenche a área delimitada por uma linha com a cor <cor da borda> com a cor <cor>.

#### **PDL** (função, 1)

Formato: X = PDL (<nº paddle>)

Função: Retorna em X o estado do paddle especificado. O número do paddle pode ser:

1, 3, 5, 7, 9, 11 – Paddles conectados na porta 1.

2, 4, 6, 8, 10, 12 – Paddles conectados na porta 2.

#### **PEEK** (função, 1)

Formato: X = PEEK (<endereço>)

Função: Retorna em X o valor do byte contido em <endereço>.

#### **PLAY** (macro declaração, 1)

Formato: PLAY <expr\$1>[,<expr\$2>[,<expr\$3>]]

Função: Toca as Notas especificadas por <expr\$> no PSG. Os comandos válidos para <expr\$> são os seguintes:



{ } n Define em n as Notas entre { }. (n=1~8, padrão é Ln)

@n Troca o instrumento (1~64)

*Para as peças de bateria, os comandos são os seguintes:*

B Bass Drum

S Snare Drum

W Tom tom

C Cymbals

H Hi hat

@Vn Seta mudança detalhada de volume (0~127)

@Nn Mantém a duração definida por n (1~64, padrão Ln)

n A enésima Nota é pausada (1~64)

! Acentua a Nota precedente

@An Define o volume para as vozes acentuadas (0~15)

**Obs.:** Tn, Vn, @Vn, Rn, X, =x; e . são idênticos aos outros instrumentos.

### **POINT** (declaração, 1)

Formato: X = POINT (X,Y)

Função: Retorna em X o código de cor do ponto (X,Y) da tela gráfica.

### **POKE** (declaração, 1)

Formato: POKE <endereço>,<dado>

Função: Escreve no <endereço> de memória um byte de dados. <dado> deve ser um valor numérico entre 0 e 255.

### **POS** (variável de sistema, 1)

Formato: X = POS(0)

Função: Armazena a posição horizontal do cursor no modo texto.

### **PRESET** (declaração, 1-2)

Formato: PRESET {(X,Y) | STEP(X,Y)} [,<cor> [,<operação lógica>]]

Função: Desliga o ponto especificado por (X,Y) na tela gráfica. “STEP”, se especificado, define o deslocamento.

### **PRINT** (declaração, 1)

Formato: PRINT [<exprA>[; | ,]<exprA>...]]

Função: Apresenta na tela os caracteres correspondentes às expressões <exprA>. “;” não gera avanço de linha e “,” avança para a posição de tabulação seguinte.

**PRINT#** (declaração, 1-D)

Formato: PRINT#<nº arq>,[<exprA>[; | ,]<exprA>...]]

Função: Escreve o valor de <exprA> no arquivo especificado. “;” não gera avanço de linha e “,” avança para a posição de tabulação seguinte.

**PRINT USING** (declaração, 1)

Formato: PRINT USING <“formato”>;<exprN>[{: | ,}<exprN>...]

PRINT USING <“formato expr\$”>

Função: Apresenta na tela os caracteres correspondentes às expressões <exprN> ou <expr\$>, formatando. “;” não gera avanço de linha e “,” avança para a posição de tabulação seguinte. Os caracteres de formatação estão descritos abaixo:

→ Formatação numérica:

# Espaço para um dígito

. Inclui ponto decimal

+ Indica + ou -; usado antes ou depois do número

- Indica -; usado depois do número

\$\$ Coloca \$ à esquerda do número

\*\* Substitui espaços à esquerda por asteriscos

\*\*\$ Coloca um \$ à esquerda precedido por asteriscos

^^^^^ Apresenta o número em Notação científica

→ Formatação alfanumérica:

\ \ Espaço para caracteres

! Espaço para um caractere

& Espaçamento variável

\_ Próximo caractere será impresso normalmente

outro Imprime caractere

**PRINT# USING** (declaração, 1-D)

Formato: PRINT#<nº arq> USING <“formato”>;<exprA>

[{: | ,}<exprA>...]

Função: Escreve o valor de <exprA> no arquivo especificado, formatando. Os caracteres de formatação são os mesmos de PRINT USING.

**PSET** (declaração, 1)

Formato: PSET {(X,Y) | STEP(X,Y)} [,<cor> [,<operação lógica>]]

Função: Desenha o ponto especificado por (X,Y) na tela gráfica. “STEP”, se especificado, define o deslocamento.

**PUT** (declaração, D)

Formato: PUT [#]<nº arq>[,<nº registro>]

Função: Grava um registro em um arquivo aleatório.

**PUT HAN** (declaração, Daewoo CPC 400/400S)

Formato: PUT HAN [(X,Y)],<código Hangul>[,<cor>[,<operação lógica>[,<modo>]]]]

Função: Apresenta um caractere coreano Hangul na tela.

<modo> define o tamanho do caractere Hangul:

0 – 16x16 pontos

1 – 16x8 pontos (apresenta apenas linhas ímpares)

2 – 16x8 pontos (apresenta apenas linhas pares)

**PUT KANJI** (declaração, 1-2-Kanji)

Formato: PUT KANJI [(X,Y)],<código JIS>[,<cor>[,<operação lógica>[,<modo>]]]]

Função: Apresenta um caractere Kanji na tela. <código JIS> pode variar de &H2120 ~ &H4F53 para JIS1 e de &H5020 ~ &H7424 para JIS2.

<modo> define o tamanho do Kanji:

0 – 16x16 pontos

1 – 16x8 pontos (apresenta apenas linhas ímpares)

2 – 16x8 pontos (apresenta apenas linhas pares)

**PUT SPRITE** (declaração, 1-2)

Formato: PUT SPRITE <plano do sprite>[,{(X,Y) | STEP(X,Y)}[,<cor>[,<nº do padrão>]]]]

Função: Apresenta um sprite na tela. “STEP”, se especificado, define o deslocamento. <plano do sprite> é um número de 0 a 31 e especifica a prioridade de exibição. Números maiores serão exibidos sobre números menores. <nº do padrão> define o padrão a ser apresentado. Pode variar a 0 a 255 para sprites 8x8 e de 0 a 63 para sprites 16x16. Se não for especificado, será igual ao <plano do sprite>.

**READ** (declaração, 1)

Formato: READ <nome da variável>[,<nome da variável>...]

Função: Lê os dados do comando DATA e os armazena nas variáveis.

**REM** (declaração, 1)

Formato: REM &lt;comentários&gt;

Função: Colocar comentários no programa.

**RENUM** (comando, 1)

Formato: RENUM [&lt;nº linha&gt;[,&lt;nº linha antigo&gt;[,&lt;incremento&gt;]]]

Função: Renumera as linhas de programa.

**RESTORE** (declaração, 1)

Formato: RESTORE [&lt;nº de linha&gt;]

Função: Especifica o número de linha DATA inicial a ser lido por READ.

**RESUME** (declaração, 1)

Formato: RESUME { [0] | NEXT | &lt;nº de linha&gt; }

Função: Finaliza rotina de tratamento de erros.

0 – a execução retorna ao mesmo comando onde houve o erro

NEXT – a execução continua no comando seguinte ao de onde houve o erro

&lt;nº de linha&gt; – A execução continuará na linha especificada

**RETURN** (declaração, 1)

Formato: RETURN [&lt;nº de linha&gt;]

Função: Retorna de uma sub-rotina.

**RIGHT\$** (função, 1)

Formato: X\$ = RIGHT\$ (&lt;expr\$&gt;,&lt;exprN&gt;)

Função: Retorna em X\$ os &lt;exprN&gt; caracteres direitos de &lt;expr\$&gt;.

**RND** (função, 1)

Formato: X = RND [&lt;exprN&gt;]

Função: Retorna em X um número aleatório entre 0 e 1. É aconselhável o uso de “-TIME” em &lt;exprN&gt; para obtenção de melhor aleatoriedade.

**RSET** (declaração, D)

Formato: RSET &lt;variável string&gt; = &lt;expr\$&gt;

Função: Armazena o conteúdo de &lt;expr\$&gt; à direita na variável string definida pela declaração FIELD.

**RUN** (comando, 1-D)

Formato: RUN [{<nº linha>} | “nome do arquivo”]

Função: Executa um programa BASIC na memória ou carrega um programa do disco e o executa. Se <nº linha> for especificado, a execução começará nessa linha.

**SAVE** (comando, 1-D)

Formato: SAVE “<nome do arquivo>” [,A]

Função: Salva em disco ou fita o programa da memória. Se “,A” for especificado, salva o programa BASIC na forma ASCII e não na forma atomizada.

**SCREEN** (declaração, 1-2-3)

Formato: SCREEN <modo tela> [,<tamanho sprite> [,<click teclas> ,<taxa cassete>[,<tipo impressora>[,<interlace>]]]]]

Função: Seleciona modo de tela e outros valores.

<modo tela> – 0 a 12, dependendo da versão do MSX.

“Screen 9” só funciona em micros coreanos ou carregados com o Hangul BASIC.

<tamanho sprite> – 0 → sprites 8x8 (padrão)

1 → sprites 8x8 ampliados para 16x16

2 → sprites 16x16

3 → sprites 16x16 ampliados p/ 32x32

<click teclas> – 0 → desliga “click”

1 → liga “click” (padrão)

<taxa cassete> – 1 → escreve a 1200 bauds (padrão)

2 → escreve a 2400 bauds

<tipo impressora> – 0 → impressora MSX (padrão)

1 → impressora não MSX

<interlace> – 0 → normal (padrão)

1 → entrelaçado (padrão Screen 0)

2 → normal (apresentação alternada)

3 → entrelaçado (apresent. alternada)

**SET ADJUST** (declaração, 2)

Formato: SET ADJUST (<coordenada X>,<coordenada Y>)

Função: Muda a localização da tela. X e Y podem variar de -7 a 8.

**SET BEEP** (declaração, 2)

Formato: SET BEEP <timbre>,<volume>

Função: Seleciona o tipo e o volume do beep. <timbre> pode variar de 1 a 4 e <volume> de 0 a 15.

**SET DATE** (declaração, 2)

Formato: SET DATE <expr\$> [,A]

Função: Altera a data do relógio. [,A] altera a data do alarme.  
<expr\$> deve conter uma especificação de data válida.

**SET HAN** (declaração, Hangul-BASIC 2nd version)

Formato: SET HAN [<tamanho>], [<tela>], [<impressora>]

Função: Define como os caracteres Hangul serão exibidos nos modos Screen 0 a 8 e na impressora.

<tamanho> – 0 → caracteres de 8x8 pontos

1 → caracteres de 8x16 pontos

<tela> – 0 → caracteres não agrupados

1 → caracteres agrupados em blocos

<impressora> 0 → caracteres não agrupados

1 → caracteres agrupados em blocos

**SET PAGE** (declaração, 2)

Formato: SET PAGE <página apresentada>,<página ativa>

Função: Seleciona páginas de vídeo. <página apresentada> é a página a ser apresentada na tela e <página ativa> é a página na qual serão executados os comandos.

**SET PASSWORD** (declaração, 2)

Formato: SET PASSWORD <expr\$>

Função: Ativa a senha. <expr\$> deve conter uma senha de no máximo 6 caracteres.

**SET PROMPT** (declaração, 2)

Formato: SET PROMPT <expr\$>

Função: Ativa um novo prompt para o BASIC. <expr\$> deve conter o novo prompt com no máximo 6 caracteres.

**SET SYSTEM** (comando, Daewoo CPC300/400/400S)

Formato: SET SYSTEM (<modo>) [CPC 300]

Função: Define como o computador inicializa.

- <modo> – 0 → inicia o software adicional em ROM
- 1 → inicia o BASIC

Formato: SET SYSTEM [<dummy>], [<tela>], [<impressora>]

[CPC 400/400S]

Função: Define os parâmetros iniciais para o sistema Hangul.

- <dummy> – ação nula para qualquer valor especificado
- <tela> –       0 → caracteres não agrupados
- 1 → caracteres agrupados em blocos
- <impressora> 0 → caracteres não agrupados
- 1 → caracteres agrupados em blocos

**SET SCREEN** (declaração, 2)

Formato: SET SCREEN

Função: Grava na SRAM do relógio os dados definidos na declaração SCREEN.

**SET TIME** (declaração, 2)

Formato: SET TIME <expr\$> [,A]

Função: Altera a hora do relógio. [,A] altera a hora do alarme.  
 <expr\$> deve conter uma especificação de hora válida.

**SET TITLE** (declaração, 2)

Formato: SET TITLE <expr\$> [,<cor do título>]

Função: Define o título e a cor da tela inicial. <expr\$> deve conter o título com 6 caracteres no máximo. <cor do título> pode variar de 1 a 4

**SET VIDEO** (declaração, 2, opcional)

Formato: SET VIDEO [<modo>[,<Ym>[,<CB>[,<sync>[,<áudio> [,<saída de vídeo>[,<controle AV>]]]]]]]

Função: Define superposição e outros modos.

<modo> pode variar de 0 a 3:

- 0 – Sincronização interna (valor padrão)
- 1 – Digitalização (sincronização externa)
- 2 – Superimpose (sincronização externa)
- 3 – Vídeo externo (sincronização externa)

<Ym> (luminância externa): 0=normal 1=meio tom  
 <CB> (color bus): 0=entrada 1=saída  
 <sync> (modo de sincronização): 0=interna 1=externa  
 <audio> – Seleciona a fonte de áudio:  
     0 – Somente o computador  
     1 – Computador + externo canal direito  
     2 – Computador + externo canal esquerdo  
     3 – Computador + externo canais direito e esquerdo  
 <saída de vídeo> – Seleciona o modo de saída de vídeo:  
     0 – RGB                          1 – Vídeo composto  
 <controle AV> – Seleciona a saída RGB para áudio e vídeo.  
     0 – Não selecionado 1 – Selecionado.

### **SGN** (função, 1)

Formato: X = SGN (<exprN>)

Função: Retorna o resultado do sinal de <exprN> em X.

–1 → Expressão negativa

0 → A expressão resultou em zero

1 → Expressão positiva

### **SIN** (função, 1)

Formato: X = SIN (<exprN>)

Função: Retorna em X o valor do seno de <exprN> (exprN deve ser expresso em radianos).

### **SOUND** (declaração, 1)

Formato: SOUND <nº registrador>,<dado>

Função: Escreve no registrador do PSG o valor de <dado>.

<nº registrador> pode variar de 0 a 13 e <dado> de 0 a 255.

### **SPACE\$** (função, 1)

Formato: X\$ = SPACE\$ (<exprN>)

Função: Retorna em X\$ uma string com <exprN> espaços.

### **SPC** (função, 1)

Formato: PRINT SPC (<exprN>)

Função: Imprime <exprN> espaços.

**SPRITE** (declaração, 1)

Formato: SPRITE {ON | OFF | STOP}

Função: Habilita, desabilita ou suspende interrupção por colisão de sprites.

**SPRITE\$** (variável de sistema, 1)

Formato: SPRITE\$ (<nº sprite>) = <expr\$>

X\$ = SPRITE\$ (<nº sprite>)

Função: Define ou lê o padrão dos sprites.

**SQR** (função, 1)

Formato: X = SQR(<exprN>)

Função: Retorna em X o valor da raiz quadrada de <exprN>.

**STICK** (função, 1)

Formato: X = STICK (<nº porta joystick>)

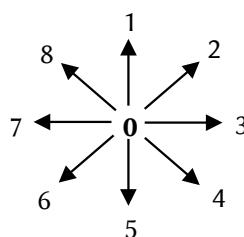
Função: Examina a direção do joystick e retorna o resultado em X.

<nº porta joystick> - 0 → Teclado

1 → porta #1

2 → porta #2

O valor de "X" está ilustrado abaixo:

**STOP** (declaração, 1)

Formato: STOP

Função: Paralisa a execução de um programa.

Formato: STOP {ON | OFF | STOP}

Função: Habilita, desabilita ou suspende interrupção pelo pressionamento das teclas CTRL+STOP.

**STRIG** (função/declaração, 1)

Formato: X = STRIG (<nº porta joystick>)

**Função:** Examina o estado dos botões de disparo e retorna o resultado em X. O valor será -1 se estiver sendo pressionado ou 0 caso contrário.

<nº porta joystick> pode ser:

- 0 = barra de espaço
- 1 = joystick na porta 1, botão A
- 2 = joystick na porta 2, botão A
- 3 = joystick na porta 1, botão B
- 4 = joystick na porta 2, botão B

**Formato:** STRIG (<nº porta joystick>) {ON | OFF | STOP}

**Função:** Habilita, desabilita ou suspende interrupção pelo pressionamento dos botões de disparo.

### **STR\$ (função, 1)**

**Formato:** X\$ = STR\$(<exprN>)

**Função:** Converte o valor de <exprN> em uma string decimal e retorna o valor obtido em X\$.

### **STRING\$ (função, 1)**

**Formato:** X\$ = STRING\$ (<exprN1>, {<expr\$> | <exprN2>})

**Função:** Retorna em X\$ uma string de comprimento <exprN1>, onde todos os caracteres são iguais, formada pelo primeiro caractere de <expr\$> ou pelo caractere cujo código ASCII está representado por <exprN2>.

### **SWAP (declaração, 1)**

**Formato:** SWAP <nome variável>,<nome variável>

**Função:** Troca o conteúdo de duas variáveis. As variáveis têm que ser do mesmo tipo.

### **TAB (função, 1)**

**Formato:** PRINT TAB(<exprN>)

**Função:** Produz <exprN> espaços para as instruções PRINT.

### **TAN (função, 1)**

**Formato:** X - TAN (<exprN>)

**Função:** Retorna em X o valor da tangente de <exprN> (exprN deve ser expresso em radianos).

**TIME** (variável de sistema, 1)

Formato: X – TIME

Função: Retorna o valor de TIME (é uma variável inteira que é continuamente incrementada 60 vezes por segundo)

Formato: TIME = <exprN>

Função: Atribui o valor de <exprN> à variável TIME. Deve ser um valor inteiro.

**TROFF** (comando, 1)

Formato: TROFF

Função: Desliga o rastreamento de linhas do programa em execução.

**TRON** (comando, 1)

Formato: TRON

Função: Liga o rastreamento de linhas do programa em execução.

**USR** (função, 1)

Formato: X = USR[<número>] (<argumento>)

Função: Executa uma rotina em assembly. <número> pode ser um valor de 0 a 9.

**VAL** (função, 1)

Formato: X = VAL (<expr\$>)

Função: Converte <expr\$> em um valor numérico e o armazena em X.

**VARPTR** (função, 1-D)

Formato: X = VARPTR (<nome variável> | #<nº arquivo>)

Função: Retorna em X o endereço onde a <variável> está armazenada ou o endereço do FCB de <nº arquivo>

**VDP** (variável de sistema, 1-2-3)

Formato: X = VDP(<nº registrador>)

VDP(<nº registrador>) = <dado>

Função: Lê ou escreve um dado em um registrador do VDP. <dado> deve ser um valor numérico entre 0 e 255.

**VPEEK** (função, 1-2)

Formato: X = VPEEK (<endereço>)

Função: Retorna em X o conteúdo do byte da VRAM especificado por <endereço>.

**VPOKE** (declaração, 1-2)

Formato: VPOKE <endereço>,<dado>

Função: Escreve no <endereço> da VRAM um byte de dados.  
 <dado> deve ser um valor numérico entre 0 e 255.

**WAIT** (declaração, 1)

Formato: WAIT <nº porta>,<exprN1>[,<exprN2>]

Função: Paralisa a execução do programa até que o valor da porta especificada coincida com o valor de <exprN1> ou <exprN2>.

**WIDTH** (declaração, 1-2)

Formato: WIDTH <número>

Função: Especifica a número de caracteres por linha nos modos texto (Screen 0 e 1).

**XOR** (operador lógico, 1)

Formato: <exprA1> XOR <exprA2>

Função: Efetua operação lógica XOR entre <exprA1> e <exprA2>.

0 xor 0 → 0	1 xor 0 → 1
0 xor 1 → 1	1 xor 1 → 0

### 3.3 – DESCRIÇÃO DE COMANDOS ESTENDIDOS

O comando CALL permite expandir indefinidamente as instruções do MSX-BASIC, permitindo acesso a novos dispositivos em cartuchos ou a novas funcionalidades. Abaixo está listada grande parte das instruções disponíveis através do comando CALL.

#### DM-System2 BASIC

(Extensão instalável para o BASIC)

BGMOFF	BLOCK	COS	FILES
BGMON	CALL	DMM	FSIZE
BGMTMP	CELLO	DMMINI	HELP
BGMTRS	CHGCPU	DMMOFF	HMMM
BGMVOL	CHGDRV	DMMON	HMMV
BGMWAIT	CHGPLT	EXT	INTWAIT
BINLOAD	COLOR=	EXTCOPY	KBOLD

KCOLOR	PACSAVE	SEOFF	UPPER
KINIT	PAUSE	SEON	VCOPY
KPRINT	PCMON	SETBIN	VDPWAIT
KPUT	PEEK	SETPLT	VMOFF
KSIZE	PEEKS	SETSE	VMON
LMMM	PEEKW	SIN	VMWAIT
LMMV	POKE	STATUS	WAIT
LOAD	POKES	SYOFF	XY
MALLOC	POKEW	SYSON	YMMM
PACLOAD	SAVE	SYSTEM	

**FM-X BASIC**

(Disponível com a interface Fujitsu MB22450 quando inserida no slot de expansão FM-X)

CALL MON                    CALL PRINTERSETUP

**FormatMaster-BASIC**

(Disponível com extensão instalável que vem no disco da Future Magazine Extra. Arqs FORMAT.BIN – FORMAT.MEM – FORMAT.TXT)

CALL FORMAT

**GR8NET BASIC**

(Disponível com a instalação do cartucho GR8NET)

DSK	FLUPDATE	NETDUMP	NETGETMAP
DSKCFG	NET	NETEXPRT	NETGETMASK
DSKFMT	NETBITOV	NETFIX	NETGETMD
DSKGETIMG	NETBLOAD	NETFKOPLL	NETGETMEM
DSKLDIR	NETBROWSE	NETFWUPDATE	NETGETMIX
DSKHELP	NETBTOV	NETGETCLK	NETGETMMV
DSKLDIR	NETCDTOF	NETGETCLOUD	NETGETNAME
DSKSETIMG	NETCFG	NETGETDA	NETGETNTP
DSKSVIMG	NETCODE	NETGETDNS	NETGETOPL
DSKSTATE	NETDHCP	NETGETGW	NETGETPATH
FLINFO	NETDIAG	NETGETHOST	NETGETPORT
FLLIST	NETDNS	NETGETIP	NETGETPSG

NETGETQSTR	NETRESST	NETSETMEM	NETSTAT
NETGETTSHN	NETSAVE	NETSETMIX	NETSYSINFO
NETGW	NETSDCRD	NETSETMMV	NETRCHKS
NETHELP	NETSETCLK	NETSETNAME	NETTELNET
NETIMPRT	NETSETCLOUD	NETSETNTP	NETTERM
NETIP	NETSETDA	NETSETOPL	NETTGTMAP
NETLDBUF	NETSETDM	NETSETPATH	NETTSYNC
NETLDRAM	NETSETDNS	NETSETPORT	NETVARBRSTR
NETMASK	NETSETGW	NETSETPSG	NETVARBSIZE
NETNTP	NETSETHOST	NETSETQSTR	NETVARRWTH
NETPLAYBUF	NETSETIP	NETSETTSHN	NETVARUDTO
NETPLAYVID	NETSETMAP	NETSNDTG	NETVER
NETPLAYWAV	NETSETMASK	NETSNDVOL	

### **Hangul-BASIC**

(Disponível em micros coreanos. Veja também PUT HAN, SET HAN e SET SYSTEM)

CLS	HELP	KLEN	REBOOT
ENG	KCHR	KMID	RTCINI
FONT	KCODE	KTYPE	VER
HANOFF	KEXT	MODE9	
HANON	KINSTR	PALETTE	

### **Hitachi-BASIC**

(Disponível em alguns micros Hitachi)

- A versão 1 está disponível no modelo MB-H1 (MSX1)
- A versão 2 está disponível no modelo MB-H2 (MSX1)
- A versão 3 está disponível no modelo MB-H3 (MSX2 )

AUTOMUTE	CMT	MON	REW
BLSCAN	CSCAN	MUTE	SCOPY
CCOPY	CSCOPY	NSCAN	STDBY
CDCOPY	FF	PAUSE	STOP
CFILES	HCOPY	PLAY	TABOFF
CHCOPY	IDTRACE	REC	TABON

**Kanji-BASIC**

(Disponível em micros japoneses MSX2+ ou superior. Veja também PUT KANJI)

AKCNV	KACNV	KLEN	PALETTE
ANK	KANJI	KMID	SJIS
CLS	KEXT	KNJ	
JIS	KINSTR	KTYPE	

**Mega Assembler**

(Disponível com a instalação do cartucho Mega-Assembler)

ASM                    START

**MSX-Aid BASIC**

(Disponível com a instalação do cartucho MSX-AID)

CFILES	MESSAGE	TRACE ON
FIND	MON	VARLIST
HELP	TRACE OFF	XREF

**MSX-Audio BASIC**

(Disponível com a instalação de cartuchos com o MSX-Audio BIOS)

APEEK	COPY PCM	MK VOICE	RECMOD
APOKE	INMK	MK VOL	REC PCM
APPEND MK	KEY OFF	PCM FREQ	SAVE PCM
AUDIO	KEY ON	PCM VOL	SET PCM
AUDREG	LOAD PCM	PITCH	STOPM
BGM	MK PCM	PLAY	TEMPER
CONT MK	MK STAT	PLAY MK	TRANSPOSE
CONVA	MK TEMPO	PLAY PCM	VOICE
CONVP	MK VEL	REC MK	VOICE COPY

**MSX-Music BASIC**

(Disponível através de cartuchos ou internamente)

AUDREG	MUSIC	STOPM	VOICE
BGM	PITCH	TEMPER	VOICE COPY
MDR (*)	PLAY	TRANSPOSE	

(\*) Disponível apenas no MSX turbo R FS-A1GT

**Network-BASIC**

(Extensão BASIC disponível apenas nos computadores MSX2 Yamaha YIS-503IIIR e YIS-805/128R2, usados em escolas da União Soviética)

BRECEIVE	NETEND	PON	SNDRUN
BSEND	NETINIT	RCVMAIL	STOP
CHECK	OFFLINE	RECEIVE	TALK
DISCOM	ONLINE	RUN	WHO
ENACOM	PEEK	SEND	
HELP	POFF	SNDCMD	
MESSAGE	POKE	SNDMAIL	

**NewModem-BASIC**

(Disponível para os modems Philips NMS 1255 e Micro Technology MT-Telcom II)

ANSWER	FILEOUT	LOGFILE	RECFILE
BREAK	GET	LS	RTSOFF
CARRIER	INIMDM	MC	RTSON
CHKMDM	INITMD	MRING	SENDFILE
CONNECT	LINEOFF	MSTART	SPEAKEROFF
DTROFF	LINEON	MSTOP	SPEAKERON
DTRON	LB	OFFHOOK	TDIAL
ECHOOFF	LEN	ONHOOK	TERMINAL
ECHOON	LO	PDIAL	

**Nextor-BASIC**

(Disponível através de cartuchos IDE com Nextor)

CURDRV	DRVINFO	MAPDRV	NEXTOR
DRIVERS	LOCKDRV	MAPDRV	USR

**Pioneer-BASIC (P-BASIC)**

(Disponível nos micros da Pioneer PX-7, PX-V7 e PX-V60)

BLIND	FRAME	MUTE	SEARCH
CHAPTER	FRAME OFF	PAN	SYMBOL
CHAPTER OFF	IMPOSE	REMOTE	VIDEO
DEF UNIV	LCOPY	SCLOAD	
EXTV	LD	SCSAVE	

**Printer-BASIC**

(Disponível nos micros da Toshiba HX20 até HX23F e HX31 até HX34)  
 LCOPY                    SPOLOFF                    SPOLON

**QuickDisk-BASIC**

(Disponível nos micros Daewoo (v1.0) e Casio, Philips e Sanyo (v1.1))

BLOAD	LOAD	QDFORMAT	RUN
BSAVE	MERGE	QDKEY	SAVE
CASQD	QDFILES	QDKILL	

**RMSX-BASIC**

(Extensão que vem com o emulador RMSX para o MSX Turbo R)

?	CHCAS	FILES	LICENSE
CASAUTOREW	CHDIR	HELP	MUTE
CASREW	CHDSK	HZ	PALETTE
CASRUN	EXIT	IOSOUND	RESET

**RookieDrive-BASIC**

(Disponível com a instalação do cartucho RookieDrive)

CREATEDISK	HELP	REBOOT	USBRESET
EJECT	INSERTDISK	USBCD	
FNAME	LOADROM	USBERROR	
FORMAT	MOUNT	USBFILES	

**SFG-BASIC**

(Disponível com a instalação do cartucho Yamaha FM Music Macro)

CANCEL	PATTERN	STOP
CLDVOICE	PHRASE	SYNCOUT
ERASE	PLAY	TEMPO
EVENT ON/OFF/STOP	RCANCEL	TIMER
INIT	REPORT	TRACK
INMKEY	RHYTHM	TRANSPOSE
INST	RSTOP	TSTOP
LENGTH	SELPATTERN	TUNE
LFO	SELVOICE	USERHYTHM
LOOK	SOUND	VLIST
MODINST	STANDBY	WAIT
ON EVENT...GOSUB	START	

**StudioFM BASIC**

(Disponível com a instalação do StudioFM para tocar músicas .MUS geradas pelo FAC Soundtracker. Usar BLOAD"SFMDRV1.BIN",R )

MFADE	MLOAD	MPLAY	MSTOP
-------	-------	-------	-------

**SVI-Modem BASIC**

(Disponível apenas no modem Spectravideo SVI-737)

COMBREAK	COMOFF	COMTERM	TDIAL
COMDTR	COMON	OFFLINE	
COM...GOSUB	COMSTAT	ONLINE	
COMINI	COMSTOP	PDIAL	

**X-BASIC**

(Disponível com a instalação do compilador em tempo real X-BASIC)

'#C	CALL BC	CALL TURBO ON
'#I	CALL RUN	CALL TURBO OFF
'#N		

**3.3.1 – Lista dos comandos**

**?**(declaração, RMSX-BASIC)

Formato: CALL ?

Função: Apresenta a ajuda para o RMSX-BASIC. É equivalente ao comando CALL HELP.

**ADJUST**(comando, Daewoo)

Formato: CALL ADJUST

Função: Habilita a interface interna de caneta ótica. Disponível apenas para MSX2 fabricados pela Daewoo.

**AKCNV**(declaração, Kanji-BASIC)

Formato: CALL AKCNV (<variável>, “<cadeia de caracteres>”)

Função: Converte caracteres de um byte em Kanji de 2 bytes.  
 <variável string> recebe os caracteres convertidos.  
 <cadeia de caracteres> são os caracteres ASCII a serem convertidos.

**ANK** (declaração, Kanji-BASIC)

Formato: CALL ANK

Função: Sai do modo Kanji (a memória usada pelo Kanji driver não é liberada).

**ANSWER** (função, New Modem BASIC)

Formato: CALL ANSWER (<velocidade>)

Função: Detecta a velocidade de uma conexão. Essa instrução funciona apenas em programas BBS. As velocidades detectadas são:

Valor	Norma	Veloc. recep.	Veloc. transm.
1	V21	300 baud	300 baud
2	V23	1200 baud	75 baud
3	V23	75 baud	1200 baud

**APEEK** (função, MSX-Audio)

Formato: CALL APEEK (<endereço>, X)

Função: Retorna em X o valor do byte correspondente ao endereço de memória do MSX-Audio. O endereço pode variar de 0000H a 7FFFH.

**APOKE** (declaração, MSX-Audio)

Formato: CALL APOKE (<endereço>, <dado>)

Função: Escreve no <endereço> da memória de áudio um byte de dados. <dado> deve ser um valor numérico entre 0 e 255. O endereço pode variar de 0000H a 7FFFH.

**APPEND MK** (declaração, MSX-Audio)

Formato: CALL APPEND MK (<nome da matriz>)

CALL APPEND MK (<endereço inicial>, <endereço final>)

CALL APPEND MK (A), onde a sequência A deve ser previamente declarada nas instruções DIM e REC MK.

Função: Acrescenta uma gravação suplementar tocada no teclado musical.

**ASM** (comando, Mega Assembler)

Formato: CALL ASM

Função: Chama o Mega Assembler sem inicializar as variáveis. Para chamar o MA inicializando as variáveis, use CALL START.

**AUDIO** (declaração, MSX-Audio)

Formato: CALL AUDIO (<modo>, <canais com instrumentos>, <canais p/ string 1>, <canais p/ string 2>, ..... , <canais p/ string 9>)

<modo> define o uso do MSX-Audio. O valor padrão é 1.

Modo	FM melodia	PCM	FM ritmo	Tipo
0	9 vozes			Normal
1	6 vozes		3 vozes	Normal
2	9 vozes	1 voz		Normal
3	6 vozes	1 voz	3 vozes	Normal
4	9 vozes			CSM
5	6 vozes		3 vozes	CSM
6	9 vozes	1 voz		CSM
7	6 vozes	1 voz	3 vozes	CSM

No modo CSM, o controle de todos os sons FM (melodia e ritmo) são inválidos. CSM significa Composite Sinusoidal Modeling. Usando todos os operadores em paralelo, esse modo pode ser usado para sintetizar a fala.

<canais com instrumentos> define quantos canais serão atribuídos a um instrumento.

<canais p/ string n> define quantos canais serão usados para cada string relacionada à melodia FM na instrução PLAY.

**AUDREG** (declaração, MSX-Music e MSX-Audio)

Formato: CALL AUDREG <registrarador>, <dado>[,<canal>]

Função: Escreve o valor de <dado> no registrador do OPLL ou do MSX-Audio. <canal> especifica o canal a ser usado (apenas MSX-Audio). Pode ser 0 ou 1, sendo que o padrão é 0.

Obs.: Necessário uso anterior de CALL MUSIC ou CALL AUDIO.

**AUTOMUTE** (comando, Hitachi-BASIC versão 2)

Formato: CALL AUTOMUTE

Função: Adiciona uma pausa de 4 segundos antes de ativar o leitor de dados interno de alguns micros Hitachi.

**BGM** (declaração, MSX-Music e MSX-Audio)

Formato: CALL BGM(n)

Função: Seta execução de comandos enquanto a música está sendo tocada. <n> pode ser 0 ou 1, conforme abaixo:  
 0 - nenhum comando pode ser executado durante a música.  
 1 - comandos podem ser executados durante música (padrão).

**BGMOFF** (declaração, DM-System2 BASIC)

Formato: CALL BGMOFF (<fade>)

Função: Silencia a música tocada pelo OPLL/MSX Music. Requer driver BGM.  
 <fade> – 1 → sem fade (parada imediada)  
 2 → com fade out

**BGMON** (declaração, DM-System2 BASIC)

Formato: CALL BGMON (<endereço inicial> [, <nº repetições>])

Função: Reproduz música usando o driver BGM. Requer driver BGM.  
 <endereço inicial> é um ponteiro para os dados BGM na Main RAM.  
 <nº repetições> é o número de vezes que a música será tocada. “0” indica repetições infinitas.

**BGMTMP** (declaração, DM-System2 BASIC)

Formato: CALL BGMTMP (<tempo>)

Função: Ajusta o “tempo” da música. Requer driver BGM.  
 <tempo> é um valor entre 0 e 255 representando a porcentagem. O valor padrão é 100.

**BGMTRS** (declaração, DM-System2 BASIC)

Formato: CALL BGMTRS (<transpose>)

Função: Ajusta a clave da música. Requer driver BGM.  
 <transpose> é um valor de um byte entre -128 e +127. O valor padrão é 0.

**BGMVOL** (declaração, DM-System2 BASIC)

Formato: CALL BGMVOL ([<Mestre>][,<OPLL>][,<PSG>][,<SCC>])

Função: Ajusta individualmente o volume dos diversos geradores de som. Pode variar de 0 a 15 para cada um, sendo que o valor padrão para todos é 15. Requer driver BGM.

**BGMWAIT** (declaração, DM-System2 BASIC)

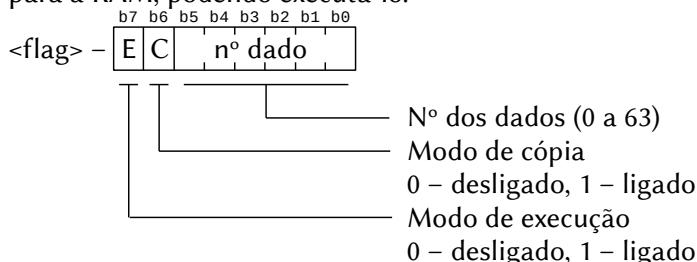
Formato: CALL BGMWAIT

Função: Pausa ou recomeça o BGM. Requer driver BGM.

**BINLOAD** (comando, DM-System2 BASIC)

Formato: CALL BINLOAD (<flag>[,<flag 2>][,<endereço de destino>][,<tamanho>])

Função: Transfere dados concatenados da tabela binária na VRAM para a RAM, podendo executá-lo.



<flag 2> – Valor de um byte que substitui a mesma flag da tabela binária. Por padrão, a flag da tabela é usada.

<endereço de destino> – Valor de 2 bytes que especifica o endereço inicial de destino dos dados.

<tamanho> – Valor de 2 bytes que especifica número de bytes a ser transferido.

Obs. – <endereço de destino> e <tamanho> devem ser omitidos quando o formato dos dados tiver o mesmo formato da instrução COPY (o primeiro byte é a coordenada X e o segundo é a coordenada Y).

**BLIND** (declaração, Pioneer-BASIC)

Formato: CALL BLIND ([<sequência>], [S | L ] )

Função: Apaga ou reabilita a apresentação da tela (apenas Screen 2). <sequência> pode variar de 0 a 9 e especifica a sequência em que a tela será apagada ou habilitada.

S – Salva a tela enquanto é apagada

L – Carrega a tela previamente salva por “S”.

**BLOAD** (comando, QuickDisk BASIC)

Formato: CALL BLOAD ("[QD[n]:]<nome do arquivo>" { [R] | [S] } [,deslocamento])

Função: Carrega o código binário do dispositivo QuickDisk.

QD[n] especifica o dispositivo QuickDisk que será usado.  
 Pode variar de 0 a 7, sendo que o padrão é 0.  
 <nome do arquivo> deve estar no formato 8.3 caracteres.  
 “,R” executa automaticamente o código binário do arquivo.  
 “,S” carrega o conteúdo na VRAM.  
 <deslocamento> indica que o programa será carregado no endereço inicial + deslocamento. Este parâmetro também afeta o endereço de execução.

#### **BLOCK** (comando, DM-System2 BASIC)

Formato: CALL BLOCK ([@]<endereço fonte>,

[@]<endereço destino>, <tamanho>)

Função: Copia dados entre a Main RAM e a VRAM. Se o <endereço> for precedido de “@”, será especificada a VRAM. Para evitar mensagens de erro, números decimais devem ser usados para endereços maiores que FFFFh (65 535).

#### **BLSCAN** (comando, Hitachi-BASIC versão 2)

Formato: CALL BLSCAN

Função: Faz com que o leitor de dados interno do micro Hitachi MB-H2 procure arquivos gravados com BSAVE e que podem ser carregados com BLOAD.

#### **BREAK** (comando, New Modem BASIC)

Formato: CALL BREAK

Função: Atribui a chamada de interrupção à tecla CODE. Isso permitirá interromper as rotinas RING e DIAL apenas pressionando a tecla CODE.

#### **BRECEIVE** (comando, Network-BASIC)

Formato: CALL BRECEIVE ([[<nome da unidade>:]

<nome do arquivo> [, <número do aluno>]  
 [, <end. inicial>] [, <end. final>] [S])

Função: Recebe dados binários na RAM ou VRAM de (outros) micros dos alunos. Pode ser usada pelo professor e pelos alunos autorizados pelo professor através de CALL ENACOM. <nome da unidade> pode ser “A:” ou “B:”; <número do aluno> pode variar de 0 a 15; <endereço inicial> e <endereço final> podem variar de &H0000 a &HFFFF e “,S” especifica VRAM. Versão curta \_BREC.

**BSAVE** (comando, QuickDisk BASIC)

Formato: CALL BSAVE ("[QD[n]:] <nome do arquivo>",  
                   <endereço inicial>, <end. final> [, <end. de execução>])  
 CALL BSAVE ("[QD[n]:] <nome do arquivo>",  
                   <endereço inicial>, <endereço final>, S)

Função: Salva uma área de memória no dispositivo QuickDisk.  
 QD[n] especifica o dispositivo QuickDisk que será usado.  
 Pode variar de 0 a 7, sendo que o padrão é 0.  
 <nome do arquivo> deve estar no formato 8.3 caracteres.  
 <endereço inicial>, <endereço final> e <endereço de  
 execução> podem variar de &H0000 a &HFFFF. Se  
 <endereço de execução> for omitido, será usado o  
 <endereço inicial> no lugar.  
 „S” é usado para salvar o conteúdo da VRAM.

**BSEND** (comando, Network-BASIC)

Formato: CALL BSEND ([[<nome da unidade>:] <nome do arquivo>]  
                   [, <nº aluno>] [, <end. inicial>] [, <end. final>] [S])

Função: Envia dados binários da RAM ou VRAM de outros  
 micros dos alunos. Veja RECEIVE para mais informações.

**CALL** (declaração, DM-System2 BASIC)

Formato: CALL CALL (<endereço>[,<AF>][,<HL>][,<DE>][,<BC>]  
                   [,<IX>][,<IY>])

Função: Chama uma rotina de linguagem de máquina na Main-  
 RAM, a menos que o endereço seja menor que 2000H  
 (abaixo disso será chamada a Main-ROM para permitir  
 acesso às rotinas da BIOS). Se <AF> for menor que 256, o  
 valor será carregado no registrador A.

**CANCEL** (declaração, SFG-BASIC)

Formato: CALL CANCEL (<número do instrumento>)

Função: Cancela um instrumento. <número do instrumento> pode  
 variar de 1 a 4. Versão curta: \_CANC.

**CARRIER** (declaração, New Modem BASIC)

Formato: CALL CARRIER (:GOSUB <número da linha>)

Função: Especifica a rotina do GOSUB a ser executada quando a  
 operadora estiver ausente por um motivo desconhecido  
 ou porque o chamador acabou de desligar. Esta instrução  
 é útil apenas em programas BBS.

**CASAUTOREW** (comando, RMSX-BASIC)

Formato: CALL CASAUTOREW [ON] | [OFF]

Função: Habilita ou desabilita a rebobinagem automática de uma imagem de fita (arquivo CAS) de volta ao início. Sem parâmetro, esta instrução alterna entre as duas opções.  
 ON – Ativa a rebobinagem automática da imagem da fita.  
 OFF – Desativa a rebobinagem automática

**CASQD** (comando, QuickDisk BASIC)

Formato: CALL CASQD [("[CAS:]<nome do arquivo 1>"]  
 [,"[QD[n]:]<nome do arquivo 2>"])]

Função: Transfere o arquivo especificado do cassete para o QuickDisk.  
 QD[n] especifica o dispositivo QuickDisk que será usado.  
 Pode variar de 0 a 7, sendo que o padrão é 0.  
 <nome do arquivo 1> é o nome do arquivo a ser copiado da fita.  
 <nome do arquivo 2> é o nome do arquivo a ser gravado no Quick Disk. O formato é limitado a 6 caracteres sem extensão. Se <nome do arquivo 2> for omitido, será repetido o <nome do arquivo 1>. Sem parâmetros, este comando transfere o arquivo da fita para o QuickDisk padrão com o mesmo nome.

**CASREW** (comando, RMSX-BASIC)

Formato: CALL CASREW

Função: Rebobina manualmente uma imagem de fita (arquivo CAS) de volta ao início.

**CASRUN** (comando, RMSX-BASIC)

Formato: CALL CASRUN [("[<letra da unidade>:] [\<caminho> \]<nome do arquivo.CAS>")]

Função: Carrega e executa arquivos contidos da imagem de fita especificada (arquivo CAS). Se <nome do arquivo.CAS> for omitido, será executado o primeiro arquivo da imagem inserida com CALL CHCAS.<letra da unidade>: pode ir de A: a H:.

**CCOPY** (comando, Hitachi-BASIC versão 3)

Formato: CALL CCOPY

Função: Envia para a impressora uma cópia mais escura de uma tela gráfica em Screens 2, 4 ou 5 simulando tons de cinza.

**CDCOPY** (comando, Hitachi-BASIC versão 3)

Formato: CALL CDCOPY

Função: Envia para a impressora uma cópia de uma tela gráfica em Screens 2, 4 ou 5 usando apenas pontos brancos e pretos.

**CELLO** (declaração, DM-System2 BASIC)

Formato: CALL CELLO (<X0>,<Y0>)-[STEP](<X1>,<Y1>)[,<n0>]  
[,<n1>][,<n2>]

Função: Altera as cores de uma área retangular especificada da tela. Não funciona nas Screens 0 a 4.

Screens 5, 6, 7: Substitui uma cor por outra  
 Screen 8: Modifica as cores de acordo com o RGB  
 Screen 10, 11, 12: Modifica os valores Y das cores  
 <X0>,<Y0> coordenadas do canto inicial do retângulo  
 <X1>,<Y1> coordenadas do canto final do retângulo  
 <n0> cor a substituir nas Screens 5 a 7 (0 ~ 15);  
     valor para adicionar ao vermelho na Scr 8 (-7 ~ + 7);  
     valor para adicionar a Y na Scr 10 a 12 (-31 ~ + 31).  
 <n1> nova cor para as Screens 5 a 7 (0 ~ 15);  
     valor para adicionar ao verde na Screen 8 (-7 ~ + 7).  
 <n2> valor para adicionar ao azul na Screen 8 (-7 ~ + 7).

**CFILES** (comando, Hitachi-BASIC, MSX Aid BASIC)

Formato: CALL CFILES [Hitachi-BASIC 2]

Função: Lista o conteúdo da fita inserida no leitor de dados interno de alguns micros Hitachi. É recomendável rebobinar a fita com o comando CALL REW antes.

Formato: CALL CFILES [MSX Aid BASIC]

Função: Lista o conteúdo da fita inserida no leitor de dados conectado ao MSX. A lista especifica se um arquivo é binário (OBJ), BASIC (BAS) ou ASCII (ASC). Retorna também o tamanho dos arquivos binários. É recomendável rebobinar a fita antes.

**CHAPTER** (declaração, Pioneer-BASIC)

Formato: CALL CHAPTER (<nº capítulo>, GOSUB <nº linha>  
 CALL CHAPTER OFF

Função: Especifica no número de linha da sub-rotina que será executada quando o capítulo <nº capítulo> for atingido.

<nº capítulo> deverá estar na faixa entre 50 e 54.000. Se “OFF” for especificado, cancela a atribuição de número de linha. Este comando é específico para uso com o Laser Vision Player LD-700 da Pioneer e não pode ser usado em conjunto com o comando FRAME.

#### **CHCAS** (comando, RMSX-BASIC)

Formato: CALL CHCAS ("[<letra da unidade>:] [\<caminho>\<nome do arquivo.CAS>")

Função: Monta (insere) a imagem de fita especificada (arquivo CAS) no leitor de cassete virtual do computador MSX1 ou MSX2 emulado em um Turbo R com o emulador rMSX.  
 <letra da unidade>: pode ir de A: até H:.

#### **CHCOPY** (comando, Hitachi-BASIC versão 3)

Formato: CALL CHCOPY

Função: Envia para a impressora uma cópia mais clara de uma tela gráfica em Screens 2, 4 ou 5 simulando tons de cinza.

#### **CHDIR** (declaração, Disk-BASIC 2nd version, RMSX-BASIC)

Formato: CALL CHDIR ([<letra da unidade>:] [\<caminho>]) [Disk-BASIC 2]

Função: Troca subdiretório. O argumento também pode ser apenas “..” ou “.” para retornar diretórios.

Formato: CALL CHDIR ([<letra da unidade>:] [\<caminho>]) [RMSX-BASIC]

Função: Altera o diretório de trabalho atual de um disco real em uma unidade do MSX Turbo R, usada como host para um computador MSX1 / MSX2 emulado nesta máquina com o emulador rMSX. O argumento também pode ser apenas “..” ou “.” para retornar diretórios.

#### **CHDRV** (comando, Disk-BASIC 2nd version)

Formato: CALL CHDRV (<letra da unidade>:)

Função: Troca o drive de acordo com “[<letra da unidade>:]”. Se o Nextor estiver instalado, o argumento pode ser substituído por um número (1 = A:; 2 = B:; etc.); caso contrário deve indicar a letra da unidade (A: até H:).

#### **CHKDSK** (comando, RMSX-BASIC)

Formato: CALL CHKDSK (ver os parâmetros abaixo)

- Função:** Monta (insere) a imagem de disco especificada (arquivo DSK) na unidade de disco virtual do computador MSX1/MSX2 emulado em um Turbo R com o emulador rMSX e/ou ativa um número de disco especificado (a unidade de disco do Turbo R também pode ser usada com um disco real).
- Para montar a imagem do disco para o número do disco atual (se o parâmetro não for fornecido ou estiver vazio, a unidade de disco real será usada):  
 CALL CHKDSK [("[" <letra da unidade>:]  
           [\" <caminho>\"] <nome do arquivo.DSK>")]
  - Para montar a imagem do disco no número especificado do disco (sem ativação):  
 CALL CHKDSK ("[" <letra da unidade>:] [\" <caminho>\"]  
           <nome do arquivo.DSK>"), <número do disco>
  - Para ativar o número do disco especificado e eventualmente montar a imagem do disco nele:  
 CALL CHKDSK (<letra da unidade>), [("[" <nome dispositivo>:] [\" <caminho>\"] <nome arq.DSK>")]

### **CHECK** (comando, Network-BASIC)

Formato: CALL CHECK ([<variável de conexão>] [, <variável de comunicação>])

- Função:** Verifica quais alunos estão conectados à rede e/ou quais alunos estão habilitados para se comunicar com outros. Esta instrução está disponível apenas para o professor. <variável de conexão> contém a representação binária dos alunos conectados / não conectados. <variável de comunicação> contém a representação binária dos alunos com comunicação estendida ativada ou desativada. Ambas são variáveis inteiras de 16 bits, onde o bit 0 está associado ao aluno 1, o bit 1 está associado ao aluno 2 e assim por diante, até o bit 14. “0” significa aluno conectado ou ativo e “1” significa aluno desconectado ou inativo.

### **CHGCPU** (comando, DM-System2 BASIC)

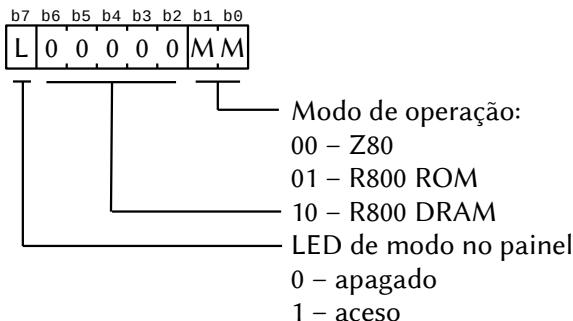
Formato: CALL CHGCPU ([<modo>][,<variável>])

- Função:** Troca ou retorna modo CPU nos MSX turbo R.

<modo> – Modo a ser aplicado.

<variável> – Valor antes de ser trocado por <modo>.

Os dois parâmetros têm o seguinte formato:



### **CHGDRV** (comando, DM-System2 BASIC)

Formato: CALL CHGDRV ([<número do drive>][,<variável>])

Função: Troca ou retorna a unidade de drive atual.

<número do drive> – deve ser um número entre 1 e 8, onde  
1 = A; 2 = B; etc.

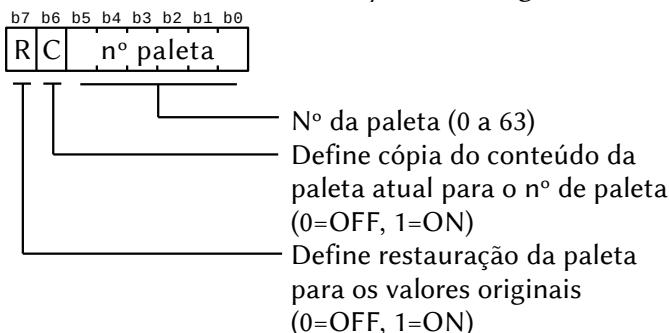
<variável> – variável numérica que receberá o tamanho do arquivo.

### **CHGPLT** (declaração, DM-System2 BASIC)

Formato: CALL CHGPLT (<número>)

Função: Altera as cores da paleta. Os dados da paleta dever ser previamente colocados na RAM.

<número> é um valor de um byte com o seguinte formato:



### **CHKHMDM** (função, New Modem BASIC)

Formato: CALL CHKMDM (<variável numérica>)

Função: Verifica se o modem está presente. Se <variável numérica> for 0, o modem foi detectado, caso contrário não há modem.

**CLDVOICE** (comando, SFG-BASIC)

Formato: CALL CLDVOICE [ ( <exibir nome> , <dispositivo> ) ]

Função: Carrega os dados de voz para a memória, a partir do cassete ou da memória do cartucho. Versão curta: \_CLDV. <opção> especifica se o nome da voz deve ser exibido durante a carga. Se for “0” o nome não será exibido e se for “1” será exibido.

<dispositivo> pode ser: 0 – Cassete; 1 – Cartucho.

**CLS** (declaração, Kanji-BASIC e Hangul-BASIC 4)

Formato: CALL CLS

Função: Limpa a tela no modo Kanji.

**CMT** (comando, Hitachi-BASIC versão 2)

Formato: CALL CMT

Função: Inicia o “Tape Utility” no computador Hitachi MB-H2.

**COLOR=** (declaração, DM-System2 BASIC)

Formato: CALL COLOR = (<nº paleta>,<nível verm.>,<nível verde>,<nível azul>)

Função: Altera as cores da paleta de uma única cor. O nível pode variar de 0 a 7 para cada cor primária. As alterações são armazenadas somente em (NEWPLT) e não na tabela de paleta VRAM.

**COMBREAK** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMBREAK ([<nº da porta> ,<nº caracteres> ] ] )

Função: Envia instrução para bloqueio de mensagens. <nº da porta> pode variar de 0 a 4 e <nº caracteres> a serem bloqueados pode variar de 3 a 32 767.

**COM GOSUB** (declaração, Modem BASIC, SVI Modem BASIC)

Formato: CALL COM ( [<nº da porta> GOSUB <nº da linha>] )

Função: Especifica a subrotina que será chamada quando ocorrer uma interrupção na RS232-C.

<nº da porta> pode variar de 0 a 4; se omitido será 0.

A subrotina iniciará no <nº da linha> especificado.

**COMINI** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMINI ([<dados>] [,<velocidade de recepção>]  
[,<velocidade de transmissão>] [,<tempo de espera>] )

Função: Inicializa o modem com os <dados> fornecidos.

<dados> é uma string alfanumérica de até 10 caracteres, cujo padrão, se omitida é “0:8N1XHNNN”. O número inicial é a porta RS232C seguida de “:” e os caracteres seguintes representam:

- 3º – Tamanho da palavra (5 a 8) ou “del”
- 4º – paridade (E-par, O-ímpar, I-ignora, N-sem paridade) ou “ins”
- 5º – Tam. bit de parada: 1- 1 bit, 2- 1,5 bits, 3- 2 bits
- 6º – XON/XOFF: X- xon, N- xoff
- 7º – H- handshaking, N- sem handshaking
- 8º – LF: A- insere LF, N- não insere LF
- 9º – LF: A- deleta LF, N- não deleta LF
- 10º – Shift in/out: S- habilita, N- desabilitada

<velocidade de recepção> pode variar de 50 a 1200. Os valores válidos são: 50, 75, 110, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200. Se omitido, assumirá 1200.

<velocidade de transmissão> pode variar de 50 a 1200. Se omitido, assume a mesma de <veloc. recepção>.

<tempo de espera> é especificado em segundos e pode variar de 0 a 255. Se omitido, assumirá 0.

**COMOFF** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMOFF ( [<“nº da porta :”>] )

Função: Desabilita a interrupção vinda da porta RS232-C.

**COMON** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMON ( [<“nº da porta :”>] )

Função: Habilita a interrupção vinda da porta RS232-C.

**COMSTAT** (função, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMSTAT ( [<“nº da porta :”>], <variável inteira> )

Função: Retorna o estado da porta RS232-C.

<“nº da porta :”> deve variar e 0: a 4:. Se omitido, será 0:.

<variável inteira> retorna os seguintes valores:

- bit 15: Erro de estouro de buffer de recebimento  
0- nenhum erro, 1- ocorreu erro
- bit 14: Erro de tempo limite  
0- nenhum erro, 1- ocorreu erro
- bit 13: Erro de enquadramento (o bit binário "0" foi  
recebido em vez do bit de parada.)  
0- nenhum erro, 1- ocorreu erro
- bit 12: Erro de saturação (dados recebidos antes do  
buffer de recebimento estar vazio)  
0- nenhum erro, 1- ocorreu erro
- bit 11: Erro de paridade  
0- nenhum erro, 1- ocorreu erro
- bit 10: Pressionamento de [CTRL] + [STOP]  
0- não pressionado, 1- pressionado
- bit 9: Reservado
- bit 8: Reservado
- bit 7: Status do sinal CS (CTS)  
0- desligado, 1- ligado
- bit 6: Temporizador/contador definido para a  
detecção de erro de tempo limite  
0- não definido, 1-definido
- bit 5: Reservado
- bit 4: Reservado
- bit 3: Status do sinal DR (DSR)  
0- desligado, 1- ligado
- bit 2: Sequência de parada detectada enquanto que  
o COMSTAT é executado  
0- não detectada, 1- detectada
- bit 1: Reservado
- bit 0: Status do sinal do CD  
0- desligado, 1- ligado

#### **COMSTOP** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMSTOP ( [<"nº da porta :">] )

Função: Suspende a interrupção vinda da porta RS232-C.

<"nº da porta :"> deve variar e 0: a 4:. Se omitido, será 0:.

**COMTERM** (comando, Modem BASIC, SVI Modem BASIC)

Formato: CALL COMTERM ( [<“nº da porta :”>] )

Função: Coloca o MSX no modo terminal. Para sair do modo terminal, pressione CTRL+STOP juntas. <“nº da porta :”> deve variar e 0: a 4:. Se omitido, será 0:. Uma vez no modo terminal, use as seguintes teclas:  
 [SHIFT]+[F1] – Exibe os códigos de controle recebidos.  
 [SHIFT]+[F2] – Exibe as teclas pressionadas.  
 [SHIFT]+[F3] – Exibe e imprime as teclas pressionadas.  
 [STOP] – Pressione e segure para enviar a sequência de interrupção ao host.

**CONNECT** (comando, New Modem BASIC)

Formato: CALL CONNECT (<variável numérica>)

Função: Estabelece conexão em um programa Terminal, com a velocidade definida em CALL INIMDM. <variável numérica> armazena o resultado da operação:  
 0 – ocorreu um erro ao tentar fazer a conexão  
 1 – operadora detectada – o modem está conectado  
 2 – A rotina foi abortada pressionando a tecla CODE  
 3 – o número de telefone está ocupado

**CONT MK** (comando, MSX-Audio)

Formato: CALL CONT MK

Função: Continua uma reprodução ou gravação do teclado musical que foi cancelada pelo comando STOPM.

**CONVA** (declaração, MSX-Audio)

Formato: CALL CONVA (<arquivo fonte>, <arquivo destino>)

Função: Converte dados PCM para dados ADPCM.  
 <arquivo fonte> e <arquivo destino> são definidos por um número que pode variar de 0 a 15.

**CONVP** (declaração, MSX-Audio)

Formato: CALL CONVP (<arquivo fonte>, <arquivo destino>)

Função: Converte dados ADPCM para dados PCM.  
 <arquivo fonte> e <arquivo destino> são definidos por um número que pode variar de 0 a 15.

**COPY PCM** (comando, MSX-Audio)

Formato: CALL COPY PCM (<arquivo fonte>, <arquivo destino>, [<offset fonte>, [<tamanho arq>, [<offset destino>]]])

Função: Copia dados ADPCM e PCM.

<arquivo fonte> e <arquivo destino> são definidos por um número que pode variar de 0 a 15.

<offset fonte> e <offset destino> define o deslocamento em unidades de 256 bytes.

<tamanho arq> é o tamanho do arquivo em bytes

**COS** (função, DM-System2 BASIC)

Formato: CALL COS (<variável>,<ângulo>,<valor>)

Função: Retorna o cosseno de um ângulo. O resultado é obtido pela multiplicação do cosseno do ângulo por um valor numérico.

<variável> – Variável numérica que receberá o resultado.

<ângulo> – é o valor do ângulo em graus.

<valor> – Número de dois bytes (valor inteiro).

**CREATEDISK** (comando, RookieDrive-BASIC)

Formato: CALL CREATEDISK (<nome do disco>)

Função: Cria uma nova imagem de disco sem formatá-la. A imagem de disco criada está cheia de caracteres 0xFFH. Instrução experimental e não totalmente implementada (limitada a discos de 720 Kbytes).

**CSCAN** (comando, Hitachi-BASIC versão 2)

Formato: CALL CSCAN

Função: Faz com que o leitor de dados interno do micro Hitachi MB-H2 procure arquivos gravados com CSAVE e que podem ser carregados com CLOAD.

**CSCOPY** (comando, Hitachi-BASIC versão 3)

Formato: CALL CSCOPY (<c1> [,<c2>, <c3>, <c4> .... <c15>] )

Função: Envia para a impressora uma cópia de uma tela gráfica em Screens 2, 4 ou 5 usando uma fórmula baseada nas cores selecionadas. A diferença com CALL SCOPY é desconhecida.

**CURDRV** (declaração, Nextor)

Formato: CALL CURDRV

Função: Apresenta a unidade de drive ativa.

**DEF UNIV** (comando, Pioneer-BASIC)

Formato: CALL DEF UNIV (<nº dispositivo>, <índice dispositivo>)

Função: Define o dispositivo a ser controlado pelo comando  
REMOTE. <nº dispositivo> pode variar de 3 a 15 e <índice dispositivo> pode variar de 1 a 255.

**DISCOM** (comando, Network-BASIC)

Formato: CALL DISCOM (<número do aluno>)

Função: Desabilita o envio de mensagens de um aluno. Esta  
instrução está disponível apenas para o professor. Por  
padrão, após a inicialização, os alunos podem enviar  
apenas mensagens ao professor. <número do aluno> pode  
variar de 1 a 15. Versão curta: \_DISC.

**DMM** (função, DM-System2 BASIC)

Formato: CALL DMM (<variável>[,<tempo>]) [, S])

Função: Executa a entrada de dispositivo e devolve o resultado em  
<variável>. Para abortar: CTRL+STOP. (Requer driver DEV).  
<variável> deve ser numérica. Os valores de retorno são:

0 – Não pressionado	10 – GRAPH	13 – ESC
1 a 8 – 8 direções	11 – STOP	14 – HOME
9 – Espaço	12 – TAB	15 – SELECT

<tempo> que o comando aguarda, em unidades de 1/60 seg.

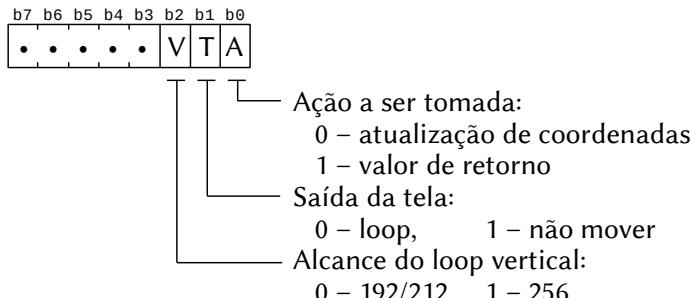
[,S] – Se especificado, o sprite definido em "CALL  
DMMINI" será movido automaticamente.

**DMMINI** (declaração, DM-System2 BASIC)

Formato: CALL DMMINI ([<modo>] [,<número do sprite>])

Função: Define a entrada de dispositivo. Quando o DM-System2 é  
inicializado, o mouse e o joypad são configurados como  
dispositivos de entrada. (Requer driver DEV).

<modo> é um valor de um byte com o seguinte formato  
(os valores padrão são 0):



<número do sprite> é um valor de 1 byte que especifica o sprite exibido durante a execução de CALL DMM. Se omitido, "0" é usado.

#### **DMMON** (comando, DM-System2 BASIC)

Formato: CALL DMMON ([<endereço>])

Função: Ativa a verificação contínua de dispositivo, colocando o resultado na área de informações do DM-System2.

<endereço> define o endereço de execução quando um evento de dispositivo for detectado. (Requer driver DEV).

#### **DMMOFF** (comando, DM-System2 BASIC)

Formato: CALL DMMOFF

Função: Desativa a verificação contínua de dispositivo. (Requer driver DEV).

#### **DRIVERS** (declaração, Nextor)

Formato: CALL DRIVERS

Função: Apresenta informações sobre os drivers disponíveis para o Nextor e MSXDOS.

#### **DRVINFO** (declaração, Nextor)

Formato: CALL DRVINFO

Função: Apresenta informações sobre todas as letras de drive disponíveis.

#### **DSK** (comando, GR8NET-BASIC)

Formato: CALL DSK

Função: Apresenta ajuda e estado e faz diagnósticos.

**DSKCFG** (comando, GR8NET-BASIC)

Formato: CALL DSKCFG (<nº máx páginas>, <nº de páginas>)

Função: Obtém ou gerecia o estado da imagem de disco.

<nº máx páginas> é uma variável que recebe o número máximo de páginas lógicas da RAM-Disk.

<nº de páginas> é uma variável ou constante que define o tamanho da RAM-Disk. Deve estar entre 0 e <nº máx páginas>.

**DSKFMT** (comando, GR8NET-BASIC)

Formato: CALL DSKFMT

Função: Inicializa a imagem em RAM-Disk.

**DSKGETIMG** (função, GR8NET-BASIC)

Formato: CALL DSKGETIMG [(<variável string>)]

Função: Obtém a localização atual da imagem de disco e retorna o caminho na <variável string>.

**DSKHELP** (declaração, GR8NET-BASIC)

Formato: CALL DSKHELP

Função: Apresenta a ajuda para a GR8NET.

**DSKLDIR** (comando, GR8NET-BASIC)

Formato: CALL DSKLDIR

Função: Carrega a imagem de disco atual no buffer da GR8NET.

**DSKSETIMG** (comando, GR8NET-BASIC)

Formato: CALL DSKSETIMG [(<caminho>)]

Função: Define a localização da imagem de acordo com <caminho>, que pode ser variável string ou expressão alfanumérica.

**DSKSVIMG** (comando, GR8NET-BASIC)

Formato: CALL DSKSVIMG [(<caminho>)]

Função: Salva a imagem de disco do buffer da GR8NET no cartão SD. Se <caminho> for omitido, será usado o caminho definido por DSKSETIMG.

**DSKSTATE** (comando, GR8NET-BASIC)

Formato: CALL DSKSTATE (<estado>,<sinalizadores>)

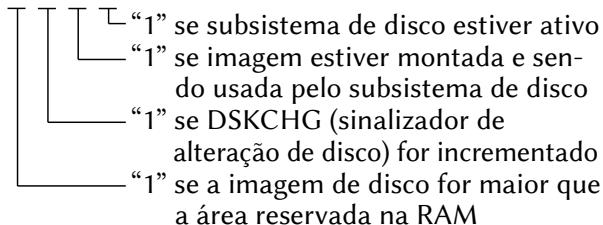
Função: Obtém ou define subsistema de disco.

<estado> define o estado do sistema. Se for “0”, o disco será

desativado; “1” ativa. A mudança entrará em vigor na próxima inicialização a quente do sistema. A inicialização por hardware forçará o retorno ao padrão.

**<sinalizadores>** retornará com os seguintes valores:

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	0	0	R	D	M	A



#### **DTROFF** (comando, New Modem BASIC)

Formato: CALL DTROFF

Função: Desabilita o sinal DTR (Data Terminal Ready).

#### **DTRON** (comando, New Modem BASIC)

Formato: CALL DTRON

Função: Habilita o sinal DTR (Data Terminal Ready).

#### **ECHOOFF** (comando, New Modem BASIC)

Formato: CALL ECHOOFF

Função: Envia caracteres apenas para a linha telefônica. A tela exibirá apenas caracteres recebidos.

#### **ECHOON** (comando, New Modem BASIC)

Formato: CALL ECHOON

Função: Habilita o envio de caracteres para a linha telefônica e simultaneamente para a tela.

#### **EJECT** (comando, RookieDrive-BASIC)

Formato: CALL EJECT

Função: Ejeta a imagem de disco atualmente inserida e exclui seu nome do arquivo USBMSX.INI.

#### **ENACOM** (comando, Network-BASIC)

Formato: CALL ENACOM (<número do aluno>)

**Função:** Habilita o envio de mensagens para um aluno. Esta instrução está disponível apenas para o professor. Por padrão, após a inicialização, os alunos podem enviar mensagens apenas para o professor. <número do aluno> pode variar de 1 a 15. Versão curta: \_ENAC.

**ENG** (comando, Hangul-BASIC 3)

**Formato:** CALL ENG

**Função:** Volta para o modo texto Screen 0.

**ERASE** (comando, SFG-BASIC)

**Formato:** CALL ERASE (<número da trilha>)

**Função:** Apaga o conteúdo da trilha especificada. <número da trilha> pode ir de 1 até o número especificado por CALL TRACK. Versão curta: \_ERAS.

**EVENT** (comando, SFG-BASIC)

**Formato:** CALL EVENT ([<nº do evento>]) ON | OFF | STOP

**Função:** Habilita, desabilita ou interrompe a interrupção por evento especificada em ON EVENT ... GOSUB. <nº do evento> pode ser:

1 ~ 4 – Interrompe quando a reprodução do instrumento especificado termina.

5 – Interrompe quando a reprodução do ritmo termina.

6 – Interrompe de acordo com o tempo programado no timer da unidade FM.

Se <nº do evento> for omitido, o comando será aplicado a todos os eventos. Versão curta: \_EVEN.

**EXIT** (comando, RMSX-BASIC)

**Formato:** CALL EXIT

**Função:** Sai do emulador rMSX e volta ao uso ‘normal’ do computador MSX Turbo R.

**EXT** (comando, DM-System2 BASIC)

**Formato:** CALL EXT ([@]<endereço fonte>,[@]<endereço destino>)

**Função:** Extrai dados compactados no formato BPE.

<endereço fonte> é o endereço dos dados compactados.

<endereço destino> é o endereço de destino dos dados descompactados.

**Obs.:** Se especificado “@”, significa VRAM.

**EXTCOPY** (comando, DM-System2 BASIC)

Formato: CALL EXTCOPY ([@]<endereço fonte>,<X>,<Y>  
[,<direção>]) [,<operador lógico>]

Função: Descompacta dados no formato BPE para uma área retangular na tela.

<endereço fonte> – Endereço dos dados compactados. Se especificado “@”, significa VRAM (máximo 64K).

<X> – Coordenada destino horizontal (0 a 511).

<Y> – Coordenada destino vertical (0 a 1023).

<direção> – é a direção de descompactação na tela.

0 – para a direita e para baixo (padrão).

1 – para a esquerda e para baixo.

2 – para a direita e para cima.

3 – para a esquerda e para cima.

<operador lógico> pode ser [T]PSET, [T]PRESET, [T]XOR,  
[T]OR ou [T]AND. O padrão é PSET.

**EXTV** (comando, Pioneer-BASIC)

Formato: CALL EXTV (<variável>)

Função: Verifica se existe sinal de vídeo externo no terminal de entrada e retorna o resultado em <variável>.

0 – Não há sinal de vídeo na entrada.

1 – Sinal externo de vídeo detectado.

**FF** (comando, Hitachi-BASIC versão 2)

Formato: CALL FF

Função: Coloca o leitor de dados embutido do computador Hitachi MB-H2 no modo de busca rápida.

**FILEOUT** (comando, New Modem BASIC)

Formato: CALL FILEOUT ("[<dispositivo>:] <nome do arquivo>]",  
<variável>)

Função: Envia um arquivo de texto ou envia diretamente o texto digitado em um terminal. <variável> contém primeiramente uma opção sobre a pergunta “More? (Y/N)” e depois armazena um código de controle. Os valores em <variável> podem ser:

0 – Esta opção está ligada.

1 – A opção está desligada (necessária para upload ASCII).

Códigos de controle:

- 0 – Texto foi enviado corretamente.
- 3 – A operação foi abortada com CTRL+C ou C.
- 7 – o arquivo não foi encontrado.

### **FILES** (declaração, DM-System2 BASIC, RMSX BASIC)

Formato: CALL FILES ("[<dispositivo>:]\\<caminho>][[\\]<nome do arquivo>]", <variável>) [DM2-BASIC]

Função: Retorna nomes de arquivos e os coloca na área de trabalho do DM-System2 (endereço 7A00h). Os nomes dos arquivos serão colocados um após o outro a cada 12 bytes. <dispositivo> pode ser drive A: a H: ou COM: para micros conectados com RS232C.

<caminho> especifica o local da pasta ou arquivo  
 <nome do arquivo> aceita caracteres coringa (\* e ?)  
 <variável> é uma variável numérica que receberá a quantidade de arquivos localizados.

Formato: CALL FILES ("[<dispositivo>:]\\<caminho>][[\\]<nome do arquivo>]) [RMSX BASIC]

Função: Lista o conteúdo de um disco real em uma unidade do MSX Turbo R, usado como host emulação de um MSX1 ou MSX2 com o emulador rMSX.  
 <dispositivo> pode ser drive A: a H:  
 <caminho> especifica o local da pasta ou arquivo.  
 <nome do arquivo> aceita caracteres coringa (\* e ?).

### **FIND** (comando, MSX-Aid BASIC)

Formato: CALL FIND ("<variável>" [, [<número da linha inicial>], [<número da linha final>], [P]])

Função: Lista parte do programa MSX-BASIC que está na memória, onde uma variável alfanumérica ou string específica é usada. <variável> deve ter um ou dois caracteres. Podem ser especificados também o <número da linha inicial> e o <número da linha final> do programa BASIC a ser listado. Se [P] for especificado, a listagem será enviada para a impressora.

### **FLINFO** (declaração, GR8NET-BASIC)

Formato: CALL FLINFO

Função: Apresenta informações sobre a memória flash serial.

**FLLIST** (declaração, GR8NET-BASIC)

Formato: CALL FLLIST

Função: Lista o conteúdo da memória flash serial.

**FLUPDATE** (comando, GR8NET-BASIC)

Formato: CALL FLUPDATE (<setor>[,F])

Função: Atualiza o conteúdo da memória flash serial. <setor> especifica o número do setor onde a atualização começará. Se o parâmetro “F” for incluído, a atualização iniciará imediatamente sem solicitar confirmação.

**FNAME** (declaração, RookieDrive-BASIC)

Formato: ?

Função: ?

**FONT** (comando, Hangul-BASIC 4)

Formato: CALL FONT

Função: Habilita alternância entre caracteres coreanos (tecla HANGUL) e os caracteres não coreanos disponíveis através das teclas KANA, CYRILLIC ou CODE. Retornará erro se usada em Screen 9.

**FORMAT** (comando, Disk-BASIC, FormatMaster-B., RookieDrive-B.)

Formato: CALL FORMAT [Disk-BASIC]

Função: Formata um disquete. Oferece duas opções:  
1 – 1 side, double track (face simples, 360K).  
2 – 2 sides,double track (face dupla, 720K).

Formato: CALL FORMAT [FormatMaster-BASIC]

Função: Formata um disquete oferecendo opções adicionais.

Necessário Disk-BASIC versão 1 (essa instrução não é compatível com o Disk-BASIC versão 2).

- 1) 40 trilhas – 8 setores por trilhas – FA.
- 2) 80 trilhas – 8 setores por trilhas – FB.
- 3) 40 trilhas – 9 setores por trilhas – F8.
- 4) 80 trilhas – 9 setores por trilhas – F9.

Formato: CALL FORMAT [RookieDrive-BASIC]

Função: Formata o disco inserido em uma unidade de disquete USB padrão conectada a uma interface Rookie Drive ou a imagem do disco inserida na interface do Rookie Drive.

- 1) 720K, format. completa.      3) 1,44M, format. completa.
- 2) 720 K, format. rápida.      4) 1,44M, format. rápida.

**FRAME** (declaração, Pioneer-BASIC)

Formato: CALL FRAME (<nº frame>, GOSUB <nº linha>)  
 CALL FRAME OFF

Função: Especifica no número de linha da sub-rotina que será executada quando o frame <nº frame> for atingido. <nº frame> deverá estar na faixa entre 50 e 54.000. Se "OFF" for especificado, cancela a atribuição de número de linha. Este comando é específico para uso com o Laser Vision Player LD-700 da Pioneer e não pode ser usado em conjunto com o comando CHAPTER.

**FSIZE** (função, DM-System2 BASIC)

Formato: CALL FSIZE ("[<dispositivo>:]\\<caminho>][[]<nome do arquivo>]",<variável>)

Função: Retorna o tamanho do arquivo.  
 <dispositivo> pode ser drive A: a H: ou COM: para micros conectados com RS232C.  
 <caminho> especifica o local da pasta ou arquivo.  
 <nome do arquivo> aceita caracteres coringa (\* e ?).  
 <variável> recebe o tamanho do arquivo.

**GET** (função, New Modem BASIC)

Formato: CALL GET (<variável>)

Função: Recupera o código ASCII de um caractere pressionado no teclado ou recebido na linha telefônica (se essa linha não tiver sido desativada com CALL LINEOFF). <variável> armazena o código ASCII. Atalhos especiais em um programa BBS: Pausa: CTRL+S ou S; continuar (após uma pausa): qualquer tecla; parar: CTRL+C ou C.

**HANOFF** (comando, Hangul-BASIC 1)

Formato: CALL HANOFF

Função: Desativa o recurso para agrupar os caracteres em blocos. (Característica dos caracteres Hangul, usados na Coréia, disponíveis depois de pressionar a tecla HANGUL). Retorna erro se usada em Screen 9.

**HANON** (comando, Hangul-BASIC 1)

Formato: CALL HANON

**Função:** Habilita o recurso para agrupar os caracteres em blocos. (Característica dos caracteres Hangul, usados na Coréia, disponíveis depois de pressionar a tecla HANGUL). Retorna erro se usada em Screen 9.

**HCOPY** (comando, Hitachi-BASIC versão 2-3)

**Formato:** CALL HCOPY

**Função:** Envia para a impressora uma cópia da tela de texto (Screens 0 ou 1), nos micros Hitachi MB-H2 e MB-H3.

**HELP** (declaração, DM-System2 BASIC, Hangul-BASIC 4, MSX Aid BASIC, Network BASIC, RMSX BASIC, RookieDrive B.)

**Formato:** CALL HELP

**Função:** Mostra ajuda no BASIC em uso.

**HIRO** (comando, MSX turbo R modelo FS-A1ST).

**Formato:** CALL HIRO

**Função:** Chama o menu para os programas em ROM no MSX turbo R modelo FS-A1ST. Para o FS-A1GT, use CALL MWR.

**HMMM** (declaração, DM-System2 BASIC)

**Formato:** CALL HMMM (<X0>,<Y0>) – [STEP](<X1>,<Y1>) TO (<X2>,<Y2>)

**Função:** Executa o comando HMMM (cópia rápida em bytes) do VDP. Disponível para as Screens 5 a 12.

<X0> – Coordenada X do primeiro ponto da área fonte.

<Y0> – Coordenada Y do primeiro ponto da área fonte.

<X1> – Coordenada X do segundo ponto da área fonte.

<Y1> – Coordenada Y do segundo ponto da área fonte.

<X2> – Coordenada X esquerda da área destino.

<Y2> – Coordenada Y superior da área destino.

STEP, se especificado, indica coordenadas relativas.

Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.

**HMMV** (declaração, DM-System2 BASIC)

**Formato:** CALL HMMV (<X0>,<Y0>) – [STEP](<X1>,<Y1>) <byte>

**Função:** Executa o comando HMMV (pintura rápida da VRAM) do VDP. Disponível para as Screens 5 a 12.

<X0> – Coordenada X do primeiro ponto da área.  
 <Y0> – Coordenada Y do primeiro ponto da área.  
 <X1> – Coordenada X do segundo ponto da área.  
 <Y1> – Coordenada Y do segundo ponto da área.  
 <byte> – Byte a ser enviado para a VRAM. Especifica um ponto para Screens 8 a 12, dois pontos para Screens 5 e 7 e quatro pontos para Screen 6.  
 STEP, se especificado, indica coordenadas relativas.  
 Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.

#### **HZ** (comando, RMSX-BASIC)

Formato: CALL HZ [50 | 60]  
 Função: Seleciona a taxa de atualização da tela (frequência VDP).  
 Sem parâmetro alterna as taxas.  
 50 – O VDP terá a frequência de 50 Hz (MSX europeu, russo ou árabe).  
 60 – O VDP terá a frequência de 60 Hz (MSX japonês, coreano ou brasileiro).

#### **IDTRACE** (comando, Hitachi-BASIC versão 2)

Formato: CALL IDTRACE  
 Função: Coloca o leitor de dados embutido do micro Hitachi MB-H2 no modo de rastreamento de ID para verificar se a fita correta foi inserida no leitor.

#### **IMPOSE** (comando, Pioneer-BASIC)

Formato: CALL IMPOSE (<modo>)  
 Função: Seleciona o modo de vídeo.  
 <modo> pode ser:  
 0 – Tela do computador (sincronização interna)  
 1 – Superimpose (vídeo composto)  
 2 – Vídeo externo

#### **INIMDM** (comando, New Modem BASIC)

Formato: CALL INIMDM (<variável>)  
 Função: Inicializa a velocidade do modem, cujo valor é armazenado em uma variável, conforme a tabela abaixo:

Valor	Norma	Veloc. recep.	Veloc. transm.	Nota
0	V21	300 baud	300 baud	Chamador
1	V21	300 baud	300 baud	Receptor
2	V23	1200 baud	75 baud	-
3	V23	75 baud	1200 baud	-
4	V23	1200 baud	75 baud	p/ conexão ruim
5	V23	75 baud	1200 baud	p/ conexão ruim
6	V23	600 baud	75 baud	-
7	V23	75 baud	600 baud	-

**INIT** (comando, SFG-BASIC)

Formato: CALL INIT

Função: Inicializa o FM Music Macro.

**INITMD** (comando, New Modem BASIC)

Formato: CALL INITMD

Função: Inicializa o protocolo de comunicação X8N1.

X = protocolo Xon / Xoff ativado

8 = 8 bits de dados

N = sem paridade

1 = 1 bit de parada

**INMK** (função, MSX-Audio)

Formato: CALL INMK [ ( [&lt;variável 1&gt;] [, [&lt;variável 2&gt;] ] [, &lt;variável 3&gt;] ) ]

Função: Informa alterações durante o uso do teclado musical.

&lt;variável 1&gt; – Número da tecla (0 a 127)

&lt;variável 2&gt; – Estado da tecla (0 se pressionada, caso contrário será 1)

&lt;variável 3&gt; – Frequência do ADPCM correspondente à tecla pressionada.

**INMKEY** (função, SFG-BASIC)

Formato: CALL INMKEY (&lt;variável&gt;)

Função: Verifica se alguma tecla do teclado musical está sendo pressionada. &lt;variável&gt; retorna o código da tecla pressionada. Se for 0, nenhuma tecla está sendo pressionada. Versão curta: \_INMK.

**INSERTDISK** (comando, RookieDrive-BASIC)

Formato: CALL INSERTDISK ("<nome do disco>")

Função: Insere uma nova imagem de disco na unidade virtual USB.

Atualmente, está limitado a imagens de disco com 720 Kbytes no máximo.

**INST** (declaração, SFG-BASIC)

Formato: CALL INST (<nº instrumento> [<nº de vozes>] [,<MIDI>]  
[,<canal da MIDI>])

Função: Define os instrumentos a serem usados pelo FM Music Macro.

Até 4 instrumentos podem ser definidos por este comando.

<nº instrumento> pode variar de 1 até 4.

<nº de vozes> especifica o número de vozes simultâneas usadas pelo instrumento. Pode variar de 1 a 8.

<MIDI> especifica se os dados serão enviados para a interface MIDI. “MIDI ON” usa a interface MIDI e “MIDI OFF” não (padrão).

<canal da MIDI> pode variar de 1 a 16. Se omitido, será usado o canal 1.

**INTWAIT** (comando, DM-System2 BASIC)

Formato: CALL INTWAIT

Função: Pausa o sistema até a próxima interrupção do DM System2. Pode ser abortada por CTRL+STOP.

**IOSOUND** (comando, RMSX-BASIC)

Formato: CALL IOSOUND [ON] | [OFF]

Função: Ativa ou desativa sons de cassetes e discos emulados.

ON – Ativa todos os sons de I/O.

OFF – Desativa todos os sons de I/O.

**JIS** (declaração, Kanji-BASIC)

Formato: CALL JIS (<variável string>,<cadeia de caracteres>)

Função: Converte o primeiro caractere de uma cadeia em um código JIS hexadecimal de 4 dígitos.

<variável string> recebe o código hexadecimal

<cadeia de caracteres> contém os caracteres a serem convertidos.

**KACNV** (declaração, Kanji-BASIC)

Formato: CALL KACNV (<variável string>, <cadeia de caracteres>)  
 Função: Converte caracteres Kanji de dois bytes em caracteres de um byte.  
           <variável string> recebe os caracteres convertidos  
           <cadeia caracteres> contém os caracteres Kanji a converter.

**KANJI** (comando, Kanji-BASIC)

Formato: CALL KANJI [<n>]  
 Função: Ativa o modo Kanji. <n> pode variar de 0 a 3, mas os modos 1 a 3 só funcionam em um MSX2 ou superior.  
           Quando no modo Kanji, pressione CTRL+ESPAÇO ou GRAPH+SELECT ativar o modo de entrada Kanji.  
           0 – 13 linhas de 32 ou 64 caracteres (16x16, 8x16)  
           1 – 13 linhas de 40 ou 80 caracteres (12x16 ou 6x16)  
           2 – 24 linhas de 32 ou 64 caracteres no modo entrelaçado (16x16, 8x16)  
           3 – 24 linhas de 40 ou 80 caracteres no modo entrelaçado (16x16, 8x16)  
 Obs.: No modo Kanji, os comandos CLS, COLOR= e SCREEN 9 são desabilitados.

**KBOLD** (declaração, DM-System2 BASIC)

Formato: CALL KBOLD ([<largura>] [,<altura>] [,<aresta X>]  
                          [,<aresta Y>] [,<sombra X>] [,<sombra Y>])  
 Função: Define o estilo dos caracteres de texto. (Requer driver FNT).  
           <largura> do caractere (1 a 16, padrão é 1)  
           <altura> do caractere (1 a 16, padrão é 1)  
           <aresta X> – Espessura da borda X (1 a 8, padrão é 1)  
           <aresta Y> – Espessura da borda Y (1 a 8, padrão é 1)  
           <sombra X> – Espessura horizontal do caractere de sombra (1 a 32, padrão é 1)  
           <sombra Y> – Espessura vertical do caractere de sombra (1 a 32, padrão é 1)  
           Se a espessura da sombra for 0, esta será determinada automaticamente.

**KCHR** (função, Hangul-BASIC 3)

Formato: CALL KCHR (<variável string>, <código hexadecimal>)  
 Função: Retorna em <variável string> o caractere coreano especificado pelo <código hexadecimal> de 4 dígitos.

**KCODE** (função, Hangul-BASIC 3)

Formato: CALL KCHR (<variável string>, <string>)

Função: Retorna em <variável string> o código hexadecimal de 4 dígitos do primeiro caractere coreano da <string>.

**KCOLOR** (declaração, DM-System2 BASIC)

Formato: CALL KCOLOR ([<cor do caractere>] [,<cor do fundo>]  
[,<cor da borda>] [,<cor da sombra>])

Função: Define as cores caracteres de texto. (Requer driver FNT).

<cor do caractere> pode variar de 0 a 15 (padrão: 15).

<cor do fundo> pode variar de 0 a 15 (padrão: 0).

<cor da borda> funciona apenas para a função “borda”.

Pode variar de 0 a 15 e o padrão é 1.

<cor da sombra> funciona apenas para a função “sombra”.

Pode variar de 0 a 15 e o padrão é 14.

**KEXT** (função, Kanji-BASIC, Hangul-BASIC 3)

Formato: CALL KEXT (<variável string>, <cadeia de caracteres>,  
<função>)

Função: Extrai apenas caracteres de 2 bytes ou de 1 byte de uma string.

<variável string> recebe os caracteres extraídos.

<cadeia de caracteres> contém os caracteres a serem extraídos.

<função> – Se for 0, apenas caracteres de um byte serão extraídos para o Kanji-BASIC ou caracteres não coreanos para o Hangul-BASIC; se for 1, apenas caracteres de 2 bytes serão extraídos para o Kanji-BASIC ou caracteres coreanos para o Hangul-BASIC.

**KEY ON/OFF** (função, MSX-Audio)

Formato: CALL KEY ON (<número de tecla>, <velocidade>)  
CALL KEY OFF (<número da tecla>)

Função: Informa se a tecla está pressionada ou liberada independente da condição real da mesma.

<número de tecla> pode variar de 0 a 127.

<velocidade> pode variar de 0 a 15 (padrão: 8).

**KINIT** (declaração, DM-System2 BASIC)

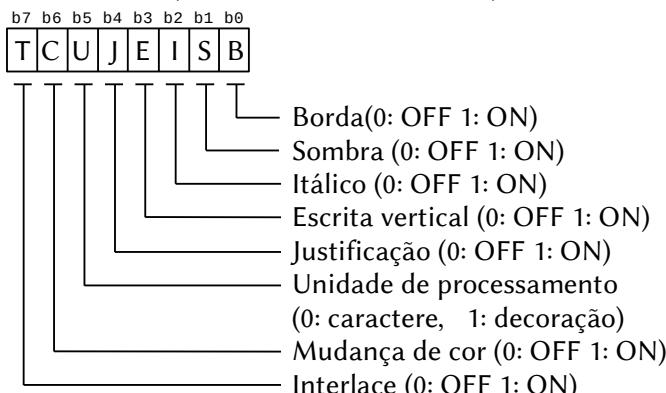
Formato: CALL KINIT

**Função:** Retorna todas as definições de texto, inclusive Kbold e Ksize, aos seus valores padrão. (Requer driver FNT).

Formato: CALL KINIT ([<configuração>][,<sombra X>][,<sombra Y>]  
[,<ítalo>] [,<tabela de cores>])

**Função:** Define todas as opções de decoração do caractere.

<configuração> é um valor de um byte, com os seguintes sinalizadores (o valor inicial de todos é 0):



**<sombra X>** e **<sombra Y>** definem a posição da sombra em relação ao canto superior esquerdo do caractere, podendo variar entre -128 e 127.

**<itálico>** define o deslocamento à direita de cada linha do caractere, incluindo a borda, podendo variar entre -128 e 127. Se omitido, serão usados os valores do sistema.

<tabela de cores> define as cores para cada linha do caractere a partir da superior, incluindo a borda. Se omitida, será usada a tabela de cores do sistema. O valor padrão é C000H.

**KINSTR** (função, Kanji-BASIC, Hangul-BASIC 3)

Formato: CALL KINSTR (<variável numérica> [<início busca>],

<string 1>, <string 2>)

**Função:** Procura a ocorrência da <string 2> na <string 1> e retorna a posição na <variável numérica>. Se não houver ocorrências, retornará 0. <índice busca> é um valor opcional e indica a posição do caractere inicial para a busca.

**KLEN** (função, Kanji-BASIC, Hangul-BASIC 3)

Formato: CALL KLEN (<variável numérica>, <cadeia de caracteres>, <função>) [Kanji-BASIC]

Formato: CALL KLEN (<variável numérica>, <cadeia de caracteres>) [Hangul-BASIC]

Função: Retorna em <variável numérica> o tamanho da <cadeia de caracteres>. Se <função> for 0 ou omitida, retorna o número total de caracteres; se for 1, retorna a quantidade de caracteres de 1 byte e se for 2 retorna a quantidade de caracteres de 2 bytes. Hangul-BASIC não permite o parâmetro <função>.

**KMID** (função, Kanji-BASIC, Hangul BASIC 3)

Formato: CALL KMID (<variável string>, <cadeia de caracteres>, <deslocamento> [, <tamanho>])

Função: Extrai <tamanho> caracteres a partir da posição <deslocamento> da <cadeia de caracteres> e coloca em <variável string>.

**KNJ** (declaração, Kanji-BASIC)

Formato: CALL KNJ (<variável string>, <cadeia de caracteres>)

Função: Atribui à <variável string> um caractere kanji equivalente ao código kanji hexadecimal de 4 dígitos especificado em <cadeia de caracteres>. Quando o código kanji for menor que 8000H, será considerado como JIS; quando for maior será considerado como shift JIS.

**KPRINT** (declaração, DM-System2 BASIC)

Formato: CALL KPRINT = (<cadeia de caracteres>, [<caractere limite>]) [, <código operação lógica>]

Função: Imprime uma string kanji na tela.

**KPUT** (declaração, DM-System2 BASIC)

Formato: CALL KPUT (<string> [,<número de caracteres>])

Função: Exibe uma cadeia de caracteres em alta velocidade. (Requer driver FNT).

<string> é a cadeia de caracteres a ser exibida.

<número de caracteres> é o número máximo de caracteres exibidos. Se omitida, todos os caracteres são exibidos.

**KPUT** (declaração, DM-System2 BASIC)

Formato: CALL KPUT (<string> [,<número de caracteres>])

Função: Exibe uma cadeia de caracteres em alta velocidade.  
(Requer driver FNT).

**KSIZE** (declaração, DM-System2 BASIC)

Formato: CALL KSIZE (<largura>,<altura>[,<separação>])

Função: Define o tamanho dos caracteres de um byte. (Requer driver FNT).

<largura> pode variar de 1 a 32 (padrão: 8).

<altura> pode variar de 1 a 64 (padrão: 16).

<separação> define o espaço entre caracteres e pode variar de 0 a 15 (padrão: 0).

**KTYPE** (função, Kanji-BASIC, Hangul-BASIC 3)

Formato: CALL KTYPE (<variável numérica>, <cadeia de caracteres>, <posição do caractere>)

Função: Retorna na <variável numérica> o valor 0 se o caractere correspondente à <posição do caractere> na <cadeia de caracteres> for de um byte e o valor 1 se o caractere for de 2 bytes.

**LB** (comando, New Modem BASIC)

Formato: CALL LB (<variável string>)[;]

Função: Envia um texto para a tela e / ou a linha telefônica de acordo com a tabela a seguir:

Eco	Linha	Tela	Telefone
-----	-------	------	----------

CALL ECHOON	CALL LINEON	Sim	Sim
-------------	-------------	-----	-----

CALL ECHOON	CALL LINEOFF	Sim	Não
-------------	--------------	-----	-----

CALL ECHOFF	CALL LINEON	Não	Sim
-------------	-------------	-----	-----

O texto pode ser pausado com CTRL+S mas não pode ser interrompido.

**LCOPY** (comando, Printer-BASIC ou Pioneer-BASIC)

Formato: CALL LCOPY [Printer-BASIC]

Função: Imprime os dados que ainda estão no buffer temporário de 32 Kbytes quando o spooler de impressão é usado.

Formato: CALL LCOPY (<modo>) [Pioneer-BASIC]

Envia para a impressora uma cópia da tela em Screen 2.

Se <modo> for 0, faz uma cópia positiva e se for 1 faz uma cópia negativa.

**LD** (comando, Pioneer-BASIC)

Formato: CALL LD

Função: Executa o software interativo presente em um disco CPE (Computer Program Encoded).

**LEN** (função, New Modem BASIC)

Formato: CALL LEN (&lt;variável string&gt;), &lt;variável numérica&gt;

Função: Retorna o comprimento de uma string, sem os caracteres de controle finais (espaço, tabulação, retorno, etc.). A &lt;variável numérica&gt; retornará o comprimento em caracteres imprimíveis da &lt;variável string&gt;.

**LENGTH** (função, SFG-BASIC)

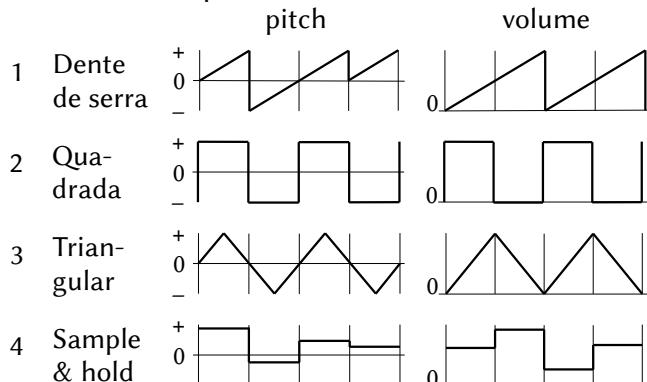
Formato: CALL LENGTH ([&lt;trilha 1&gt;] [,&lt;trilha 2&gt;] ... [,&lt;trilha 8&gt;])

Função: Retorna o tamanho dos dados em uma trilha musical. As unidades retornadas correspondem a 1/192 de uma Nota inteira. &lt;trilha 1&gt; a &lt;trilha 8&gt; são variáveis numéricas.

**LFO** (declaração, SFG-BASIC)Formato: CALL LFO (<nº forma de onda> [,<velocidade>]  
[,<trêmolo>] [,<vibrato>])

Função: Define os dados do LFO (Low Frequency Oscilator).

&lt;nº forma de onda&gt; pode variar de 1 a 4:



\* Os valores “Sample &amp; hold” são aleatórios.

&lt;velocidade&gt; especifica a frequência do LFO em relação ao volume. Pode variar de 1 a 100. Quanto maior, mais alta a frequência e a velocidade.

<trêmolo> especifica a modulação em relação ao volume.

Pode variar de 1 a 100. Quanto maior, mais o volume será alterado.

<vibrato> especifica a modulação em relação à frequência (pitch). Pode variar de 1 a 100. Quanto maior, mais o pitch será alterado.

#### **LICENSE** (declaração, RMSX-BASIC)

Formato: CALL LICENSE

Função: Exibe informações de licença sobre a versão usada do emulador rMSX, desenvolvido pelo finlandês NYYRIKKI para computadores Turbo R.

#### **LINEOFF** (comando, New Modem BASIC)

Formato: CALL LINEOFF

Função: Desliga a linha telefônica mas mantém a conexão ativa.

#### **LINEON** (comando, New Modem BASIC)

Formato: CALL LINEON

Função: Liga a linha telefônica.

#### **LMMM** (declaração, DM-System2 BASIC)

Formato: CALL LMMM (<X0>,<Y0>)-[STEP](<X1>,<Y1>) TO  
<X2>,<Y2> [,<operador lógico>]

Função: Executa o comando LMMM (cópia lógica em pontos) do VDP. Disponível para as Screens 5 a 12.

<X0> – Coordenada X do primeiro ponto da área fonte.

<Y0> – Coordenada Y do primeiro ponto da área fonte.

<X1> – Coordenada X do segundo ponto da área fonte.

<Y1> – Coordenada Y do segundo ponto da área fonte.

<X2> – Coordenada X esquerda da área destino.

<Y2> – Coordenada Y superior da área destino.

STEP, se especificado, indica coordenadas relativas.

Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.

<operador lógico> pode ser [T]PSET, [T]PRESET, [T]XOR, [T]OR ou [T]AND. O padrão é PSET.

#### **LMMV** (declaração, DM-System2 BASIC)

Formato: CALL LMMV (<X0>,<Y0>)-[STEP](<X1>,<Y1>), <cor>  
[,<operador lógico>]

**Função:** Executa o comando LMMV (cópia lógica em pontos do VDP para a VRAM). Disponível para as Screens 5 a 12.  
 <X0> – Coordenada X do primeiro ponto da área destino.  
 <Y0> – Coordenada Y do primeiro ponto da área destino.  
 <X1> – Coordenada X do segundo ponto da área destino.  
 <Y1> – Coordenada Y do segundo ponto da área destino.  
 STEP, se especificado, indica coordenadas relativas.  
 Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.  
 <cor> especifica a cor do retângulo a ser pintado.  
 <operador lógico> pode ser [T]PSET, [T]PRESET, [T]XOR,  
 [T]OR ou [T]AND. O padrão é PSET.

## **LO** (comando, New Modem BASIC)

**Formato:** CALL LO (<variável string>)[:]<variável>

**Função:** Envia um texto para a tela e / ou a linha telefônica de acordo com a tabela a seguir:

Eco	Linha	Tela	Telefone
CALL ECHOON	CALL LINEON	Sim	Sim
CALL ECHOON	CALL LINEOFF	Sim	Não
CALL ECHOOFF	CALL LINEON	Não	Sim

Eventuais caracteres recebidos são ignorados, exceto as teclas para pausar, abortar e continuar. Os buffers das teclas permanecem vazios; nenhuma tecla é armazenada. Os códigos de controle finais são enviados com o texto. O parâmetro <variável> retorna o resultado da execução:

0 = O texto foi enviado corretamente

3 = a operação foi abortada com CTRL+C ou C

## **LOAD** (comando, DM-System2 BASIC, QuickDisk BASIC)

**Formato:** CALL LOAD ("[<dispositivo>:]<caminho>[\\""  
 <nome do arquivo>","[@]<endereço destino>  
 [<tamanho>][,<offset>]) [DM-System2 BASIC]

**Função:** Lê um arquivo ou parte dele.

<dispositivo> pode ser drive A: a H: ou COM: para micros conectados com RS232C.

<caminho> especifica o local da pasta ou arquivo.

<nome do arquivo> é o arquivo a ser lido.

<endereço destino> é o endereço de destino dos dados. Se precedido por “@” significa VRAM.

<tamanho> especifica a quantidade de bytes a ler.

<offset> especifica o deslocamento no arquivo origem.

Formato: CALL LOAD ("[QD[n]:]<nome do arquivo>"[,R])  
[QuickDisk-BASIC]

Função: Carrega um arquivo não binário do dispositivo Quick Disk especificado. Pode ser um arquivo BASIC em modo tokenizado ou em texto ASCII.

QD[n] especifica o dispositivo QuickDisk que será usado.  
Pode variar de 0 a 7, sendo que o padrão é 0.

<nome do arquivo> deve estar no formato 8.3 caracteres.

[,R], se especificado, executa o arquivo BASIC logo após o carregamento.

### **LOAD PCM** (comando, MSX-Audio)

Formato: CALL LOAD PCM (<“nome arquivo”>, <número arq>)

Função: Carrega dados ADPCM e PCM do disco.

<nome arquivo> – Nome do arquivo no disco.

<número arq> – Número do arquivo na memória de áudio.  
Pode variar de 0 a 15.

### **LOADROM** (comando, RookieDrive-BASIC)

Formato: CALL LOADROM ("<nome do arquivo>")

Função: Carrega um arquivo ROM de 8kb, 16kb ou 32kb na RAM e inicia sua execução reinicializando o computador. O arquivo ROM deve estar localizado no diretório raiz do dispositivo USB. <nome do arquivo> deve estar no formato 8.3.

### **LOCKDRV** (comando, Nextor)

Formato: CALL LOCKDRV (<letra de drive>: N)

Função: Bloqueia ou desbloqueia letras de drive, ou apresenta a lista de drives bloqueados. Se “N” for 0, desbloqueia o drive; qualquer outro número bloqueia.

### **LOGFILE** (comando, New Modem BASIC)

Formato: CALL LOGFILE (<nome do arquivo>)[],<variável>

Função: Armazena em um arquivo de texto tudo o que está na tela. <nome do arquivo> deve ser uma variável string.

**LOOK** (função, SFG-BASIC)

Formato: CALL LOOK ([<instrumento 1>] [,<instrumento 2>]  
[,<instrumento 3>] [,<instrumento 4>])

Função: Retorna o estado do instrumento, se está sendo tocado ou não. Os parâmetros <instrumento x> são variáveis numéricas. Se o instrumento não foi definido por \_INST, retorna 0.

**LS** (comando, New Modem BASIC)

Formato: CALL LS (<variável string>)[;],<variável>

Função: Envia um texto para a tela e / ou a linha telefônica de acordo com a tabela a seguir:

Eco	Linha	Tela	Telefone
CALL ECHOON	CALL LINEON	Sim	Sim
CALL ECHOON	CALL LINEOFF	Sim	Não
CALL ECHOOFF	CALL LINEON	Não	Sim

Eventuais caracteres recebidos são ignorados, exceto as teclas para pausar, abortar e continuar. Os buffers das teclas permanecem vazios; nenhuma tecla é armazenada. Os códigos de controle finais são enviados com o texto. O parâmetro <variável> retorna o resultado da execução:  
 0 = O texto foi enviado corretamente  
 3 = a operação foi abortada com CTRL+C ou C

**MALLOC** (comando, DM-System2 BASIC)

Formato: CALL MALLOC ([<número de páginas>][,<variável>])

Função: Habilita ao acesso à Memória Mapeada.

<número de páginas> é número de páginas a serem alocadas. Se for 0, as páginas alocadas serão liberadas. Se omitido, o número atual de páginas alocadas retornará em <variável>.

<variável> é uma variável numérica que conterá o número de páginas efetivamente alocadas.

**MAPDRV** (comando, Nextor)

Formato: CALL MAPDRV (<drive> [, <partição> [, <dispositivo>  
[,<slot>|0]]])

Função: Mapeia uma unidade de drive no sistema Nextor.  
 <drive> letra ou número da unidade a ser mapeada

<partição> 0 – A unidade será mapeada a partir do setor zero absoluto do dispositivo.

1 – primeira partição primária.

2 a 4 – referem a partições estendidas 2.1 a 2.4 se a partição 2 for estendida; caso contrário se referem a partições primárias.

5 em diante referem a partições estendidas.

<dispositivo> – Índice do dispositivo (1 a 7)

<slot> – Número do slot (0 a 3). Se o slot for expandido, use a fórmula <slot principal> + 4 \* <subslot>. Se “0” for especificado, o slot da unidade primária será selecionado.

### **MAPDRV<sub>L</sub>** (comando, Nextor)

Formato: CALL MAPDRV<sub>L</sub> (<drive> [, <partição> [, <dispositivo> [, <slot>|0]]])

Função: Mapeia uma unidade de drive no sistema Nextor e bloqueia o drive especificado. Os parâmetros são idênticos a MAPDRV.

### **MC** (declaração, New Modem BASIC)

Formato: CALL MC (<variável string>)

Função: Converte os caracteres alfabéticos da <variável string> para maiúsculas.

### **MDR** (comando, MSX turbo R modelo FS-A1GT)

Formato: CALL MDR

Função: Ativa a saída do MSX-MUSIC para a interface MIDI. Somente MSX turbo R modelo FS-A1GT.

### **MEMINI** (comando, 2)

Formato: CALL MEMINI [(tamanho da RAM disk)]

Função: Ativa a RAM disk nos 32K inferiores de memória.

### **MERGE** (comando, QuickDisk BASIC)

Formato: CALL MERGE ("[QD[n]:]<nome do arquivo>")

Função: Mescla um programa BASIC ou DATA salvo em ASCII no dispositivo QuickDisk com o programa que está na memória do MSX.

QD[n] especifica o dispositivo QuickDisk que será usado.

Pode variar de 0 a 7, sendo que o padrão é 0.

<nome do arquivo> deve estar no formato 8.3 caracteres.

#### **MESSAGE** (declaração, MSX-Aid BASIC, Network BASIC)

Formato: CALL MESSAGE [MSX-Aid BASIC]

Função: Exibe uma mensagem encorajadora para os programadores que usam o MSX-Aid.

Formato: CALL MESSAGE (<mensagem>, [<número do aluno>]) [Network BASIC]

Função: Envia uma mensagem de até 56 caracteres para um aluno específico. Esta instrução está disponível apenas para o professor. <número do aluno> pode variar de 1 a 15. Pode ser usada a versão curta: \_MESS.

#### **MFADE** (declaração, StudioFM BASIC)

Formato: CALL MFADE (<grau de fade>)

Função: Produz um fade-out ao interromper a reprodução de arquivos .MUS no Studio FM. <grau de fade> pode variar entre 0 e 255, sendo que 0 não há fade e 255 produzirá o fade-out mais longo.

#### **MFILES** (comando, 2)

Formato: CALL MFILES

Função: Lista os arquivos da RAM disk dos 32K inferiores de memória.

#### **MK PCM** (declaração, MSX-Audio)

Formato: CALL MK PCM (<número arq>)

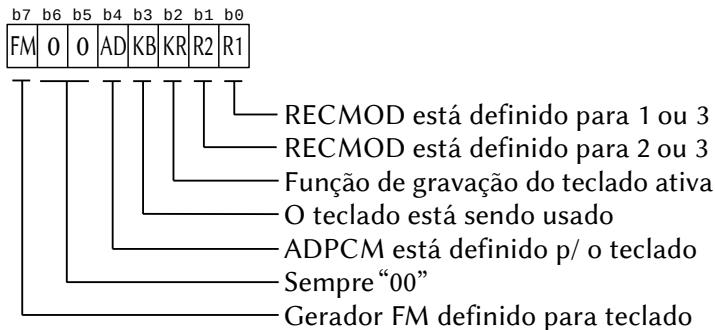
CALL MK PCM OFF

Função: Define qual arquivo ADPCM será tocado como instrumento. Se OFF, cancela o instrumento previamente definido. <número arq> pode variar de 0 a 15.

#### **MK STAT** (função, MSX-Audio)

Formato: CALL MK STAT (<variável>)

Função: Retorna o estado de gravação/reprodução do teclado musical. <variável> é um valor numérico definido de acordo com a figura abaixo:

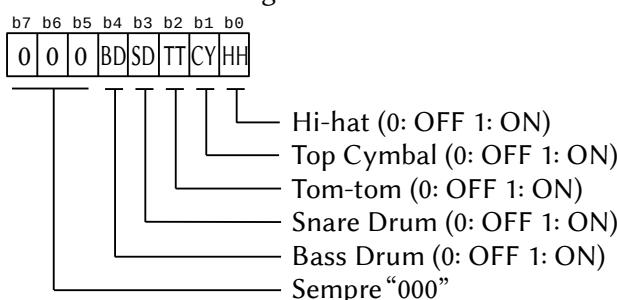
**MK TEMPO** (declaração, MSX-Audio)

Formato: CALL MK TEMPO (<velocidade>, <mapa de percussão>)

Função: Especifica a velocidade de gravação/reprodução do teclado musical ou ativa a função metrônomo. Neste caso, é necessário a definição prévia do comando AUDIO. Este comando afeta a velocidade das instruções MK PLAY, MK REC e MK APPEND.

<velocidade> deve estar no intervalo 25 ~ 360, sendo que o valor inicial é 120.

<mapa de percussão> é um valor numérico definido de acordo com a figura abaixo.

**MK VEL** (declaração, MSX-Audio)

Formato: CALL MK VEL (<velocidade>)

Função: Especifica a velocidade, ou força de pressão, que é aplicada sobre uma tecla do teclado musical.

<velocidade> pode variar de 0 a 15, sendo que o valor inicial é 8.

**MK VOICE** (declaração, MSX-Audio)

Formato: CALL MK VOICE (<parâmetros>)

Função: Define o instrumento a ser associado ao teclado musical.  
 <parâmetros> é o símbolo de @ seguido de uma variável simples que define o número do instrumento, podendo variar de 0 a 63. Se não houver @, a variável será assumida como sendo uma matriz, onde os valores em sequência definem o instrumento.

**MK VOL** (declaração, MSX-Audio)

Formato: CALL MK VOL (<volume>)

Função: Define o volume associado ao teclado musical.  
 <volume> pode variar de 0 a 63.

**Mkdir** (comando, Disk-BASIC 2nd version)

Formato: CALL MKDIR (<subdiretório>)

Função: Cria o <subdiretório> com o nome especificado.

**MKill** (comando, 2)

Formato: CALL MKILL (“<nome do arquivo>”)

Função: Apaga o arquivo <nome do arquivo> da RAM disk dos 32K inferiores de memória.

**MLOAD** (comando, StudioFM BASIC)

Formato: CALL MLOAD (“<nome do arquivo>”, <endereço>)

Função: Carrega uma música no formato Studio FM (.MUS).

**MNAME** (comando, 2)

Formato: CALL MNAME (“<nome arquivo 1>” AS “<nome arquivo 2>”)

Função: Renomeia o arquivo <nome arquivo 1> com <nome arquivo 2> na RAM disk dos 32K inferiores de memória.

**MODE9** (comando, Hangul-BASIC 4)

Formato: CALL MODE9

Função: Vai para Screen 9 usando automaticamente Width 80.

**MODINST** (declaração, SFG-BASIC)

Formato: CALL MODINST (<instrumento> [<voz>] [<transposição>]  
 [<volume>] [<portamento>] [<veloc portamento>]  
 [<sustentação>] [<modo disparo>] [<LFO sync>]  
 [<trêmolo>] [<vibrato>] )

- Função:** Altera os dados dos instrumentos. Versão curta: \_MODI.
- <instrumento> especifica o tom do instrumento definido por \_INST.
- <voz> especifica o tom do instrumento.
- 1 ~ 48 – Seleciona tom da ROM do FM Sound Synthesizer. 47 e 48 são reservados e não contêm tons.
  - 49 ~56 – Seleciona tons definidos por \_SEL.
- <transposição> permite que os vários instrumentos sejam transpostos separadamente. Pode ir de -12 a +12 em intervalos de meio-passo.
- <volume> define o volume separadamente para cada instrumento. Pode variar de 0 a 100, sendo 100 o volume máximo.
- <portamento> pode assumir dois valores:
- 0 – portamento apenas durante a reprodução
  - 1 – portamento todo o tempo
- <veloc portamento> pode variar de 0 a 100, sendo 100 o mais lento. 0 desliga o portamento.
- <sustentação> pode assumir dois valores:
- 0 – Tempo de sustentação padrão
  - 1 – o tempo da sustentação é dobrado
- <modo disparo> determina se haverá disparo quando as teclas pressionadas forem soltas.
- 0 – “attack” quando as teclas forem soltas
  - 1 – Sem “attack” durante a reprodução legada
- <LFO sync> determina se haverá sincronização com o LFO quando as teclas pressionadas forem soltas.
- 0 – Sem sincronização
  - 1 – o LFO inicia a forma de onda sempre que uma tecla pressionada for solta.
- <trêmolo> especifica o grau do trêmolo. Pode variar de 0 a 100, sendo 100 o maior grau de sensibilidade. (apesar da faixa 0 a 100, há apenas 4 graus de trêmolo).
- <vibrato> especifica o grau do vibrato. Pode variar de 0 a 100, sendo 100 o maior grau de sensibilidade. (apesar da faixa 0 a 100, há apenas 8 graus de vibrato).

**MON** (comando, FM-X BASIC, Hitachi BASIC, MSXaid BASIC)  
**Formato:** CALL MON [FM-X BASIC]

- Função: Inicia o monitor quando o micro Fujitsu FM-X está conectado à máquina FM-7 com a interface MB22 450.
- Formato: CALL MON [Hitachi BASIC 1-2]
- Função: Inicia a linha de comando do System Monitor Utility nos micros Hitachi MB-H1 e MB-H2. Para obter a lista de todos os comandos disponíveis neste utilitário, digite H na linha de comandos.
- Formato: CALL MON [MSXAid BASIC]
- Função: Inicia o monitor interno do utilitário MSX-Aid. Para ajuda, insira um endereço da RAM e depois pressione F6.

#### **MOUNT** (comando, RookieDrive-BASIC)

Formato: ?

Função: ?

#### **MPLAY** (declaração, StudioFM BASIC)

Formato: CALL MPLAY (<endereço da memória>)

Função: Reproduz uma música no formato StudioFM (.MUS) carregada na memória com \_MLOAD.

#### **MRING** (comando, New Modem BASIC)

Formato: CALL MRING (<variável numérica>)

Função: Verifica se o telefone está tocando. <variável numérica> pode retornar os seguintes valores:  
 0 – o telefone está tocando.  
 1 – o telefone não está tocando.  
 2 – Rotina interrompida (tecla CODE pressionada).

#### **MSTART** (comando, New Modem BASIC)

Formato: CALL MSTART (<variável numérica>)

Função: Prepara o modem para comunicação de dados. Se <variável numérica> retornar 0, está tudo certo; caso contrário houve erro.

#### **MSTOP** (comando, New Modem BASIC, StudioFM BASIC)

Formato: CALL MSTOP [New Modem BASIC]

Função: Interrompe as funções do modem.

Formato: CALL MSTOP [StudioFM BASIC]

Função: Interrompe a reprodução de uma música no formato StudioFM (.MUS) tocada em segundo plano.

**MUSIC** (comando, MSX-Music)

**Função:** Inicia o MSX-MUSIC e determina quais vozes serão usadas e de que forma.

<n1> pode ser:

0 – Seleciona modo melodia puro (n3~n9 podem ser especificados)

1 – Seleciona modo melodia + bateria (n3~n6 podem ser especificados)

**< n3 > até < n9 > podem ser:**

### 1 – Selecciona melodía

2 – Selecciona batería

**MUTE** (comando, Hitachi-BASIC, Pioneer-BASIC, RMSX-BASIC)

Formato: CALL MUTE [Hitachi-BASIC 2]

**Função:** Adiciona uma pausa de 4 segundos antes de gravar dados no gravador interno do micro Hitachi HB-M2.

Formato: CALL MUTE [R | L] [Pioneer-BASIC]

CALL MUTE OFF

**Função:** Torna mudo os canais de áudio direito (R), esquerdo (L) ou ambos se não houver especificação de canal. Se OFF for especificado, a função de mute é cancelada.

Formato: CALL MUTE [ON] | [OFF] [RMSX-BASIC]

**Função:** Habilita ou desabilita a saída de áudio. Se o parâmetro for omitido, apenas inverte o estado.

**MWP** (comando MSX turbo R modelo ES-A1GT)

Formato: CALL MWP

**Função:** Chama o menu para os programas em ROM no MSX turbo R modelo FS-A1GT. Para o FS-A1ST, use CALL HIRO

**NET** (comando GR8NET-BASIC)

Formato: CALL NET

Função: Apresenta ajuda do GR8NET.

**NETBITOV** (comando, GR8NET-BASIC)

Formato: CALL NETBITOV(<página>,<endereço>,<banco de VRAM>,<endereço de VRAM>)

Função: Transfere a imagem do ícone da GR8NET para a VRAM.  
 <página> é o número da página lógica dos dados do ícone  
 <endereço> é o endereço dos dados do ícone só pode  
 variar de 6000H a 7FFFH.  
 <banco de VRAM> deve ser 0 para 0000H~FFFFH ou 1  
 para 1000H~1FFFFH.  
 <endereço de VRAM> é o endereço dentro do banco de  
 VRAM selecionado.

### **NETBLOAD (comando, GR8NET-BASIC)**

Formato: CALL NETBLOAD (<url>,<flag execução>,<página lógica>,  
 <endereço GR8NET>)

Função: Carregar arquivo binário do cartão SD ou do servidor da  
 web remoto usando HTTP.  
 <url> é a string URL para acesso remoto. Para a primeira  
 partição do cartão SD, use “SDC://”.  
 <flag execução> indica a ação a ser tomada para arquivos  
 executáveis.  
 0 – os dados não serão carregados (valor padrão).  
 1 – os dados serão carregados mas não executados.  
 2 – o arquivo será carregado e executado.  
 <página lógica> da GR8NET (00H a 7FH).  
 <endereço GR8NET> pode variar de 6000H a 7FFFH.

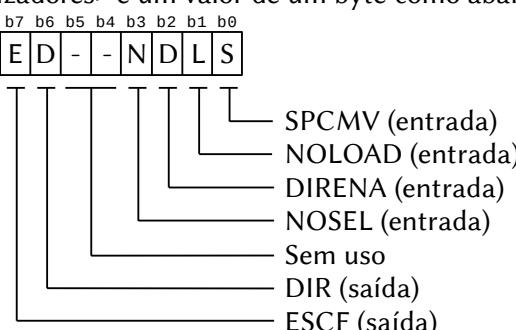
### **NETBROWSE (comando, GR8NET-BASIC)**

Formato: CALL NETBROWSE (<url>,<sinalizadores>)

Função: Chama o navegador de internet e de cartão SD.

<url> é a string URL inicial.

<sinalizadores> é um valor de um byte como abaixo:



**SPCMV:** se este bit estiver setado, quando o usuário pressionar a tecla SPACE na seleção, a url será carregada na RAM do GR8NET, mas nenhuma ação será executada e o navegador sai;

**NOLOAD:** se este bit estiver setado, o navegador não carregará a url selecionada na RAM do GR8NET.

**DIRENA:** se este bit estiver setado, ao pressionar a barra de espaço no diretório este será selecionado e o navegador será encerrado; se este bit estiver resetado, ao pressionar espaço no diretório o conteúdo será carregado e a navegação continua.

**NOSEL:** se setado, não força a página de seleção de dispositivo de origem (Internet/cartão SD), e a navegação prossegue diretamente para o dispositivo identificado pela cadeia de URL.

**DIR:** este bit retorna setado se o conteúdo for uma entrada de diretório ou uma página com lista de diretórios gerado pelo servidor WEB.

**ESCF:** este bit retorna setado quando o navegador for encerrado com a tecla ESC.

#### **NETBTOV** (comando, GR8NET-BASIC)

Formato: CALL NETBTOV (<banco VRAM>,<offset endereço>)  
 Função: Move dados binários do buffer da GR8NET para a VRAM.  
     <banco VRAM> deve ser 0 para 0000H~FFFFH ou 1 para 10000H~1FFFFH.  
     <offset endereço> é o offset do endereço especificado no cabeçalho do arquivo binário.

#### **NETCDTOF** (comando, GR8NET-BASIC)

Formato: CALL NETCDTOF  
 Função: Copia a configuração do DHCP para a configuração do endereço IP fixo .

#### **NETCFG** (comando, GR8NET-BASIC)

Formato: CALL NETCFG  
 Função: Ativa a configuração interativa da GR8NET.

**NETCODE** (função, GR8NET-BASIC)

Formato: CALL NETCODE ( < código\_erro >, [ < http\_oper > ] )

Função: Retorna o estado da última operação e o código de resposta HTTP.

**NETDHCP** (comando, GR8NET-BASIC)

Formato: CALL NETDHCP

Função: Executa busca DHCP e faz sua configuração dinâmica.

**NETDIAG** (comando, GR8NET-BASIC)

Formato: CALL NETDIAG (<V>)

Função: Liga/desliga modo diagnóstico. Se <V> for 0, desliga; caso contrário liga.

**NETDNS** (comando, GR8NET-BASIC)

Formato: CALL NETDNS [ ( [<A>], [<B>], [<C>], [<D>] ) ]

Função: Obtém o IP do domínio DNS atual. Se não houver argumentos, imprime o endereço na tela.

**NETDUMP** (comando, GR8NET-BASIC)

Formato: CALL NETDHCP

Função: Executa busca DHCP e faz sua configuração dinâmica.

**NETEND** (comando, Network-BASIC)

Formato: CALL NETEND

Função: Desabilita a rede MSX (MSX Network). Versão curta: \_NETE.

**NETEXPRT** (comando, GR8NET-BASIC)

Formato: CALL NETEXPRT

Função: Criar programa BASIC contendo dados de configuração do GR8NET.

**NETFIX** (comando, GR8NET-BASIC)

Formato: CALL NETFIX

Função: Configurar informações de endereço IP fixo da rede.

**NETKOPLLR** (comando, GR8NET-BASIC)

Formato: CALL NETKOPLLR

**Função:** Carregar a ROM do OPLL (MSX-Music) na memória mapeada. Este comando é direcionado para o software que é executado nos modos mapper 1 a 6 da GR8NET (mapper de jogos) e não pode ser executado no modo 8 quando o MSX-Music ROM está disponível no subslot 3 da GR8NET.

#### **NETFWUPDATE** (comando, GR8NET-BASIC)

**Formato:** CALL NETFWUPDATE ([<argumento>])

**Função:** Atualiza o firmware da GR8NET. Se <argumento> for omitido ou for 0, apenas exibe informações sobre o firmware atual. Se for 1, atualiza apenas o firmware principal e se for 3 (três) atualiza também a área de configuração.

#### **NETGETCLK** (função, GR8NET-BASIC)

**Formato:** CALL NETGETCLK ([<fonte>], <frequência>)

**Função:** Retorna a frequência do relógio. Se <fonte> retornar zero, a frequência será a do bus principal do MSX; se for diferente de zero, retornará a frequência do oscilador interno da GR8NET. <frequência> deverá ser uma variável numérica.

#### **NETGETCLOUD** (comando, GR8NET-BASIC)

**Formato:** CALL NETGETCLOUD

**Função:** Imprime o estado do volume virtual do GR8cloud na tela.

#### **NETGETDA** (função, GR8NET-BASIC)

**Formato:** CALL NETGETDA (<nº adaptador>, <adaptadores ativos>)

**Função:** Retorna o número do adaptador padrão em <nº adaptador> e a lista de adaptadores ativos em <adaptadores ativos>. <nº adaptador> deve ser uma variável numérica.

<adaptadores ativos> deve ser uma variável numérica onde os bits 0 a 3 receberão o estado dos adaptadores (o bit respectivo estará setado se o dispositivo estiver ativo).

#### **NETGETDNS** (comando, GR8NET-BASIC)

**Formato:** CALL NETGETDNS ([<A>], [<B>], [<C>], [<D>])

**Função:** Obtém o endereço DNS do IP fixo. [<A>], [<B>], [<C>] e [<D>] devem ser variáveis numéricas.

**NETGETGW** (função, GR8NET-BASIC)

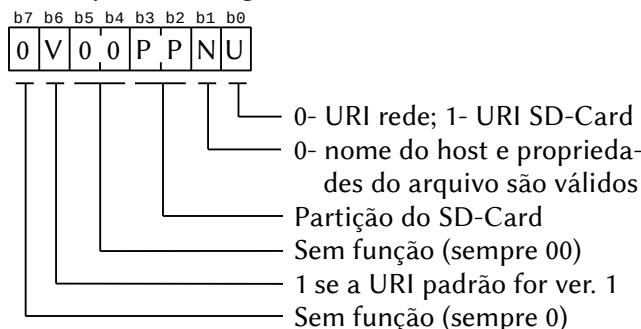
Formato: CALL NETGETGW ([<A>], [<B>], [<C>], [<D>])

Função: Obtém endereço IP fixo do gateway. [<A>], [<B>], [<C>] e [<D>] devem ser variáveis numéricas.

**NETGETHOST** (comando, GR8NET-BASIC)

Formato: CALL NETGETHOST (<sinalizador>, <nome> | <A>, <B>, <C>, <D>)

Função: Obtém o nome e endereço IP do host remoso. <A>, <B>, <C> e <D> e <sinalizador> devem ser variáveis numéricas e <nome> deve ser uma variável alfanumérica. <sinalizador> é um byte com a seguinte estrutura:

**NETGETIP** (função, GR8NET-BASIC)

Formato: CALL NETGETIP ([<A>], [<B>], [<C>], [<D>])

Função: Obtém o endereço do IP fixo. [<A>], [<B>], [<C>] e [<D>] devem ser variáveis numéricas.

**NETGETMAP** (função, GR8NET-BASIC)

Formato: CALL NETGETMAP (<sinalizadores>)

Função: Obtém o tipo de memória mapeada e outros dados. <sinalizadores> deve ser uma variável numérica de 16 bits, onde os bits 0 a 7 contêm a página lógica atual da memória mapeada e os bits 8, 13 e 14 são bits do registrador de modo do sistema.

**NETGETMASK** (função, GR8NET-BASIC)

Formato: CALL NETGETMASK ([<A>], [<B>], [<C>], [<D>])

Função: Obtém a máscara do endereço de IP fixo. [<A>], [<B>], [<C>] e [<D>] devem ser variáveis numéricas.

#### **NETGETMD** (função, GR8NET-BASIC)

Formato: CALL NETGETMD (<página lógica>, <endereço>, variável)

**Função:** Obtém uma palavra de 4 bytes (32 bits) da memória, converte e armazena em uma variável BASIC.

<página lógica> é o número da página lógica no banco 1 da GR8NET (6000-7FFF).

`<endereço>` é o endereço visível pelo Z80

<variável> é uma variável BASIC capaz de acomodar o valor lido (simples ou dupla precisão).

#### **NETGETMEM** (função, GR8NET-BASIC)

Formato: CALL NETGETMEM (<página lógica>, <endereço>, [<A>], [<B>], [<C>], [<D>])

Funcão: Lê uma sequêcia de 4 bytes na memória.

<página lógica> número da página lógica no banco 1 da GR8NET (6000H~7FFFH).

**<endereço>** é o endereço de memória visível para o Z80.

<A> – Endereço, <B> – Endereço+1, etc.

#### **NETGETMIX (função, GR8NET-BASIC)**

Formato: CALL NETGETMIX ([<número>])

**Função:** Retorna a configuração o mixer de áudio. Se o bit 15 for 0, o áudio é mono, se for 1 é estéreo. Se <número> for omitido, a configuração será impressa na tela.

**<número>** é um valor numérico de 16 bits:

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	PSG	Y8950	OPLL	Wave	SCC	PCM					

Cada 2 bits representam o seguinte:

00 – Mudo 10 – Canal direito

01 – Canal esquierdo      11 – Ambos os canais

#### **NETGETMMV** (função, GR8NET-BASIC)

Formato: CALL NETGETMMV ([<página inicial usuário>], [<topo RAM>], [<página inicial imagem disco>], [<máximo de páginas>], [<página inicial RAM Y8950>])

**Função:** Retorna a configuração do gerenciador de memória. Todos são valores numéricos e qualquer um pode ser omitido. Se todos forem omitidos, o comando imprime os valores na tela.

**NETGETNAME** (função, GR8NET-BASIC)

Formato: CALL NETGETNAME ([<nome de arquivo>])

Função: Retorna o nome de arquivo do recurso remoto.

**NETGETNTP** (função, GR8NET-BASIC)

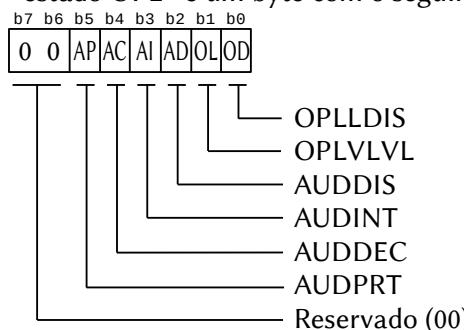
Formato: CALL NETGETNTP ([<A>], [<B>], [<C>], [<D>])

Função: Obtém as propriedades no servidor NTP na configuração de endereço de IP fixo. [<A>], [<B>], [<C>] e [<D>] devem ser variáveis numéricas.

**NETGETOPL** (função, GR8NET-BASIC)

Formato: CALL NETGETOPL (<estado OPL>, <num páginas sample RAM>, <tamanho sample RAM>)

Função: Obtém o estado do OPLL/Y8950 e o tamanho da Sample RAM.  
 <estado OPL> é um byte com o seguinte formato:



OPLLDIS – 0 – Saída OPLL está ativa (padrão).  
 1 – Saída OPLL desligada.

OPLVLVL – 0 – Volume normal OPLL/Y8950 (padrão).  
 1 – Volume duplicado OPLL/Y8950.

AUDDIS – 0 – MSX-Audio está habilitado (padrão).  
 1 – MSX-Audio desabilitado.

AUDINT – 0 – Interrup. MSX-Audio habilitada (padrão).  
 1 – Interrupções MSX-Audio desabilitada.

AUDDEC – 0 – MSX-Audio desconfigurado/indisponível.  
 1 – MSX-Audio configurado em AUDPRT.

AUDPRT – 2 – MSX-Audio configurado porta C0-C1.  
 1 – MSX-Audio configurado porta C2-C3.

<num páginas sample RAM> alocadas (8K cada).

<tamanho sample RAM> requerida em páginas de 8K.

**NETGETPATH** (função, GR8NET-BASIC)

Formato: CALL NETGETPATH ([<caminho>])

Função: Retorna o <caminho> do recurso remoto. <caminho> deve ser uma variável alfanumérica. Se for omitido, imprime o caminho na tela.

**NETGETPORT** (função, GR8NET-BASIC)

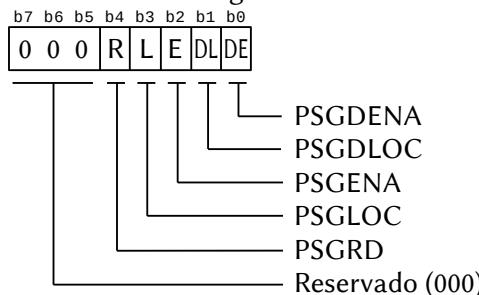
Formato: CALL NETGETPORT ([<porta remota>], [<porta local>])

Função: Retorna a <porta remota> e a <porta local> (devem ser variáveis numéricas).

**NETGETPSG** (função, GR8NET-BASIC)

Formato: CALL NETGETPSG ([<sinalizadores>])

Função: Retorna alguns dados do PSG. <sinalizadores> é um byte de dados com o seguinte formato:



Onde:

- PSGDENA e PSGDLOC são o estado inicial desejado do PSG (consulte \_NETSETPSG);
- PSGENA e PSGLOC são o estado real (1 = ativado) e o local (0 = 0xA0, 1 = 0x10) do PSG;
- Se PSGRD for 1, os registradores do PSG podem ser lidos e se for 0 o PSG está no modo somente gravação.

**NETGETQSTR** (função, GR8NET-BASIC)

Formato: CALL NETGETQSTR ([<string de consulta>])

Função: Retorna a <string de consulta> definida para o recurso remoto. Se o argumento for omitido, imprime o resultado na tela.

**NETGETTSHN** (função, GR8NET-BASIC)

Formato: CALL NETGETTSHN ([<servidor de horário>])

Função: Retorna o nome do host do <servidor de horário>. Se o argumento for omitido, imprime o resultado na tela.

**NETGW** (função, GR8NET-BASIC)

Formato: CALL NETGW ([<A>], [<B>], [<C>], [<D>])

Função: Retorna a configuração do gateway atual. <A>, <B>, <C> e <D> devem ser variáveis numéricas.

**NETHELP** (comando, GR8NET-BASIC)

Formato: CALL NETHELP <comando>

Função: Mostra ajuda para o <comando> GR8NET especificado. Se <comando> for omitido, imprime a listagem de todos os comandos disponíveis.

**NETIMPRT** (comando, GR8NET-BASIC)

Formato: CALL NETIMPRT

Função: Preencher as variáveis de sistema da GR8NET com dados do programa BASIC criado por NETEXPRT.

**NETINIT** (comando, Network-BASIC)

Formato: CALL NETINIT

Função: Inicializa a rede MSX. Usar somente após CALL NETEND, pois a rede é inicializada automaticamente ao ligar o micro. Versão curta: \_NETI

**NETIP** (função, GR8NET-BASIC)

Formato: CALL NETIP ([<A>], [<B>], [<C>], [<D>])

Função: Obtém o endereço IP do adaptador atual. <A>, <B>, <C> e <D> devem ser variáveis numéricas.

**NETLDBUF** (comando, GR8NET-BASIC)

Formato: CALL NETLDBUF (<página do adaptador>, <endereço do adaptador>, <tamanho do bloco>, <endereço da RAM>, [<tipo de mapper>])

Função: Copiar dados da memória principal para o buffer do adaptador.

**NETLDRAM** (comando, GR8NET-BASIC)

Formato: CALL NETLDRAM (<página do adaptador>, <endereço do adaptador>, <tamanho bloco>, <endereço RAM>)

Função: “Descarrega” dados do buffer do adaptador para a memória principal. <endereço adaptador> deve estar entre &H6000 e &H7FFF.

**NETMASK** (função, GR8NET-BASIC)

Formato: CALL NETMASK ([<A>], [<B>], [<C>], [<D>])

Função: Obtém a máscara de sub-rede do adaptador atual.

<A>, <B>, <C> e <D> devem ser variáveis numéricas.

**NETNTP** (função, GR8NET-BASIC)

Formato: CALL NETNTP (<A>, <B>, <C>, <D>, <TZF>)

Função: Obtém a configuração efetiva do servidor NTP. <A>, <B>, <C>, <D> e <TZF> devem ser variáveis ou constantes numéricas. Se o bit 7 de TZF estiver setado, o RTC será sincronizado com o servidor NTP. Os bits 0 a 6 representam um valor positivo ou negativo (-64 [40H] a +63 [3FH]) e definem o fuso horário em incrementos de 15 minutos.

**NETPLAYBUF[#]A** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYBUF[#]A (<página lógica>, <endereço>, <tamanho>)

Função: Define o endereço e o tamanho inicial do buffer nº [#] para o PCM. <página lógica> deve estar entre 00H~7FH, <endereço> entre 6000H~7FFFH e <tamanho> deve ser calculado para não exceder a memória de 1 MB da GR8NET. [#] deve estar entre 0 e 9.

**NETPLAYBUF[#]C** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYBUF[#]C

Função: Continuar a reprodução reabastecendo o buffer PCM nº [#], que deve estar entre 0 e 9.

**NETPLAYBUF[#]P** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYBUF[#]P (<tamanho>, <frequencia>)

Função: Iniciar a reprodução dos dados pré-armazenados no buffer PCM nº [#]. <tamanho> pode ser 8 ou 16 bits e <frequência> pode variar de 1 a 65.536. [#] deve estar entre 0 e 9. Para prevenir o esvaziamento do buffer deve ser usado o comando \_NETPLAYBUF[#]C.

**NETPLAYBUF[#]R** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYBUF[#]R

Função: Resetar o mecanismo de reprodução do buffer nº [#], que deve estar entre 0 e 9.

**NETPLAYBUF[#]S** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYBUF[#]S (<estado>)

Função: Retornar o estado da reprodução do buffer nº [#].

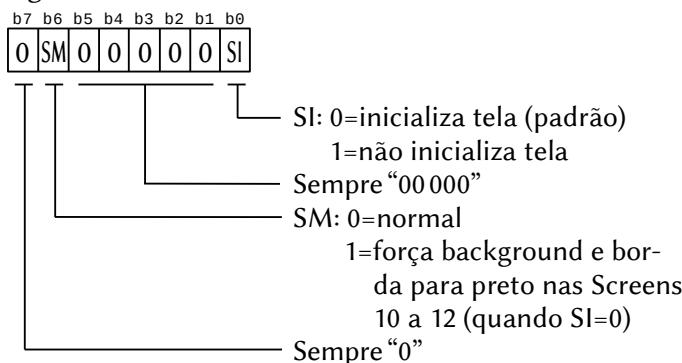
<estado> deve ser uma variável numérica. Se retornar -1, a reprodução terminou e se retornar 0 os dados ainda estão sendo reproduzidos. [#] deve estar entre 0 e 9.

**NETPLAYVID** (comando, GR8NET-BASIC)

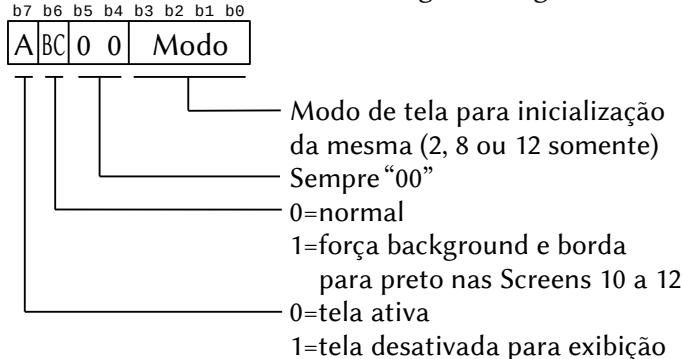
Formato: CALL NETPLAYVID (<caminho> [,<sinalizadores>])

CALL NETPLAYVID (<modo de tela>)

Função: Reproduzir vídeo a partir do cartão SD. Funciona de dois modos, dependendo do primeiro argumento. Se for string, esta especificar o <caminho> do arquivo de vídeo no cartão SD. <sinalizadores> é um valor de 8 bits cujos significados estão descritos abaixo:



Se o primeiro argumento for inteiro, seus 8 bits mais baixos são sinalizadores com os seguintes significados:



**NETPLAYWAV** (comando, GR8NET-BASIC)

Formato: CALL NETPLAYWAV (<caminho>)

Função: Reproduzir áudio no formato wave. <caminho> é um nome ou variável string que identifica a localização da URI do arquivo wave remoto.

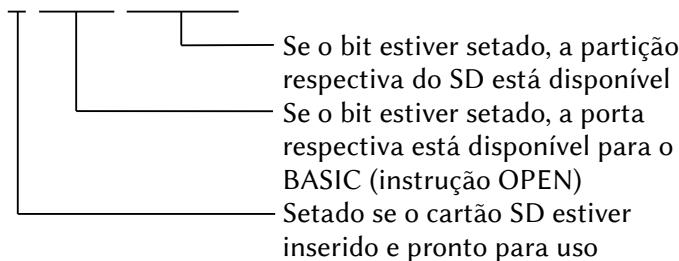
**NETRESST** (comando, GR8NET-BASIC)

Formato: CALL NETRESST (<sinalizadores>)

CALL NETRESST (<URI entrada>, <URI saída>,  
<sinalizadores>, <tamanho>)

Função: Retornar o estado do recurso. Se o primeiro argumento for uma variável inteira, os 8 bits mais baixos conterão os sinalizadores conforme descrito abaixo. Se for alfanumérica, <URI entrada> e <URI saída> conterão os respectivos caminhos e <tamanho> é uma variável numérica que retorna o tamanho do recurso. <sinalizadores> é uma variável inteira cujos 8 bits mais baixos são mapeados como se segue:

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
S	F2	F1	F0	P3	P2	P1	P0

**NETSAVE** (comando, GR8NET-BASIC)

Formato: CALL NETSAVE

Função: Salvar a configuração atual na página de config. da ROM.

**NETSDCRD** (comando, GR8NET-BASIC)

Formato: CALL NETSDCRD (<página lógica>, <endereço>, <setor>,  
<quantidade de setores a ler>)

Função: Ler setores do cartão SD. <página lógica> é o número da página no banco 1 da GR8NET (6000H~7FFFH), <endereço> é o endereço visível para o Z80 e <setor> é o número do primeiro setor a ser lido.

**NETSETCLK** (comando, GR8NET-BASIC)

Formato: CALL NETSETCLK (<fonte>)

Função: Define a fonte de frequência para medição de velocidade.

Se <fonte> for 0, será usada a frequência do barramento interno do MSX; se for diferente de 0, será usada a frequência do oscilador interno da GR8NET (3.579545 MHz).

**NETSETCLOUD** (comando, GR8NET-BASIC)

Formato: CALL NETSETCLOUD (<nome host : porta>, <senha>)

CALL NETSETCLOUD (<sinalizador de ativação>)

Função: Configurar acesso ao volume virtual da GR8NET. <nome host : porta> pode ter até 70 caracteres, com o número de porta separado por dois pontos. A <senha> de acesso pode ter até 16 caracteres. Para habilitar o subsistema GR8cloud, <sinalizador de ativação> deve conter o valor numérico 1, mas o volume só estará totalmente acessível após a reinicialização.

**NETSETDA** (comando, GR8NET-BASIC)

Formato: CALL NETSETDA (<nº adaptador>)

Função: Define o número do adaptador padrão. <nº adaptador> deve ser um valor de 0 a 3.

**NETSETDM** (comando, GR8NET-BASIC)

Formato: CALL NETSETDM (<página lógica>,<endereço>,<variável>)

Função: Obtém o valor de uma variável BASIC, converte em um valor de 32 bits e o armazena na memória.

<página lógica> número da página lógica no banco 1 da GR8NET (6000H~7FFFH).

<endereço> é o endereço de memória visível para o Z80. <variável> pode ser um número, uma expressão ou uma variável numérica de qualquer tipo.

**NETSETDNS** (comando, GR8NET-BASIC)

Formato: CALL NETSETDNS ([<A>], [<B>], [<C>], [<D>])

Função: Define endereço de IP fixo. Ao menos um dos valores <A>, <B>, <C> ou <D> deve ser definido.

**NETSETGW** (comando, GR8NET-BASIC)

Formato: CALL NETSETGW ([<A>], [<B>], [<C>], [<D>])

Função: Define endereço de IP fixo do gateway. Ao menos um dos valores <A>, <B>, <C> ou <D> deve ser definido.

**NETSETHOST** (comando, GR8NET-BASIC)

Formato: CALL NETSETHOST (<URI>)

CALL NETSETHOST (<A>, <B>, <C>, <D>)

Função: Define o nome do host remoto e, se necessário, executa uma consulta DNS simples. A <URI> deve ser escrita sem definição de protocolo e sem a barra final (ex. “www.gr8bit.ru”)

**NETSETIP** (comando, GR8NET-BASIC)

Formato: CALL NETSETIP ([<A>], [<B>], [<C>], [<D>])

Função: Define endereço de IP fixo. Ao menos um dos valores <A>, <B>, <C> ou <D> deve ser definido.

**NETSETMAP** (comando, GR8NET-BASIC)

Formato: CALL NETSETMAP [<A>, <M>, <MRPD>])

Função: Define o tipo de memória mapeada e reinicia o sistema.

<A> identifica o tipo de memória mapeada e a localização do conjunto de registradores especiais.

<M> 0 – Leitura desativada

1 – Leitura ativada

2 – Detecção automática (padrão)

<MRPD> RAM mapeada com o bit de desativação pendente (0 – Ativar; 1 – Desativar)

**NETSETMASK** (comando, GR8NET-BASIC)

Formato: CALL NETSETMASK ([<A>], [<B>], [<C>], [<D>])

Função: Define a máscara dp endereço de IP fixo. Ao menos um dos valores <A>, <B>, <C> ou <D> deve ser definido.

**NETSETMEM** (comando, GR8NET-BASIC)

Formato: CALL NETSETMEM (<página lógica>, <endereço>, [<A>], [<B>], [<C>], [<D>])

Função: Grava uma sequência de 4 bytes na memória.  
 <página lógica> número da página lógica no banco 1 da GR8NET (6000H~7FFFH).  
 <endereço> é o endereço de memória visível para o Z80.  
 <A> – Endereço, <B> – Endereço+1, etc.

#### **NETSETMIX** (comando, GR8NET-BASIC)

Formato: CALL NETSETMIX (<number>)  
 CALL NETSETMIX (<string>)

Função: Configura o mixer de áudio.

<number> – Valor numérico de 16 bits:

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	PSG	Y8950	OPLL	Wave		SCC			PCM			

Cada 2 bits representam o seguinte:

00 – Mudo	10 – Canal direito
01 – Canal esquerdo	11 – Ambos os canais

Se for string de 6 caracteres, cada um significará:

M – Mudo	R – Canal direito
L – Canal esquerdo	B – Ambos os canais

Outro caractere, a configuração será preservada.

#### **NETSETMMV** (comando, GR8NET-BASIC)

Formato: CALL NETSETMMV (<variável numérica>)

Função: Define o valor do gerenciador de memória. É possível gerenciar apenas página inicial de RAM protegida pelo usuário.

#### **NETSETNAME** (comando, GR8NET-BASIC)

Formato: CALL NETSETMMV (<nome do arquivo>)

Função: Define o nome do arquivo do recurso remoto. O tamanho máximo do nome de arquivo é de 63 caracteres.

#### **NETSETNTP** (comando, GR8NET-BASIC)

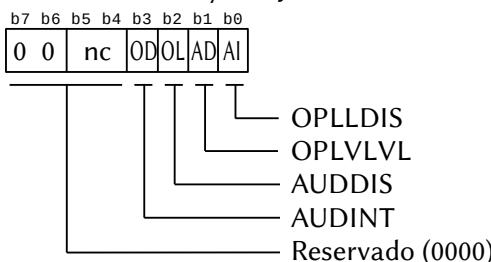
Formato: CALL NETSETNTP (<A>, <B>, <C>, <D>, <TZF>)

Função: Define as propriedades do servidor NTP dentro da configuração de IP fixo e dos sinalizadores de configuração de hora. <A>, <B>, <C> e <D> definem o endereço IP do servidor NTP e <TZF> é o sinalizador de atualização do fuso horário (ver comando NETNTP).

**NETSETOPL** (comando, GR8NET-BASIC)

Formato: CALL NETSETOPL (<sinalizadores>, <tamanho memória>)

Função: Ativa ou desativa o OPLL / Y8950, controla a duplicação da amplitude de saída e define o tamanho da memória de áudio do Y8950. <tamanho memória> define o tamanho da memória de áudio em incrementos de 8 Kbytes, sendo que o valor máximo e padrão é 32 ( $32 \times 8 = 256\text{K}$ ). <sinalizadores> é um valor de 1 byte cuja estrutura está descrita abaixo.



OPLLDIS – 0 – Ativar saída OPLL.

1 – Desativar saída OPLL.

OPLVVL – 0 – Volume normal OPLL/Y8950 (padrão).

1 – Volume duplicado OPLL/Y8950.

AUDDIS – 0 – habilita MSX-Audio (padrão).

1 – Desabilita MSX-Audio.

AUDINT – 0 – habilita interrup. MSX-Audio (padrão).

1 – Desabilita interrupções MSX-Audio.

**NETSETPATH** (comando, GR8NET-BASIC)

Formato: CALL NETSETPATH (<caminho>)

Função: Define o caminho do recurso remoto. <caminho> é um nome ou variável string com comprimento máximo de 239 caracteres e sem barra final e representa o valor absoluto.

**NETSETPORT** (comando, GR8NET-BASIC)

Formato: CALL NETSETPORT (<porta remota>, <porta local>)

Função: Define os números das portas de comunicação na estrutura URI padrão. Se <porta local> for 0, será usado número dinâmico de porta (valor padrão). Este comando não verifica a validade das portas.

**NETSETPSG** (comando, GR8NET-BASIC)

Formato: CALL NETSETPSG (<valor>)

Função: Configura o PSG.

<valor> é uma variável ou constante bitmap, onde o conjunto de bits 0 define se o PSG deve ser ativado na (re)configuração e o conjunto de bits 1 designa a localização porta para o valor 0x10 se, no reset, a porta for 0xA0 (PSG espelhado embutido). Se o argumento for omitido, será executada a reconfiguração do PSG.

**NETSETQSTR** (comando, GR8NET-BASIC)

Formato: CALL NETSETQSTR (<parâmetro>)

Função: Define a sequência de consultas para processamento de recursos remotos. <parâmetro> é uma variável ou constante string que deve começar com o caractere "?" e ter, no máximo, 63 caracteres.

**NETSETTSNH** (comando, GR8NET-BASIC)

Formato: CALL NETSETTSNH (<nome>)

Função: Define o nome do servidor de hora. <nome> é uma variável ou constante string que deve ter, no máximo, 63 caracteres.

**NETSNDTG** (comando, GR8NET-BASIC)

Formato: CALL NETSNDTG (<nº arq>, <A>, <B>, <C>, <D>, <RP>)

Função: Envia datagrama / dados pendentes para o host remoto. <nº arq> é o número do arquivo BASIC. <A>, <B>, <C> e <D> representam o endereço IP do dispositivo remoto (podem ser omitidos). RP é o número da porta do dispositivo remoto e também pode ser omitido.

**NETSNDVOL** (comando, GR8NET-BASIC)

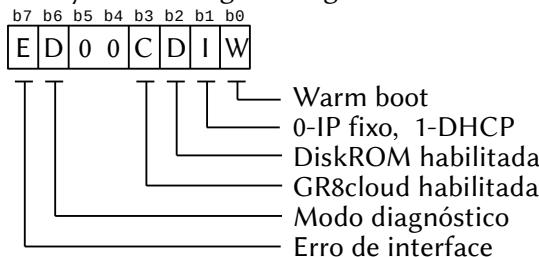
Formato: CALL NETSNDVOL (<principal>, <SCC>, <forma de onda>, <PCM>, <OPLL>, <Y8950>, <PSG>)

Função: Lê ou altera o volume dos geradores de áudio. Todos os argumentos devem estar na faixa de 0 (mudo) até 128 (volume máximo) e qualquer um pode ser omitido.

**NETSTAT** (comando, GR8NET-BASIC)

Formato: CALL NETSTAT

Função: Exibe informações de estado do adaptador. O último valor impresso na tela, identificado como “Mode:”, é um valor de um byte com o seguinte significado:



### **NETSYSINFO** (comando, GR8NET-BASIC)

Formato: CALL NETSYSINFO (<versão MSX>, <frequência clock>, <performance ciclo T>, <versão VDP>, <taxa vertical/tamanho VRAM>)

Função: Retorna informações e dados de desempenho do sistema.  
 <versão MSX> – 0=MSX1; 1=MSX2; 3=MSX2+, 4=MSX TR.

Para MSX turbo R, bit5=0→R800; bit5=1→Z80 e  
 bit6=0→modo DRAM; bit6=1→modo ROM

<frequência clock> retorna clock do slot (3 579 560)

<performance ciclo T> retorna o total de vezes que uma  
 instrução de 51 ciclos T (mais 8 do ciclo M1) é  
 executada em um segundo ( $60\ 671^*(51+8)$ )

<versão VDP> – 0=TMS; 1=V9938; 2=V9958

<taxa vertical/tamanho VRAM> – Valor de dois bytes,  
 onde o byte mais baixo retorna a taxa de quadro

(0=60 Hz, 1=50 Hz, 255=erro) e o byte mais alto  
 retorna o tamanho da VRAM em bloco de 4K (1=8K;  
 2=16 K; 4=32 K; 8=64 K; 16=128 K; 255 – Erro).

### **NETRCHKS** (comando, GR8NET-BASIC)

Formato: CALL NETRCHKS (<bloco de dados>, <endereço>, <número de bytes> <[, checksum]>)

Função: Calcula o checksum de 16 bits do conteúdo do buffer da RAM do <bloco de dados> no banco 1 da GR8NET. Se a variável <checksum> for fornecida, receberá a soma de verificação, caso contrário, a soma será impressa na tela.

**NETTELNET** (comando, GR8NET-BASIC)

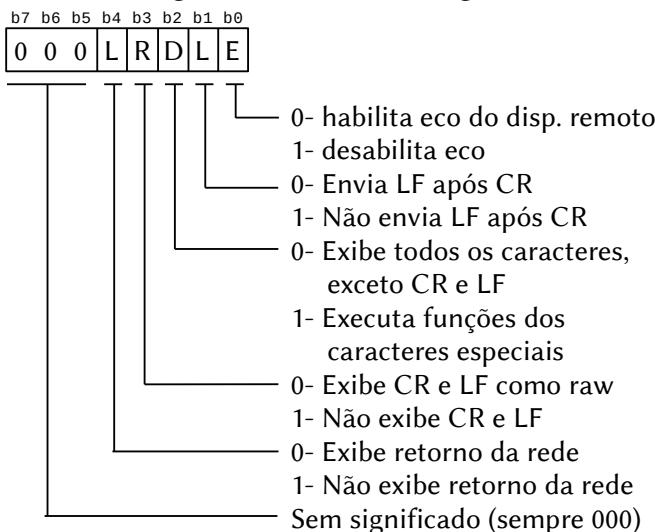
Formato: CALL NETTELNET ([<url | IP : porta>], <sinalizador>)

Função: Executar sessão de telnet usando TCP. <sinalizador> é um valor de um byte onde apenas o bit 1 tem significado. Se for 1, significa que o aplicativo telnet não adiciona o caractere LF após o CR; se for 0, ao pressionar RETURN será enviado CR+LF ao host remoto.

**NETTERM** (comando, GR8NET-BASIC)

Formato: CALL NETTERM ([<url | IP : porta>], <sinalizadores>)

Função: Executar sessão de telnet usando TCP. Este comando não realiza tradução especial de código ESC (&H1B). <sinalizadores> deve ser mantido em 0 se o dispositivo remoto executar echo o que recebe de volta para a GR8NET. O significado dos bits é o seguinte:

**NETTGTMAP** (comando, GR8NET-BASIC)

Formato: CALL NETTGTMAP [<A>, <M>, <MRPD>]

Função: Define o tipo de memória mapeada. A diferença de NETSETMAP e que este comando não reinicia a máquina.

<A> identifica o tipo de memória mapeada e a localização do conjunto de registradores especiais.

<M> 0 – Leitura desativada  
 1 – Leitura ativada  
 2 – Detecção automática (padrão)  
 <MRPD> RAM mapeada com o bit de desativação  
 pendente (0 – Ativar; 1 – Desativar)

#### **NETTSYNC** (comando, GR8NET-BASIC)

Formato: CALL NETTSYNC

Função: Exibe e sincroniza a hora do sistema.

#### **NETVARBRSTR** (função, GR8NET-BASIC)

Formato: CALL NETVARBRSTR (<variável alfanumérica>)

Função: Obtém a string da URL do local selecionado pelo usuário no navegador e a armazena em <variável alfanumérica>. Se o tamanho exceder 254 caracteres será gerado erro.

#### **NETVARBSIZE** (função, GR8NET-BASIC)

Formato: CALL NETVARBSIZE (<variável numérica>)

Função: Obtém o tamanho dos dados carregados em bytes e o armazena em <variável numérica>.

#### **NETVARRWTH** (comando, GR8NET-BASIC)

Formato: CALL NETVARRWTH (<valor atual>, <novo limite>)

Função: Definir o limite da janela RX da rede. <valor atual> deve ser uma variável numérica que recebe o tamanho atual (por padrão é 0). <novo limite> pode ser variável ou constante numérica entre 0 e 2047.

#### **NETVARUDTO** (comando, GR8NET-BASIC)

Formato: CALL NETVARUDTO (<valor atual>, <novo limite>)

Função: Definir tempo limite do pacote UDP para operações DHCP e DNS. <valor atual> é uma variável que recebe o valor atual do tempo limite e da contagem de novas tentativas de solicitação DHCP. <novo limite> é variável ou constante, definindo novo valor de tempo limite e contagem de novas tentativas de solicitação DHCP. Os bits 7~0 identificam o valor do tempo limite UDP (0-255), em períodos de 100 ms. O padrão é 20 (2s). Os bits 11~8 identificam o número de tentativas de solicitação de DHCP tentadas quando o GR8NET é inicializado.

**NETVER** (função, GR8NET-BASIC)

Formato: CALL NETVER

Função: Retorna a versão do firmware da GR8NET.

**NEXTOR** (comando, Nextor)

Formato: CALL NEXTOR

Função: Apresenta a lista de comandos adicionada pelo Nextor.

**NSCAN** (comando, Hitachi-BASIC versão 2)

Formato: CALL NSCAN

Função: Faz com que o leitor de dados embutido do micro Hitachi MB-H2 procure partes vazias na fita.

**OFFHOOK** (comando, New Modem BASIC)

Formato: CALL OFFHOOK

Função: Retira o fone do telefone do gancho.

**OFFLINE** (comando, Network-BASIC, SVI-Modem BASIC)

Formato: CALL OFFLINE [Network-BASIC]

Função: Desconecta o micro da rede. Esta instrução está disponível apenas para os alunos e precisa ser precedida por CALL NETINIT. Versão curta: \_OFFL.

Formato: CALL OFFLINE [SVI-Modem BASIC]

Função: Coloca o modem no modo offline.

**ON EVENT (n) GOSUB** (declaração, SFG-BASIC)

Formato: CALL ON EVENT (<nº do evento>) GOSUB

Função: Define a subrotina que será executada quando algum evento específico ocorrer. <nº do evento> pode ser:

1 ~ 4 – Interrompe quando a reprodução do instrumento especificado termina.

5 – Interrompe quando a reprodução do ritmo termina.

6 – Interrompe de acordo com o tempo programado no timer da unidade FM.

Se <nº do evento> for omitido, o comando será aplicado a todos os eventos. A prioridade dos eventos é a seguinte:

1 <sup>a</sup> – BASIC	5 <sup>a</sup> – Instrumento 4
------------------------	--------------------------------

2 <sup>a</sup> – Instrumento 1	6 <sup>a</sup> – Ritmo
--------------------------------	------------------------

3 <sup>a</sup> – Instrumento 2	7 <sup>a</sup> – Temporizador
--------------------------------	-------------------------------

4 <sup>a</sup> – Instrumento 3	
--------------------------------	--

Versão curta: \_ON EVEN (<nº do evento>) GOSUB.

**ONHOOK** (comando, New Modem BASIC)

Formato: CALL ONHOOK

Função: Coloca o fone do telefone no gancho.

**ONLINE** (comando, Network-BASIC, SVI-Modem BASIC)

Formato: CALL ONLINE [Network-BASIC]

Função: Conecta o micro na rede. Esta instrução está disponível apenas para os alunos. Eventualmente pode ser necessário executar CALL NETINIT antes. Versão curta: \_ONLI.

Formato: CALL ONLINE [SVI-Modem BASIC]

Função: Coloca o modem no modo online.

**PACLOAD** (comando, DM-System2 BASIC)

Formato: CALL PACLOAD (&lt;endereço PAC&gt;, [@]&lt;endereço de destino&gt; [, &lt;comprimento&gt;])

Função: Lê dados do SRAM do PAC (Pana Amusement Cartridge). &lt;endereço PAC&gt; é o endereço absoluto do cartucho PAC e pode variar de 0000H a 1FFDH.

&lt;endereço de destino&gt; é o endereço para onde os dados lidos serão transferidos. "@" significa VRAM.

&lt;comprimento&gt; é a quantidade de bytes lidos. Se omitido, serão lidos 1024 bytes (1 bloco).

**PACSAVE** (comando, DM-System2 BASIC)

Formato: CALL PACSAVE ([@]&lt;endereço inicial&gt;, &lt;endereço PAC&gt; [, &lt;comprimento&gt;])

Função: Escreve dados na SRAM do PAC (Pana Amusement Cartridge). &lt;endereço inicial&gt; é o endereço de onde os dados serão lidos. Se precedido por "@", significa VRAM.

&lt;endereço PAC&gt; é o endereço absoluto do cartucho PAC e pode variar de 0000H a 1FFDH.

&lt;comprimento&gt; é a quantidade de bytes a escrever. Se omitido, serão escritos 1024 bytes (1 bloco).

**PALETTE** (declaração, 3, Kanji-BASIC, Hangul-BASIC, RMSX BASIC)

Formato: CALL PALETTE (&lt;nº paleta&gt;, &lt;R&gt;, &lt;G&gt;, &lt;B&gt;)

[MSX-BASIC version 3, Kanji-BASIC, Hangul-BASIC]

Função: Especifica as cores para a paleta. &lt;nº paleta&gt; pode variar de 0 a 15 e "&lt;R&gt;, &lt;G&gt;, &lt;B&gt;" de 0 a 7. Todas as versões do BASIC possuem a mesma sintaxe, exceto RMSX BASIC.

Formato: CALL PALETTE <paleta/monitor> [RMSX BASIC]

Função: Seleciona a emulação de paleta ou monitor a ser usada no computador MSX1 emulado em uma máquina Turbo R pelo emulador rMSX. <paleta/monitor> pode ser MSX1, MSX2, GREEN or GRAY.

### PAN (declaração, Pioneer-BASIC)

Formato: CALL PAN (<eixo X>, <volume>, <string>)

Função: Gera som de acordo com os parâmetros fornecidos.

<eixo X> – Localização do som gerado. Pode variar de 0 a 255, sendo que 0 corresponde à extrema esquerda e 255 à extrema direita.

<volume> pode variar de 0 a 7.

<string> – Macrocomandos idênticos aos da instrução PLAY para o PSG, à exceção de V, S, M e X. A string pode conter até 79 caracteres.

### PATTERN (declaração, SFG-BASIC)

Formato: CALL PATTERN (<nº padrão>, <variável alfanum (n)>)

Função: Define os padrões dos ritmos através de uma matriz alfanumérica de uma dimensão (vetor). Versão curta: \_PATT. <nº padrão> pode ser 7 ou 8 (apenas dois padrões podem ser definidos).

<variável alfanum (n)> aponta para um vetor cujos índices definem aspectos variados:

xx\$(0) define o tamanho:

“3” – um quarto de Nota vezes 3

“4” – um quarto de Nota vezes 4

“8” – um quarto de Nota vezes 8

xx\$(1) define “close high-hat”

xx\$(2) define “open high hat”

xx\$(3) define “bass drum”

xx\$(4) define “high tomtom”

xx\$(5) define “low tomtom”

“close high-hat” e “open high hat” não podem ser tocados simultaneamente.

Nota: A string para os índices (1) a (5) deve ser composta por uma sequência de “0”s e “1”s que representam unidades de 1/12 de um quarto de Nota. O tamanho depende do valor de xx\$(0): “3” para 36 caracteres, “4” para 48 e “8” para 96.

**PAUSE** (comando, MSX-BASIC 4, ChakkariCopy BASIC,  
DM-System2 BASIC, Hitachi BASIC v2)

Formato: CALL PAUSE (<tempo>) [MSX-BASIC 4, DM-System2 BASIC]

Função: Pausa a execução do programa em BASIC. <tempo> é especificado em milissegundos e pode variar de 0 a 65.535. Pode ser abortado por CTRL+STOP.

Formato: CALL PAUSE [ChakkariCopy BASIC]

Função: Coloca o cartucho Chakkari Copy no modo de pausa.

Formato: CALL PAUSE [Hitachi BASIC 2]

Função: Coloca o leitor de dados interno do micro Hitachi MB-H2 no modo de pausa.

**PCM FREQ** (comando, MSX-Audio)

Formato: CALL PCM FREQ (<frequência>)

Função: Define a frequência de amostragem para o ADPCM. <frequência> pode variar de 1.800 a 49.716 Hz.

**PCM VOL** (comando, MSX-Audio)

Formato: CALL PCM VOL (<volume>)

Função: Define o volume de reprodução para o ADPCM e PCM. <volume> pode variar 0 a 63. Os valores iniciais são 55 para ADPCM e 32 para PCM.

**PCMON** (declaração, DM-System2 BASIC)

Formato: CALL PCMON ([@]<endereço inicial>, [@]<endereço final>,<taxa>[,<loop>])

Função: Reproduz dados através do PCM no MSX2 em diante. Requer driver PCM.

<endereço inicial> dos dados a reproduzir. Se precedido por “@”, significa VRAM.

<endereço final> dos dados a reproduzir. Se precedido por “@”, significa VRAM.

<taxa> pode ser: 0 → 15,75 KHz 2 → 5,25 KHz  
1 → 7,875 KHz 3 → 3,9375 KHz

<loop> define a quantidade de vezes que os dados serão reproduzidos. Pode variar de 1 a 255. 0 = loop infinito.

**PCMPLAY** (declaração, 4)

Formato: CALL PCMPLAY (@<endini>,<endfim>,<samp.rate>[,S])

Função: Reproduz dados PCM armazenados na RAM ou VRAM.  
 <samp. rate> pode ser 0 a 3.  
 <endini> e <endfim> são os endereços inicial e final.  
 [,S] especifica VRAM.

#### **PCMREC** (comando, 4)

Formato: CALL PCMREC (@<endini>,<endfim>,<samp.rate>,  
 [[<nível de disparo>],[<salvamento>],S])

Função: Grava dados PCM na RAM ou VRAM.  
 <endini> e <endfim> podem variar de 0000H a FFFFH,  
 <samp.rate> de 0 a 3,  
 <nível de disparo> de 0 a 127  
 <salvamento> pode ser 0 (não salva) ou 1 (salva na RAM)  
 [,S] grava na VRAM.

#### **PDIAL** (comando, New Modem BASIC, SVI Modem BASIC)

Formato: CALL PDIAL(<variável string>), <variável numérica>  
 [New Modem BASIC]

Função: Chama um número de telefone via discagem por pulso.  
 <variável de string> armazena o número de telefone, onde  
 só são permitidos números e o caractere “-”, que correspon-  
 de a uma espera de 1 segundo. Se <variável numérica> re-  
 tornar 0, o valor de entrada está correto; se for “1” não está.

Formato: CALL PDIAL(“<número telefone>”) [SVI Modem BASIC]

Função: Chama um número de telefone específico via discagem  
 por pulso. <número telefone> deve estar entre aspas e só  
 são permitidos caracteres numéricos.

#### **PEEK** (função, DM-System2 BASIC, Network-BASIC)

Formato: CALL PEEK ([@]<endereço>[,<variável>])  
 [DM-System2 BASIC]

Função: Lê um byte de qualquer área da Main RAM, VRAM ou  
 Memória Mapeada.  
 <endereço> – é o endereço a ser lido. Se for maior que  
 65 534, serão usadas as especificações da Memória  
 Mapeada. Se for precedido de “@”, indica VRAM.  
 <variável> é uma variável numérica que receberá o valor do  
 byte lido. Se omitida, o valor será apresentado na tela.

Formato: CALL PEEK (<variável>,<endereço>[,<nº estudante>] [,<N>])  
 [Network-BASIC]

Função: Lê um byte de dados da NetRAM (aluno ou professor) ou da NetRAM / RAM (somente professor).  
 <variável> recebe o valor do byte lido.  
 <endereço> deve estar entre 7800H e 7FFFH para NetRAM.  
 <número do aluno> é um número entre 1 e 15. Somente o professor pode usar esse parâmetro.  
 <N> deve ser usado para ler um endereço no NetRAM.  
 Útil apenas para o professor.

#### **PEEKS** (função, DM-System2 BASIC)

Formato: CALL PEEKS ([@]<endereço>,<tamanho>,<variável string>)  
 Função: Lê vários bytes consecutivos na Main RAM, VRAM ou Memória Mapeada, os converte em caracteres e os armazena em uma variável string.  
 <endereço> – é o endereço a ser lido. Se for maior que 65 534, serão usadas as especificações da Memória Mapeada. Se for precedido de “@”, indica VRAM.  
 <tamanho> – Número de bytes a serem lidos, podendo variar de 1 a 255.  
 <variável string> recebe os bytes lidos.

#### **PEEKW** (função, DM-System2 BASIC)

Formato: CALL PEEKW ([@]<endereço>[,<variável>])  
 Função: Lê dois bytes consecutivos na Main RAM, VRAM ou Memória Mapeada.  
 <endereço> – é o endereço a ser lido. Se for maior que 65 534, serão usadas as especificações da Memória Mapeada. Se for precedido de “@”, indica VRAM.  
 <variável> recebe o valor lido. Se não especificado, o valor lido será apresentado na tela no formato hexa.

#### **PHRASE** (macro-declaração, SFG-BASIC)

Formato: CALL PHRASE (<nº trilha>,<lista reprodução>[,<marca>] )  
 Função: Escreve os dados de reprodução de audio na trilha especificada. Versão curta: \_PHRA.  
 <nº trilha> pode variar de 1 até o valor definido por \_PLAY.  
 <marca> é um número que pode variar de 1 a 254. Se for omitido, será considerado igual a <nº trilha>.

<lista reprodução> contém os macrocomandos musicais.

A~G	Toca Nota cifrada com duração n (1~64, padrão 4).
# ou +	Sustenido.
-	Bemol.
!	Retorna a Nota ao valor original (comandos K e S).
On	Oitava (n → 1 a 8; o padrão é 3).
Nn	Pitch (n → 25 a 120).
Ln	Duração (length) da Nota (n → 1 a 64, padrão: 4)
.	Aumento da duração em 50%.
Rn	Pausa de duração n (n → 1 a 64, o padrão é 4).
Wn	Duração da Nota em unidades de 1/96 (n: 1 a 96).
Tn	Tempo (n → 1 a 200, especificado por _TEMPO).
Vn	Volume (n → 1 a 100, o padrão é 50)
&	Ligadura
Mn	Período em unidades de n/4 (n → 3 a 8)
/ /	A string entre duas barras será considerada um bloco de duração especificada por Mn.
Sn	“n” especifica o número de “Sharps” (#).
Kn	“n” especifica o número de “Flats” (b).
%n	“Staccato” e “tenuto”. “n” especifica a proporção de tempo em que a Nota que será tocada (0% a 100%).
[ ]	Acorde. Strings entre aspas e separadas por vírgula dentro do colchete são tocadas simultaneamente.
{ }n	Define em n as Notas entre { }. (n=1~64, padrão é Ln)

### **PITCH** (declaração, MSX-Music)

Formato: CALL PITCH (<n>)

Função: Ajuste fino do som. <n> pode variar de 410 a 459, sendo que o valor padrão é 440 (Nota LÁ central).

### **PLAY** (comando, MSX-Music/Audio, Hitachi-BASIC, SFG-BASIC)

Formato: CALL PLAY (<n>,<variável numérica>) [Music/Audio]

Função: Retorna na <variável numérica> o estado da voz <n> do OPPLL (tocando[-1] ou não [0]). Se <n> for 0, todas as vozes são checadas; 1 a 9 checa a voz respectiva.

Formato: CALL PLAY [Hitachi-BASIC 2]

Função: Coloca o leitor de dados interno do micro Hitachi MB-H2 no modo de reprodução. Esta instrução não suporta arquivos no modo ASCII (BASIC ou dados).

Formato: CALL PLAY (<instrumento>, <faixa> [,<marca>] )  
 [SFG-BASIC]

Função: Reproduz a música previamente escrita com a instrução CALL PHRASE. <instrumento> é um número de 1 a 4, <faixa> é um número de 1 a 9, sendo que 2 a 8 devem ser previamente definidos pela instrução CALL TRACK e 9 quando a reprodução for pelo teclado musical, e <marca> é um número entre 1 e 255 para especificar a marca CALL PHRASE usada para reprodução. Se omitido, é considerado o mesmo número de <track>.

### **PLAY MK** (declaração, MSX-Audio)

Formato: CALL PLAY MK (<nome da matriz>)  
 CALL PLAY MK (<endereço inicial>, <endereço final>)  
 CALL PLAY MK (A), onde a sequência A deve ser previamente declarada nas instruções DIM e REC MK.  
 Função: Reproduz arquivo gravado pelo teclado musical.

### **PLAY PCM** (declaração, MSX-Audio)

Formato: CALL PLAY PCM (<número arquivo>, <offset>,  
 <tamanho>, <frequência amostragem>)  
 Função: Reproduz um arquivo de áudio através do PCM / ADPCM.  
 <número arquivo> – Número do arquivo de áudio (0 a 15)  
 <offset> – Deslocamento em unidades de 256 bytes  
 <tamanho> – Tamanho em bytes do arquivo de áudio  
 <frequência de amostragem> – pode variar de 1.800 a 49.716 Hz para o ADPCM e de 1.800 a 16.000 Hz para o PCM.

### **POKE** (declaração, DM-System2 BASIC, Network-BASIC)

Formato: CALL POKE ([@]<endereço>,<valor>) [DM-System2 BASIC]  
 Função: Escreve um byte em qualquer área da Main RAM, VRAM ou Memória Mapeada.  
 <endereço> – é o endereço a ser lido. Se for maior que 65 534, serão usadas as especificações da Memória Mapeada. Se for precedido de "@", indica VRAM.  
 <valor> é o valor a ser escrito. Deve ser em decimal entre 0 e 255, podendo também ser uma expressão.

**Formato:** CALL POKE (<valor>,<endereço>[,<nº estudante>] [,<N>])  
 [Network-BASIC]

**Função:** Escreve um byte de dados na NetRAM (aluno ou professor) ou na NetRAM / RAM (somente professor).  
 <valor> deve ser um número decimal entre 0 e 255.  
 <endereço> deve estar entre 7800H e 7FFFH para NetRAM.  
 <número do aluno> é um número entre 1 e 15. Somente o professor pode usar esse parâmetro.  
 <N> deve ser usado para escrever um endereço na NetRAM. Útil apenas para o professor.

**POKES** (declaração, DM-System2 BASIC)

**Formato:** CALL POKES ([@]<endereço>,<string>)

**Função:** Converte uma string em uma sequência de bytes e os grava na Main RAM, VRAM ou Memória Mapeada.  
 <endereço> – é o endereço de início de escrita. Se for maior que 65 534, serão usadas as especificações da Memória Mapeada. “@”, indica VRAM.  
 <string> = sequência de caracteres a ser convertida em bytes.

**POKEW** (declaração, DM-System2 BASIC)

**Formato:** CALL POKEW ([@]<endereço>,<valor>)

**Função:** Escreve dois bytes consecutivos na Main RAM, VRAM ou Memória Mapeada.  
 <endereço> – é o endereço a ser escrito. Se for maior que 65 534, serão usadas as especificações da Memória Mapeada. Se for precedido de “@”, indica VRAM.  
 <valor> é o valor a ser escrito. Deve ser em decimal entre 0 e 65.535, podendo também ser uma expressão.

**PON** (comando, Network-BASIC)

**Formato:** CALL PON

**Função:** Inicia a pesquisa dos alunos. Esta instrução está disponível apenas para o professor.

**PRINTERSETUP** (comando, FM-X BASIC)

**Formato:** CALL PRINTERSETUP

**Função:** Permite imprimir hiragana e caracteres gráficos na impressora conectada ao Fujitsu FM-7 quando este for conectado ao micro FM-X usando a interface MB22 450.

**QDFILES** (comando, QuickDisk BASIC)

Formato: CALL QDFILES [("QD[n]":)]

Função: Lista o conteúdo do dispositivo Quick Disk especificado em formato longo, com os nomes dos arquivos, os atributos e os tamanhos dos arquivos. Os atributos listados são os seguintes:  
 01 – Arquivo binário da MainRAM.  
 02 – BASIC em formato tokenizado.  
 03 – BASIC ou DATA no formato ASCII.  
 0B – Arquivo binário da VRAM.

QD[n] especifica o dispositivo QuickDisk que será usado.  
 Pode variar de 0 a 7, sendo que o padrão é 0.

**QDFORMAT** (comando, QuickDisk BASIC)

Formato: CALL QDFORMAT

Função: Formata um QuickDisk excluindo todos os arquivos existentes. Os dados são gravados em uma pista espiral em um disco de 2,8 polegadas. Um QuickDisk pode salvar no máximo 20 arquivos e tem capacidade de 64 Kbytes cada lado, com capacidade máxima de 128 Kbytes.

**QDKEY** (comando, QuickDisk BASIC)

Formato: CALL QDKEY (<parâmetro>)

Função: Modifica o conteúdo das teclas de função, exceto F7, quando uma unidade de QuickDisk está conectada. Quando um MSX for inicializado com uma unidade QuickDisk conectada, o conteúdo da maioria das teclas de função é modificado. CALL QDKEY permite alternar entre os conteúdos.

Tecla	Conteúdo	Novo comando
F1	_RUN	CALL RUN
F2	_LOAD	CALL LOAD
F3	_BLOAD	CALL BLOAD
F4 (*)	list	LIST
F5 (*)	run	RUN + [RETURN]
F6 (**)	color 15,4,7	COLOR 15,4,7 + [RETURN]
F7	_QDKEY	CALL QDKEY
F8	_SAVE ("QD:	CALL SAVE ("QD:
F9	_BSAVE ("QD:	CALL BSAVE ("QD:
F10	_QDFILES	CALL QDFILES

(\*) Geralmente inalterado

(\*\*) Inalterado em máquinas japonesas, coreanas, Philips VG-8000 e VG-8010 (não na versão 8010F), Sanyo PHC-28S <parâmetro> – Se for 0, o conteúdo padrão das teclas será recarregado (exceto F7). Sem parâmetro ou qualquer outro número carrega o conteúdo do QuickDisk.

#### **QDKILL** (comando, QuickDisk BASIC)

Formato: CALL QDKEY (["QD[n]"]<nome do arquivo>")

Função: Exclui o último arquivo do QuickDisk. A tentativa de excluir outro arquivo retornará mensagem de erro.

QD[n] especifica o dispositivo QuickDisk que será acessado. Pode variar de 0 a 7, sendo que o padrão é 0. <nome do arquivo> é o arquivo a ser excluído e deve estar no formato 8.3 caracteres.

#### **RAMDISK** (comando, Disk-BASIC 2nd version)

Formato: CALL RAMDISK (<tamanho máx>,[<tamanho criado>])

Função: Cria uma RAMDISK com <tamanho máximo> e opcionalmente retorna o <tamanho criado> de fato. A RAMDISK é acessada através do drive H:

#### **RCANCEL** (comando, SFG-BASIC)

Formato: CALL RCANCEL

Função: Cancela os instrumentos de ritmo. Usando esta instrução, o total de vozes simultâneas passa de 6 para 8.

#### **RCVMAIL** (comando, Network-BASIC)

Formato: CALL RCVMAIL (<número do aluno>)

Função: Recebe dados da caixa de correio de envio de um aluno na caixa de correio de recebimento do professor. Esta instrução está disponível apenas para o professor. Caixas de correio são áreas especiais de 256 bytes reservadas na NetRAM do professor e do aluno. <número do aluno> pode variar de 1 a 15. Versão curta: \_RCVM.

#### **REBOOT** (comando, Hangul-BASIC 4, Rookie Drive BASIC)

Formato: CALL REBOOT

Função: Provoca uma reinicialização “a quente” do sistema.

**REC** (comando, Hitachi-BASIC versão 2)

Formato: CALL REC

Função: Coloca o leitor de dados interno do micro Hitachi MB-H2 no modo de gravação. Esta instrução não suporta arquivos no modo ASCII (BASIC ou dados).

**REC MK** (comando, MSX-Audio)

Formato: CALL REC MK (<nome da matriz>)

CALL REC MK (<endereço inicial>, <endereço final>)

Função: Grava arquivo tocado pelo teclado musical.

**REC PCM** (comando, MSX-Audio)

Formato: CALL REC PCM (<número arquivo> [,SYNC][,<Offset>]  
[,<tamanho>] [,<freq de amostragem>])

Função: Gravar áudio na memória através do microfone.

<número arquivo> – Arquivo a ser gravado (0 a 15).

SYNC – Se for 0, o MSX-Audio aguarda até que um sinal de áudio seja detectado. Se for 1, a gravação inicia imediatamente.

<Offset> – Deslocamento em unidades de 256 bytes.

<tamanho> – Tamanho do arquivo de áudio.

<freq de amostragem> – pode variar de 1.800 a 49.716 Hz para o ADPCM e de 1.800 a 16.000 Hz para o PCM.

**RECEIVE** (comando, Network-BASIC)

Formato: CALL RECEIVE ([[<letra da unidade>:] <nome do arquivo>], <número do aluno>)

Função: Recebe o programa BASIC de um computador do aluno.

Esta instrução pode ser usada pelo professor e pelos alunos que foram autorizados pelo professor com CALL ENACOM. <letra da unidade> pode ser “A:” ou “B:” e pode ser usado apenas pelo professor. <número do aluno> pode variar de 1 a 15. Versão curta: \_RECE.

**RECFILE** (comando, New Modem BASIC)

Formato: CALL RECFILE(<variável string>), <variável numérica>

Função: Recebe um arquivo usando um protocolo específico <variável string> contém o nome do arquivo a ser recebido (pode incluir o nome da unidade, se omitido, o arquivo será salvo na unidade ativa atual). Se já existir um arquivo

com o mesmo nome no disco, a primeira letra será substituída por "\$", o que ocorrerá até o quarto caractere.

<variável numérica> armazena o protocolo:

0 – Xmodem ou Xmodem-1K.

3 – Ymodem (permite receber vários arquivos simultâneos).

Ao retornar, <variável numérica> conterá o estado:

0 – Recebimento foi feito corretamente.

1 – Sinal caiu (geralmente a conexão está interrompida).

2 – Tempo limite atingido (o download não foi iniciado).

3 – Operação abortada com CTRL+X.

4 – Muitas pausas (tempos de espera).

5 – Não usado (sem efeito).

6 – Disco cheio.

7 – Arquivo não encontrado.

8 – Erro de gravação (disco protegido contra gravação ou não há disco).

9 – Arquivo vazio.

10 – Excesso de tentativas.

### **RECMOD** (comando, MSX-Audio)

Formato: CALL RECMOD (<modo de gravação>)

Função: Define o modo de gravação para o teclado musical.

<modo de gravação> é um valor de 0 a 3:

0 – Mudo (não grava).

1 – Grava a melodia tocada no teclado (padrão).

2 – Grava, em outra área, a reprodução de uma melodia já gravada.

3 – Grava a performance e a reprodução de uma melodia já gravada.

### **REMOTE** (comando, Pioneer-BASIC)

Formato: CALL REMOTE (<nº dispositivo>, <string>)

Função: Controla dispositivos externos.

<nº dispositivo> pode variar de 0 a 15, mas os dispositivos de 0, 1 e 2 já estão atribuídos:

0 – Pioneer Laser Vision Player LD-700

1 – Pioneer Laser Vision Player LD-1100

2 – Pioneer Component Display SD-26

Os comandos de 3 a 15 devem ser atribuídos com CALL DEF UNIV.

<string> contém um código de caracteres de até 16 comandos de acordo com a tabela a seguir (o caractere “+” pode ser omitido):

*Funções do modelo LD-700 (dispositivo 0):*

A+	48	Repete A	M+	58	Multi-veloc. à frente
A-	44	Repete B	M-	55	Multi-veloc. reversa
C+	47	Inc. multi-veloc.	P+	17	Toca (Play)
C-	46	Dec. multi-veloc.	P@	16	Rejeita
D+	43	Apresenta nº de quadro/capítulo	P/	18	Pausa
F+	10	Busca rápida	S+	54	Congela / quadro a quadro p/ frente
F-	11	Busca ráp. Rev.	S-	50	Congela / quadro a quadro reverso
L+	4B	Áudio 1 / esq.	T+	51	Avanço rápido (3x)
L-	49	Áudio 2 / dir.	T-	59	Retroc. rápido (3x)
L@	4A	Estéreo	X+	45	Limpa (Clear)

*Funções do modelo LD-1100 (dispositivo 1):*

D+	Apresenta nº quadro	P+	Toca (Play)
D-	Apresenta nº capítulo	P@	Rejeita
F+	Busca rápida	P/	Pausa
F-	Busca rápida rev.	S+	Congela / quadro a quadro p/ frente
L+	Áudio 1 / esquerda	S-	Congela / quadro a quadro reverso
L-	Áudio 2 / direita	T+	Avanço rápido (3x)
M+	Busca lenta à frente	T-	Retrocesso rápido (3x)
M-	Busca lenta reversa		

*Funções do modelo SD-26 (dispositivo 2):*

1	01	Canal A	F+	10	Incrementa canal (+)
2	02	Canal B	F-	11	Decrementa canal (-)
3	03	Canal C	K1	0C	Entrada: TV
4	04	Canal D	K2	0D	Entrada: Video-Disc
5	05	Canal E	K3	0E	Entrada: Vídeo 1
6	06	Canal F	K4	0F	Entrada: Vídeo 2
7	07	Canal G	O@	1C	Liga / desliga
8	08	Canal H	V+	0A	Aumenta volume (+)
9	00	Canal I	V-	0B	Dimunui volume (-)
0	00	Canal J		48	Sleep
C-	46	Canal K		49	Mudo
C+	47	Canal L		4A	Display call
4D		Canal M			
4E		Canal N			
4F		Canal O			
50		Canal P			

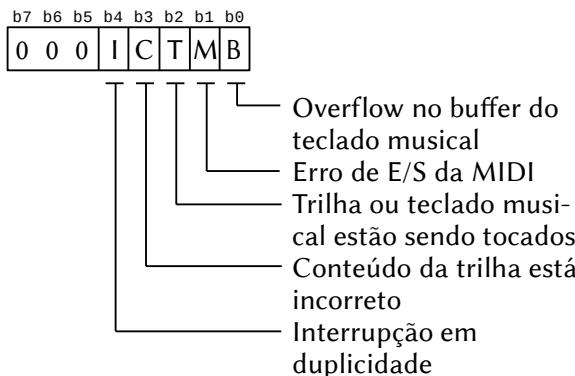
*Outras funções:*

M@	ID	Liga/desliga tape monitor
P-	15	Reverse play (para tape-deck)
W		Espera vídeo (para laser vision player)
R+	14	Grava (para tape-deck)
R-	12	Grava mudo (para tape-deck)

**REPORT** (Variável de sistema, SFG-BASIC)

Formato: CALL REPORT ( [<sinalizador de erros>] [,<nº marca>]  
[,<nº de repetições restantes>] )

Função: Retorna as variáveis de sistema. Versão curta: \_REPO.  
<sinalizador de erros> é uma variável inteira onde apenas os 5 bits mais baixos são válidos:



< n° marca> é uma variável inteira que retorna o número de marca da última seção reproduzida.

<nº de repetições restantes> retorna o número de vezes que o ritmo ainda será reproduzido.

**RESET** (comando, RMSX BASIC)

Formato: CALL RESET

Função: Reinicia o computador MSX1 ou MSX2 emulado em uma máquina Turbo R pelo emulador rMSX. É uma reinicialização a quente, como com DEFUSR=0: X=USR(0).

**REW** (comando, Hitachi-BASIC versão 2)

Formato: CALL REW

Função: Faz com que o leitor de dados interno do micro Hitachi MB-H2 rebobine a fita.

**RHYTHM** (declaração, SFG-BASIC)

Formato: CALL RHYTHM (<nº de repetições> [,<nº da marca>])  
 Função: Reproduz os padrões de ritmo selecionados pelo comando \_SELPATTERN. Versão curta: \_RHYT.  
 <nº de repetições> especifica o número de repetições em unidades de 1/4 de Nota.  
 <nº da marca> pode variar de 1 a 254. Se omitido, será usado o valor 10.

**RMDIR** (comando, Disk-BASIC 2nd version)

Formato: CALL RMDIR (<subdiretório>)  
 Função: Remove o <subdiretório> especificado.

**RSTOP** (declaração, SFG-BASIC)

Formato: CALL RSTOP  
 Função: Interrompe a reprodução do ritmo. Versão curta: \_RSTO.

**RTCINI** (comando, Hangul-BASIC 3)

Formato: CALL RTCINI  
 Função: Reseta o conteúdo da SRAM do RTC para o padrão inicial correspondente ao MSX1.

**RTSOFF** (comando, New Modem BASIC)

Formato: CALL RTSOFF  
 Função: Desliga a onda transportadora (RTS = Request To Send). Esta instrução funciona apenas quando o sinal DTR (Data Terminal Ready) está ativo (CALL DTRON).

**RTSON** (comando, New Modem BASIC)

Formato: CALL RTSON  
 Função: Liga a onda transportadora (RTS = Request To Send). Esta instrução funciona apenas quando o sinal DTR (Data Terminal Ready) está ativo (CALL DTRON).

**RUN** (comando, Network-BASIC, QuickDisk-BASIC, X-BASIC)

Formato: CALL RUN [[(<número do aluno>), [<número da linha>])] [Network-BASIC]  
 Função: Executa o programa BASIC que está na memória do micro de um aluno. Esta instrução está disponível apenas para o professor. <número do aluno> pode variar de 0 a 15. Se

omitido ou igual a 0, os programas de todos os micros serão executados. O programa será executado a partir da linha <número da linha>, se especificado.

Formato: CALL RUN ("[QD[n]:] <nome do arquivo>") [QuickDisk-BASIC]

Função: Carrega um arquivo BASIC do dispositivo Quick Disk especificado na memória MSX e o executa.

QD[n] especifica o dispositivo QuickDisk que será usado.

Pode variar de 0 a 7, sendo que o padrão é 0.

<nome do arquivo> deve estar no formato 8.3 caracteres.

Formato: CALL RUN [X-BASIC]

Função: Compila e executa o programa BASIC presente na memória do MSX.

### **SAVE** (comando, DM-System2 BASIC, QuickDisk-BASIC)

Formato: CALL SAVE ("[<dispositivo>:][]<caminho>[]<nome do arquivo>[@]<endereço fonte>,<tamanho>,[<offset>]) [DM-System2 BASIC]

Função: Salva dados em um novo arquivo ou em algum ponto de um arquivo já existente.

<dispositivo> pode ser drive A: a H: ou COM: para micros conectados com RS232C.

<caminho> especifica o local da pasta ou arquivo.

<nome do arquivo> é o arquivo a ser salvo.

<endereço destino> é o endereço de origem dos dados. Se precedido por "@" significa VRAM.

<tamanho> especifica a quantidade de bytes a salvar.

<offset> especifica o deslocamento no arquivo destino.

Formato: CALL SAVE ("[QD[n]:]<nome do arquivo>[,A]) [QuickDisk-BASIC]

Função: Salva dados da memória ou um programa BASIC em um dispositivo QuickDisk. Os dados serão sempre salvos em texto ASCII. O programa BASIC pode ser salvo em texto ASCII ou tokenizado.

QD[n] especifica o dispositivo QuickDisk que será usado.

Pode variar de 0 a 7, sendo que o padrão é 0.

<nome do arquivo> deve estar no formato 8.3 caracteres.

[,A], se especificado, salva o arquivo BASIC na forma de texto ASCII.

**SAVE PCM** (comando, MSX-Audio)

Formato: CALL SAVE PCM (<nome arquivo>, <número arq>)

Função: Salvar arquivo de áudio no disco.

<nome arquivo> – Nome do arquivo a ser gravado no disco.

<número arq> – Número do arquivo na memória de áudio.

Pode variar de 0 a 15.

**SCLOAD** (comando, Pioneer-BASIC)

Formato: CALL SCLOAD [<nome de arquivo>]

Função: Carrega dados do cassete para a VRAM para apresentação na tela (disponível apenas para Screen 2).

**SCOPY** (comando, Hitachi-BASIC versão 3)

Formato: CALL SCOPY (<c1> [,<c2>, <c3>, <c4> .... <c15>] )

Função: Envia para a impressora uma cópia de uma tela gráfica em Screens 2, 4 ou 5 usando uma fórmula baseada nas cores selecionadas. Mesmo que CALL CSCOPY.

**SCSAVE** (comando, Pioneer-BASIC)

Formato: CALL SCSAVE (<nome de arquivo>, [<baud rate>])

Função: Grava dados da VRAM no cassete. <baud rate> pode ser 1 (para 1200 bauds) ou 2 (para 2400 bauds). Se não especificado, será usado o baud rate definido em SCREEN. Comando disponível apenas para Screen 2.

**SEARCH** (comando, Pioneer-BASIC)

Formato: CALL SEARCH (<tipo>, { F | C }, <nº quadro ou capítulo>)

Função: Busca no Laser Vision Player o quadro ou capítulo especificados. <tipo> pode ser 0 para o LD-700 ou 1 para LD-1100. “F” busca quadro (frame) e “C” busca capítulo. <nº quadro ou capítulo> pode variar entre 0 e 54 000.

**SELPATTERN** (declaração, SFG-BASIC)

Formato: CALL SELPATTERN (<nº do padrão>)

Função: Seleciona os padrões de ritmo para reprodução. Versão curta: \_SELP.

<nº do padrão> pode variar de 1 a 8, sendo 1 a 6 os padrões da ROM e 7 e 8 os padrões definidos pelo comando \_PATTERN. Os padrões da ROM são:

- |                |            |
|----------------|------------|
| 1 – 16 beats.  | 4 – Rock.  |
| 2 – Slow rock. | 5 – Disco. |
| 3 – Waltz.     | 6 – Swing. |

#### **SELVOICE** (declaração, SFG-BASIC)

Formato: CALL SELVOICE ( [<voz 1>] [,<voz 2>] ... [,<voz 8>] )

Função: Seleciona até 8 vozes escolhidas dos dados carregados pelo comando \_CLDVOICE e os executa. <voz x> deve corresponder ao número da voz criada pelo FM Voicing Program. Se as vozes forem definidas pelo comando \_MODISNT, os números serão 49 a 56 (estes números serão usados por padrão se os parâmetros de voz forem omitidos).

#### **SEND** (comando, Network-BASIC)

Formato: CALL SEND [(<nome da unidade>:] <nome do arquivo> [, <número do aluno>])]

Função: Envia o programa BASIC para (outros) computadores dos alunos. Esta instrução pode ser usada pelo professor e pelos alunos que foram autorizados pelo professor com CALL ENACOM. <nome da unidade> pode ser “A:” ou “B:” e <número do aluno> pode variar de 0 a 15. Se <nome do arquivo> for omitido, o programa BASIC que está na memória do micro remetente será enviado.

#### **SENDFILE** (comando, New Modem BASIC)

Formato: CALL SENDFILE(<variável string>), <variável numérica>

Função: Envia um arquivo usando um protocolo específico.  
 <variável string> contém o nome do arquivo a ser enviado  
 (pode incluir o nome da unidade, se omitido, o  
 arquivo será lido da unidade ativa atual).

<variável numérica> armazena o protocolo a ser usado:

- 1 – Xmodem
- 2 – Ymodem-1K
- 3 – Ymodem (permite apenas um arquivo por vez)

Ao retornar, <variável numérica> conterá o estado:

- 0 – Envio foi bem sucedido
- 1 – Sinal caiu (geralmente por conexão interrompida)
- 2 – Tempo limite atingido (o upload não foi iniciado)
- 3 – Operação abortada com CTRL+X
- 4 – Muitas pausas (tempos de espera)
- 5 – Não usado (sem efeito)
- 6 – Disco cheio
- 7 – Arquivo não encontrado
- 8 – Erro de gravação (disco protegido contra gravação ou não há disco)
- 9 – Arquivo vazio
- 10 – Excesso de tentativas

#### **SEOFF** (declaração, DM-System2 BASIC)

Formato: CALL SEOFF

Função: Interrompe a reprodução do efeito sonoro. Requer driver SE.

#### **SEON** (declaração, DM-System2 BASIC)

Formato: CALL SEON (<número>)

Função: Reproduz um efeito sonoro a partir de uma tabela.  
Requer driver SE.

<número> é o efeito sonoro a ser reproduzido. Pode variar de 0 a 255, sendo que 0 interrompe a reprodução.

#### **SET PCM** (comando, MSX-Audio)

Formato: CALL SET PCM (<número arq>, <número dispositivo>, <modo>, <parâmetro 1>, <parâmetro 2>, <freq amostragem>)

Função: Define parâmetros para os arquivos de áudio. Os parâmetros são definidos para os seguintes comandos:

CONVA	CONVP	COPY PCM
-------	-------	----------

LOAD PCM	MK PCM	PLAY PCM
----------	--------	----------

REC PCM	SAVE PCM	
---------	----------	--

<número arq> – Número do arquivo na memória de áudio.

Pode variar de 0 a 15.

<número dispositivo> – Segue a tabela abaixo:

disp.	Nome disp.	Modo	Parâmetro 1	Parâmetro 2
0	RAM externa	0/1	–	Tamanho
5	VRAM	0/1	Endereço	Tamanho

O endereço e o tamanho são definidos em unidades de 256 bytes.

<modo> – 0 – ADPCM, 1 – PCM

<freq de amostragem> – pode variar de 1.800 a 49.716 Hz para o ADPCM e de 1.800 a 16.000 Hz para o PCM.

#### **SETBIN** (comando, DM-System2 BASIC)

Formato: CALL SETBIN (@<endereço>)

Função: Especifica o endereço inicial da tabela binária de acordo com o sistema binário.

<endereço> é o endereço inicial da tabela binária. O bit menos significativo é ignorado. É necessário o uso de "@" na frente de <endereço>, caso contrário, ocorrerá erro porque a tabela não pode ser colocada na Main RAM.

#### **SETPLT** (comando, DM-System2 BASIC)

Formato: CALL SETPLT (<endereço>)

Função: Define o endereço inicial da tabela de paletas de cores. A tabela tem 32 bytes e o endereço padrão é C0000H.

#### **SETSE** (comando, DM-System2 BASIC)

Formato: CALL SETSE (<endereço>)

Função: Define o endereço inicial da tabela de efeitos sonoros. (Requer driver SE).

<endereço> é o endereço inicial da tabela (0 a FFFFH), sendo que o valor na inicialização é C000H.

#### **SIN** (função, DM-System2 BASIC)

Formato: CALL SIN (<variável>,<ângulo>,<valor>)

Função: Retorna o seno de um ângulo. O resultado é obtido pela multiplicação do seno do ângulo por um valor numérico.

<variável> – Variável numérica que receberá o resultado.

<ângulo> – é o valor do ângulo em graus.

<valor> – Número de dois bytes (valor inteiro).

**SJIS** (declaração, Kanji-BASIC)

Formato: CALL SJIS (<variável string>, <caracteres Kanji>)

Função: Converte um caractere em código JIS para um valor de 4 dígitos hexadecimais.

<variável string> – Recebe os 4 dígitos hexadecimais em ASCII

<caracteres Kanji> – Cadeia de caracteres Kanji de 2 bytes onde apenas o primeiro será convertido.

**SNDCMD** (comando, Network-BASIC)

Formato: CALL SNDCMD (<instrução>, [<número de aluno>])

Função: Envia instruções BASIC para o computador do aluno e as executa. CHR\$(13) é enviada ao final da instrução e é possível enviar várias separando-as com CHR\$(13). Esta instrução está disponível apenas para o professor. <número de aluno> pode variar de 1 a 15. Versão curta: \_SNDC.

**SNDMAIL** (comando, Network-BASIC)

Formato: CALL SNDMAIL (<número do aluno>)

Função: Envia dados da caixa de correio do professor para a caixa de correio de uma aluno. Esta instrução está disponível apenas para o professor. Caixas de correio são áreas especias de 256 bytes reservadas na NetRAM do professor e do aluno.

<número do aluno> pode variar de 1 a 15. Vs curta: \_SNDM.

**SNDRUN** (comando, Network-BASIC)

Formato: CALL SNDRUN ([[<nome da unidade>:] <nome do arquivo>] [, <número do aluno>] )

Função: Envia o programa BASIC para o computador do aluno e o executa. Esta instrução está disponível apenas para o professor. Se um aluno já tiver um programa BASIC na memória, ele será apagado e o aluno receberá um novo. <nome da unidade> pode ser “A:” ou “B:” e <número do aluno> pode variar de 0 a 15. Se <nome do arquivo> for omitido, o programa BASIC que está na memória do micro remetente será enviado. Versão curta \_SNDR.

**SOUND** (declaração, SFG-BASIC)

Formato: CALL SOUND (<nº instrumento>, <modo de controle>

[,<pitch>] [,<ajuste fino>] [,<velocidade>]  
[,<volume>] )

Função: Controla os instrumentos diretamente.  
 <nº instrumento> escolhe o instrumento dentre aqueles definidos pela instrução \_INST.  
 <modo de controle> pode ser:  
 0 – Sem key on / key off-line  
 1 – Key on (a Nota é audível)  
 2 – Key off (a Nota está em volume zero)  
 <pitch> pode variar de 25 a 120.  
 <ajuste fino> do pitch. Pode variar de 0 a 100.  
 <volume> pode variar de 0 a 100, sendo 100 o volume máximo (padrão).

#### **SPEAKEROFF** (comando, New Modem BASIC)

Formato: CALL SPEAKEROFF

Função: Desliga o alto-falante.

#### **SPEAKERON** (comando, New Modem BASIC)

Formato: CALL SPEAKERON

Função: Liga o alto-falante.

#### **SPOLOFF** (comando, Printer-BASIC)

Formato: CALL SPOLOFF

Função: Desativa o spooler de impressão mas não esvazia o buffer de 32 Kbytes. Para limpar o buffer temporário, é necessário o uso de LPRINT ou LLIST.

#### **SPOLON** (comando, Printer-BASIC)

Formato: CALL SPOLON

Função: Ativa o spooler de impressão, reservando um buffer de 32 Kbytes para o mesmo.

#### **STANBY** (declaração, SFG-BASIC)

Formato: CALL STANDBY

Função: Interrompe a reprodução temporariamente. Versão curta: \_STAN.

#### **START** (comando, Mega Assembler, SFG-BASIC)

Formato: CALL START [Mega Assembler]

Função: Chama o Mega Assembler inicializando as variáveis do mesmo. Para chamar o MA sem inicializar, use \_ASM.

**Formato:** CALL START [SFG-BASIC]

**Função:** Retoma a reprodução interrompida por \_STANDBY.

**Versão curta:** \_STAR.

**STATUS** (declaração, DM-System2 BASIC)

**Formato:** CALL STATUS

**Função:** Exibe a lista dos drivers instalados para o DM-System2.

**STDBY** (comando, Hitachi-BASIC versão 2)

**Formato:** CALL STDBY

**Função:** Coloca o leitor de dados interno do micro Hitachi MB-H2 no modo de espera/suspensão para economizar bateria.

**STOP** (comando, Hitachi-BASIC, Network-BASIC, SFG-BASIC)

**Formato:** CALL STOP [Hitachi-BASIC 2]

**Função:** Interrompe o movimento da fita no leitor de dados interno do micro Hitachi MB-H2.

**Formato:** CALL STOP (<número do aluno>) [Network-BASIC]

**Função:** Interrompe o programa BASIC em execução no micro do aluno. Esta instrução está disponível apenas para o professor. <número do aluno> pode variar de 1 a 15. Se for omitido ou igual a zero, a execução será interrompida em todos os micros dos alunos.

**Formato:** CALL STOP (<instrumento>) [SFG-BASIC]

**Função:** Suspende a reprodução de um instrumento específico e, opcionalmente, a digitalização do teclado musical (quando atribuído a um instrumento em vez de uma faixa pela instrução CALL PLAY). <instrumento> deve ser um número entre 1 e 4.

**STOPM** (declaração, MSX-Audio, MSX-Music)

**Formato:** CALL STOPM

**Função:** Interrompe a música tocada pelo MSX-Audio ou MSX-Music.

**SYMBOL** (declaração, Pioneer-BASIC)

**Formato:** CALL SYMBOL (X, Y), CHR\$(<código do caractere>), [<hor>], [<vert>], [<cor>], [<rotação>]

**Função:** Apresenta um caractere em Screen 2 nas coordenadas (X, Y). Os parâmetros opcionais são os seguintes:

<código do caractere> – Código ASCII do caractere  
 <hor> – Multiplicador de tamanho horizontal. Pode estar entre 1 e 32. Se omitido, o valor usado será 1.  
 <vert> – Multiplicador de tamanho vertical. Pode estar entre 1 e 24. Se omitido, o valor usado será 1.  
 <cor> – Código de cor de 0 a 15. Se omitido, será usada a cor definida pelo comando COLOR.  
 <rotação> define a rotação do caractere:  
     0 – Sem rotação.  
     1 – Rotação de 90 graus à direita.  
     2 – Rotação de 180 graus à direita.  
     3 – Rotação de 270 graus à direita.

#### **SYNCOUT** (comando, SFG-BASIC)

Formato: CALL SYNCOUT

Função: Envia um sinal de sincronização para o gravador de fita cassete. Versão curta: \_SYNC.

#### **SYSOFF** (comando, DM-System2 BASIC)

Formato: CALL SYSOFF

Função: Desinstala o DM-System2 BASIC e volta para o MSX-BASIC padrão.

#### **SYSON** (comando, DM-System2 BASIC)

Formato: CALL SYSON

Função: Inicializa o DM-System2 BASIC.

#### **SYSTEM** (comando, Disk-BASIC, DM-System2 BASIC)

Formato: CALL SYSTEM [Disk-BASIC]

Função: Chama o MSXDOS.

Formato: CALL SYSTEM [("["<dispositivo>:]["<caminho>"][[\]<nome de arquivo>])"] [Disk-BASIC 2]

Função: Chama o MSXDOS2, opcionalmente executando o arquivo especificado ou entrando no subdiretório.

Formato: CALL SYSTEM [DM-System2 BASIC]

Função: Desinstala o DM-System2 e volta ao MSX-BASIC padrão.

Se precedido por CALL SYSOFF, desinstala o DM-System2 e chama o MSXDOS.

**TABOFF** (comando, Hitachi-BASIC versão 3)

Formato: CALL TABOFF

Função: Desativa o aplicativo Drawing Tablet no micro Hitachi MB-H3.

**TABON** (comando, Hitachi-BASIC versão 3)

Formato: CALL TABON

Função: Inicia o aplicativo Drawing Tablet no micro Hitachi MB-H3.

**TALK** (declaração, Network BASIC)

Formato: CALL TALK (<mensagem>, [<número micro>])

Função: Envia uma mensagem de até 56 caracteres para o professor ou outro aluno. Esta instrução está disponível apenas para os alunos. <número micro> pode variar de 1 a 15 e pode ser obtido através de CALL WHO. Se for 0, a mensagem será enviada para o professor. Se, após enviar a mensagem, <número micro> contiver 255, houve falha, se contiver 0, a mensagem foi enviada com sucesso.

**TDIAL** (comando, New Modem BASIC, SVI Modem BASIC)

Formato: CALL TDIAL (<variável string>), <variável numérica>

[New Modem BASIC]

Função: Chama um número de telefone específico via discagem por tom. Esta instrução pode ser usada apenas em um programa Terminal. <variável de string> armazena o número de telefone a ser chamado, onde são permitidos apenas os caracteres “0 123 456 789-AaBbCcDd \*#” (o caractere “-” corresponde a uma espera de 1 segundo). <variável numérica> retorna o estado: se for 0, o valor de entrada está correto; se for “1” não está.

Formato: CALL TDIAL(“<número telefone>”) [SVI Modem BASIC]

Função: Chama um número de telefone específico via discagem por tom. <número telefone> deve estar entre aspas e são permitidos apenas os caracteres “0 123 456 789AaBbCcDd ”.

**TEMPER** (declaração, MSX-Music e MSX-Audio)

Formato: CALL TEMPER (<n>)

Função: Define o modo bateria para o OPLL. <n> pode variar de 0 a 21, cujo significado é o seguinte:

- |                              |                             |
|------------------------------|-----------------------------|
| 0 – Pythograph.              | 11 – Ritmo puro Cis+ (B-).  |
| 1 – Mintone.                 | 12 – Ritmo puro D+ (H-).    |
| 2 – Welkmeyster.             | 13 – Ritmo puro Es+ (C-).   |
| 3 – Welkmeyster (ajustado).  | 14 – Ritmo puro E+ (Cis-).  |
| 4 – Welkmeyster (separado).  | 15 – Ritmo puro F+ (D-).    |
| 5 – Kilanbuger.              | 16 – Ritmo puro Fis+ (Es-). |
| 6 – Kilanbuger (ajustado).   | 17 – Ritmo puro G+ (E-).    |
| 7 – Velotte Young.           | 18 – Ritmo puro Gis+ (F-).  |
| 8 – Lamour .                 | 19 – Ritmo puro A+ (Fis-).  |
| 9 – Ritmo perfeito (padrão). | 20 – Ritmo puro B- (G-).    |
| 10 – Ritmo puro C+ (A-).     | 21 – Ritmo puro H- (Gis-).  |

### **TEMPO** (declaração, SFG-BASIC)

Formato: CALL TEMPO (<valor de tempo>)

Função: Define o “tempo” em unidades de quartos de Nota que serão reproduzidas em um minuto. Pode variar de 0 a 200, sendo que 0 interrompe a reprodução. Versão curta: \_TEMP.

### **TERMINAL** (comando, New Modem BASIC)

Formato: CALL TERMINAL (<variável numérica>)

Função: Permite comunicar com um BBS. Quase todas as teclas pressionadas são enviadas pela linha telefônica e o que vem da linha telefônica é exibido na tela. Esta instrução pode ser usada apenas em um programa Terminal.

<variável numérica> armazena o estado:

1 – O sinal caiu (geralmente a conexão está interrompida).

5 – Foi recebido o caractere de login automático (do BBS).

11 – Foi pressionada a tecla HOME para voltar ao BASIC.

Pode ser usada para voltar ao menu Terminal.

220 – Foram pressionadas GRAPH+I para voltar ao BASIC.

Podem ser usadas para enviar manualmente o nome

e a senha para um BBS sem login automático.

### **TIMER** (comando, SFG-BASIC)

Formato: CALL TIMER (<período> [,<nº de marca>] )

Função: Inicia e define o período do timer.

<período> é definido em unidades de 1/100 segundos e pode variar e 1 a 24.000.

<nº de marca> pode ser qualquer número entre 1 e 254. Se omitido, será usado o número 11.

**TRACE OFF** (comando, MSX Aid BASIC)

Formato: CALL TRACE OFF

Função: Interrompe o rastreamento de execução do programa.

**TRACE ON** (comando, MSX Aid BASIC)

Formato: CALL TRACE ON

Função: Inicia o rastreamento da execução do programa da mesma forma que a instrução TRON mas, sempre que a execução saltar para outra linha, envia o número da linha executada para a impressora.

**TRACK** (declaração, SFG-BASIC)

Formato: CALL TRACK (<nº de trilhas>)

Função: Define o número de trilhas usadas por \_PHRASE ou \_PLAY. <nº de trilhas> por variar de 1 a 8; se omitido será usado o valor 1. Versão curta: \_TRAC.

**TRANSPOSE** (declaração, MSX-Music e MSX-Audio, SFG-BASIC)

Formato: CALL TRANSPOSE (<n>) [MSX Music/Audio]

Função: Muda de clave. <n> pode variar de -12 799 a +12 799, sendo que 100 unidades correspondem a meio tom. O valor padrão é 0.

Formato: CALL TRANSPOSE (<n>) [SFG-BASIC]

Função: Muda de clave. <n> pode variar de -12 a +12 em incrementos de meio tom. O valor padrão é 0.

**TSTOP** (comando, SFG-BASIC)

Formato: CALL TSTOP

Função: Interrompe o timer. Versão curta: \_TSTO. A instrução \_INIT também interrompe o timer.

**TUNE** (comando, SFG-BASIC)

Formato: CALL TUNE (<valor numérico>)

Função: Sintoniza o sistema FM Tone Generation com os outros instrumentos. <valor numérico> pode variar de -100 a +100, valor este que corresponde a um semitom.

**UPPER** (função, DM-System2 BASIC)

Formato: CALL UPPER (<variável>, <string alfanumérica>)

Função: Converte os caracteres alfabéticos da <string alfanumérica> para maiúsculas e retorna em <variável>.

**USBCD** (comando, RookieDrive-BASIC)

Formato: CALL USBCD ("<diretório>")

Função: Troca o diretório ativo no dispositivo USB.

**USBERROR** (declaração, RookieDrive-BASIC)

Formato: CALL USBERROR

Função: Exibe o código de erro armazenado sempre que uma transação USB falha por qualquer motivo. Apenas o erro da última transação USB executada fica armazenado.

**USBFILES** (comando, RookieDrive-BASIC)

Formato: CALL USBFILES

Função: Exibe a lista de imagens de disco que estão no diretório raiz da unidade virtual USB. A execução desta instrução coloca o disco no estado "off-line". Para voltar ao estado "on-line", use CALL INSERTDISK ou CALL REBOOT.

**USBRESET** (comando, RookieDrive-BASIC)

Formato: CALL USBRESET

Função: Repete o procedimento de inicialização que é realizado no momento em que uma unidade de disquete USB padrão é conectada a uma interface do Rookie Drive.

**USERHYTHM** (declaração, SFG-BASIC)

Formato: CALL USERHYTHM

Função: Habilita os instrumentos de ritmo (bateria) para uso.

Estes instrumentos usam duas vozes FM; por isso o número de vozes disponíveis cai de 8 para 6 com o ritmo habilitado.

Versão curta: \_USER.

**USR** (comando, Nextor)

Formato: CALL USR (<endereço de execução>, [<registradores>] )

Função: Chama uma rotina em Assembler, opcionalmente carregando os registradores com valores específicos antes. <endereço de execução> é o endereço inicial da rotina. Se for especificado “-1”, a rotina apenas retornará sem erro (útil para detectar o Nextor no BASIC).

<registradores> é um apontador para um buffer de 12 bytes onde os valores dos registradores são especificados na sequência “F, A, C, B, E, D, L, H, IXl, IXh, IYl, IYh”.

**VARLIST** (comando, MSX Aid BASIC)

Formato: CALL VARLIST [([<variável>] [, P])]

Função: Exibe uma lista com todas as variáveis já usadas pelo programa MSX-BASIC que está na memória. Se <variável> for especificada (1 ou 2 caracteres) serão listados os números de linha com a variável em questão. Se o segundo caractere for um asterisco (\*), serão consideradas todas as variáveis que começam com o primeiro caractere. Com o parâmetro P, os dados serão enviados para a impressora. Sem nenhum parâmetro, a lista completa será exibida na tela.

**VCOPY** (declaração, DM-System2 BASIC)

Formato: CALL VCOPY (<X0>,<Y0>)-(<X1>,<Y1>)[,<PgF>] TO  
                   (<X2>,<Y2>-<X3>,<Y3>)[,<PgD>] [,<R>]  
                   [ON (<X4>,<Y4>)] [,<operador lógico>]

Função: Copia uma área retangular da VRAM para outra com zoom in/out e rotação.

<X0> – Coordenada X do primeiro ponto da área fonte.

<Y0> – Coordenada Y do primeiro ponto da área fonte.

<X1> – Coordenada X do segundo ponto da área fonte.

<Y1> – Coordenada Y do segundo ponto da área fonte.

<PgF> – página fonte da VRAM

<X2> – Coordenada X do primeiro canto da área destino.

<Y2> – Coordenada Y do primeiro canto da área destino.

<X3> – Coordenada X do canto oposto da área destino.

<Y3> – Coordenada Y do canto oposto da área destino.

<PgD> – página destino da VRAM

<R> – Rotação em graus no sentido horário

<X4> – Coordenada X do eixo de rotação (X2 é padrão)

<Y4> – Coordenada Y do eixo de rotação (Y2 é padrão)

Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.

<LO> é o operador lógico e pode ser [T]PSET, [T]PRESET, [T]XOR, [T]OR ou [T]AND. O padrão é PSET.

**VDPWAIT** (comando, DM-System2 BASIC)

Formato: CALL VDPWAIT

Função: Espera até o VDP terminar de executar o comando.

**VER** (declaração, Hangul-BASIC 4)

Formato: CALL VER

Função: Apresenta a versão do Hangul-BASIC.

**VERIFY** (comando, Disk-BASIC)

Formato: CALL VERIFY [ON | OFF]

Função: Ativa ou desativa a verificação de escrita no disco.

**VIDEO** (função, Pioneer-BASIC)

Formato: CALL VIDEO (<variável>)

Função: Retorna em <variável> o tipo de seleção de vídeo atualmente ativo. O valor retornado pode ser:  
 0 – Tela do computador (sincronização interna)  
 1 – Superimpose  
 2 – Vídeo externo

**VLIST** (declaração, SFG-BASIC)

Formato: CALL VLIST

Função: Apresenta a tabela de instrumentos na tela.

**VMOFF** (comando, DM-System2 BASIC)

Formato: CALL VMOFF [(<nº do segmento>)]

Função: Aborta a operação de macro.

**VMON** (comando, DM-System2 BASIC)

Formato: CALL VMON (<endereço inicial>,[<valor inicial>])

Função: Operação de macro do processamento do VDP.  
 <endereço inicial> especifica o início do código macro.  
 <valor inicial>, se especificado, faz com que a operação de macro só seja iniciada após seu armazenamento na variável de macro do VDP.

**VMWAIT** (comando, DM-System2 BASIC)

Formato: CALL VMWAIT

Função: Coloca o sistema em espera até a operação de macro do VDP ser concluída. CTRL+STOP podem ser usadas para sair deste comando.

**VOICE** (declaração, MSX-Music e MSX-Audio)

Formato: CALL VOICE ([@<n1>],[@<n2>], .... [@<n9>])

Função: Especifica os instrumentos que serão usados em cada voz.  
 <nx> pode variar de 0 a 63. O valor padrão é 0.

**VOICE COPY** (declaração, MSX-Music e MSX-Audio)

Formato: CALL VOICE COPY (@<n1>,-<n2>)

Função: Copia dados referentes aos instrumentos de/para uma variável matriz tipo DIM A%(16). <n1> é a fonte e <n2> o destino. <n1> pode variar de 0 a 63 e <n2> só pode ser 63, ou <n1> e <n2> podem ser uma variável matriz.

**WAIT** (comando, DM-System2 BASIC, SFG-BASIC)

Formato: CALL WAIT (<tempo>) [DM-System2 BASIC]

Função: Espera um tempo definido. Pode ser abortado por CTRL+STOP. <tempo> é definido em unidades de 1/60 segundos e pode variar de 0 a 32 767.

Formato: CALL WAIT (<nº evento>) [SFG-BASIC]

Função: Suspende a interrupção quando a melodia está sendo reproduzida. <nº evento> pode ser:

1 ~ 4 – Suspende durante a reprodução do instrumento respectivo.

5 – Suspende durante a reprodução do ritmo.

6 – Suspende até o tempo do timer zerar.

**WHO** (declaração, Network BASIC)

Formato: CALL WHO (<número micro>)

Função: Retorna o número do micro na rede MSX. <número micro> pode variar de 0 a 15, onde 0 é o micro do professor.

**XREF** (declaração, MSX Aid BASIC)

Formato: CALL XREF {[<número da linha>] [, P]}

Função: Exibe uma lista com todas as linhas vinculadas de um programa MSX-BASIC que está na memória (Instruções GOSUB, GOTO, RESUME, RESTORE, RETURN). <número da linha> é usado para limitar a lista para um número de linha especificado. Com o parâmetro P, os dados serão enviados para a impressora. Sem nenhum parâmetro, a lista completa será exibida na tela.

**XY** (comando, DM-System2 BASIC)

Formato: CALL XY (<coordenada X>,<coordenada Y>)

Função: Altera as coordenadas do acumulador gráfico.

**YMMM** (declaração, DM-System2 BASIC)

Formato: CALL YMMM (<X0>,<Y0>)-[STEP](<X1>,<Y1>) TO  
<X2>,<Y2>)

Função: Executa o comando YMMM (cópia rápida em bytes na direção Y) do VDP. Disponível para as Screens 5 a 12.  
 <X0> – Coordenada X do primeiro ponto da área fonte.  
 <Y0> – Coordenada Y do primeiro ponto da área fonte.  
 <X1> – Coordenada X do segundo ponto da área fonte.  
 <Y1> – Coordenada Y do segundo ponto da área fonte.  
 <X2> – Coordenada X esquerda da área destino.  
 <Y2> – Coordenada Y superior da área destino.  
 STEP, se especificado, indica coordenadas relativas.  
 Obs.: <X> pode variar de 0 a 511 e <Y> de 0 a 1023.

### 3.4 – CÓDIGOS DE ERRO DO MSX-BASIC

Nº	Original Inglês	Português
01	NEXT without FOR	NEXT sem FOR
02	Syntax error	Erro de sintaxe
03	RETURN without GOSUB	RETURN sem GOSUG
04	Out of DATA	Sem 'DATA'
05	Illegal function call	Chamada ilegal de função
06	Overflow	Overflow
07	Out of memory	Falta memória
08	Undefined line number	Número de linha não definido
09	Subscript out of range	Índice fora do limite
10	Redimensioned array	Array redimensionado
11	Division by zero	Divisão por zero
12	Illegal direct	Direto ilegal
13	Type mismatch	Tipo desigual
14	Out of string space	Falta área para string
15	String too long	String muito longa
16	String formula too complex	String muito complexa
17	Can't CONTINUE	Não pode continuar
18	Undefined user function	Função de usuário não definida
19	Device I/O error	Erro de dispositivo I/O
20	Verify error	Verificar erro
21	No RESUME	Sem RESUME

22	RESUME without error	RESUME sem erro
23	Unprintable error	Erro indefinido
24	Missing operand	Falta operando
25	Line buffer overflow	Linha muito longa
26~49	Unprintable error	Erro indefinido
50	FIELD overflow	Campo maior
51	Internal error	Erro interno
52	Bad file number	Número de arquivo ruim
53	File not found	Arquivo não encontrado
54	File already open	Arquivo já aberto
55	Input past end	Fim de arquivo
56	Bad file name	Nome de arquivo ruim
57	Direct statement in file	Comando direto no arquivo
58	Sequential I/O only	Acesso sequencial somente
59	File not OPEN	Arquivo não aberto
60	Bad FAT	Erro na FAT
61	Bad file mode	Modo de arquivo errado
62	Bad drive name	Nome de drive errado
63	Bad sector	Setor com erro
64	File still open	Arquivo já aberto
65	File already exists	Arquivo já existe
66	Disk full	Disco cheio
67	Too many files	Diretório cheio
68	Disk write protected	Disco protegido contra escrita
69	Disk I/O error	Erro de I/O de disco
70	Disk offline	Sem disco
71	RENAME across disk	RENAME em discos diferentes
72	File write protected	Arq. protegido contra escrita
73	Directory already exists	Diretório já existe
74	Directory not found	Diretório não encontrado
75	RAM disk already exists	RAMDISK já existe
76	Invalid device driver *	Driver inválido
77	Invalid device or LUN *	Dispositivo ou LUN inválido
78	Invalid partition number *	Número de partição inválido
79	Partition already in use *	Partição já em uso
80~255	Unprintable error	Erro indefinido

Obs. Os códigos marcados com “\*” (76 a 79) são exclusivos do Nextor.

## 4 – MSXDOS

NOME DO COMANDO (tipo do comando, versão do COMMAND)

Formato: Formatos válidos para o comando

Função: Forma de operação do comando

Comandos internos são comandos executados diretamente pelo COMMAND.COM, e os externos são carregados do disco.

A versão do COMMAND assinala a versão para a qual o comando está implementado. Valores separados por “–” indicam que há diferenças de sintaxe ou comportamento para versões diferentes. A seguir há uma curta descrição das versões.

- 1 - MSXDOS versão 1.0.
- 2 - MSXDOS versão 2.0 (Command até versão 2.3).
- 2.41 - MSXDOS versão 2.0 (Command versão 2.41).
- N - NEXTOR.
- K - Necessário Kanji-ROM.

### 4.1 – NOTAÇÕES DE FORMATO

<nome do arquivo> – Nome de arquivo na forma: A:\dir1\dir2\arquivo.ext

<nome arquivo composto> – Vários nomes de arquivos no formato acima

<caminho> – Caminho na forma: A:\dir1\dir2\

[ ] delimita parâmetro opcional.

| significa que apenas um dos itens pode ser utilizado (OR).

{ } delimita opção.

Caracteres entre parênteses após algumas opções de alguns comandos indicam a versão do COMMAND para a qual aquela opção está disponível.

Um <dispositivo> pode ser:

CON	Console (Teclado)	NUL	Nulo
CRT	Vídeo	AUX	Auxiliar
PRN	Impressora	COM	Porta serial

Ou qualquer outro que esteja instalado.

#### **4.1.1 – Descrição das extensões de nomes de arquivos**

- ACC Arquivos de dados de acompanhamento do Music Creator.
- APT Arquivo de dados de padrões do Studio FM.
- ARC Arquivo(s) compactado(s) no formato ARC pela System Enhancement Associates (SEA). As ferramentas para extrair são UNARC.COM (v1.6) e UNP.COM (v1.0 por Pierre Gielen).
- ARC Arquivo(s) compactado(s) no formato ARC russo, incompatível com o formato ARC da SEA. As ferramentas para extrair são XARC.COM (v1.01) e ARCDE.COM (v1.03).
- ARJ Arquivo(s) compactado(s) no formato ARJ. As ferramentas para extrair são UNARC.COM (v1.10) e UNP.COM (v1.0 por Pierre Gielen).
- ASC Texto plano (formato ASCII) que pode conter um programa ou dados BASIC.
- ASM Arquivos com texto Assembler.
- ASN Arquivos de atribuição para MIDI Blaster
- BAS Arquivo de texto BASIC tokenizado. Podem ser executados a partir do MSX-DOS com o comando BASIC nome.bas.
- BAT Arquivos em lote (texto simples) interpretados pelo MSX-DOS.
- BGM Arquivo de música binário MuSICA. MuSICA é um software desenvolvido pela ASCII para criar músicas em 17 vozes com PSG, FM e SCC da Konami.
- BGM Arquivo de música MSX-FAN. Não deve ser confundido com arquivos MuSICA. Músicas neste formato estavam contidas em todas as suas revistas em disco. Mais tarde um reproduutor específico foi lançado que até suportava a reprodução em MIDI.
- BGM Arquivo carregável MSX-MUSIC criado pelo utilitário BIT2BGM.COM de Uwe Schröder que converte arquivos musicais Synth Saurus.

- BIN** Arquivo binário criado com a instrução BASIC BSAVE. Carrega com BLOAD. O cabeçalho tem um comprimento de 7 bytes (FEh + endereço inicial + endereço final + endereço de execução). Pode conter código de máquina e/ou dados.
- BMP** Arquivo de imagem em formato bit map. Pode ser visualizado em Screen 7 ou 8 com BMP.COM (v1.01 da SEIGA).
- BOK** Arquivo de imagens do MSXView.
- BTM** Arquivos Batch suportados por MSX-DOS 2 v.2.40 ou posterior.
- CAS** Imagem de programas ou dados de fita cassete. O SofaCas (PC) permite converter software em fita para arquivo CAS e também reproduzi-lo usando uma saída de som de PC ligado na entrada de cassete MSX. No MSX turbo R, podemos usar TRCAS (de Martos) para rodar CAS tocado por SofaCas.
- CMP** Imagem de Screen 5 compactada, incluindo paleta, criada com DD-Graph (Graphic Dot Designer).
- CMP** Imagem compactada, incluindo paleta, criada com GIOS (Graphical Input/Outuput System).
- COM** Comando contendo um executável binário no MSX-DOS. Também pode ser um arquivo executável compactado com POPCOM.COM (v1.0 por Perpermint-Star).
- CPM** Arquivo .COM renomeado para .CPM, para ser usado em alguns emuladores de CPM. Basta renomeá-los como .COM para poder executá-los.
- DRM** Arquivo para o editor de bateria do tracker First Rate Music Hall.
- DAT** Arquivo de configuração do Sintetizador para MIDI Blaster ou arquivo de dados para outros softwares.
- DSK** A imagem de disco para emuladores. É necessário uma ferramenta específica para executar ou gravar no disco normal. Pode ser executado no MSX real com SofaRunit ou usando o comando EMUFILE do Nextor.
- DUA** Arquivo de dados duplo Music-BOX (melodia + samples)

- EDI Arquivo para o editor musical do tracker First Rate Music Hall.
- EMx Imagem de disco para o emulador de disquete (HDDEMUCOM) para MSX Turbo R de Tsuyoshi. A estrutura interna é a mesma dos arquivos DSK. Os discos protegidos têm informações adicionais armazenadas em arquivos com extensão HED.
- EVA Arquivo de vídeo em formato EVA.
- EVG Arquivo de dados de evento do cartucho Yamaha SFG-05.
- FM Arquivo MSX-MUSIC BASIC.
- FMP Arquivo MSX-MUSIC BASIC.
- FMS Arquivo de som Synth Saurus.
- FNT Arquivo de fonte para o utilitário Scroll Power.
- FON Fonte do MSXView.
- G9B Biblioteca de formatos gráficos para GFX-9000.
- GE5 Sinônimo para .SC5. Veja a descrição do arquivo .SCx.
- GEN Texto simples que contém o código-fonte do assembly Z80, usado com o compilador GEN80.
- GIF Graphics Interchange Format. Pode ser visualizado com GIFI.COM (por Kakami Hiroyuki) e convertidos para o formato MSX com ENGIN.COM (v1.2 por Pierre Gielen). Ver também SHOWEM.COM (por Steven van Loef) e GIFTDUMP.COM (por Francesco Duranti).
- GLx Arquivo de imagem do Graph Saurus no mesmo formato usado pela instrução BASIC COPY.
- GRA Arquivo de imagem em formato QLD. Pode ser visualizado pelo aplicativo BLS.COM (v2.00 da SEIGA).
- GRP Sinônimo para .SC2. Ver a descrição do arquivo .Scx. Também pode ser uma imagem compactada para Graph Saurus.
- GZ Arquivo compactado no formato GZIP. A ferramenta para extrair é o GUNZIP.COM.

- HLP Arquivo de ajuda MSX-DOS 2 (texto simples).
- INS Arquivo para o editor de instrumentos do tracker First Rate Music Hall.
- IPS Patch para arquivo.
- ISH Arquivo compactado.
- JPG Arquivo de imagem compactado no formato JPEG. Alguns visualizadores podem mostrar imagens até 1024x1024: JPD.COM (v0.23 da API), JLD.COM (v1.11 da SEIGA) ou BLS.COM (v2.00 da SEIGA). O arquivo JPEG pode ser produzido em MSX a partir de imagens SCREEN 12 com JSV.COM (v0.1 por SEIGA).
- KSS Arquivo de música MSX que contém também o código do reprodutor. Use KSSPLAY.COM (por NYYRIKKI) para reproduzir.
- LDR Arquivo em BASIC tokenizado, geralmente usado para carregar e executar um programa que consiste em vários arquivos BASIC.
- LHA Arquivo(s) compactado(s) no formato LHA. As ferramentas para criar um arquivo LHA são LHPACK.COM (v1.03 por H.Saito) ou LHA.COM (v1.05a por Kyouju). Para extrair, use PMM.COM (v1.20 por lita), LHARC.COM e LHEXT.COM (v1.33 por Kyouju).
- LPF Arquivo Loop para o utilitário Scroll Power.
- LZH Sinônimo de LHA.
- MAG Arquivo de imagem Maki-chan V2 usado no PC-9801 e Sharp X68000. Visualizável com BLS.COM (v2.00 por SEIGA).
- MAX Sinônimo para MAG. MSX-DOS
- MBK Arquivo Samplekit para o tracker de música MoonBlaster.
- MBM Arquivo de música para o tracker de música MoonBlaster.
- MBS Arquivo de amostra para o tracker de música MoonBlaster.
- MBV Arquivo de voz para o tracker de música MoonBlaster.
- MBW Arquivo Wave para o tracker de música MoonBlaster.
- MCM Arquivo de música Micro Cabin. Reproduzido por MCDRV.EXE.

- MDT Arquivo de dados de música do MSX Music-System
- MDX Arquivo de música em um formato do Sharp X68 000. Pode ser reproduzido por MPX2.COM (quando o driver instalado com MXDRV.COM). Os arquivos PDX opcionais são amostras PCM. Requer o cartucho YAMAHA SFG-01/05 ou o cartucho MFP PCM.
- MEG Texto simples com o código-fonte em Assembly Z80 do Mega Assembler. Também usada para imagens Mega-Rom.
- MEL Arquivo de dados de melodia Music-BOX.
- MFM Música FM para MoonBlaster.
- MGS Arquivo de música em formato desenvolvido pela AIN. Reproduzido por MGSEL.COM (quando o driver instalado com MGSDRV.COM).
- MID Arquivo padrão MIDI (pode ser reproduzido usando a interface MIDI ou o software MoonSound).
- MIF Arquivo de imagem compactado (MSX Image File). Pode ser visualizado por MIFVIEW.COM.
- MIO Arquivo de música MIODRV. Reproduzido por MIODRV Player.
- MKI Arquivo de imagem Maki-chan V1 usado no Sharp X68 000. Visualizável com BLS.COM (v2.00 por SEIGA).
- MOD Arquivo Amiga MOD (pode ser tocado no MSX turbo R ou MoonSound).
- MP3 Arquivo MPEG Audio Layer III. MP3s podem ser reproduzidos com o reprodutor Sunrise MP3, MPX Cartridge r1.1 da Junsoft ou SE-ONE da TMT Logic.
- MPK Música Music Player K-kaz. Requer WAMPK Player.
- MSx Arquivo de pontuação do Synth Saurus.
- MSD Arquivo de música de origem MuSICA (MML). Também pode ser usado o compilador alternativo KINROU4 (por Masarun).
- MUE Arquivo de música do HAL Music Editor.

- MUS Arquivo de música FAC Soundtracker.
- MUS Arquivo MML de origem MGSDRV. Precisa ser compilado em um arquivo MGS com MGSC.COM. OTOH, MGSCR.COM pode descompilar arquivos MGS de volta para a fonte MUS.
- MUS Arquivo de música Studio FM (não recomendado)
- MWK Sample kit para MoonSound Wave.
- MWM Música para MoonBlaster (MoonSound Wave).
- OPX Driver de música para o OPLL.
- PAC Dump da SRAM (salvar jogos) do cartucho PAC ou FM-PAC.
- PAT Arquivo de padrões do Studio FM.
- PCM Arquivo samples de sons para MSX-Turbo R.
- PCK Arquivo empacotado para o tracker First Rate Music Hall. Inclui 4 sons com todos os instrumentos e dados de bateria.
- PCT Arquivos de página do Dynamic Publisher.
- PDX Arquivo de amostra PCM opcional usado com um arquivo MDX. Reproduzido com PDXLOAD.COM da AIN.
- PIC Imagem gerada pelo Phillips Video Graphics. Sinônimo de .SC8. Também é o formato de imagem específico do X68 000, sendo visualizável com BLS.COM (v2.00 por SEIGA).
- PLx Arquivo de paleta de cores Graph Saurus em formato Raw (contém 8 conjuntos de paletas com dois bytes por cor 'RG' e 'OB'). É um complemento para o respectivo arquivo .SRx.
- PMA Arquivo(s) compactado(s) no formato PMARC. As ferramentas para criar o arquivo são PMARC.COM, PMARC2.COM (v2.0 por Sybex) e UNP.COM (v1.0 por Pierre Gielen). Extrair com PMM.COM (v1.20 por lita), PMEXE.COM (v2.0) e PMEXT.COM. O PMEXT foi portado para Windows (v1.21 por Yoshihiko Mino).
- PRO Arquivo de música para o Pro-Tracker (por Tyfoon Soft).
- PSG Arquivo de sample para o PSG Sampler.

- RDT Arquivo de dados de ritmo do MSX Music-System.
- RLT Arquivos de dados em tempo real do Music Creator.
- ROM Imagem bruta de ROM.
- RTM Arquivo de ritmo do Synth Saurus.
- S1x Contém as linhas ímpares de uma imagem entrelaçada. Para obter mais informações, consulte o arquivo SCx.
- S3M Ver arquivo MOD.
- SAM Arquivo de dados de amostra Music-BOX (usado como arquivo drumkit no Music Creator)
- SBM Dados de música para chips de som SCC e PSG.
- SBS Dados de instrumentos para chips de som SCC e PSG.
- SCx Arquivo de imagem binário Screen-x. Pode ter um arquivo .S1x complementar que conterá as linhas extras entrelaçadas para dobrar a resolução vertical. Usado por editores de imagem ou instrução BLOAD com o parâmetro ,S. Imagens SC2 podem ser visualizadas sob MSX-DOS com SC2VIEW.COM (por GDX), e arquivos SC5 para SCC com BLS.COM (por SEIGA).
- SCR Imagem Screen-2 criada com Graphos III. Arquivo executável com carregador que produz um efeito. Pode ser executado no MSX-BASIC com BLOAD ,R.
- SDT Arquivo de dados de som MSX Music-System (= dados de voz)
- SDT Arquivo de música SCMD para MSX feito um compilador MML para Windows. Pode ser reproduzido com SC.COM.
- SEE Efeitos sonoros usados pelo “Sound Effect Editor” (Shareware por Fuzzy Logic).
- SEQ Arquivos de dados de sequência do Music Creator.
- SFM Arquivo de música Studio FM.
- SMx Arquivo de samples do FAC Soundtracker.
- SMP Arquivo de samples para Covox / SIMPL ou MSX Turbo R.

- SNG Arquivo de música para o editor SCC-Musixx da Tyfoon Soft.
- SPT Arquivos de dados de tempo do Music Creator.
- SPT Arquivo de texto para o utilitário Scroll Power.
- SRx Arquivo de imagem Graph Saurus. Requer o respectivo arquivo .PLx. Pode ser opcionalmente compactado em tempo de execução. Arquivos descompactados podem ser carregados no MSX-BASIC com um BLOAD "FILE.SRx", S, mas a paleta externa deverá ser carregada com OPEN#n.
- STP Arquivos de carimbo do Dynamic Publisher. Contém uma imagem que pode ser carregada em uma página.
- TLx Arquivo de blocos do Graph Saurus.
- TSR Programas “Terminate and Stay Resident” para serem usados com MemMan 2.0 e superior.
- TXT Arquivo de texto simples geralmente codificado em ASCII, Ank ou JIS.
- VCD Arquivo de voz para o MSX Voice Recorder (HAL Laboratory).
- VCD Arquivo de voz do MuSICA.
- VGM Arquivo de música que suporta muitos chips de som. É reproduzido por VGMPPLAY.COM (por Laurens Holst).
- VOC Arquivos de dados de voz do Music Creator.
- VOC Arquivo de dados de voz Studio FM.
- VOG Arquivo de dados de voz Yamaha SFG-05.
- WAV Arquivo de sample de som. Pode ser reproduzido com o cartucho MPX r1.1 da Junsoft.
- WB Arquivo de projeto Assembler. Usado pelo assembler “The WBASS2 Z80 Assembler”.
- XM Ver arquivo MOD.
- XPC Arquivo de patch ROM para EXECROM.COM (A&L Software).

**ZIP** Arquivo(s) compactado(s) em formato ZIP por compactadores de PC. A melhor ferramenta para extrair é SUZ.COM (v1.3 por Loutrax).

## 4.2 – DESCRIÇÃO DOS COMANDOS

### **ALIAS** (interno, 2.41)

Formato: ALIAS [/P] [nome] [=] [valor] | /R | {/L | /S} <nome arquivo>

Função: Apresenta ou define comando alias.

[/P] pausa a listagem ao completar uma tela.

[/R] remove todos os alias definidos.

[/L] carrega um alias definido em <nome do arquivo>.

[/S] salva o alias corrente no arquivo <nome do arquivo>.

[nome] é o nome do novo comando.

[valor] é o comando ou string que será atribuída a [nome].

<nome do arquivo> é o arquivo em disco onde serão gravados ou de onde serão recuperados os alias definidos.

### **ASSIGN** (interno ,2)

Formato: ASSIGN [d1: [d2:]]

Função: Redireciona acesso ao drive d1: para o drive d2:

### **ATDIR** (interno, 2)

Formato: ATDIR +|-H [/H] [/P] <nome do arquivo>

Função: Ativa/desativa atributos de diretório oculto.

[/P] pausa as mensagens de erro ao completar uma tela.

+H marca arquivo como oculto.

-H desliga o atributo de arquivo oculto, devendo obrigatoriamente ser seguido por /H.

### **ATTRIB** (interno, 2-2.41)

Formato: ATTRIB {+|-H | +|-R | +|-S | +|-A} [/H] [/P] <nome arquivo>

Função: Altera atributos de arquivo oculto (H) somente leitura (R), arquivo de sistema (S, 24.1 somente) ou arquivado (A, 2.41 somente). “-H” deve ser usando com “/H”.

[/P] pausa as mensagens de erro ao completar uma tela.

**BASIC** (interno, 1)

Formato: BASIC [<nome prog>]

Função: Transfere o controle ao interpretador BASIC e opcionalmente carrega e executa o programa <nome prog>.

**BEEP** (interno, 2.41)

Formato: BEEP

Função: Gera um beep.

**BOOT** (interno, 2.41)

Formato: BOOT [drive]

Função: Troca o drive de boot do MSXDOS a partir do BASIC.

**BUFFERS** (interno, 2)

Formato: BUFFERS [número]

Função: Apresenta ou define o número de buffers de I/O do sistema.

**CD** (interno, 2)

Formato: CD [[d:][caminho] | -]

CHDIR [[d:][caminho] | -]

Função: Apresenta ou troca o subdiretório corrente. Se “-” for especificado, retorna ao diretório anterior.

**CDD** (interno, 2.41)

Formato: CDD [[d:][caminho] | -]

Função: Apresenta ou troca o subdiretório e o drive correntes. Se “-” for especificado, retorna ao drive/diretório anterior.

**CDPATH** (interno, 2.41)

Formato: CDPATH [[+|-] [d:] caminho [[d:] caminho... ]]]

Função: Apresenta ou define o caminho de procura.

**CHDIR** (interno, 2)

Formato: O mesmo que o comando CD.

Função: A mesma que o comando CD.

**CHKDSK** (interno, 2)

Formato: CHKDSK [d:] [/F]

Função: Checa a integridade dos arquivos no disco. Se [/F] for especificado, os arquivos não serão corrigidos; apenas a informação sobre a falha de integridade será mostrada.

**CLS** (interno, 2)

Formato: CLS

**Função:** Limpa a tela.

## COLOR (interno, 2.41)

Formato: COLOR <cor frente> [ <cor fundo> [ <cor borda> ] ]

**Função:** Troca as cores da tela.

### **COMMAND2** (interno, 2)

Formato: COMMAND2 [comando]

**Função:** Executa um comando.

### **CONCAT** (interno, 2-2.41)

Formato: CONCAT [/H] [/S] [/P] [/A] [/B] [/V] <arquivos fonte>  
<arquivos destino>

Função: Concatena todos os arquivos fonte em um único arquivo.

[/H] Arquivos ocultos também serão concatenados

[S] Arquivos de sistema também serão concatenados (somente 2.41)

[/P] Pausa as mensagens ao completar uma tela

[/B] Concatena sem interpretação

[/A] Reverte o efeito de [/B]

[A] Reverte o efeito de [B]  
[V] Verifica arquivo concatenado criado

COPY (intero 1-2-241)

Função: Copia arquivos

[/H] Arquivos ocultos também serão copiados (?)

[/H] Arquivos ocultos também serão copiados (2)  
[/S] Arquivos de sistema também serão copiados (2.41)

[/P] Pausa as mensagens ao completar uma tela

[/I] Pausa as mensagens ao completar uma tela  
[/A] Faz cópia ASCII (acrescenta Ctrl+Z no fim do arquivo)

[B] Reverte o efeito de [A]

[B] Reverte o efeito de [A]  
[V] Verifica arquivo copiado

[V] Verifica arquivo copiado  
[T] Altera a data e hora do arquivo

[/I] Altera a data e hora do arquivo copiado para a atual

**CPU** (interno, 2.41)

Formato: CPU [número]

**Função:** Apresenta ou troca a CPU para o MSX turbo R (0=Z80; 1=R800 ROM; 2=R800 DRAM).

**DATE** (interno, 1-2.41)

Formato: DATE [data]

Função: Apresenta ou altera a data do sistema. [data] deve estar no formato “mm-dd-aaaa” ou no formato definido p/ SET DATE.

**DEL** (interno, 1)

Formato: DEL [/S] [/H] [/P] <nome de arquivo composto>

ERA [/S] [/H] [/P] <nome de arquivo composto>

ERASE [/S] [/H] [/P] <nome de arquivo composto>

Função: Deleta um ou mais arquivos.

[/S] Arquivos de sistema também serão deletados (2.41)

[/H] Arquivos ocultos também serão deletados

[/P] Pausa as mensagens ao completar uma tela

**DELALL** (externo, N)

Formato: DELALL <letra de drive>:

Função: Formatação rápida para uma unidade de drive.

**DEVINFO** (externo, N)

Formato: DEVINFO <slot do driver>-[<subslot do driver>]

Função: Apresenta informações sobre dispositivos controlados pelo Nextor.

**DIR** (interno, 1-2-2.41)

Formato: DIR [/S] [/H] [/W] [/P] [/2] [<nome arquivo composto>]

Função: Apresenta os nomes dos arquivos do disco.

[/S] Arquivos de sistema também serão listados (2.41)

[/H] Arquivos ocultos também serão listados

[/W] Lista apenas os nomes dos arquivos

[/P] Pausa a listagem ao completar uma tela

[/2] Lista em duas colunas (2.41)

**DISKCOPY** (externo, 2)

Formato: DISKCOPY [d1: [d2:]] [/X]

Função: Copia um disco inteiro (d1:) para outro (d2:)

[/X] Suprime as mensagens durante a cópia

**DRIVERS** (externo, N)

Formato: DRIVERS

Função: Apresenta informações sobre os drivers disponíveis para o Nextor e MSXDOS.

**DRVINFO** (externo, N)

Formato: DRVINFO

Função: Apresenta informações sobre todas as letras de drive disponíveis.

**DSKCHK** (interno, 2.41)

Formato: DSKCHK [ON | OFF]

Função: Apresenta ou define o estado de checagem do disco.

**ECHO** (interno, 1)

Formato: ECHO [texto]

Função: Imprime um texto durante a execução de um arquivo em lote com alimentação de linha no final.

**ECHOS** (interno, 1)

Formato: ECHOS [texto]

Função: Imprime um texto durante a execução de um arquivo em lote sem alimentação de linha no final.

**ELSE** (interno, 2.41)

Formato: ELSE [comando]

Função: Execução condicional de comando. Sem o parâmetro [comando], alterna o Command Mode entre ON/OFF.

**END** (interno, 2.41)

Formato: END

Função: Termina um arquivo em lote (batch).

**ENDIF** (interno, 2.41)

Formato: ENDIFF [comando]

Função: Aumenta um nível e restaura o Command Mode.

**ERA** (interno, 1)

Formato: O mesmo que o comando DEL.

Função: A mesma que o comando DEL.

**ERASE** (interno, 1)

Formato: O mesmo que o comando DEL.

Função: A mesma que o comando DEL.

**EXIT** (interno, 2)

Formato: EXIT [número]

Função: Sai do programa executado pelo comando COMMAND2.  
[número] é o código de erro do usuário (o padrão é 0).

**FASTOUT** (externo, N)

Formato: FASTOUT [ON | OFF]

Função: Liga ou desliga a saída rápida para a rotina STROUT, ou apresenta o estado atual de STROUT.

**FIXDISK** (externo, 2)

Formato: FIXDISK [d:] [/S]

Função: Atualiza um disco para o formato MSXDOS2.  
[/S] Atualização completa.

**FORMAT** (interno, 1-2.41)

Formato: FORMAT [d:] (1)

FORMAT [d: [opção [/X]]] (2.41)

Função: Formata um disco. Se [opção] for especificada, formata com essa opção, sem apresentar lista de opções. Para um MSX padrão, [opção] = 1 formata em face simples e 2 formata em face dupla.  
[/X] Inicia formatação imediata, sem apresentar mensagem.

**FREE** (interno, 2.41)

Formato: FREE [d:]

Função: Apresenta os espaços total, livre e usado do disco.

**GOSUB** (interno, 2.41)

Formato: GOSUB ~label

Função: Executa uma sub-rotina dentro de um arquivo em lote (batch).

**GOTO** (interno, 2.41)

Formato: GOTO ~label

Função: Salta para a label dentro de um arquivo em lote (batch).

**HELP** (interno, 2)

Formato: HELP [<nome do arquivo>]

Função: Apresenta o arquivo de ajuda <nome do arquivo>.HLP ou lista todos se não houver argumento.

**HISTORY** (interno, 2.41)

Formato: HISTORY [/P]

Função: Apresenta o histórico de comandos.

[/P] Pausa o histórico ao completar uma tela

**IF** (interno, 2.41)

Formato: IF [NOT] EXIST [d:][<caminho>] <nome do arquivo> [THEN] <comando>

ou

IF [NOT] <expr1> == | EQ | LT | GT <expr2> [AND | OR | XOR [NOT] <expr3> == | EQ | LT | GT <expr4> [AND | OR | XOR ...]] [THEN] <comando>

Função: Executa comando se a equação dada for verdadeira.

EQ Equivalência (igualdade)

LT Menor que

GT Maior que

**IFF** (interno, 2.41)

Formato: IFF [NOT] EXIST [d:][<caminho>] <nome do arquivo> [THEN] <comando>

:

ENDIFF [<comando>]

ou

IFF [NOT] <expr1> == | EQ | LT | GT <expr2> [AND | OR | XOR [NOT] <expr3> == | EQ | LT | GT <expr4> [AND | OR | XOR ...]] [THEN] <comando>

:

ENDIFF [<comando>]

Função: Liga o Command Mode se a equação dada for verdadeira e desliga caso contrário.

EQ Equivalência (igualdade)

LT Menor que

GT Maior que

**INKEY** (interno, 2.41)

Formato: INKEY [<string>] %%<variável de ambiente>

Função: Lê o valor de uma tecla pressionada e armazena o valor lido na <variável de ambiente>.

**INPUT** (interno, 2.41)

Formato: INPUT [<string>] %%<variável de ambiente>

Função: Lê uma string do teclado ou dispositivo e armazena o valor lido na <variável de ambiente>.

**KMODE** (externo, 2-K)

Formato: KMODE [modo | OFF] [/S] [d:]

Função: Seleciona ou desliga o modo Kanji ou atualiza o boot para que instale automaticamente o Kanji driver.

[/S] Atualiza o código de inicialização do drive [d:].

**LOCK** (externo, N)

Formato: LOCK [<letra de drive>: [ON|OFF] ]

Função: Bloqueia ou desbloqueia letras de drive, ou apresenta a lista de drives bloqueados.

**MAPDRV** (externo, N)

Formato: MAPDRV [/L] <drive>: <partição> | d | u [<índice disp> – <índice LUN>][<slot driver>[–<subslot driver>]]]

Função: Mapeia uma unidade de drive no sistema Nextor.

[/L] Bloqueia a unidade logo após o mapeamento

<drive> letra da unidade a ser mapeada

<partição> 0 – A unidade será mapeada a partir do setor zero absoluto do dispositivo.

1 – primeira partição primária

2 a 4 – referem a partições estendidas 2.1 a 2.4 se a partição 2 for estendida; caso contrário se referem a partições primárias.

5 em diante referem a partições estendidas.

d – a unidade padrão será mapeada

u – a unidade não será mapeada

**MD** (interno, 2)

Formato: MD [d:] <caminho>

MKDIR [d:] <caminho>

Função: Cria um subdiretório.

**MEMORY** (interno, 2.41)

Formato: MEMORY [/K] [/P]

Função: Apresenta informações sobre a RAM do sistema.

[/K] Apresenta em Kbytes.

[/P] Pausa as mensagens ao completar uma tela.

**MKDIR** (interno, 2)

Formato: O mesmo que o comando MD.

Função: A mesma que o comando MD.

**MODE** (interno, 1-2.41)

Formato: MODE <nº de caracteres> [<linhas>]

Função: Altera o número de caracteres por linha horizontal (1, 2 e 2.41) e o número de linhas de tela (somente 2.41).

**MORE** (externo, 1-2.41)

Formato: <comando> | MORE

Função: Comando de exibição. A saída do <comando> é redirecionada para o comando MORE. Ao final da tela a exibição é pausada com a mensagem MORE até uma tecla ser pressionada. <ESC> ou <N> abortam o comando.

**MOVE** (interno, 2)

Formato: MOVE [/H] [/P] [/S] <nome do arquivo> <caminho>

Função: Move arquivos para outra parte do disco.

[/H] Arquivos ocultos também serão movidos.

[/S] Arquivos de sistema também serão movidos (2.41).

[/P] Pausa as mensagens ao completar uma tela.

**MVDIR** (interno, 2)

Formato: MVDIR [/H] [/P] <nome do arquivo> <caminho>

Função: Move diretórios para outra parte do disco.

[/H] Diretórios ocultos também serão movidos.

[/P] Pausa as mensagens ao completar uma tela.

**NSYSVER** (externo, N)

Formato: NSYSVER <versão maior>.<versão secundária>

Função: Altera número da versão do DOS retornada pelo sistema.

**PATH** (interno, 2)

Formato: PATH [[+ | -] [d:]<caminho> [[d:]<caminho> ...]]]

Função: Apresenta ou define o caminho de procura para os arquivos de execução tipo .COM e .BAT.

- + Deleta os caminhos com o mesmo nome e os recria.
- Deleta os caminhos especificados.

Sem +/-, deleta todos os caminhos existentes e cria o caminho especificado.

**PAUSE** (interno, 2)

Formato: PAUSE [comentário]

Função: Interrompe a execução de um arquivo em lote (batch) até que uma tecla seja pressionada.

**POPD** (interno, 2.41)

Formato: POPD [/N]

Função: Recupera o drive e o diretório correntes.

[/N] Somente o último drive e diretório são removidos da lista

**PUSHD** (interno, 2.41)

Formato: PUSHD [d:] [<caminho>]

Função: Troca o diretório e drive padrão, salvando os correntes.

**RAMDISK** (interno, 2)

Formato: RAMDISK [=] [<tamanho>[K]] [/D]

Função: Apresenta o tamanho ou cria uma RAMDISK.

[/D] Deleta a RAMDISK existente e cria outra.

**RALLOC** (externo, N)

Formato: RALLOC [<letra de drive>: [ON|OFF] ]

Função: Ativa ou desativa a redução de espaço de alocação para uma unidade de drive, ou apresenta a lista de drives com alocação reduzida.

**RD** (interno, 2)

Formato: RD [/H] [/P] <nome do arquivo>

RMDIR [/H] [/P] <nome do arquivo>

Função: Remove um ou mais subdiretórios.

[/H] Arquivos ocultos também serão movidos

[/P] Pausa as mensagens ao completar uma tela

**REM** (interno, 1)

Formato: REM [comentários]

Função: Insere comentários em um arquivo em lote (batch).

**REN** (interno, 1-2.41)

Formato: REN [/H] [/P] [/S] <nome arquivo 1> <nome arq 2>  
RENAME [/H] [/P] [/S] <nome arq 1> <nome arq 2>

Função: Renomeia o arquivo <nome arq 1> com <nome arq 2>.   
[/H] Arquivos ocultos também serão renomeados  
[/S] Arquivos de sistema também serão renomeados (2.41)  
[/P] Pausa as mensagens ao completar uma tela

**RENAME** (interno, 1)

Formato: O mesmo que o comando REN.

Função: A mesma que o comando REN.

**RESET** (interno, 2.41)

Formato: RESET

Função: Reseta o sistema.

**RETURN** (interno, 2.41)

Formato: RETURN [~label]

Função: Retorna de uma sub-rotina em um arquivo em lote (batch).

**RMDIR** (interno, 2)

Formato: O mesmo que o comando RD.

Função: A mesma que o comando RD.

**RNDIR** (interno, 2)

Formato: RNDIR [/H] [/P] <nome diretório 1> <nome diretório 2>

Função: Renomeia o subdiretório <nome diretório 1> com <nome diretório 2>.  
[/H] Arquivos ocultos também serão renomeados.

[/P] Pausa as mensagens ao completar uma tela.

**SET** (interno, 2-2.41)

Formato: SET [/P] [nome] [=] [valor]

Função: Define ou apresenta itens de ambiente.

[/P] Pausa as mensagens ao completar uma tela  
 Os valores padrão são os seguintes:  
 EXPAND = ON (2.41)  
 SEPAR = ON (2.41)  
 ALIAS = ON (2.41)  
 REDIR = ON  
 LOWER = ON (2.41)  
 UPPER = OFF  
 ECHO = OFF  
 EXPERT = ON (2.41)  
 PROMPT = %\_CWD%> (modificado no 2.41)  
 CDPATH = ; (2.41)  
 PATH = ;  
 TIME = 24  
 DATE = yy-mm-dd  
 TEMP = A:\  
 HELP = A:\HELP  
 SHELL = A:\COMMAND2.COM

#### **SHIFT** (interno, 2.41)

Formato: SHIFT [/<número>]

Função: Desloca os argumentos do arquivo em lote uma posição para a esquerda. Se /<número> for especificado, o argumento nesta posição será o primeiro a ser deslocado; argumentos anteriores não serão afetados.

#### **THEN** (interno, 2.41)

Formato: THEN [<comando>]

Função: Executa um comando (THEN é ignorado).

#### **TIME** (interno, 1)

Formato: TIME [<hora>]

Função: Apresenta ou altera a hora do sistema.

#### **TO** (interno, 2.41)

Formato: TO <parte\_nome\_subdirerório> [/N] [/X | F | P | L]

TO [d:] /S [/H]

TO [d:] ...

TO [d:]-n

TO [d:]\  
 TO [d:]<nome\_diretório> /M | C [/H]  
 TO [d:]<nome\_diretório> /D  
 TO [d:]<nome antigo> <nome novo> /R  
 TO [d:]<dir\_fonte> <dir\_destino> /V

- Função: Troca, cria, deleta, renomeia ou remove um diretório.
- [/N] Lista os diretórios contendo <parte\_nome\_subdir>.
  - [/X] Apenas nomes exatos são procurados.
  - [/F] Procura apenas no início no nome.
  - [/P] Procura por todo o nome.
  - [/L] Procura apenas no final do nome.
  - [/S] Procura todos os diretórios e cria o arquivo TO.LST.
  - [/H] Faz /S procurar também por arquivos ocultos e /M ou /C criarem diretório oculto.
  - [/M] Cria novo diretório.
  - [/C] Cria novo diretório e entra nele.
  - [/D] Remove diretório.
  - [/R] Renomeia diretório.
  - [/V] Move subdiretório.
  - n Nível dos subdiretórios.
  - \ Vai para o diretório raiz.

#### **TREE** (interno, 2.41)

- Formato: TREE [d:] [<caminho>] [/P] [/?]
- Função: Apresenta a árvore de diretórios no disco.
- [/P] Pausa a listagem ao completar uma tela.
  - [/] Apresenta uma tela de ajuda.

#### **TYPE** (interno, 1-2.41)

- Formato: TYPE [/S] [/H] [/P] [/B] <nome arquivo> [“>” <dispositivo>]  
 TYPE <dispositivo> “>” <nome do arquivo>
- Função: Apresenta dados de um arquivo ou dispositivo.
- [/S] Arquivos de sistema também serão apresentados (somente 2.41).
  - [/H] Arquivos ocultos também serão apresentados.
  - [/P] Pausa a apresentação ao completar uma tela.
  - [/B] Desabilita a checagem de códigos de controle.

#### **TYPEWW** (externo, 1-2.41)

- Formato: TYPEWW <nome do arquivo> [/S] [/H] [/B]

- Função:** Apresenta dados de um arquivo. Ao contrário do comando TYPE, <nome do arquivo> não pode ser ambíguo.
- [**/S**] Arquivos de sistema também serão apresentados (somente 2.41).
  - [**/H**] Arquivos ocultos também serão apresentados.
  - [**/P**] Pausa a apresentação ao completar uma tela.

### **UNDEL** (externo, 2)

- Formato:** UNDEL [<nome do arquivo>]  
**Função:** Recupera arquivos deletados.

### **VER** (interno, 2)

- Formato:** VER  
**Função:** Apresenta a versão do sistema.

### **VERIFY** (interno, 2)

- Formato:** VERIFY [ON | OFF]  
**Função:** Apresenta ou altera o estado de verificação de escrita.

### **VOL** (interno, 2)

- Formato:** VOL [d:] [<nome do volume>]  
**Função:** Apresenta ou altera o nome de volume do disco.

### **XCOPY** (externo, 2)

- Formato:** XCOPY [<nome do arquivo> [<nome do arquivo>]] [  
                   /T] [  
                   /A] [  
                   /M] [  
                   /S] [  
                   /E] [  
                   /P] [  
                   /W] [  
                   /V]
- Função:** Copia arquivos e diretórios. As opções são:
- [**/T**] Altera a data do arquivo copiado para a atual
  - [**/A**] Apenas arquivos com atributo “arquivo” setado são copiados.
  - [**/M**] Similar a /A, mas o atributo “arquivo” é resetado após a cópia.
  - [**/S**] Subdiretórios também são copiados.
  - [**/E**] Faz /S criar todos os subdiretórios, mesmo vazios.
  - [**/P**] Pausa após copiar cada arquivo.
  - [**/W**] Pausa após copiar alguns arquivos.
  - [**/V**] Verifica arquivos copiados.

**XDIR** (externo, 2)

Formato: XDIR [<nome do arquivo>] [/H]

Função: Lista todos os arquivos do subdiretório corrente, em árvore.

[/H] Arquivos ocultos também serão listados.

**Z80MODE** (externo, N)

Formato: Z80MODE [<slot do driver>[- <subslot do driver>]]

[ON|OFF]

Função: Ativa ou desativa modo de acesso Z80 para o driver MSXDOS especificado.

**4.3 – CHAMADAS PARA A BDOS****4.3.1 – Manipulação de I/O****CONIN** (01H)

Função: Entrada de teclado.

Setup: Nenhum.

Retorno: A – Código do caractere do teclado.

Nota: Entrada de caractere com espera e eco na tela. As seguintes sequências de controle são checadas por esta rotina:

CTRL+C → Retorna o sistema ao nível de comandos.

CTRL+P → Liga o eco para a impressora. Tudo o que for escrito na tela sairá na impressora.

CTRL+N → Desliga o eco para a impressora.

CTRL+S → Interrompe a apresentação dos caracteres até que uma tecla seja pressionada.

**CONOUT** (02H)

Função: Apresenta na tela o caractere contido no registrador E. As sequências de controle descritas acima são checadas.

Entrada: E – Código do caractere.

Retorno: Nenhum.

**AUXIN** (03H)

Função: Entrada externa de dispositivo auxiliar (modem, por exemplo). As quatro sequências de controle são checadas.

Entrada: Nenhum.

Retorno: A – Código de caractere do dispositivo auxiliar.

**AUXOUT** (04H)

- Função: Saída para dispositivo externo. Esta função checa as quatro sequências de controle.
- Entrada: E – Código do caractere a enviar.
- Retorno: Nenhum.

**LSTOUT** (05H)

- Função: Saída de caractere para a impressora. Esta função checa as quatro sequências de controle.
- Entrada: E – Código do caractere a ser enviado.
- Retorno: Nenhum.

**DIRIO** (06H)

- Função: Entrada ou saída de string. Não suporta caracteres de controle, mas checa as quatro sequências de controle.
- Entrada: E – Código do caractere a ser impresso no monitor.  
Se for FFH, o caractere será recebido.
- Retorno: Se E for FFH na entrada o código ASCII da tecla retornará em A. Se A retornar 00H, não foi pressionada nenhuma tecla.

**DIRIN** (07H)

- Função: Lê um caractere do teclado (com espera) e imprime na tela. Esta função não suporta caracteres de controle.
- Entrada: Nenhum.
- Retorno: A – Código ASCII do caractere lido.

**INNOE** (08H)

- Função: Lê um caractere do teclado (com espera) mas não imprime na tela. Esta função não suporta caracteres de controle.
- Entrada: Nenhum.
- Retorno: A – Código ASCII do caractere lido.

**STRROUT** (09H)

- Função: Saída de string. O caractere ASCII 24H (\$) marca o final da string e não será impresso na tela. Esta função checa as quatro sequências de controle.
- Entrada: DE – Endereço inicial da string a ser enviada.
- Retorno: Nenhum.

**BUFIN** (0AH)

- Função: Entrada de string. A leitura dos caracteres termina ao ser pressionada a tecla RETURN. Se o número de caracteres ultrapassar o máximo apontado por DE, estes serão ignorados e será emitido um "beep" para cada caractere extra. Esta função checa as quatro sequências de controle.
- Entrada: DE deve apontar para um buffer com a seguinte estrutura:  
 $DE+0 \rightarrow$  número de caracteres a ler.  
 $DE+1 \rightarrow$  número de caracteres efetivamente lidos.  
 $DE+2$  em diante: códigos dos caracteres lidos.
- Retorno: O segundo byte do buffer apontado por DE contém o número de caracteres efetivamente lidos e do terceiro byte em diante estão os códigos dos caracteres lidos.

**CONST** (0BH)

- Função: Checa o status do teclado. Esta função checa as quatro sequências de controle.
- Entrada: Nenhum.
- Retorno: Se alguma tecla foi pressionada, o registrador A retorna com FFH, caso contrário, retorna com o valor 00H.

**4.3.2 – Definição e leitura de parâmetros****TERMO** (00H)

- Função: Reset do sistema. Quanto esta função for chamada sob o DOS, provocará a recarga do MSXDOS. Quando for chamada sob o DISK-BASIC, provocará um reset total.
- Entrada: Nenhum.
- Retorno: Nenhum.

**CPMVER** (0CH)

- Função: Leitura da versão do sistema. No caso do MSX, retornará sempre o valor 0022H, indicando compatibilidade com o CP/M 2.2.
- Entrada: Nenhum.
- Retorno: HL – Versão do sistema

**DSKRST** (0DH)

Função: Reset do disco. Todos os buffers são apagados (FCB, DPB, etc.), o drive corrente será o A: e a DTA ficará em 0080H.

Entrada: Nenhum.

Retorno: Nenhum.

**SELDSK** (0EH)

Função: Selecionar o drive corrente. O número do drive corrente é armazenado no endereço 0004H.

Entrada: E – Número do drive (A:=00H, B:=01H, etc.).

Retorno: A – Número de drives disponíveis (1 a 8).

**LOGIN** (18H)

Função: Leitura dos drives conectados ao micro.

Entrada: Nenhum.

Retorno: HL – Drives conectados

H – Sempre “00 000 000”

L – |b7|b6|b5|b4|b3|b2|b1|b0|

drive: |H:|G:|F:|E:|D:|C:|B:|A:|

O bit conterá 0 se o drive não estiver conectado e 1 se estiver. Se B: = 1 e A: = 0 (b1=1 e b0=0), significa que há apenas um drive físico funcionando como A: e B:

**CURDRV** (19H)

Função: Leitura do drive corrente (atual).

Entrada: Nenhum.

Retorno: A – Número do drive corrente (A:=00H; B:=01H, etc.).

**SETDTA** (1AH)

Função: Seta o endereço para transferência de dados.

Entrada: DE – Endereço inicial da DTA (Disk Transfer Adress).

Retorno: Nenhum.

Nota: No reset do sistema, a DTA é setada em 0080H.

**ALLOC** (1BH)

Função: Leitura de informações sobre o disco.

Entrada: E – Número do drive desejado (0=corrente; 1=A;; etc.).

Retorno: A – FFH se a especificação de drive for inválida;  
caso contrário:

A – Número de setores lógicos por cluster;  
 BC – Tamanho do setor em bytes (normalmente 512);  
 DE – Número total de clusters no disco;  
 HL – Número de clusters livres (não usados);  
 IX – Endereço inicial do DPB na RAM;  
 IY – Endereço inicial da FAT na RAM.

#### **GDATE** (2AH)

Função: Retorna a data do sistema.  
 Entrada: Nenhum.  
 Retorno: HL – Ano (1980 a 2079);  
           D – Mês (1=janeiro, 2=fevereiro, etc.);  
           E – Dia do mês (1 a 31)  
           A – Dia da semana (0=domingo, 1=segunda, etc.).

#### **SDATE** (2BH)

Função: Modificar a data do sistema.  
 Entrada: HL – Ano (1980 a 2079).  
           D – Mês (1=janeiro, 2=fevereiro, etc.).  
           E – Dia do mês (1 a 31).  
 Retorno: A = 00H se a especificação de data for válida;  
              FFH se a especificação for inválida.

#### **GTIME** (2CH)

Função: Retorna a hora do sistema.  
 Entrada: Nenhum.  
 Retorno: H – Horas;  
           L – Minutos;  
           D – Segundos;  
           E – Centésimos de segundo.

#### **STIME** (2DH)

Função: Modificar a hora do sistema.  
 Entrada: H – Horas;  
           L – Minutos;  
           D – Segundos;  
           E – Centésimos de segundo.  
 Retorno: A = 00H se a especificação de hora for válida;  
              FFH se a especificação for inválida.

**VERIFY** (2EH)

- Função: Verificação de escrita no disco.  
 Entrada: E = 0 → desativa o modo de verificação de escrita no disco.  
 $E \neq 0$  → ativa a verificação de escrita no disco.  
 Retorno: Nenhum.

**4.3.3 – Leitura/escrita absoluta de setores****RDABS** (2FH)

- Função: Leitura de setores lógicos do disco. Os setores lidos são colocados a partir da DTA.  
 Entrada: DE – Número do primeiro setor lógico a ler;  
           H – Número de setores a ler;  
           L – Número do drive (0=A; 1=B; etc.).  
 Retorno: A = 0 → Leitura bem-sucedida;  
           A ≠ 0 → Código de erro.

**WRABS** (30H)

- Função: Escrita de setores lógicos no disco. Os dados a serem escritos no disco serão lidos na RAM a partir da DTA.  
 Entrada: DE – Número do primeiro setor lógico a ser escrito;  
           H – Número de setores a escrever;  
           L – Número do drive (0=A; 1=B; etc.).  
 Retorno: A = 0 → Escrita bem-sucedida;  
           A ≠ 0 → Código de erro.

**4.3.4 – Acesso aos arquivos usando o FCB****FOPEN** (0FH)

- Função: Abrir arquivo (FCB).  
 Entrada: DE – Endereço inicial de um FCB não aberto.  
 Retorno: A = 0 → Operação bem-sucedida;  
           A ≠ 0 → Código de erro.

**FCLOSE** (10H)

- Função: Fechar arquivo (FCB).  
 Entrada: DE – Endereço inicial de um FCB aberto.  
 Retorno: A = 0 → Operação bem-sucedida;  
           A ≠ 0 → Código de erro.

**SFIRST** (11H)

Função: Procurar o primeiro arquivo. Esta função aceita caracteres coringa (\* e ?).

Entrada: DE – Endereço inicial de um FCB não aberto.

Retorno: A = 0 → Arquivo encontrado;

A ≠ 0 → Arquivo não encontrado.

**SNEXT** (12H)

Função: Procurar o próximo arquivo. Aceita caracteres coringa (\* e ?).

Entrada: Nenhum.

Retorno: A = 0 → Arquivo encontrado;

A ≠ 0 → Arquivo não encontrado.

**FDEL** (13H)

Função: Apagar arquivos. Caracteres coringa (\* e ?) podem ser usados.

Entrada: DE – Endereço inicial de um FCB aberto.

Retorno: A = 0 → Operação bem-sucedida;

A ≠ 0 → Código de erro.

**RDSEQ** (14H)

Função: Leitura sequencial.

Entrada: DE – Endereço inicial de um FCB aberto.

Bloco atual no FCB – Bloco inicial para leitura.

Registro atual no FCB – Registro inicial para leitura.

Retorno: A = 0 → Leitura bem-sucedida;

A ≠ 0 → Código de erro.

**WRSEQ** (15H)

Função: Escrita sequencial.

Entrada: DE – Endereço inicial de um FCB aberto.

Bloco atual no FCB – Bloco inicial para escrita.

Registro atual no FCB – Registro inicial para escrita.

128 bytes iniciais da DTA – Dados a serem escritos.

Retorno: A = 0 → Escrita bem-sucedida;

A ≠ 0 → Código de erro.

**FMAKE** (16H)

Função: Criar arquivos.

Entrada: DE – Endereço inicial de um FCB não aberto.

Retorno: A = 0 → Operação bem-sucedida;

A ≠ 0 → Código de erro.

**FREN** (17H)

- Função: Renomear arquivos. O caractere coringa "?" pode ser usado para renomear vários arquivos simultaneamente.
- Entrada: DE – Endereço inicial do FCB com o nome do arquivo a ser renomeado. Na primeira posição do FCB deve ser colocado o número do drive seguido do nome do arquivo a ser renomeado. A partir do 18º byte (FCB+11H) até o 28º deve ser colocado o novo nome do arquivo.
- Retorno: A = 0 → Operação bem-sucedida;  
A ≠ 0 → Código de erro.

**RDRND** (21H)

- Função: Leitura aleatória. O registro lido será colocado na área indicada pela DTA e tem o tamanho fixo de 128 bytes.
- Entrada: DE – Endereço inicial de um FCB aberto.  
Registro aleatório no FCB – Nº do registro a ler.
- Retorno: A = 0 → Leitura bem-sucedida;  
A ≠ 0 → Código de erro.

**WRRND** (22H)

- Função: Escrita aleatória.
- Entrada: DE – Endereço inicial de um FCB aberto.  
Registro aleatório no FCB – Nº do registro a escrever.  
128 bytes a partir da DTA – Dados a serem escritos.
- Retorno: A = 0 → Escrita bem-sucedida;  
A ≠ 0 → Código de erro.

**FSIZE** (23H)

- Função: Ler o tamanho do arquivo. O tamanho retorna nos três primeiros bytes no campo de tamanho do arquivo aleatório do FCB em incrementos de 128 bytes. Assim, se um arquivo contiver de 1 a 128 bytes, o valor retornado será 1; se contiver de 129 a 256 bytes, o valor será 2, e assim por diante.
- Entrada: DE – Endereço inicial de um FCB aberto.
- Retorno: A = 0 → Operação bem-sucedida;  
A ≠ 0 → Código de erro.

**SETRND** (24H)

- Função: Setar campo do registro aleatório.

- Entrada:** DE – Endereço inicial de um FCB aberto.  
 Bloco atual no FCB – Número do bloco desejado.  
 Registro atual no FCB – Número do registro desejado.
- Retorno:** A posição do registro atual desejada, calculada a partir do registro e bloco contidos no FCB, é colocada no campo de registro aleatório. Os três primeiros bytes registro aleatório contêm valores válidos.

#### **WRBLK (26H)**

- Função:** Escrita aleatória em bloco. O número do registro aleatório é automaticamente incrementado depois da escrita, e seu tamanho pode variar de 1 até 65 535 bytes.
- Entrada:** DE – Endereço inicial de um FCB aberto.  
 HL – Número de registros a serem escritos.  
 DTA – Dados a serem escritos.  
 Tamanho do registro no FCB – Tamanho dos registros a serem escritos.  
 Registro aleatório no FCB – Número do primeiro registro a ser escrito.
- Retorno:** A = 0 → Escrita bem-sucedida;  
 A ≠ 0 → Código de erro.

#### **RDBLK (27H)**

- Função:** Acesso aleatório em bloco.
- Entrada:** DE – Endereço inicial de um FCB aberto.  
 HL – Número de registros a serem lidos.  
 DTA – Endereço inicial para os dados lidos.  
 Tamanho do registro no FCB – Tamanho dos registros a serem lidos.  
 Registro aleatório no FCB – Número do primeiro registro a ser lido.
- Retorno:** A = 0 → Leitura bem-sucedida;  
 A ≠ 0 → Código de erro.  
 HL – Número de registros efetivamente lidos, caso o fim de arquivo seja atingido antes de todos os registros serem lidos.

**WRZER** (28H)

- Função: Escrita aleatória com bytes 00H. Esta função é igual à 22H (WRRND), exceto pelo fato de preencher os registros restantes do arquivo com bytes 00H, se o registro especificado não for o último do arquivo.
- Entrada: DE – Endereço inicial de um FCB aberto.  
 Registro aleatório no FCB – Registro a ser escrito.  
 128 bytes a partir da DTA – Dados a serem escritos.
- Retorno: A = 0 → Escrita bem-sucedida;  
 A ≠ 0 → Código de erro.

**4.3.5 – Funções adicionadas pelo MSXDOS2****DPARM** (31H)

- Função: Lê os parâmetros do disco.
- Entrada: DE – Endereço inicial de um buffer de 32 bytes.  
 L – Número do drive (0=corrente, 1=A; etc.).
- Retorno: A – Código de erro (se for 0, não houve erro).  
 DE – Endereço inicial do buffer de parâmetros.
- |        |  |
|--------|--|
| +0     | Número do drive físico (1=A; etc.)       |
| +1~2   | Tamanho setor em bytes (normalmente 512) |
| +3     | Número de setores por cluster            |
| +4~5   | Número de setores reservados             |
| +6     | Número de FATs (normalmente 2)           |
| +7~8   | Número de entradas do diretório          |
| +9~10  | Número total de setores lógicos          |
| +11    | ID do disco                              |
| +12    | Número de setores por FAT                |
| +13~14 | Primeiro setor do diretório              |
| +15~16 | Primeiro setor da área de dados          |
| +17~18 | Número máximo de clusters                |
| +19    | Dirty disk flag                          |
| +20~23 | Volume ID (-1 = sem ID de volume)        |
| +24~31 | Reservado (normalmente 0)                |

**FFIRST** (40H)

- Função: Procura primeira entrada no diretório.

**Entrada:** DE – Endereço inicial do FIB ou de uma string ASCII "drive/path/arquivo", podendo conter os caracteres coringa "?" e "\*".  
 HL – Endereço inicial do nome de arquivo (somente quando DE apontar para o FIB).  
 B – Atributos para procura (igual ao do diretório).  
 IX – Endereço inicial de um novo FIB.  
**Retorno:** A – Código de erro (se for 0, não houve erro).  
 IX – Endereço inicial do novo FIB preenchido.

#### **FNEXT** (41H)

**Função:** Procura próxima entrada do diretório. Esta função só deve ser usada após a função 40H. Ela aceita os caracteres coringa "?" e "\*" definidos em 40H e procura todos os arquivos que tenham partes de seu nome iguais especificadas através dos caracteres coringa, um após outro.  
**Entrada:** IX – Endereço inicial do FIB.  
**Retorno:** A – Código de erro (se for 0, não houve erro).  
 IX – Endereço inicial do novo FIB preenchido.

#### **FNEW** (42H)

**Função:** Procura nova entrada.  
**Entrada:** DE – Endereço inicial do FIB ou de uma string ASCII "drive/path/arquivo". Se houver caracteres coringa no nome de arquivo, eles serão substituídos por caracteres apropriados. Se o bit "diretório" estiver setado na entrada (registrador B), será criado um subdiretório. Os outros bits serão copiados.  
 HL – Endereço inicial de um nome de arquivo (somente se DE apontar para o FIB).  
 B – b0~b6 – Atributos;  
 b7 – Cria nova flag.  
 IX – Endereço inicial do novo FIB contendo o nome de arquivo padrão.  
**Retorno:** A – Código de erro (se for 0, não houve erro)  
 IX – Endereço inicial do FIB preenchido com a nova entrada.

**OPEN** (43H)

Função: Abre arquivo handle. Se o bit "herdável" de A estiver setado, o arquivo handle deverá ser aberto por outro processo (ver função 60H).

Entrada: DE – Endereço inicial do FIB ou string ASCII "drive/path/arquivo".

A – Modo abertura:

- b0=1 – Não escrita;
- b1=1 – Não leitura;
- b2=1 – herdável;
- b3~b7 – Devem ser "0".

Retorno: A – Código de erro (se for 0, não houve erro).

B – Novo arquivo handle.

**CREATE** (44H)

Função: Criar arquivo handle. O arquivo criado por esta função será automaticamente aberto (função 43H)

Entrada: DE – Drive/path/arquivo ou string ASCII.

A – Modo abertura:

- b0=1 – Não escrita;
- b1=1 – Não leitura;
- b2=1 – herdável;
- b3~b7 – Devem ser "0".

B – b0~b6 = atributos;

b7 = cria nova flag.

Retorno: A – Código de erro (se for 0, não houve erro).

B – Novo arquivo handle.

**CLOSE** (45H)

Função: Fechar arquivo handle.

Entrada: B – Arquivo handle a fechar.

Retorno: A – Código de erro (se for 0, não houve erro).

**ENSURE** (46H)

Função: Proteger arquivo handle (o apontador do arquivo corrente não poderá ser modificado).

Entrada: B – Arquivo handle a ser protegido.

Retorno: A – Código de erro (se for 0, não houve erro).

**DUP** (47H)

Função: Duplicar arquivo handle.  
 Entrada: B – Arquivo handle.  
 Retorno: A – Código de erro (se for 0, não houve erro).  
 B – Novo arquivo handle.

**READ** (48H)

Função: Ler de um arquivo handle. As quatro sequências de controle (Ctrl+P, Ctrl+N, Ctrl+S e Ctrl+C) são checadas.  
 Entrada: B – Arquivo handle.  
 DE – Endereço inicial do buffer.  
 HL – Número de bytes a ler.  
 Retorno: A – Código de erro (se for 0, não houve erro).  
 HL – Número de bytes efetivamente lidos.

**WRITE** (49H)

Função: Escrever por um arquivo handle. Se o fim de arquivo for encontrado, ele será estendido até o valor necessário.  
 Entrada: B – Arquivo handle.  
 DE – Endereço inicial do buffer.  
 HL – Número de bytes a escrever.  
 Retorno: A – Código de erro (se for 0, não houve erro).  
 HL – Número de bytes efetivamente escritos.

**SEEK** (4AH)

Função: Mover apontador do arquivo handle.  
 Entrada: B – Arquivo handle.  
 A – Código do método:  
     0 – Relativo ao início do arquivo;  
     1 – Relativo à posição corrente;  
     2 – Relativo ao final do arquivo.  
 DE:HL – Sinalização de offset.  
 Retorno: A – Código de erro (se for 0, não houve erro).  
 DE:HL – Novo apontador de arquivo.

**IOCTL** (4BH)

Função: Controle para dispositivos de I/O.  
 Entrada: B – Arquivo handle.

A – Código de subfunção:

- 00H – Ler status do arquivo handle;
- 01H – Setar modo ASCII/binário;
- 02H – Testa se disposit. está pronto p/ entrada;
- 03H – Testa se disposit. está pronto p/ saída;
- 04H – Calcula tamanho da screen.

DE – Outros parâmetros.

Retorno: A – Código de erro (se for 0, não houve erro).

DE – Outros valores de retorno.

Nota: Se A for igual a 0 na entrada, então o registrador DE deve ser carregado com os seguintes parâmetros:

→ Para dispositivos:

- b0=1 – Dispositivo de entrada;
- b1=1 – Dispositivo de saída;
- b2~b4 – Reservados;
- b5=1 – Modo ASCII;
- =0 – Modo binário;
- b6=1 – Fim de arquivo;
- b7=1 – Dispositivo (sempre 1);
- b8~b15 – Reservados.

→ Para arquivos:

- b0~b5 – Número do drive (0=A; etc.);
- b6=1 – Fim de arquivo;
- b7=0 – Arquivo de disco (sempre 0);
- b8~b15 – Reservados.

No retorno, DE apresentará os mesmos valores. Se A=1, deve ser especificado apenas o bit 5 de DE; os demais bits serão ignorados. Se A for igual a 2 ou 3, o registrador E retornará com o valor 00H se o dispositivo não estiver pronto e com FFH se estiver pronto. Se A for igual a 4, DE retornará com o valor lógico do tamanho da tela para o arquivo handle (D=número de linhas e E=número de colunas). Para dispositivos que não a tela, DE retornará com o valor 0000H.

## **HTEST (4CH)**

Função: Testar arquivo handle.

Entrada: B – Arquivo handle.

DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".

Retorno: A – código de erro (se for 0, não houve erro).

B = 00H – Não é o mesmo arquivo;

FFH – É o mesmo arquivo.

**DELETE** (4DH)

- Função: Apagar arquivo ou subdiretório. Um subdiretório só poderá ser apagado se não contiver nenhum arquivo. Se um nome de dispositivo for especificado, não retornará erro, mas o dispositivo não será "apagado".
- Entrada: DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".
- Retorno: A – Código de erro (se for 0, não houve erro).

**RENAME** (4EH)

- Função: Renomear arquivo ou subdiretório.
- Entrada: DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".  
HL – Apontador para o novo nome em ASCII.
- Retorno: A – Código de erro (se for 0, não houve erro).

**MOVE** (4FH)

- Função: Mover arquivo ou subdiretório. Um arquivo não poderá ser movido se o arquivo handle respectivo estiver aberto. O FIB do arquivo movido não será atualizado.
- Entrada: DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".  
HL – Apontador para nova string path em ASCII.

**ATTR** (50H)

- Função: Setar ou ler atributos de um arquivo. Os atributos de um arquivo não podem ser modificados se o arquivo handle correspondente estiver aberto.
- Entrada: DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".  
A = 0 – lê atributos;  
      1 – Escreve atributos.  
L – Novo byte de atributos (se A = 1).
- Retorno: A – Código de erro (se for 0, não houve erro).  
L – Byte de atributos atual.

**FTIME** (51H)

- Função: Ler ou setar data e hora em um arquivo.
- Entrada: DE – Apontador p/ o FIB ou p/ string "drive/path/arquivo".  
A = 0 – Ler data e hora;  
      1 – Setar data e hora.  
IX – Nova hora (se A=1).  
HL – Nova data (se A=1).

Retorno: A – Código de erro (se for 0, não houve erro).  
 DE – Hora do arquivo corrente.  
 HL – Data do arquivo corrente.

#### **HDELETE (52H)**

Função: Apagar arquivo handle. Se houver outro arquivo handle aberto para o mesmo arquivo, esse não poderá ser apagado.  
 Entrada: B – Arquivo handle.  
 Retorno: A – Código de erro (se for 0, não houve erro).

#### **HRENAM (53H)**

Função: Renomear arquivo handle. O arquivo não poderá ser renomeado se houver outro arquivo handle aberto para o mesmo arquivo. Esta função é idêntica à função 4EH, exceto pelo fato do registrador HL não poder apontar para um FIB.  
 Entrada: B – Arquivo handle.  
 HL – Novo nome de arquivo em ASCII.  
 Retorno: A – Código de erro (se for 0, não houve erro).

#### **HMOVE (54H)**

Função: Mover arquivo handle. O arquivo não poderá ser movido se houver outro arquivo handle aberto para o mesmo arquivo. Esta função é idêntica à função 4FH, exceto pelo fato do registrador HL não poder apontar para um FIB.  
 Entrada: B – Arquivo handle.  
 HL – Nova path em ASCII.  
 Retorno: A – Código de erro (se for 0, não houve erro).

#### **HATTR (55H)**

Função: Ler ou setar atributos do arquivo handle. O byte de atributos não poderá ser modificado se houver outro arquivo handle aberto para o mesmo arquivo.  
 Entrada: B – arquivo handle.  
 A – 0 = ler atributos; 1 = setar atributos.  
 L – novo byte de atributos (se A=1).  
 Retorno: A – código de erro (se for 0, não houve erro).  
 L – Byte de atributos corrente.

**HFTIME** (56H)

**Função:** Ler ou alterar hora e data do arquivo handle. Se houver outro arquivo handle aberto para o mesmo arquivo, a data e a hora não poderão ser modificadas. Esta função é idêntica à função 51H, exceto pelo fato de não haver apontador; somente o arquivo handle.

**Entrada:** B – Arquivo handle.

A – 0 = ler data e hora; 1 = setar data e hora.

IX – Nova hora (se A=1).

HL – Nova data (se A=1).

**Retorno:** A – Código de erro (se for 0, não houve erro).

DE – Hora corrente do arquivo.

HL – Data corrente do arquivo.

**GETDTA** (57H)

**Função:** Ler o endereço da DTA (Disk Transfer Area).

**Entrada:** Nenhum.

**Retorno:** DE – Endereço inicial da DTA.

**GETVFY** (58H)

**Função:** Ler flag de verificação de escrita.

**Entrada:** Nenhum.

**Retorno:** B – 0 = verificação de escrita desativada;  
1 = verificação de escrita ativada.

**GETCD** (59H)

**Função:** Ler diretório ou subdiretório corrente.

**Entrada:** B – número do drive (0=corrente; 1=A; etc.).

DE – endereço inicial de um buffer de 64 bytes.

**Retorno:** A – código de erro (se for 0, não houve erro).

DE – aponta para o buffer preenchido. Não são incluídos o nome do drive e o caractere "\". Se não houver diretório corrente, o buffer será preenchido com bytes 00H.

**CHDIR** (5AH)

**Função:** Trocar o subdiretório corrente.

**Entrada:** DE – String ASCII "drive/path/nome".

**Retorno:** A – Código de erro (se for 0, não houve erro).

**PARSE** (5BH)

Função: Analisa pathname (nome do caminho).

Entrada: B – Flag do nome do volume (bit 4).

    bit4 = 0 → string "drive/path/arquivo".

    1 → string "drive/volume".

DE – String ASCII para análise.

Retorno: A – Código de erro (se for 0, não houve erro).

DE – Apontador para o caractere de finalização.

HL – Apontador para o início do último item.

B – Flags de análise.

    b0=1 se algum caractere apontar para outro drive;

    b1=1 se algum diretório path for especificado;

    b2=1 se nome de drive for especificado;

    b3=1 se arquivo mestre for especificado no último item;

    b4=1 se extensão do nome do arquivo for especificada no último item;

    b5=1 se o último item for ambíguo;

    b6=1 se o último item for "." ou "..";

    b7=1 se o último item for "..".

C – drive lógico (1=A; etc.).

Nota: Como exemplo, para a string "A:\XYZ\P.Q /F", DE apontará para o espaço em branco antes de "/F" e HL apontará para "P".

**PFILE** (5CH)

Função: Analisar nome do arquivo.

Entrada: DE – string ASCII a ser analisada, sem especificação de drive. Podem ser usados caracteres coringa (? e \*).

HL – apontador para um buffer de 11 bytes.

Retorno: A – sempre 00H.

DE – apontador para o caractere final.

HL – apontador para o buffer preenchido.

B – flags de análise. Os valores são idênticos aos da função 5BH, exceto que os bits 0, 1 e 2 sempre serão 0.

**CHKCHR** (5DH)

Função: Checa caractere. Caracteres de 16 bits também são checados.

Entrada: D – Flags do caractere.

    b0=1 para suprimir o caractere. Neste caso, o caractere retornado em E será sempre o mesmo.

    b1=1 se for o primeiro byte de um caractere de 16 bits;

b2=1 se for o segundo byte de um caractere de 16 bits;  
 b3=1 nome do volume ou, preferencialmente, nome  
 de arquivo;  
 b4=1 caractere de arquivo/volume não válido;  
 b5~b7 são reservados (sempre 0).

E – Caractere a ser checado.

- Retorno: A – Sempre 00H.  
 D – Flags atualizadas do caractere.  
 E – Caractere checado.

#### **WPATH (5EH)**

Função: Ler string path completa, sem especificação de drive e caractere "\". Para maior confiabilidade, chamar primeiro a função 40H ou 41H e depois chamar WPATH duas vezes, já que outras funções podem alterar os dados.

- Entrada: DE – Apontador para um buffer de 64 bytes.  
 Retorno: A – Código de erro (se for 0, não houve erro).  
 DE – Buffer preenchido com a string path completa.  
 HL – Apontador para o início do último item.

#### **FLUSH (5FH)**

Função: Descarregar buffers de disco.

- Entrada: B – Especificação de drive (0=corrente; 1=A:, etc. Se for FFH, descarrega todos os drives).  
 D = 00H – Somente descarregar;  
 FFH – Descarregar e invalidar.

Retorno: A – Código de erro (se for 0, não houve erro).

#### **FORK (60H)**

Função: Ramificar arquivos em árvore.

Entrada: Nenhum.

- Retorno: A – código de erro (se for 0, não houve erro).  
 B – ID do processo de ramificação.

Nota: Novos arquivos handle são criados e os arquivos handle correntes que estão abertos no modo herdá vel (ver função 43H) são copiados para os novos arquivos handle. Os arquivos handle padrão (00H~05H) são copiados impreterivelmente. Pelo fato de haver uma cópia dos arquivos handle originais, se algum deles for fechado, poderá ser reaberto sem problemas.

**JOIN** (61H)

- Função: Reunir arquivos em árvore. Esta função retorna para o arquivo handle original o arquivo handle que foi copiado pela função anterior.
- Entrada: B – ID do processo de ramificação (ou 0).
- Retorno: A – Código de erro (se for 0, não houve erro).  
 B – Código de erro primário do ramo.  
 C – Código de erro secundário do ramo.
- Nota: O arquivo copiado é automaticamente fechado e o arquivo handle original é reativado. Se o registrador B for 00H na Entrada, uma reinicialização parcial do sistema é feita: todos os arquivos handle copiados são fechados e os arquivos handle originais são reativados. Se esta função for chamada pelo endereço F37DH, os registradores B e C não retornarão o código de erro. Ver a função 62H.

**TERM** (62H)

- Função: Finalizar com código de erro.
- Entrada: B – Código de erro para finalização.
- Retorno: Nenhum.
- Nota: A operação desta função é diferente conforme o endereço de chamada (0005H para MSXDOS ou F37DH para DISK-BASIC). Se for chamada por 0005H, a rotina de saída deve ser definida pela função 63H com o código de erro especificado (0 no caso de código de erro secundário) e se não houver rotina de saída definida pelo usuário, o sistema fará um salto para o endereço 0000H, provocando uma partida a quente do DOS. O interpretador de comandos somente imprimirá a mensagem de erro na tela somente se esta estiver entre 20H e FFH. Se esta função for chamada por F37DH, o controle será passado para o Interpretador BASIC que imprimirá a mensagem de erro.

**DEFAB** (63H)

- Função: Definir rotina de abortamento (saída). Somente disponível se for chamada por 0005H.
- Entrada: DE – Endereço inicial da rotina de abortamento; o endereço padrão é 0000H.
- Retorno: A – Sempre 00H.

**DEFER** (64H)

Função: Definir rotina do usuário para erro de disco.

Entrada: DE – Endereço inicial da rotina de erro de disco. O valor padrão é 0000H.

Retorno: A – Sempre 00H.

Nota: A especificação dos parâmetros e resultados da rotina criada são os seguintes:

Entrada: A – Código de erro;  
 B – Número do drive físico;  
 C – b0=1 se for erro de escrita;  
       b1=1 se ignorar o erro (não recomendado);  
       b2=1 se for sugerida abortagem automática;  
       b3=1 se o número do setor é válido;

DE – Número do setor do disco (se b3 de C for 1).

Retorno: A = 0 – Chama rotina de erro do sistema;  
 1 – Aborta;  
 2 – Tenta novamente;  
 3 – Ignora.

**ERROR** (65H)

Função: Pegar código de erro antecipadamente para prevenir o tipo de erro que poderá ocorrer na próxima chamada de função.

Entrada: Nenhum.

Retorno: A – Sempre 00H.  
 B – Código de erro da função.

**EXPLN** (66H)

Função: Retornar mensagem do código de erro.

Entrada: B – Código de erro.  
 DE – Apontador para um buffer de 64 bytes.

Retorno: A – Sempre 00H.  
 B – Código de erro ou 00H.  
 DE – Buffer preenchido com a mensagem de erro no formato ASCII.

**FORMAT** (67H)

Função: Formatar um disco.

Setup: B – número do drive (0=corrente; 1=A:, etc.).

A = 00H – Retorna mensagem de escolha;  
 01H~09H – Formata com esta escolha;  
 0AH~0DH – Illegal;  
 FEH – Atualiza parâmetros para MSXDOS2;  
 FFH – Atualização total para MDXDOS2.

HL – Apontador para o buffer (se A = 1~9).

DE – Tamanho do buffer (se A = 1~9).

Retorno: A – Código de erro (se for 0, não houve erro).  
 B – Slot da mensagem escolhida (só se A=0 na entrada).  
 HL – Endereço da mensagem escolhida (só se A=0).

### **RAMD (68H)**

Função: Criar ou apagar a ramdisk no drive “H:”.

Entrada: B = 00H – Apaga a ramdisk;  
 01H~FEH – Cria nova ramdisk com xxH segmentos  
 lógicos de 16K.  
 FFH – Retorna tamanho da ramdisk em segmentos  
 de 16K.

Retorno: A – Código de erro (se for 0, não houve erro).  
 B – Tamanho da ramdisk.

### **BUFFER (69H)**

Função: Alocar buffers (cada um tem 16K).

Entrada: B – 0 = retorna número de buffers alocados;  
 1 a 20 (01H a 14H) = aloca o nº especificado de buffers;  
 21 ou mais (mais que 15H) = inválido.

Retorno: A – Código de erro (se for 0, não houve erro).  
 B – Número total de buffers alocados.

### **ASSIGN (6AH)**

Função: Atribuir drive lógico a um drive físico.

Entrada: B – Número do drive lógico (1=A:, 2=B:, etc.).  
 0 – Cancela todas as atribuições (para D = 0);  
 1 a 7 – Atribui/cancela drive lógico respectivo.  
 D – Número do drive físico (1=A:, 2=B:, etc.);  
 0 – Cancela atribuição (para B = 1 a 7);  
 1 a 7 – Atribui drive físico respectivo;  
 FFH – Apenas retorna drive lógico em D.

Retorno: A – Código de erro (se for 0, não houve erro).  
 D – Número do drive físico.

**GENV** (6BH)

Função: Ler item externo.

Entrada: HL – Apontador para nome string em ASCII.  
DE – Apontador do buffer para string de valor.  
B – Tamanho do buffer. Se o buffer for pequeno, o valor  
de retorno será truncado e terminado em 00H. Um  
buffer de 255 bytes sempre será suficiente.

Retorno: A – Código de erro (se for 0, não houve erro).

DE – Apontador para o buffer preenchido.

**SENV** (6CH)

Função: Setar item externo.

Entrada: HL – Apontador para o nome em ASCII.  
DE – Apontador para a string de valor a ser setado. Deve  
ter até 255 caracteres e ser terminado em 00H. Se a  
string for nula, o item externo será removido.

Retorno: A – Código de erro (se for 0, não houve erro).

**FENV** (6DH)

Função: Procurar item externo.

Entrada: DE – Número do item externo.  
HL – Apontador do buffer para o nome em ASCII.

Retorno: A – Código de erro (se for 0, não houve erro).

HL – Apontador para o buffer preenchido, cujo fim é  
marcado com um byte 00H.

**DSKCHK** (6EH)

Função: Ativar ou desativar checagem do disco. Quando a checagem  
estiver ativa, o sistema recarregará o boot, a FAT, o FIB, o  
FCB, etc. Do disco toda vez que este for trocado.

Entrada: A = 00H – Ler valor de checagem do disco;  
01H – Setar valor de checagem do disco.  
B = 00H – Ativa (se A=01H);  
01H – Desativa (se A=01H).

Retorno: A – Código de erro (se for 0, não houve erro).

B – valor de checagem do disco corrente.

**DOSVER** (6FH)

- Função: Ler o número da versão do MSXDOS. Os valores retornados nos registradores BC e DE estarão em BCD. Assim, se a versão for 2.34, por exemplo, o valor retornado será 0234H. Para compatibilidade com o MSXDOS1 verifique primeiro se houve algum erro (A≠0).
- Entrada: Nenhum.
- Retorno: A – Código de erro (se for 0, não houve erro).  
BC – Versão do DOS Kernel.  
DE – Versão do MSXDOS2.SYS.

**REDIR** (70H)

- Função: Ler ou setar o estado de redirecionamento. O efeito desta função é temporário, no caso de A=01H e B=00H na entrada.
- Entrada: A = 00H – ler estado de redirecionamento;  
01H – setar estado de redirecionamento.  
B – Novo estado:  
b0 – Entrada padrão.  
b1 – Saída padrão.

**4.3.6 – Funções adicionadas pelo NEXTOR****FOUT** (71H)

- Função: Ativa ou desativar o modo STROUT rápido. Quando ativado, apenas os primeiros 511 caracteres serão impressos.
- Entrada: A – 00H = Obter o modo STROUT rápido;  
01H = Definir modo STROUT rápido.  
B – 00H = Desabilitar (apenas se A = 01H);  
FFH = Habilitar (somente se A = 01H).
- Saída: A – Código de erro (se for 0, não houve erro).  
B – Modo STROUT rápido atual.

**ZSTROUT** (72H)

- Função: Imprimir uma string terminada em zero. Esta função é afetada pelo modo STROUT rápido.
- Entrada: DE – Endereço da string.
- Saída: A = 0 (nunca retorna um erro).

**RDDRV** (73H)

- Função:** Ler os setores absolutos da unidade. Esta função é capaz de ler setores independentemente do sistema de arquivos visualizado na unidade (FAT12, FAT16 ou um sistema de arquivos desconhecido), e mesmo quando não há nenhum sistema de arquivos. Os setores lidos serão colocados a partir da DTA do disco atual.
- Entrada:** A – Número da unidade (0 = A:, 1=B:, etc.).  
 B – Número de setores para ler.  
 HL:DE – Número do setor.
- Saída:** A – Código de erro (se for 0, não houve erro).

**WRDRV** (74h)

- Função:** Gravar setores absolutos no disco. Esta função grava setores independentemente do sistema de arquivos na unidade (FAT12, FAT16 ou um sistema de arquivos desconhecido), e também quando não há nenhum sistema de arquivos. Os setores serão gravados a partir da DTA do disco atual.
- Entrada:** A – Número da unidade (0 = A:, 1=B:, etc.).  
 B – Número de setores para ler.  
 HL:DE – Número do setor.
- Saída:** A – Código de erro (se for 0, não houve erro).

**RALLOC** (75H)

- Função:** Obter ou definir vetor de modo de informação de alocação reduzida. O vetor atribui um bit para cada drive; o bit 0 de L é para A:, o bit 1 de L é para B:, etc. Este bit é 1 se o modo de alocação reduzida estiver atualmente habilitado (ao obter o vetor) ou a ser habilitado (ao definir o vetor) para o inversor, 0 quando o modo está desabilitado ou a ser desabilitado.
- Entrada:** A = 00H – Obter vetor atual;  
 01H – Definir vetor.  
 HL – Novo vetor (apenas se A = 01H).
- Saída:** A – 0 (nunca retorna um erro).  
 HL – Vetor atual.

**DSPACE** (76H)

- Função:** Obter informações de espaço em disco. O valor extra em BC será diferente de zero apenas quando a unidade de alocação mínima da unidade não for um número inteiro em Kbytes.

Entrada: E – número da unidade (0 = padrão, 1 = A ;, etc)  
 A = 00H – Obter espaço livre;  
 01H – Obter espaço total.

Saída: A – Código de erro (se for 0, não houve erro).  
 HL:DE – Espaço em Kbytes.  
 BC – Espaço extra em bytes.

### **LOCK (77H)**

Função: Bloquear / desbloquear uma unidade ou obter o estado de bloqueio de uma unidade. Quando uma unidade está bloqueada, o Nextor presumirá que a mídia nessa unidade nunca será alterada e, portanto, nunca solicitará ao driver associado o status de alteração da mídia; resultando assim em um aumento geral na velocidade de acesso à mídia. Isso é útil ao usar dispositivos removíveis como o dispositivo de armazenamento principal.

Entrada: E – unidade física (0 = A ;, 1 = B ;, etc).  
 A = 00H – Obter status de bloqueio;  
 01H – Definir status de bloqueio.  
 B = 00H – Desbloquear unidade (apenas se A = 01H);  
 FFH – Travar unidade (somente se A = 01H).  
 Saída: A – Código de erro (se for 0, não houve erro).  
 B – Estado de bloqueio atual, igual ao da entrada.

### **GDRV R (78H)**

Função: Obter informações sobre um driver de dispositivo.

Entrada: A – Índice do driver (0 para especificar slot e segmento).  
 D – Número do segmento do driver, FFH para drivers na ROM (apenas se A = 0).  
 HL – Ponteiro para buffer de dados de 64 bytes.

Saída: A – Código de erro (se for 0, não houve erro).  
 HL – Aponta para buffer preenchido com dados do driver.  
 +0: Número do slot do driver.  
 +1: Número do segmento do driver, FFH se o driver estiver embutido em um Nextor ou ROM Kernel MSX-DOS (sempre FFH na versão atual).  
 +2: Número de letras de unidade atribuídas a este driver no momento da inicialização.

- +3: Primeira letra de unidade atribuída a este driver no momento da inicialização (A: = 0, etc). Não utilizado se nenhuma unidade for atribuída no momento da inicialização.
- +4: Sinalizadores do driver:
  - bit 7: 1 → Driver do Nextor.
  - 0 → Driver MSX-DOS (embutido na ROM do Kernel MSX-DOS).
  - bits 6-3: Não utilizados, sempre “0000”.
  - bit 2: 1 → Driver implementa rotina DRV\_CONFIG.
  - bit 1: Não utilizado, sempre zero.
  - bit 0: 1 → Driver baseado em dispositivo.
  - 0 → Driver baseado em unidade.
- +5: Número da versão principal do driver.
- +6: Número da versão secundária do driver.
- +7: Número de revisão do driver.
- +8: Nome do driver, justificado à esquerda, completado com espaços (32 bytes).
- +40 ~ +63: Reservado (sempre zero).

### **GDLI** (79H)

- Função: Obter informações sobre uma letra de unidade.
- Entrada: A – Unidade física (0 = A :; 1 = B :; etc).  
HL – Ponteiro para buffer de dados de 64 bytes.
- Saída: A – Código de erro (se for 0, não houve erro).  
HL – Aponta para buffer preenchido.
- +0: Status da unidade:
    - 0 – Não atribuído
    - 1 – Atribuído a um dispositivo de armazenamento conectado a um driver Nextor ou MSX-DOS.
    - 2 – Não utilizado.
    - 3 – Um arquivo é montado na unidade.
    - 4 – Atribuído ao disco RAM (todos os outros campos serão zero).
  - +1: Número do slot do driver.
  - +2: Número do segmento do driver (FFH se o driver estiver embutido no Kernel Nextor ou DOS).
  - +3: Número relativo da unidade dentro do driver (para drivers baseados em unidade apenas FFH se driver for baseado em dispositivo).

- +4: Índice de dispositivo (apenas para drivers baseados em dispositivo; 0 para drivers MSX-DOS).
  - +5: Índice de unidade lógica (apenas para drivers baseados em dispositivo; 0 para drivers MSX-DOS).
  - +6 ~ +9: Número do primeiro setor do dispositivo (apenas para drivers baseados em dispositivo; se for driver MSX-DOS retornará 0).
  - +10 ~ +63: Reservado (sempre zero).
- Se um arquivo for montado na unidade, as informações retornadas no buffer de terão a seguinte forma:
- +1: Unidade onde o arquivo montado está localizado (0 = A:, 1 = B:, etc)
  - +2: bit 0 → 0 – Leitura e gravação; 1 – Só leitura.
  - +3: Sempre 0.
  - +4: Nome do arquivo em formato de impressão (até 12 caracteres, mais um zero de terminação).

## **GPART (7AH)**

- Função:** Obter informações sobre uma partição de dispositivo.  
Apenas para drivers baseados em dispositivo.
- Entrada:**
- A – Número do slot do driver.
  - B – N° do segmento do driver, FFH p/ drivers na ROM.
  - D – Índice do dispositivo.
  - E – Índice de unidade lógica.
  - H – Número da partição primária (1 a 4).
  - H:7 = 0 – Obter informações sobre a partição;
  - 1 – Obter o número do setor do dispositivo que contém a entrada da tabela de partição.
  - L – Número de partição estendida (0 para uma entrada na tabela de partição primária).
- Saída:**
- A – Código de erro (se for 0, não houve erro).  
→ Se as informações da partição forem solicitadas:
  - B – Código do tipo de partição:
    - 0 – Nenhum (a partição especificada não existe).
    - 1 – FAT12.
    - 4 – FAT16, menor que 32 MB (obsoleto).
    - 5 – Estendido (manipula mais de 4 partições).
    - 6 – FAT16 (CHS).
    - 14 – FAT16 (LBA).
  - C – Byte de status da partição.

**HL:DE** – Número inicial do setor absoluto da partição.

**IX:IY** – Tamanho da partição em setores.

→ Se o número do setor da entrada da tabela de partição for solicitado:

**HL:DE** – Número do setor do dispositivo que contém a entrada da tabela de partição.

### **CDRVR** (7BH)

**Função:** Chamar uma rotina em um driver de dispositivo. Esta função funciona no modo MSX-DOS 1.

**Entrada:** A – Número do slot do driver.

B – Nº do segmento do driver, FFH p/ drivers na ROM.

DE – Endereço de rotina.

HL – Apontador para um buffer de 8 bytes com os valores de entrada. A ordem dos registradores no buffer deve ser a seguinte: F, A, C, B, E, D, L, H.

**Saída:** A – Código de erro (se for 0, não houve erro).

BC, DE, HL – Resultados da rotina.

IX – Valor de AF retornado pela rotina.

### **MAPDRV** (7CH)

**Função:** Mapear uma letra de unidade para um driver de dispositivo

**Entrada:** A – unidade física (0 = A ;, 1 = B ;, etc)

B – Ação a realizar:

0 – Remover o mapeamento da unidade

1 – Mapear a unidade para seu estado padrão

2 – Mapear a unidade usando dados de mapeamento específicos

3 – Montar um arquivo na unidade

HL – Se B=2:

Endereço de um buffer de 8 bytes com dados de mapeamento com a seguinte estrutura:

+0 – Número do slot do driver

+1 – Número do segmento do driver (FFH se o driver estiver embutido em uma ROM do Kernel Nextor)

+2 – Número do dispositivo

+3 – Número da unidade lógica

+4 ~ +7 – Setor inicial

Se B=3:

Endereço de nome de arquivo ou FIB.

D – Tipo de montagem de arquivo (se B = 3)  
 0: Automático (somente leitura se o arquivo tiver esse atributo definido, leitura e gravação de outra forma)  
 1: Somente leitura

Saída: A – Código de erro (se for 0, não houve erro).

### **Z80MODE (7DH)**

Função: Habilitar ou desabilitar o acesso do Z80 para um driver.  
 Esta função funciona apenas em micros MSX Turbo R.

Entrada: A – Número do slot do driver  
 B = 00H – Obter modo de acesso Z80 atual  
 01H – Definir modo de acesso Z80  
 D = 00H – Desabilitar o modo de acesso Z80 (Se B=01H).  
 FFH – Habilitar modo de acesso Z80 (Se B=01H).

Saída: A – Código de erro (se for 0, não houve erro).  
 D – Modo de acesso Z80 atual para o driver especificado (igual ao da entrada).

### **GETCLUS (7EH)**

Função: Obter informações para um cluster em uma unidade FAT.

Entrada: A – Número da unidade (0 = padrão, 1 = A: etc.).  
 DE – Número do cluster.

HL – Apontador para um buffer de 16 bytes.

Saída: A – Código de erro (se for 0, não houve erro).

HL – Apontador para o buffer preenchido:

- +0 – Setor da FAT com a entrada do cluster (2 bytes).
- +2 – Offset no setor FAT c/ entrada do cluster (0-511).
- +4 – Primeiro setor de dados ao ref. ao cluster(4 bytes).
- +8 – Valor da entrada FAT para o cluster (2 bytes).
- +10: – Tamanho de um cluster em setores (1 byte).

+11: – Sinalizadores (1 byte):

bit 0 = 1 se a unidade for FAT12.

bit 1 = 1 se a unidade for FAT16.

bit 2 = 1 se a entrada FAT para o cluster for uma  
 entrada ímpar (apenas FAT12).

bit 3 = 1 se o cluster é o último de um arquivo.

bit 4 = 1 se o cluster estiver livre.

bits 5-7: Não utilizados, sempre zero.

+12 ~ +15: Não utilizados, sempre zero.

O valor da entrada FAT para o cluster tem os seguintes significados:

0 → cluster livre.

0FF8H–0FFFH p/ FAT12 e FFF8H–FFFFH p/ FAT16 →  
→ cluster é o último de um arquivo.

Outro valor → número do próximo cluster onde os dados para um arquivo continuam.

#### 4.4 – CÓDIGOS DE ERRO DO MSXDOS

Nº	Original inglês	Português
50	FIELD overflow	Campo maior
51	Internal error	Erro interno
52	Bad file number	Número de arquivo inválido
53	File not found	Arquivo não encontrado
54	File open	Arquivo aberto
55	End of file	Fim de arquivo
56	Bad file name	Nome de arquivo ruim
57	Direct statement in file	Comando direto no arquivo
58	Sequential I/O only	Somente acesso sequencial
59	File not OPEN	Arquivo não aberto
60	Disk error	Erro de disco
61	Bad file mode	Modo de arquivo errado
62	Bad drive name	Nome de drive errado
63	Bad sector	Setor com erro
64	File still open	Arquivo já aberto
65	File already exists	Arquivo já existe
66	Disk full	Disco cheio
67	Too many files	Diretório cheio
68	Write protected disk	Disco protegido contra escrita
69	Disk I/O error	Erro de I/O em disco
70	Disk offline	Sem disco
71	RENAME across disk	RENAME em discos diferentes

## 4.5 – CÓDIGOS DE ERRO DO MSXDOS2

### 4.5.1 – Erros de Disco

Nº	Original inglês	Português
FFH	Incompatible disk	Disco incompatível
FEH	Write error	Erro de escrita
FDH	Disk error	Erro de disco
FCH	Not ready	Não pronto
FBH	Verify error	Verificar erro
FAH	Data error	Erro de dados
F9H	Sector not found	Setor não encontrado
F8H	Write protected disk	Disco protegido contra escrita
F7H	Unfomatted disk	Disco não formatado
F6H	Not a DOS disk	Disco não DOS
F5H	Wrong disk	Disco errado
F4H	Wrong disk for file	Disco errado para arquivo
F3H	Seek error	Erro de procura
F2H	Bad file allocation table	Tabela de alocação de arquivos inválida
F1H	No message	Sem mensagem
F0H	Cannot format this drive	Este drive não pode ser formatado

### 4.5.2 – Erros das Funções do MSXDOS

DFH	Internal error	Erro interno
DEH	Not enough memory	Memória insuficiente
DDH	-	
DCH	Invalid MSX-DOS call	Chamada inválida do MSXDOS
DBH	Invalid drive	Drive inválido
DAH	Invalid filename	Nome de arquivo inválido
D9H	Invalid pathname	Nome da path inválido
D8H	Pathname too long	Nome da path muito longo
D7H	File not found	Arquivo não encontrado
D6H	Directory not found	Diretório não encontrado
D5H	Root directory full	Diretório raiz cheio
D4H	Disk full	Disco cheio
D3H	Duplicate filename	Nome de arquivo duplicado

D2H	Invalid directory move	Movimentação de diretório inválida
D1H	Read only file	Arquivo somente de leitura
D0H	Directory not empty	Diretório não vazio
CFH	Invalid attributes	Atributos inválidos
CEH	Invalid . or .. operation	Operação com . ou .. inválida
CDH	System file exists	Arquivo de sistema existe
CCH	Directory exists	Diretório existe
CBH	File exists	Arquivo existe
CAH	File already in use	Arquivo já em uso
C9H	Cannot transfer above 64K	Não pode transferir mais de 64K
C8H	File allocation error	Erro de alocação de arquivo
C7H	End of file	Fim de arquivo
C6H	File access violation	Violação de acesso ao arquivo
C5H	Invalid process id	Processo ID inválido
C4H	No spare file handles	Não há arquivos handle disponíveis
C3H	Invalid file handle	Arquivo handle inválido
C2H	File handle not open	Arquivo handle não aberto
C1H	Invalid device operation	Operação de dispositivo inválida
C0H	Invalid environment string	String inválida
BFH	Environment string too long	String muito longa
BEH	Invalid date	Data inválida
BDH	Invalid time	Hora inválida
BCH	RAM disk already exists	RAMDISK já existe
BBH	RAM disk does not exist	RAMDISK não existe
BAH	File handle has been deleted	Arquivo handle foi deletado
B9H	Internal error	Erro interno
B8H	Invalid sub-function number	Número de subfunção inválido

#### 4.5.3 – Erros Adicionados pelo Nextor

B6H	Invalid device driver	Driver de dispositivo incorreto
B5H	Invalid device or LUN	Dispositivo ou LUN inválido
B4H	Invalid partition number	Número de partição inválido
B3H	Partition is already in use	Partição já em uso
B2H	File is mounted	Arquivo está montado
B1H	Bad file size	Tamanho de arquivo incorreto
B0H	Invalid cluster number	Número de cluster inválido

#### 4.5.4 – Erros de Término de Programas

9FH	Ctrl-STOP pressed	CTRL+STOP pressionadas
9EH	Ctrl-C pressed	CTRL+C pressionadas
9DH	Disk operation aborted	Operação de disco abortada
9CH	Error on standard output	Erro na saída padrão
9BH	Error on standard input	Erro na entrada padrão

#### 4.5.5 – Erros de Comando

8FH	Wrong version off COMMAND	Versão errada do COMMAND
8EH	Unrecognized command	Comando não reconhecido
8DH	Command too long	Comando muito longo
8CH	Internal error	Erro interno
8BH	Invalid parameter	Parâmetro inválido
8AH	Too many parameters	Excesso de parâmetros
89H	Missing parameter	Falta parâmetro
88H	Invalid option	Opção inválida
87H	Invalid number	Número inválido
86H	File for HELP not found	Arquivo HELP não encontrado
85H	Wrong version of MSX-DOS	Versão errada do MSXDOS
84H	Cannot concatenate destination file	Arquivo de destino não pode ser concatenado
83H	Cannot create destination file	Arquivo de destino não pode ser criado
82H	File cannot be copied onto itself	Arquivo não pode ser copiado nele mesmo
81H	Cannot overwrite previous destination file	Arquivo de destino não pode ser previamente escrito

## 5 – SYMBOS

### 5.1 – ROTINAS DO KERNEL

#### 5.1.1 – Kernel Restarts

##### **RST 08H (MSGSLP)** – Message\_Sleep\_And\_Receive

Descrição: Verifica se há uma nova mensagem de outro processo. Se não houver mensagem, o processo será colocado em modo de espera, até que uma mensagem esteja disponível.

Entrada: IXI – ID do processo do receptor (o seu).  
IXh – ID do processo do remetente ou –1 para verificar mensagens de qualquer processo.  
IY – Apontador para buffer de mensagem (14 bytes).

Saída: IXI – 0 → Nenhuma mensagem disponível;  
1 → Mensagem recebida).  
IXh – ID do processo remetente (se IXI = 1).

Registradores: AF, BC, DE, HL.

##### **RST 10H (MSG SND)** – Message\_Send

Descrição: Envia uma mensagem para outro processo. Se a fila de mensagens estiver cheia ou o receptor não existir, ela não será enviada. A mensagem deve estar entre C000H e FFFFH e pode ter um tamanho máximo de 14 bytes.

Entrada: IXI – ID do processo do remetente (seu próprio)  
IXh – ID do processo do receptor  
IY – Apontador para a mensagem (1-14 bytes)

Saída: IXI – 0 → A fila de mensagens está cheia,  
1 → Mensagem enviada com sucesso,  
2 → Processo receptor não existe

Registradores: AF, BC, DE, HL.

##### **RST 18H (MSGGET)** – Message\_Receive

Descrição: Verifica se há uma nova mensagem de outro processo. O buffer de mensagem deve ter um tamanho de 14 bytes e ser sempre colocado entre C000H e FFFFH.

Entrada: IXI – ID do processo do receptor (o seu)  
IXh – ID do processo do remetente ou –1 para verificar mensagens de qualquer processo.  
IY – Apontador para buffer de mensagem (14 bytes)

Saída:       $|XI| = 0 \rightarrow$  Nenhuma mensagem disponível;  
                $1 \rightarrow$  Mensagem recebida).

$|Xh|$  – ID do processo remetente (se  $|XI| = 1$ ).

Registradores: AF, BC, DE, HL.

#### **RST 20H (BNKSCL) – Banking\_SlowCall**

Descrição: Chama uma rotina, que é colocada no primeiro banco de RAM. Todos os registradores serão transferidos sem modificações de e para a rotina. O endereço da rotina deve ser especificado logo após o comando RST.

Entrada:  $(SP + 0)$  – Endereço de destino.

AF, BC, DE, HL, IX, IY – Regs para a rotina de destino.

Saída: AF, BC, DE, HL, IX, IY – Regs da rotina de destino.

Registradores: –

#### **RST 28H (BNKFCL) – Banking\_FastCall**

Descrição: Chama uma rotina, que é colocada no primeiro banco RAM. DE, IX e IY serão transferidos sem modificações de e para a rotina. É mais rápida do que RST 20H (BNKSCL).

Entrada: HL – Endereço de destino.

DE, IX, IY – Registradores para a rotina de destino.

Saída: DE, IX, IY – Registradores da rotina de destino.

Registradores: AF, BC, HL.

#### **RST 30H (MTSOFT) – Multitasking\_SoftInterrupt**

Descrição: Libera o tempo de CPU para o sistema. O processo que chamou essa função é marcado como “ocioso”.

Entrada: –

Saída: –

Registradores: –

#### **RST 38H (MTHARD) – Multitasking\_HardInterrupt**

Descrição: Entrada do manipulador de interrupção de hardware, que é chamada 50, 60 ou 300 vezes por segundo, dependendo do computador. Não deve ser chamada pelo usuário.

Entrada: –

Saída: –

Registradores: –

### 5.1.2 – Comandos do Kernel (Gerenciamento Multitarefa)

Os comandos do Kernel são acionados por meio de uma mensagem, que deve ser enviada com RST 10H (MSGSEND) para o processo do Kernel, que sempre tem o ID 1.

#### ID: 001 (MSC\_KRL\_MTADDP) – Multitasking\_Add\_Process\_Command

Descrição: Adiciona um novo processo com uma determinada prioridade e o inicia imediatamente. Os processos de aplicação geralmente serão iniciados com a prioridade 4. A pilha deve sempre ser colocada entre C000H e FFFFH (Área de transferência da RAM), com a seguinte estrutura:

```
ds 128; buffer de pilha
stack_pointer: dw 0; valor inicial para IX
               dw 0 ; valor inicial para HL
               dw 0 ; valor inicial para DE
               dw 0 ; valor inicial para BC
               dw 0 ; valor inicial para AF
               dw process_start ; end. início processo
process_id db 0 ; Kernel escreve o ID aqui
```

Biblioteca: SyKernel\_MTADDP

Mensagem: 00 1B 001.

01 1W Endereço da pilha.

03 1B Prioridade (1 – Mais alta, 7 – Mais baixa).

04 1B Banco de RAM (0~15).

Resposta: Consulte MSR\_KRL\_MTADDP.

#### ID: 002 (MSC\_KRL\_MTDELP) – Multitasking\_Delete\_Process\_Command

Descrição: Interrompe um processo existente e o exclui.

Biblioteca: SyKernel\_MTDELP.

Mensagem: 00 1B 002.

01 1B ID do processo.

Resposta: Consulte MSR\_KRL\_MTDELP.

#### ID: 003 (MSC\_KRL\_MTADDT) – Multitasking\_Add\_Timer\_Command

Descrição: Adiciona um novo timer e o inicia imediatamente. Os timers serão chamados 50 ou 60 vezes por segundo, dependendo da frequência do VSYNC da tela.

Biblioteca: SyKernel\_MTADDT.

Mensagem: 00 1B 003.

01 1W Endereço da pilha.

04 1B Banco da RAM (0~15).

Resposta: Consulte MSR\_KRL\_MTADDT.

**ID: 004 (MSC\_KRL\_MTDELT)** – Multitasking\_Delete\_Timer\_Command

Descrição: Interrompe um timer existente e o exclui.

Biblioteca: SyKernel\_MTDELT.

Mensagem: 00 1B 004.

01 1B ID do timer.

Resposta: Consulte MSR\_KRL\_MTDELT.

**ID: 005 (MSC\_KRL\_MTSLPP)** – Multitasking\_Sleep\_Process\_Command

Descrição: Coloca um processo existente no modo de hibernação.

Biblioteca: SyKernel\_MTSLPP.

Mensagem: 00 1B 005.

01 1B ID do processo.

Resposta: Consulte MSR\_KRL\_MTSLPP.

**ID: 006 (MSC\_KRL\_MTWAKP)** –

– Multitasking\_WakeUp\_Process\_Command

Descrição: Acorda um processo que estava adormecido.

Biblioteca: SyKernel\_MTWAKP.

Mensagem: 00 1B 006.

01 1B ID do processo.

Resposta: Consulte MSR\_KRL\_MTWAKP.

**ID: 007 (MSC\_KRL\_TMAADDT)** – Timer\_Add\_Counter\_Command

Descrição: Adiciona um contador para um processo. É preciso reservar um byte em qualquer lugar da memória, que será incrementado a cada [P5]/50 ou [P5]/60 segundos.

Biblioteca: SyKernel\_TMAADDT

Mensagem: 00 1B 007

01 1W Endereço do byte do contador.

03 1B Byte contador do banco de RAM (0~15).

04 1B ID do Processo.

05 1B Velocidade (contador inc. cada x/50, x/60 seg.)

Resposta: Consulte MSR\_KRL\_TMAADDT

**ID: 008 (MSC\_KRL\_TMDELT)** – Timer\_Delete\_Counter\_Command

Descrição: Interrompe o contador especificado.

Biblioteca: SyKernel\_TMDELT.

Mensagem: 00 1B 008.

01 1W Endereço do byte do contador.

03 1B Byte contador do banco de RAM (0~15).

Resposta: Consulte MSR\_KRL\_TMDELT.

**ID: 009 (MSC\_KRL\_TMDELP)** –

– Timer\_Delete\_AllProcessCounters\_Command

Descrição: Interrompe todos os contadores de um processo.

Biblioteca: SyKernel\_TMDELP.

Mensagem: 00 1B 009.

01 1B ID do processo.

Resposta: Consulte MSR\_KRL\_TMDELP.

**ID: 010 (MSC\_KRL\_MTPRIO)** –

– Multitasking\_Process\_Priority\_Command

Descrição: Altera a prioridade de um processo. Um processo pode alterar sua própria prioridade.

Biblioteca: SyKernel\_MTPRIO.

Mensagem: 00 1B 010.

01 1B ID do processo.

01 1B Nova Prioridade (1 – Mais alta, 7 – Mais baixa).

Resposta: Consulte MSR\_KRL\_MTPRIO.

### 5.1.3 – Respostas do Kernel (Gerenciamento Multitarefa)

As respostas do Kernel vêm na forma de mensagens, que devem ser recebida com RST 18H (MSG SND) ou RST 08H (MSG SLP) do processo do Kernel, que sempre tem o ID 1.

**ID: 129 (MSR\_KRL\_MTADDP)** – Multitasking\_Add\_Process\_Response

Descrição: O Kernel envia esta mensagem após tentar adicionar um novo processo (consulte MSC\_KRL\_MTADDP). Novos processos não devem ser adicionados antes de receber esta mensagem.

Mensagem: 00 1B 129.

01 1B estado de erro (0 – Sucesso, 1 – Falha).

02 1B ID do Processo (se P1 = 0).

**ID: 130 (MSR\_KRL\_MTDELP)** – Multitasking\_Delete\_Process\_Response

Descrição: O Kernel envia esta mensagem após excluir um processo existente (consulte MSC\_KRL\_MTDELP).

Mensagem: 00 1B 130.

**ID: 131 (MSR\_KRL\_MTADDT)** – Multitasking\_Add\_Timer\_Response

Descrição: O Kernel envia esta mensagem após tentar adicionar um novo timer (consulte MSC\_KRL\_MTADDT). Novos timers não devem ser adicionados até receber esta mensagem.

Mensagem: 00 1B 131.

01 1B estado de erro (0 – Sucesso, 1 – Falha).

02 1B ID do temporizador (se P1 = 0).

**ID: 132 (MSR\_KRL\_MTDELT)** – Multitasking\_Delete\_Timer\_Response

Descrição: O Kernel envia esta mensagem após excluir um timer existente (consulte MSC\_KRL\_MTDELT).

Mensagem: 00 1B 132.

**ID: 133 (MSR\_KRL\_MTSLPP)** – Multitasking\_Sleep\_Process\_Response

Descrição: O Kernel envia esta mensagem após colocar um processo em hibernação (consulte MSC\_KRL\_MTSLPP).

Mensagem: 00 1B 133.

**ID: 134 (MSR\_KRL\_MTWAKP)** –

– Multitasking\_WakeUp\_Process\_Response

Descrição: O Kernel envia esta mensagem após acordar um processo (consulte MSC\_KRL\_MTWAKP).

Mensagem: 00 1B 134.

**ID: 135 (MSR\_KRL\_TMADDT)** – Timer\_Add\_Counter\_Response

Descrição: O Kernel envia esta mensagem após tentar adicionar um novo contador (consulte MSC\_KRL\_TMADDT).

Mensagem: 00 1B 135.

01 1B estado de erro (0 – Sucesso, 1 – Falha).

**ID: 136 (MSR\_KRL\_TMLELT)** – Timer\_Delete\_Counter\_Response

Descrição: O Kernel envia esta mensagem após deletar um contador (consulte MSC\_KRL\_TMLELT).

Mensagem: 00 1B 136.

**ID: 137 (MSR\_KRL\_TMDELP) –**

– Timer\_Delete\_AllProcessCounters\_Response

Descrição: O Kernel envia esta mensagem após excluir todos os contadores de um processo (consulte MSC\_KRL\_TMDELP).

Mensagem: 00 1B 137.

**ID: 138 (MSR\_KRL\_MTPRIO) –**

– Multitasking\_Process\_Priority\_Response

Descrição: O Kernel envia esta mensagem após alterar a prioridade de um processo (consulte MSC\_KRL\_MTPRIO).

Mensagem: 00 1B 138.

### **5.1.4 – Funções do Kernel (Gerenciamento de Memória)**

Todas as funções de memória do Kernel devem ser chamadas com RST 20H (BNKSCL) ou RST 28H (BNKFCL). Muitas funções só podem ser chamadas uma vez ao mesmo tempo, então elas são protegidas com um mecanismo de sinalização. O processo de chamada será alterado para o modo inativo, enquanto a função estiver trabalhando para outro processo.

**MEMSUM (8100H) – Memory\_Summary**

Descrição: Retorna o tamanho total da memória ( $= D * 65\,536 + 65\,536$ ) e a quantidade de bytes ( $= E * 65\,536 + IX$ ) que ainda estão disponíveis.

Como chamar: Id hl, 8100H : rst 28H

Entrada: –

Saída: E, IX – Memória livre em bytes.

D – Número de bancos de 64K de RAM estendida.

Registradores: A, BC, IY.

**MEMINF (8121H) – Memory\_Information**

Descrição: Pesquisa a maior área livre dentro de um banco de 64K. Se o Banco RAM (A = 0) não for especificado, o sistema pesquisará a maior área dentro de toda a memória.

Como chamar: rst 20H : dw 8121H

Entrada: A – Banco RAM (1~15, 0 significa qualquer banco)

E – tipo de memória:

0 → Total (área de código).

1 → Dentro de um bloco de 16K (área de dados).

2 → Dentro do último bloco de 16K (área de transferência).

Saída: BC – Comprimento da maior área livre.  
 A, HL – Memória livre total em bytes.  
 Registradores: F, DE

#### **MEMGET (8118H) – Memory\_Get**

Descrição: Reserva memória em qualquer banco ou em um banco especial de RAM. Se o tipo de memória for 1, ele será reservada dentro de um bloco de 16K; se for 2, dentro do último bloco de 16k (área de transferência).

Como chamar: rst 20H : dw 8118H

Entrada: A – Banco RAM (1~15, 0 significa qualquer banco).

E – Tipo de memória:  
 0 → total (área de código).  
 1 → dentro de um bloco de 16K (área de dados).  
 2 → dentro do último bloco de 16K (área de transferência).

BC – Comprimento em bytes.

Saída: A – Banco RAM (1~15).

HL – Endereço.

CY – Estado de erro (CY=1 → Memória livre insuficiente).

Registradores: BC, DE.

#### **MEMFRE (811BH) – Memory\_Free**

Descrição: Libera a memória especificada.

Como chamar: rst 20H : dw 811BH

Entrada: A – Banco RAM (1~15).

HL – Endereço.

BC – Comprimento em bytes.

Saída: –

Registradores: AF, BC, E, HL.

#### **MEMSIZ (811EH) – Memory\_Resize**

Descrição: Altera o tamanho de uma área de memória reservada.

Sempre funcionará se o novo comprimento for menor que o anterior. Se o usuário já iniciou outros aplicativos, há poucas chances de aumentar uma área de memória reservada existente.

Como chamar: rst 20H : dw 811EH

Entrada: A – Banco RAM (1~15).  
 HL – Endereço.  
 BC – Comprimento antigo em bytes.  
 DE – Novo comprimento em bytes.  
 Saída: CY – Estado de erro (CY=1 -> memória livre insuficiente).  
 Registradores: AF, BC, DE, HL.

### 5.1.5 – Funções do Kernel (Bancos de Memória)

#### **BNKRWD** (8124H) – Banking\_ReadWord

Descrição: Lê uma palavra de um endereço em qualquer Banco RAM.  
 Como chamar: rst 20H : dw 8124H  
 Entrada: A – Banco RAM (0~15).  
 HL – Endereço.  
 Resultado: BC – Conteúdo de A, HL.  
 HL – Endereço + 2.  
 Registradores: –

#### **BNKWWD** (8127H) – Banking\_WriteWord

Descrição: Grava uma palavra em um endereço em qualquer Banco RAM.  
 Como chamar: rst 20H : dw 8127H  
 Entrada: A – Banco RAM (0~15).  
 HL – Endereço.  
 BC – Word.  
 Saída: HL – Endereço + 2.  
 Registradores: BC.

#### **BNKRBT** (812AH) – Banking\_ReadByte

Descrição: lê um byte de um endereço em qualquer Banco RAM.  
 Como chamar: rst 20H : dw 812AH  
 Entrada: A – Banco RAM (0~15).  
 HL – Endereço.  
 Saída: B – Conteúdo de A, HL.  
 HL – Endereço + 1.  
 Registradores: –

#### **BNKWBT** (812DH) – Banking\_WriteByte

Descrição: grava um byte em um endereço em qualquer Banco RAM.  
 Como chamar: rst 20H : dw 812DH

Entrada: A – Banco RAM (0~15).

HL – Endereço.

B – Byte.

Saída: HL – Endereço + 1.

Registradores: BC.

#### **BNKCOP (8130H) – Banking\_Copy**

Descrição: Copia uma área de memória de um endereço em qualquer Banco RAM para qualquer outro lugar na memória. O nibble baixo do registro A (bit 0-3) especifica o banco fonte e o nibble alto (bit 4-7) o banco de destino.

Como chamar: rst 20H : dw 8130H

Entrada: A – bit0–3: Banco RAM de origem (0~15).

bit4–7: Banco de memória RAM de destino (0~15).

HL – Endereço de origem.

DE – Endereço de destino.

BC – Comprimento.

Saída: –

Registradores: AF, BC, DE, HL.

#### **BNKGET (8133H) – Banking\_GetBank**

Descrição: Devolve o número do Banco RAM onde está o processo rodando.

Como chamar: rst 20H : dw 8133H.

Entrada: –

Saída: A – Banco RAM (1~15).

Registradores: F.

#### **BNK16C (8142H) – Banking\_Call\_Application16KRoutine**

Descrição: Permite executar uma rotina de aplicativo no primeiro banco de RAM. A rotina deve ser colocada em um banco de 16K que será alterado para 4000H–7FFFH antes da rotina ser chamada.

Como chamar: ld hl, 8142H : rst 28H

Entrada: IX – Apontador para estrutura de dados (deve estar entre C000H~FFFFH):

00 1B Banco de memória RAM da rotina (0~15).

01 1W Endereço de rotina.

03 1W Endereço da pilha temporária.

DE, IY – Serão entregues sem modificações para a rotina.

Saída: DE, IX, IY – Retornarão sem modificações pela rotina.  
 Registradores: AF, BC, HL.

#### **BNKCLL (FF03H) – Banking\_Interbank\_Call**

Descrição: Alterna para uma rotina em outro banco de memória RAM de 64K. Assim é possível ter áreas de código em vários bancos de 64K e alternar facilmente entre eles.

Como chamar: call FF03H

Entrada: IX – Endereço de rotina.

B – Banco de RAM da rotina (0~15).

IY – Endereço da pilha de rotinas.

DE, HL – Será entregue sem modificações para a rotina.

Saída: –

Registradores: AF, BC, IY

#### **BNKRET (FF00H) – Banking\_Interbank\_Return**

Descrição: Retorna de uma rotina dentro de outro banco de 64K para o chamador no banco principal. Consulte BNKCLL.

Como chamar: jp FF00H

Entrada: C, DE, HL, IX – Serão entregues sem modificações para o chamador.

Saída: –

Registradores: AF, B, IY.

#### **5.1.6 – Funções do Kernel (Diversas)**

##### **MTGCNT (8109H) – Multitasking\_GetCounter**

Descrição: Devolve o contador do sistema ( $= IY * 65\,536 + IX$ ) e o contador do processo ocioso. O contador do sistema é incrementado 50 ou 60 vezes por segundo. O processo ocioso aumenta seu contador a cada 64 microsegundos, quando possui o tempo de CPU. Com esses dois contadores, pode ser calculado o uso da CPU da seguinte forma:  
 $CPU\_usage = 100\% * (idle\_counter\_NEW - idle\_counter\_OLD) / ((system\_counter\_NEW - system\_counter\_OLD) * 312)$ .

Como chamar: ld hl, 8109H : RST 28H

Entrada: –

Saída: IY, IX – Contador do sistema.

DE – Contador ocioso.

Registradores: –

## 5.2 – GERENCIADOR DA ÁREA DE TRABALHO

O gerenciador de área de trabalho é responsável por todas as ações que ocorrem na tela de vídeo. Os comandos do gerenciador são acionados por meio de uma mensagem, que deve ser enviada com RST 10H (MSG SND) para o processo do gerenciador, que sempre tem o ID 2.

### **ID: 032 (MSC\_DSK\_WINOPN) – Window\_Open\_Command**

Descrição: Abre uma nova janela. O registro de dados deve ser colocado na área de transferência (entre C000H e FFFFH).

Biblioteca: SyDesktop\_WINOPN.

Mensagem: 00 1B 032.

01 1B Banco RAM do registro de dados da janela (0~8).

02 1W Endereço de registro de dados de janela  
(C000H~FFFFH).

Resposta: Consulte MSR\_DSK\_WOPNER e MSR\_DSK\_WOPNOK.

### **ID: 033 (MSC\_DSK\_WINMEN) – Window\_Redraw\_Menu\_Command**

Descrição: Redesenha a barra de menu de uma janela. Só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WINMEN.

Mensagem: 00 1B 033.

01 1B ID da janela.

### **ID: 034 (MSC\_DSK\_WININH) – Window\_Redraw\_Content\_Command**

Descrição: Redesenha um, todos ou um número especificado de controles dentro do conteúdo da janela. É idêntico a MSC\_DSK\_WINDIN, com a exceção de que só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WININH.

Mensagem: 00 1B 034.

01 1B ID da janela.

02 1B -1 → ID de controle ou nº negativo de controles  
000~239 → O controle com o ID especificado  
será redesenhado.

240~254 → Redesenha os controles -P2 a partir  
do controle P3. Por exemplo, se P2  
for -3 (253) e P3 for 5, os controles 5,  
6 e 7 serão redesenhados.

255 → Redesenha todos os controles dentro da  
janela atual.

→ Se P2 estiver entre 240 e 254:

03 1B ID do primeiro controle a ser redesenhadado.

**ID: 035 (MSC\_DSK\_WINTOL) – Window\_Redraw\_Toolbar\_Command**

Descrição: Redesenha um, todos ou um número especificado de controles dentro da barra de ferramentas da janela. Só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WINTOL

Mensagem: 00 1B 035.

01 1B ID da janela.

02 1B -1 → ID de controle ou nº negativo de controles  
000~239 → O controle com o ID especificado  
será redesenhadado.

240~254 → Redesenha os controles -P2 a partir  
do controle P3. Por exemplo, se P2  
for -3 (253) e P3 for 5, os controles 5,  
6 e 7 serão redesenhadados.

255 → Redesenha todos os controles dentro da  
janela atual.

→ Se P2 estiver entre 240 e 254:

03 1B ID do primeiro controle a ser redesenhadado.

**ID: 036 (MSC\_DSK\_WINTIT) – Window\_Redraw\_Title\_Command**

Descrição: Redesenha a barra de título de uma janela. Só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WINTIT

Mensagem: 00 1B 036

01 1B ID da janela.

**ID: 037 (MSC\_DSK\_WINSTA) – Window\_Redraw\_Statusbar\_Command**

Descrição: Redesenha a barra de estado de uma janela. Só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WINSTA.

Mensagem: 00 1B 037.

01 1B ID da janela.

**ID: 038 (MSC\_DSK\_WINMVX) – Window\_Set\_ContentX\_Command**

Descrição: Se o tamanho do conteúdo da janela for maior do que a parte visível, este comando permite definir o deslocamento horizontal. Só funciona se a janela estiver em foco.

Biblioteca: SyDesktop\_WINMVX.

Mensagem: 00 1B 038.

01 1B ID da janela.

02 1W Novo deslocamento X do conteúdo da janela.

**ID: 039 (MSC\_DSK\_WINMVY) – Window\_Set\_ContentY\_Command**

Biblioteca: SyDesktop\_WINMVY

Descrição: Se o tamanho do conteúdo da janela for maior do que a parte visível, este comando permite definir o deslocamento vertical. Só funciona se a janela estiver em foco.

Mensagem: 00 1B 039.

01 1B ID da janela.

02 1W Novo deslocamento Y do conteúdo da janela.

**ID: 040 (MSC\_DSK\_WINTOP) – Window\_Focus\_Command**

Descrição: Leva a janela para a posição frontal na tela. Sempre funciona.

Biblioteca: SyDesktop\_WINTOP.

Mensagem: 00 1B 040.

01 1B ID da janela.

**ID: 041 (MSC\_DSK\_WINMAX) – Window\_Size\_Maximize\_Command**

Descrição: Maximiza uma janela. Uma janela maximizada não pode ser movida para outra posição da tela. Este comando só funciona se a janela estiver minimizada ou restaurada.

Biblioteca: SyDesktop\_WINMAX.

Mensagem: 00 1B 041.

01 1B ID da janela.

**ID: 042 (MSC\_DSK\_WINMIN) – Window\_Size\_Minimize\_Command**

Descrição: Minimiza uma janela. Ele vai desaparecer da tela só pode ser acessada pelo usuário através da barra de tarefas. Só funciona se a janela estiver maximizada ou restaurada.

Biblioteca: SyDesktop\_WINMIN.

Mensagem: 00 1B 042.

01 1B ID da janela.

**ID: 043 (MSC\_DSK\_WINMID) – Window\_Size\_Restore\_Command**

Descrição: Restaura a janela ou o tamanho da janela. Só funciona se a janela estiver maximizada ou minimizada.

Biblioteca: SyDesktop\_WINMID

Mensagem: 00 1B 043

01 1B ID da janela.

#### **ID: 044 (MSC\_DSK\_WINMOV) – Window\_Set\_Position\_Command**

Descrição: Move a janela para outra posição na tela. Só funciona se a janela não estiver maximizada.

Biblioteca: SyDesktop\_WINMOV.

Mensagem: 00 1B 044.

01 1B ID da janela.

02 1W Nova posição X da janela.

04 1W Nova posição Y da janela.

#### **ID: 045 (MSC\_DSK\_WINSIZ) – Window\_Set\_Size\_Command**

Descrição: Redimensiona uma janela. O tamanho sempre se refere ao conteúdo visível da janela, não à janela inteira, incluindo o controle elementos. Este comando sempre funciona.

Biblioteca: SyDesktop\_WINSIZ.

Mensagem: 00 1B 045.

01 1B ID da janela.

02 1W Nova largura da janela.

04 1W Nova altura da janela.

#### **ID: 046 (MSC\_DSK\_WINCLS) – Window\_Close\_Command**

Descrição: Fecha a janela. Sempre funciona.

Biblioteca: SyDesktop\_WINCLS.

Mensagem: 00 1B 046.

01 1B ID da janela.

#### **ID: 047 (MSC\_DSK\_WINDIN) –**

**- Window\_Redraw\_ContentExtended\_Command**

Biblioteca: SyDesktop\_WINDIN

Descrição: Redesenha um, todos ou um número especificado de controles dentro do conteúdo da janela. Este comando é idêntico ao MSC\_DSK\_WININH, com a exceção de que sempre funciona, mas com menor velocidade.

Mensagem: 00 1B 047.

01 1B ID da janela.

- 02 1B ID de controle, -1 (todos) ou número negativo de controle.  
 000~239 → O controle com o ID especificado será redesenhado.  
 240~254 → Redesenha os controles -P2 a partir do controle P3. Por exemplo, se P2 for -3 (253) e P3 for 5, os controles 5, 6 e 7 serão redesenhados.  
 255 → Redesenha todos os controles dentro da janela atual.  
 → Se P2 estiver entre 240 e 254:  
 03 1B ID do primeiro controle a ser redesenhado.

**ID: 048 (MSC\_DSK\_DSRSRV)** – Desktop\_Service\_Command

Descrição: Por favor, leia a descrição do serviço “Gerenciador da Área de Trabalho” para mais informações.

Biblioteca: Consultar SERVIÇOS DO GERENCIADOR DA ÁREA DE TRABALHO.

Mensagem: 00 1B 048.

01 1B ID do serviço.

02~05 Veja a descrição do serviço “Gerenciador da Área de Trabalho”.

Resposta: Consultar MSR\_DSK\_DSRSRV.

**ID: 049 (MSC\_DSK\_WINSLD)** – Window\_Redraw\_Slider\_Command

Descrição: Redesenha os dois controles deslizantes da janela, com os quais o usuário pode rolar o conteúdo. Só funciona se a janela estiver em foco

Biblioteca: SyDesktop\_WINSLD.

Mensagem: 00 1B 049.

01 1B ID da janela.

**ID: 050 (MSC\_DSK\_WINPIN)** –

– Window\_Redraw\_ContentÁrea\_Command

Descrição: Este comando funciona como MSC\_DSK\_WINDIN, mas atualiza apenas uma área especificada dentro do conteúdo da janela. Para mais informações, consulte MSC\_DSK\_WINDIN e MSC\_DSK\_WININH. Este comando sempre funciona.

Biblioteca: SyDesktop\_WINPIN

Mensagem: 00 1B 050

01 1B ID da janela.

02 1B ID de controle, -1 (todos) ou número negativo de controle.

000~239 → O controle com o ID especificado será redesenhadado.

240~254 → Redesenha os controles –P2 a partir do controle P3. Por exemplo, se P2 for -3 (253) e P3 for 5, os controles 5, 6 e 7 serão redesenhadados.

255 → Redesenha todos os controles dentro da janela atual.

04 1W Início da área X dentro do conteúdo da janela.

06 1W Início da área Y (vertical).

08 1W Largura X da área (horizontal).

10 1W Altura Y da área (vertical).

→ Se P2 estiver entre 240 e 254:

03 1B ID do primeiro controle que deve ser redesenhadado.

#### **ID: 051 (MSC\_DSK\_WINSIN) –**

– Window\_Redraw\_SubControl\_Command

Descrição: Este comando funciona como MSC\_DSK\_WINDIN, mas atualiza apenas um subcontrole dentro de uma coleção de controles (não suporta o redesenho de vários subcontroles). Consultar MSC\_DSK\_WINDIN. Este comando sempre funciona.

Biblioteca: SyDesktop\_WINSIN.

Mensagem: 00 1B 051.

01 1B ID da janela.

02 1B ID de coleta de controle.

03 1B ID do subcontrole dentro da coleção de controles.

#### **5.2.1 – Respostas do Gerenciador da Área de Trabalho**

##### **ID: 160 (MSR\_DSK\_WOPNER) – Window\_OpenError\_Response**

Descrição: Não foi possível abrir a janela, pois o número máximo de janelas (32) já foi alcançado.

Mensagem: 00 1B 160.

**ID: 161 (MSR\_DSK\_WOPNOK) – Window\_OpenOK\_Response**

Descrição: A janela foi aberta. O gerenciador de área de trabalho retorna o ID respectivo, que será necessário para todos os comandos relativos à nova janela

Mensagem: 00 1B 161.  
04 1B ID da janela.

**ID: 162 (MSR\_DSK\_WCLICK) – Window\_UserAction\_Response**

Descrição: O gerenciador de área de trabalho envia esta mensagem para o aplicativo, se o usuário fizer uma interação com a janela ou com os controles da mesma.

Mensagem: 00 1B 162.  
01 1B ID da janela.  
02 1B Tipo de ação:  
05 – O botão Fechar foi clicado ou ALT + F4 foi pressionado (DSK\_ACT\_CLOSE).  
06 – A entrada do menu foi clicada (DSK\_ACT\_MENU). P8 conterá o valor de entrada do menu.  
14 – Um controle do conteúdo da janela foi clicado e/ou modificado com o teclado ou mouse (DSK\_ACT\_CONTENT). P8 conterá o valor de controle e P4/6 a posição do mouse, se o usuário o usou.  
15 – Um controle da barra de ferramentas da janela foi clicado e/ou modificado com o teclado ou mouse (DSK\_ACT\_TOOLBAR). P8 conterá o valor de controle e P4/6 a posição do mouse, se o usuário o usou.  
16 – O usuário pressionou alguma tecla sem modificar nenhum controle (DSK\_ACT\_KEY). P4 conterá o código ASCII.

→ se P2 for 14 ou 15:

03 1B Subespecificação de ação:  
00 – O botão esquerdo do mouse foi clicado (DSK\_SUB\_MLCLICK).  
01 – O botão direito do mouse foi clicado (DSK\_SUB\_MRCLICK).  
02 – O botão esquerdo do mouse clicado duas vezes (DSK\_SUB\_MDCLICK).

- 03 – O botão central do mouse foi clicado (DSK\_SUB\_MMCLICK).
- 07 – Uma tecla foi pressionada (DSK\_SUB\_KEY)
  - Se P2 for 14 ou 15 e P3 estiver entre 0 e 3:
- 04 1W Posição X do mouse de (dentro do conteúdo da janela / barra de ferramentas).
- 06 1W Posição Y do mouse.
- se P2 for 14 ou 15 e P3 for 7, ou se P2 for 16:
- 04 1B Código ASCII da tecla pressionada.
- Se P2 for 6, 14 ou 15:
- 08 1W Valor de entrada do menu ou valor de controle.

**ID: 163 (MSR\_DSK\_DSRSRV)** – Desktop\_Service\_Response

Descrição: Por favor, leia a descrição do serviço gerenciador de área de trabalho para mais informações.

- Mensagem: 00 1B 163.
- 01 1B ID do serviço.
  - 02~05 Consultar a descrição do serviço do gerenciador de área de trabalho.

**ID: 164 (MSR\_DSK\_WFOCUS)** – Window\_Focus\_Response

Descrição: O gerenciador de área de trabalho envia esta mensagem para o aplicativo se o estado do foco de uma janela mudou.

- Mensagem: 00 1B 164.
- 01 1B ID da janela.
  - 02 1B Estado:
    - 0 – A janela perdeu posição de foco.
    - 1 – A janela recebeu posição de foco.

**ID: 165 (MSR\_DSK\_CFOCUS)** – Control\_Focus\_Response

Descrição: O gerenciador de área de trabalho envia esta mensagem para o aplicativo, se outro controle dentro de uma janela obtiver o foco. O ID de controle não é seu número dentro do grupo de controle (começando com 1).

- Mensagem: 00 1B 165.
- 01 1B ID da janela.
  - 02 1B ID do novo controle de foco (começando com 1).
  - 03 1B Razão para mudança de foco:
    - 0 – O usuário clicou no controle com o mouse ou usou a roda do mouse.
    - 1 – O usuário pressionou a tecla TAB.

**ID: 166 (MSR\_DSK\_WRESIZ)** – Window\_Resize\_Response

Descrição: O gerenciador de área de trabalho envia esta mensagem para o aplicativo se o usuário redimensionar a janela. A mensagem também será enviada, se a janela for maximizada ou restaurada.

Mensagem: 00 1B 166  
01 1B ID da janela.

**ID: 167 (MSR\_DSK\_WSCROL)** – Window\_Scroll\_Response

Descrição: O gerenciador de área de trabalho envia esta mensagem para o aplicativo se o usuário rolar o conteúdo da janela (com as barras de rolagem, com os botões de seta ou com a roda do mouse).

Mensagem: 00 1B 167.  
01 1B ID da janela.

### **5.2.2 – Serviços do Gerenciador da Área de Trabalho**

**ID: 001 (DSK\_SRV\_MODGET)** – DesktopService\_ScreenModeGet

Descrição: Retorna a resolução da tela atual e o número de possíveis cores.

Biblioteca: SyDesktop\_MODGET

Mensagem: 00 1B 048  
01 1B 001

Resposta: 00 1B 163  
01 1B 001  
02 1B Modo de tela (depende da plataforma).  
 PCW 0 – 720 x 255, 2 cores (modo padrão PCW)  
 CPC: 1 – 320 x 200, 4 cores (modo padrão CPC)  
 2 – 640 x 200, 2 cores  
 EP: 1 – 320 x 200, 4 cores (modo padrão EP)  
 2 – 640 x 200, 2 cores  
 MSX: 5 – 256 x 212, 16 cores  
 6 – 512 x 212, 4 cores  
 7 – 512 x 212, 16 cores (padrão MSX)  
 G9K: 8 – 384 x 240, 16 cores  
 9 – 512 x 212, 16 cores (padrão G9K)  
 10 – 768 x 240, 16 cores  
 11 – 1024x 212, 16 cores

→ Se G9K:

03 1B Largura da área de trabalho virtual:

- 0 – Sem área de trabalho virtual.
- 1 – 512.
- 2 – 1000.

**ID: 002 (DSK\_SRV\_MODSET)** – DesktopService\_ScreenModeSet

Descrição: Define a resolução da tela e o número de cores possíveis.

Biblioteca: SyDesktop\_MODSET.

Mensagem: 00 1B 048

01 1B 002

02 1B Modo de tela (bit0~6). Os modos disponíveis dependem da plataforma do computador.

PCW 0 – 720 x 255, 2 cores (modo padrão PCW)

CPC: 1 – 320 x 200, 4 cores (modo padrão CPC)  
2 – 640 x 200, 2 cores

EP: 1 – 320 x 200, 4 cores (modo padrão EP)  
2 – 640 x 200, 2 cores

MSX: 5 – 256 x 212, 16 cores  
6 – 512 x 212, 4 cores

7 – 512 x 212, 16 cores (padrão MSX)

G9K: 8 – 384 x 240, 16 cores

9 – 512 x 212, 16 cores (padrão G9K)

10 – 768 x 240, 16 cores

11 – 1024x 212, 16 cores

→ Se G9K:

03 1B Largura da área de trabalho virtual:

- 0 – Sem área de trabalho virtual.
- 1 – 512.
- 2 – 1000.

Resposta: Nenhuma.

**ID: 003 (DSK\_SRV\_COLGET)** – DesktopService\_ColourGet

Descrição: Retorna a definição de uma cor. Observe que o intervalo sempre será de 4096.

Biblioteca: SyDesktop\_COLGET.

Mensagem: 00 1B 048.

01 1B 003.

02 1B Número da cor (0~15).

- Resposta: 00 1B 163.  
 01 1B 003.  
 02 1B Número da cor (0~15).  
 03 1B bit0-3: componente azul (0~15).  
              bit4-7: componente verde (0~15).  
 04 1B bit0-3: componente vermelho (0~15).

**ID: 004 (DSK\_SRV\_COLSET)** – DesktopService\_ColourSet

Descrição: Define uma cor. O intervalo sempre será de 4096.

Biblioteca: SyDesktop\_COLSET.

- Mensagem: 00 1B 048.  
 01 1B 004.  
 02 1B Número da cor (0~15).  
 03 1B bit0-3: componente azul (0~15).  
              bit4-7: componente verde (0~15).  
 04 1B bit0-3: componente vermelho (0~15).

Resposta: Nenhuma.

**ID: 008 (DSK\_SRV\_DSKBGR)** – DesktopService\_RedrawBackground

Descrição: Reinicializa e redesenha o plano de fundo da área de trabalho.

Biblioteca: SyDesktop\_DSKBGR.

- Mensagem: 00 1B 048.  
 01 1B 008.

Resposta: Nenhuma.

**ID: 009 (DSK\_SRV\_DSKPLT)** – DesktopService\_RedrawComplete

Descrição: Reinicializa o plano de fundo da área de trabalho e redesenha toda a tela. O fundo, a barra de tarefas e todas as janelas serão atualizadas.

Biblioteca: SyDesktop\_DSKPLT

- Mensagem: 00 1B 048.  
 01 1B 009.

Resposta: Nenhuma.

## 5.2.3 – Funções do Gerenciador da Área de Trabalho

**BUFPUT (814EH)** – Clipboard\_Put

Descrição: Copia os dados para a área de transferência. Se a área de transferência já contiver dados contidos, eles serão excluídos antes da cópia.

Como chamar: `rst 20H : dw 814EH`

Entrada: IX – Endereço dos dados de origem.

E – Banco de RAM de dados de origem (0~15).

IY – Comprimento dos dados de origem.

D – Tipo da fonte de dados:

1 – Texto.

2 – Gráfico (estendido).

3 – Lista de itens (\* formato ainda não definido \*).

4 – Atalho do ícone da área de trabalho.

Saída: CY – Estado de erro (0 – Ok, 1 – Memória cheia)

Registradores: AF, BC, DE, HL.

#### **BUFGET (8151H) – Clipboard\_Get**

Descrição: Copia dados da área de transferência para a área de memória de destino. Isso só será feito se a área de transferência contiver dados do tipo solicitado e se os dados dentro da área de transferência não forem maiores do que a área de destino.

Como chamar: `rst 20H : dw 8151H`

Entrada: IX – Endereço de destino

E – Banco de memória RAM de destino (0~15)

IY – Comprimento máximo da área de destino

D – Tipo de dados necessários

Saída: CY = 0 → Dados copiados.

IY – Comprimento dos dados copiados.

CY = 1 → Erro na cópia.

A = 0 → Área de transferência vazia.

1 → Tipo incorreto de dados.

2 → Os dados são muito grandes.

Registradores: AF, BC, DE, HL .

#### **BUFSTA (8154H) – Clipboard\_Status**

Descrição: Lê o estado da área de transferência (tipo de dados e tamanho). O endereço e o banco de RAM também são retornados, mas o aplicativo não deve acessá-los diretamente pois podem ser alterados por outros processos.

Como chamar: `rst 20H : dw 8154H`

Entrada: –

Saída: D – Tipo de dados (0 – Área de transferência está vazia)  
 IY – Comprimento de dados  
 IX – Endereço de dados  
 E – Banco de dados RAM (0~15)

Registradores: –

## 5.2.4 – Registros de Gerenciamento da Área de Trabalho

### 5.2.4.1 – Janela de Registro de Dados

- 00 1B Estado: 0 – Fechado. 1 – Normal.  
 2 – Maximizado. 3 – Minimizado.  
 +128 → Janela centralizada (resetado após a abertura).
- 01 1B bit0: Exibir ícone de aplicativo de 8x8 pontos (na borda superior esquerda).  
 bit1: Janela redimensionável.  
 bit2: Botão de fechamento da tela.  
 bit3: Barra de ferramentas de exibição (abaixo da barra de menu).  
 bit4: Exibir barra de título.  
 bit5: Exibir barra de menu (abaixo da barra de título).  
 bit6: Barra de estado de exibição (na parte inferior da janela).  
 bit7: Usado internamente (definido como 0).
- 02 1B bit0: Ajustar o tamanho X do conteúdo da janela para o tamanho X da janela.  
 bit1: Ajustar o tamanho Y do conteúdo da janela para o tamanho Y da janela.  
 bit2: A janela não será exibida na barra de tarefas.  
 bit3: A janela não é móvel.  
 bit4: A janela é uma janela modal: outras janelas, que apontam para ela (ver byte 51), não podem obter a posição de foco.  
 bit5: Reservado (definido como 0).  
 bit6: Usado internamente (definido como 0).  
 bit7: Usado internamente (definido como 0).
- 03 1B ID de processo do proprietário do Windows.
- 04 2W Posição X/Y, se a janela não estiver maximizada.
- 08 2W Tamanho X/Y, se a janela não estiver maximizada.

- 12 2W Deslocamento X/Y do conteúdo da janela exibida.
- 16 2W Tamanho X/Y total do conteúdo da janela.
- 20 2W Tamanho X/Y mínimo possível da janela.
- 24 2W Tamanho X/Y máximo possível da janela.
- 28 1W Endereço do ícone do aplicativo (objeto gráfico).
- 30 1W Endereço do texto da linha de título (terminado por 0).
- 32 1W Endereço do texto da linha de estado (terminado por 0).
- 34 1W Endereço do “MENU DATA RECORD”.
- 36 1W Endereço do REGISTRO DE DADOS DO GRUPO DE CONTROLE do conteúdo da janela.
- 38 1W Endereço do REGISTRO DE DADOS DO GRUPO DE CONTROLE do conteúdo da barra de ferramentas.
- 40 1W Altura da barra de ferramentas.
- 42 9B Reservado (usado durante o tempo de execução).
- 51 1B “0” ou número da janela modal + 1
- 52 140B Reservado (usado durante o tempo de execução).

#### **5.2.4.2 – Registro de Dados do Grupo de Controle**

- 00 1B Número de controles (deve ser > 0). O plano de fundo também deve ser preenchido.
- 01 1B ID de processo do proprietário do grupo de controle
- 02 1W Endereço do CONTROL DATA RECORDS
- 04 1W Endereço da posição / tamanho REGRA DE CÁLCULO DADOS REGISTRO (“0” significa “não recalcular”).
- 06 2B Não usado, definido como “0”.
- 08 1B Objeto para clicar, quando o usuário pressionar RETURN (1~255; 0 – Não definido). Funciona apenas para o conteúdo da janela, não para a barra de ferramentas).
- 09 1B Objeto para clicar, quando o usuário pressionar ESC (1~255; 0 – Não definido). Funciona apenas para o conteúdo da janela, não para a barra de ferramentas).
- 10 4B Reservado, definido como “0”.
- 14 1B Objeto de foco (1~255, 0 – Sem foco em qualquer objeto). Apenas para o conteúdo da janela.
- 15 1B Não usado, definido como “0”.

### 5.2.4.3 – Registros dos Dados de Controle

- [Número de controles] \* [
- 00 1W (ID de controle) / (valor). Isso será enviado para o aplicativo se o usuário clicar ou modificar o controle
  - 02 1B TIPO DE CONTROLE. Os IDs estão entre 0 e 63 (IDs > 63 serão ignorados). Se o bit6 ou 7 for setado, o objeto ficará oculto. Para mostrá-lo novamente, os dois bits devem ser levados a “00”.
  - 03 1B Número do banco, onde o registro de dados de controle estendido está localizado (0~15). “-1” significa que o controle está no mesmo banco da janela.
  - 04 1W Parâmetro para especificar as propriedades de controle ou, se se uma palavra (dois bytes) não for suficiente, pode ser um Apontador para o registro de dados de controle estendido.
  - 06 2W Posição X/Y do controle (relacionado à borda superior esquerda do conteúdo ou barra de ferramentas). Se a janela estiver usando uma REGRA DE CÁLCULO DE REGISTRO DE DADOS, pode ser escrito “0” aqui.
  - 10 2W Tamanho X/Y do controle (relacionado à borda superior esquerda do conteúdo ou barra de ferramentas). Se a janela estiver usando uma REGRA DE CÁLCULO DE REGISTRO DE DADOS, pode ser escrito “0” aqui.
  - 14 2B Não usado, definido como “0”.
- ]

### 5.2.4.4 – Registro de Dados de Regra de Cálculo

- 00 1W Posição X (parte estática).
- 02 1B Multiplicador de tamanho X da janela.
- 03 1B Divisor de tamanho X da janela.
- 04 1W Posição Y (parte estática).
- 06 1B Multiplicador de tamanho Y da janela.
- 07 1B Divisor de tamanho Y da janela.
- 08 1W Tamanho X (parte estática).
- 10 1B Multiplicador de tamanho X da janela.
- 12 1B Divisor de tamanho X da janela.
- 13 1W Tamanho Y (parte estática).
- 14 1B Multiplicador de tamanho Y da janela.
- 15 1B Divisor de tamanho Y da janela.

O cálculo é:

[posição ou tamanho] – parte\_estática + tamanho\_da\_janela \* multiplicador / divisor

## 5.3 – TIPOS DE CONTROLE

### 5.3.1 – Pintura

#### ID: 00 (PLF) – paint\_area

Descrição: Preenche uma área com uma cor especificada.

Parâmetro: bit0–3: Caneta.

bit7: 0 → 4 cores indexado, 1 → 16 cores.

Registro de dados: –

Tamanho: Não limitado.

#### ID: 01 (PLT) – paint\_text

Descrição: Plota um texto com a fonte padrão do sistema com 4 ou 16 cores para plano de fundo e de frente. O bit5 do byte3 pode ser usado para aumentar o desempenho no MSX. Se o fundo já estiver preenchido com a cor de fundo, este bit pode ser setado para acelerar a saída de texto.

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1W Endereço de texto (terminado por 0)

03 1B bit0–1: Alinhamento:

0 → Esq, 1 → Dir, 2 → Centro.

bit5: Se 1, não preparar fundo (só MSX).

bit7: 0 → 4 cores indexado.

1 → 16 cores.

→ Se modo de 4 cores:

02 1B bit0–1: Papel; bit2–3: Caneta.

bit7: Se 1, preencher fundo.

→ Se modo de 16 cores:

02 1B bit0–3: Papel; bit4–7: Caneta.

03 1B bit6: Se 1, preencher fundo.

Tamanho: A largura não é limitada, mas a altura deve ser igual à altura da fonte atual. Se o texto for maior do que a largura do controle, este só será cortado se a opção "preencher fundo" estiver ativada.

**ID: 02 (PLR) – paint\_frame**

Descrição: Desenha um quadro. Opcionalmente, a área interna pode ser preenchida.

Parâmetro: bit7: 0 → 4 cores indexado, 1 → 16 cores.

bit6: Se 1, preencher a área dentro do quadro.

→ Se modo de 4 cores:

bit4–5: Caneta de área dentro do quadro  
(usado apenas se bit6 = 1).

bit0–1: Caneta da linha superior e esquerda.

bit2–3: Caneta da linha inferior e direita.

→ Se modo de 16 cores:

bit0–3: Caneta de área dentro do quadro  
(usado apenas, se bit6 = 1).

bit8–11: Caneta da linha superior e esquerda.

bit12–15: Caneta da linha inferior e direita.

Registro de dados: –

Tamanho: Igual ou maior que 3x3.

**ID: 03 (PLX) – paint\_frame\_with\_title**

Descrição: Desenha um quadro com um título de texto. As linhas têm uma distância de 3 pontos até a borda do controle. A área interna do quadro não será preenchido.

Parâmetro: Apontador para o registro de dados.

Registro de dados: 00 1W Endereço de texto (terminado por 0)

02 1B bit7: 0 → 4 cores indexado, 1 → 16 cores.

→ Se modo de 4 cores:

02 1B bit0–1: Papel do texto indexado.

bit2–3: Caneta de texto e linha indexadas.

→ se o modo de 16 cores:

02 1B bit0–3: Caneta de linha.

03 1B bit0–3: Papel de texto.

bit4–7: Caneta de texto.

Tamanho: Igual ou maior que 16x16.

**ID: 04 (PLP) – paint\_progress**

Descrição: Traça uma barra de progresso. O segundo byte do parâmetro especifica o progresso em passos de 1/255.

Parâmetro: bit0–1: Cor indexada da linha superior e esquerda.  
 bit2–3: Cor indexada da linha inferior e direita.  
 bit4–5: Cor indexada da área preenchida do quadro.  
 bit6–7: Cor indexada da área vazia dentro do quadro.  
 bit8–15: Progresso (0 – 0%, 255 – 100%)

Registro de dados: –

Tamanho: Igual ou maior que 3x3

#### **ID: 05 (PLA) – paint\_text\_with\_alternative\_font**

Descrição: Plota um texto com uma fonte alternativa. A fonte deve ser colocada na mesma área de 16K e banco de RAM do texto. Para a descrição de como uma fonte é armazenada na memória, consulte “FONTES” abaixo.

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1W Endereço de texto (terminado por 0)

- 02 1B bit0–1: Papel (no modo 4 cores).  
 bit2–3: Caneta (no modo 4 cores).  
 bit0–3: Papel (no modo 16 cores).  
 bit4–7: Caneta (no modo 16 cores).
- 03 1B bit0–1: Alinhamento:  
 0 → Esq, 1 → Dir, 2 → Centro.  
 bit7: 0 → 4 cores, 1 → 16 cores.
- 04 1W Endereço da fonte.

Tamanho: A largura não é limitada, mas a altura deve ser igual à altura da fonte atual; se o texto for maior do que a largura do controle, só será cortado, se a opção “preencher fundo” estiver ativada

#### **ID: 06 (PLC) – paint\_text\_with\_control\_codes**

Descrição: Plota um texto, que pode incluir códigos de controle (0~31). Os seguintes códigos de controle são atualmente aceitos:

- 00 – Fim do texto
- 01 – Define a cor do texto  
 (bit0–3 → Papel, bit4–7 → caneta)
- 02 – Define a fonte. Parâmetro: 1 palavra de 2 bytes  
 (endereço da fonte; deve ser colocado na mesma área de 16K e banco de RAM que o texto. Se o endereço for –1, a fonte padrão será usada).

- 03 – Ativa o modo sublinhado.
- 04 – Desliga o modo sublinhado.
- 05 – Insere espaço adicional entre o caractere atual e o próximo. Parâmetro: 1 byte (quantidade de pontos).
- 06 a 07 – Ainda não suportado (será ignorado).
- 08 a 11 – Pula os próximos bytes ((código-8)\*2+1 bytes).
- 12 a 31 – Insere espaço adicional entre o caractere atual e o próximo (código-8 pontos).

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1W Endereço de texto (terminado por 0).  
                   02 1W Número máximo de bytes (códigos de controle incluídos).  
                   04 1W Endereço da fonte (-1 → Padrão).  
                   06 1B bit0-3: Papel, bit4-7: Caneta.  
                   07 1B bit0: Se 1, sublinhado.

Tamanho: Não limitado.

### 5.3.2 – Gráficos

#### ID: 08 (ICN) – graphic\_simple

Descrição: Plota um gráfico. O controle deve ter o mesmo tamanho do gráfico.

Parâmetro: Endereço gráfico.

Registro de dados: –

Tamanho: Igual ao objeto gráfico.

#### ID: 09 (ICT) – graphic\_with\_text

Descrição: Plota um gráfico com uma ou duas linhas de texto abaixo. É usado para exibir ícones. Quando há um 0 em vez de um endereço de texto, a linha ficará vazia. O gráfico em si deve ter um tamanho de 24x24.

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1W Endereço do gráfico (gráfico padrão) ou do cabeçalho gráfico (gráfico estendido).

02 1W “0” ou endereço de texto para a linha 1 (terminado por 0).

04 1W “0” ou endereço de texto para linha 2 (terminado por 0)

06 1B bit4: 0 → Gráfico Padrão, 1 → Estendido.  
 bit5: 0 → 4 cores indexadas, 1 → 16 cores.  
 bit6: “1”, se opções estendidas  
 bit7: “1”, se o ícone pode ser movido pelo usuário  
 → Se modo de texto de 4 cores:  
 06 1B bit0–1: Papel, bit2–3: Caneta.  
 → Se modo de texto de 16 cores:  
 07 1B bit0–3: Papel, bit4–7: Caneta.  
 → se opções estendidas:  
 08 1B bit0: “1”, se este ícone pode ser marcado.  
 bit1: “1”, se este ícone estiver marcado.

Tamanho: 48x40.

#### **ID: 10 (ICX) – graphic\_extended**

Descrição: Plota um gráfico com um cabeçalho estendido. O controle deve ter o mesmo tamanho do gráfico.

Parâmetro: Endereço do cabeçalho gráfico

Registro de dados: –

Tamanho: Igual ao do objeto gráfico.

#### **5.3.3 – Botões**

##### **ID: 16 (BTN) – button\_simple**

Descrição: Plota um botão com um texto centralizado dentro. A cor indexada 2 é usada para o fundo, a cor indexada 1 para o texto e linhas direita / inferior, e a cor indexada 3 para as linhas esquerda / superior.

Parâmetro: Endereço de texto (terminado por 0).

Registro de dados: –

Tamanho: A largura não é limitada, mas a altura deve ser sempre 12.

##### **ID: 17 (BTC) – button\_check**

Descrição: Plota uma caixa de seleção seguida por uma linha de texto. O byte de estado contém 1, se a caixa estiver marcada, caso contrário, contém 0.

Parâmetro: Apontador para registro de dados

Registro de dados: 00 1W Endereço do byte de estado (este byte pode ser 0 ou 1).

02 1W Endereço do texto (terminado por 0).

04 1B bit0–1: Papel de texto indexado,  
bit2–3: Caneta de texto indexado.

Tamanho: A largura não é limitada, mas a altura deve ser sempre 8.

#### **ID: 18 (BTR) – button\_radio**

Descrição: Plota um botão de opção seguido por uma linha de texto. Se o byte global de estado tiver o mesmo valor que o próprio estado, o botão de opção está marcado. O buffer de coordenadas armazena as coordenadas do botão de opção selecionado de 4 bytes e deve conter “-1, -1, -1, -1” no início. Os botões de opção são conectados entre si e devem apontar para o mesmo byte de estado global e mesmo buffer de coordenadas.

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1W Endereço do byte de estado global.  
02 1W Endereço do texto (terminado por 0).  
04 1B bit0–1: Papel de texto indexado,  
bit2–3: Caneta de texto indexado.  
05 1B Valor do próprio estado.  
06 1W Apontador para um buffer de coordenadas globais de 4 bytes.

Tamanho: A largura não é limitada, mas a altura deve ser sempre 8.

#### **ID: 19 (BTP) – button\_hidden**

Descrição: Define a área na qual o usuário pode clicar. Nada é exibido.

Parâmetro: –

Registro de dados: –

Tamanho: Não limitado.

#### **ID: 20 (BTT) – button\_tabs**

Descrição: Plota uma linha de tabulação. Se a largura for definida como -1, o sistema irá calcular a largura necessária e sobrecreverá -1 com o valor correto.

Parâmetro: Apontador para registro de dados.

Registro de dados: 00 1B Número de guias.

01 1B bit0–1: Papel indexado.  
bit2–3: Caneta indexada.  
bit4–5: Cor indexada das linhas esquerda / superior.  
bit6–7: Cor indexada das linhas direita / inferior.

- 02 1B Guia selecionada.
- 03 1W Endereço de texto do título da guia “1”  
(terminado por 0).
- 05 1B –1 ou largura do título da guia “1”.
- 06 1W Endereço de texto do título da guia “2”  
(terminado por 0).
- 08 1B –1 ou largura do título da guia “2”
- : :
- ?? 1W Endereço de texto do título da guia “n”  
(terminado por 0).
- ?? 1B –1 ou largura do título da guia “n”

Tamanho: A largura não é limitada, mas a altura deve ser sempre 11.

### 5.3.4 – Diversos

## ID: 24 (SLD) – slider simple

**Descrição:** Plota um controle deslizante. Pode ser usado para controlar um valor ou para mover dentro de uma janela ou lista.

Parâmetro: Apontador para registro de dados

Registro de dados: 00 1B bit0: 0 → Alinhamento vert., 1 → horiz.

bit l: 0 → Controle de valor,  
1 → Controle de tempo

01 1B Não usado, definido como 0

02 1W Valor / posição atual

04 1W Valor / posição máxima (0~máximo)

06 1B Aumento de valor, se o usuário clicar no botão para baixo / para a esquerda

07 1B Diminuição do valor, se o usuário clicar no botão para cima / para a direita

**Tamanho:** Dependendo do alinhamento, um componente tem um mínimo de 24 pontos; o outro deve ser sempre 8.

ID: 25 (SUP) = Control collection

**Descrição:** Representa uma coleção de subcontroles. Uma coleção de controle não pode conter outra coleção de controles.

Parâmetro: Apontador para registro de dados

Registro de dados: 00 1W Apontador de para registro de dados do grupo de subcontrole

- 02 1W Largura total da área de coleta do controle.
- 04 1W Altura total da área de coleta do controle.
- 06 1W Deslocamento X atual.
- 08 1W Deslocamento Y atual.
- 10 1B bit0: “1”, se o controle deslizante X deve ser aplicado  
bit1: “1”, se o controle deslizante Y deve ser aplicado

Tamanho: Se os controles deslizantes estiverem ativados, o tamanho deve ser maior que 32x32; não há outras limitações.

### 5.3.5 – Entrada de Texto

#### ID: 32 (TXL) – textinput\_line

Descrição: Plota uma linha de entrada de texto. O usuário pode usar várias funções-chave para editar o texto (veja abaixo), bem como um menu de contexto, que abre com um clique direito do mouse. Se o usuário modificou o texto, o bit7 do byte 12 do registro de dados será setado em 1.

Funções principais de edição:

- SHIFT + LEFT / RIGHT: (des)selecionar partes do texto.
- CTRL + LEFT / RIGHT: pula palavra esquerda / direita.
- CTRL + UP / DOWN: pula para o início / fim da linha.
- CTRL + A: seleciona o texto completo.
- CTRL + C: copiar o texto selecionado.
- CTRL + X: cortar texto selecionado (copiar e excluir).
- CTRL + V: colar o texto copiado.

Parâmetro: Apontador para registro de dados

Registro de dados: 00 1W Endereço de texto. Pode estar em qualquer lugar dentro de um segmento de 16K da área de dados.

- 02 1W Primeiro caractere apresentado.
- 04 1W Posição do cursor.
- 06 1W Número de caracteres selecionados:  
0 → Sem seleção, <0 → o cursor é colocado no final da seleção, >0 → o cursor é colocado no início da seleção.
- 08 1W comprimento do texto atual.

- 10 1W Comprimento máximo de texto possível, sem incluir o 0 do final do texto.
- 12 1B bit0: = “1”, se for senha (todos os caracteres aparecerão como ‘\*’).  
 bit1: Texto é somente leitura.  
 bit2: Usar cores alternativas.  
 bit7: =“1”, se o texto foi modificado.  
 → Se forem usadas cores alternativas:  
 13 1B bit0-3: Papel de texto.  
 bit4-7: Caneta de texto.  
 14 1B bit0-3: Caneta linha superior e esquerda.  
 bit4-7: Caneta da linha inferior e direita.

Tamanho: A largura não é limitada, mas a altura deve ser sempre 12.

### **ID: 33 (TXB) – textinput\_box**

**Descrição:** Plota uma caixa de entrada de texto. O usuário pode usar várias funções para editar o texto (são iguais às do comando anterior ID:32 TXL), bem como um menu de contexto, que abre com um clique direito do mouse. Se o usuário modificou o texto, o bit7 do byte 12 do registro de dados será setado 1. Os dados do registro contém várias variáveis internas usadas que não devem ser modificadas pelo aplicativo. Na inicialização, todas as variáveis internas devem ser definidas como 0.

**Parâmetro:** Apontador para registro de dados.

- Registro de dados:**
- 00 1W Endereço de texto. Pode estar em qualquer lugar dentro de um segmento de 16K da área de dados.
  - 02 1W Não usado.
  - 04 1W Posição do cursor (dentro do texto completo)
  - 06 1W Número de caracteres selecionados:  
 0 → Sem seleção, <0 → o cursor é colocado no final da seleção, >0 → o cursor é colocado no início da seleção.
  - 08 1W Comprimento do texto atual.
  - 10 1W Comprimento máximo de texto possível, sem incluir o 0 do final do texto.

- 12 1B bit1: Texto é somente leitura.  
          bit2: Usar cores alternativas.  
          bit3: Usar fonte alternativa.  
          bit7: = “1”, se o texto foi modificado.
- 13 1B bit0–3: Papel de texto,  
          bit4–7: Caneta de texto.  
          (somente ao usar cores alternativas)
- 14 1B Não usado.
- 15 1W Endereço de fonte de (somente se usar  
          fonte alternativa)
- 17 1B Reservado, definido como 0.
- 18 1W Número atual de linhas.
- 20 1W Largura máxima de uma linha em pontos  
          para quebra de linha (-1 = ilimitado).
- 22 1W Número máximo total de linhas.
- 24 1W Usado internamente. Tamanho X da área  
          visível. (-8 = forçar reformatação).
- 26 1W Usado internamente. Tamanho Y da área  
          visível.
- 28 1W Endereço do registro de dados.
- 30 1W Usado internamente. Tamanho X total.
- 32 1W Usado internamente. Tamanho Y total.
- 34 1W Usado internamente. Deslocamento X da  
          área visível.
- 36 1W Usado internamente. Deslocamento Y da  
          área visível.
- 38 1B bit0: quebra de linha (0 → Na borda da  
          janela, 1 – Na posição máxima de pixel,  
          ver byte 20).  
          bit1: Deve sempre ser “1”.
- 39 1B Largura da tabulação:  
          (1~255; 0 → Sem tabulação).
- 40 4B Buffer de mensagens para comandos de  
          controle adicionais.
- 44 4B Reservado, definido como 0.
- 48 [Número máximo de linhas]W  
          Tabela de tamanho de linhas. Esta tabela  
          contém uma palavra de 2 bytes para cada

linha com o tamanho em caracteres; que também pode incluir códigos de retorno de carro / alimentação de linha (CR + LF) no fim de uma linha. O bit15 será “1” se uma linha contiver os códigos CR + LF.

**Comandos:** O controle da caixa de entrada de texto suporta funções adicionais, que podem ser acessadas enviando códigos de teclado especiais para o controle, enquanto este tiver posição de foco. Os seguintes comandos estão disponíveis:

29 – Obtém a posição do cursor.

Saída: (buffer + 0) → coluna (começando em 0).  
 (buffer + 2) → linha (começando em 0).

30 – O texto foi modificado; este comando força o controle a reformatar e atualizar o texto.

31 – Define a posição do cursor e a seleção do texto; a área visível de entrada de texto será rolada para a nova posição, se necessário

Entrada: (buffer + 0) → Nova posição do cursor.  
 (buffer + 2) → Novo número de caracteres selecionados.

### 5.3.6 – Listas

#### ID: 40 (LST) – list\_title

**Descrição:** Plota a linha de título de uma lista.

**Parâmetro:** Apontador para registro de dados.

**Registro de dados:** 00 1W Número de linhas.

02 1W Primeira linha exibida da lista.

04 1W Apontador para o registro de dados do conteúdo da lista.

06 2B Não usado, definido como 0.

08 1B Número de colunas (1-64).

09 1B bit0-5: Índice da coluna classificada.

bit6: Lista de classificação no início.

bit7: Ordem de classificação:

0 → crescente, 1 → decrescente.

10 1W Apontador para registro de dados das colunas.

12 1W Última linha clicada.

14 1B bit0: =1, se o controle deslizante da lista for exibido  
 bit1: =1, se multisseleções são possíveis.

15 1B Não usado, definido como.

Registro da coluna: [Número de colunas] \* [

00 1B bit0-1: Alinhamento:

(0 → Esq., 1 → Dir., 2 → Centro)

bit2-3: Tipo:

0 → Texto, 1 → gráfico,

2 → Nº 16 bits, 3 → Nº 32 bits

01 1B Não usado, definido como 0.

02 1W Largura desta coluna em pontos.

04 1W Endereço de texto do título (term. por 0)

06 2B Não usado, definido como 0.

]

Registro da lista: [Número de linhas] \* [

1W bit0-12 → Valor desta linha

bit13 → Cor da primeira linha

(1 – Alternativa)

bit14 → =0, é usado internamente para "atualizar seleção"

bit15 → =1, se esta linha estiver marcada

[Número de colunas] \*

1W texto / endereço de dados ou valor para esta célula

]

Tamanho: A largura não é limitada, mas a altura deve ser sempre 10.

#### **ID: 41 (LSI) – list\_content**

Descrição: Plota a própria lista sem o título.

Parâmetro: Apontador para registro de dados.

Registro de dados: (ver ID 40).

Tamanho: A largura deve ser igual ou maior que 11 e a altura deve ser igual ou maior que 16.

#### **ID: 42 (LSP) – list\_dropdown**

Descrição: Traça uma lista suspensa. Apenas uma linha da lista será exibida. Se o usuário clicar neste controle, a lista aparecerá e o usuário poderá escolher uma das entradas.

Parâmetro: Apontador para registro de dados

Registro de dados: (ver ID 40)

12 1W Última linha clicada (Sempre representa a linha selecionada).

14 1B bit0: =1, se o controle deslizante da lista for exibido. Deve ser “1”, se a lista tiver mais de 10 entradas.

bit1: Sempre “0” (sem multisseleção)

Tamanho: A largura deve ser igual ou maior que 11 e a altura deve sempre ser 10.

#### **ID: 43 (LSC) – list\_complete**

Descrição: Plota o título da lista e a própria lista juntos. É a combinação de ID 40 e ID 41.

Parâmetro: Apontador para registro de dados

Registro de dados: (ver ID 40)

Tamanho: A largura deve ser igual ou maior que 11 e a altura deve ser igual ou maior que 26.

#### **5.3.7 – Menus PullDown**

00 1W Número de entradas

[Número de entradas] \* [

00 1W bit0: =1, se a entrada do menu estiver ativa. As entradas desativadas não podem ser clicadas pelo usuário e aparecerão em uma cor diferente.  
 bit1: =1, se houver marca de seleção atrás da entrada.  
 bit2: =1, se a entrada abrir um submenu.  
 bit3: =1, se não houver nenhuma entrada, mas uma linha separadora.

02 1W Endereço de texto (terminado por 0). Se o bit3 da palavra anterior for “1”, deve ser usado “0” aqui.

04 1W Valor, se a entrada for clicável, ou endereço do registro de dados do submenu, se o bit2 da primeira palavra estiver definido.

06 1W Reservado, definido como 0.

]

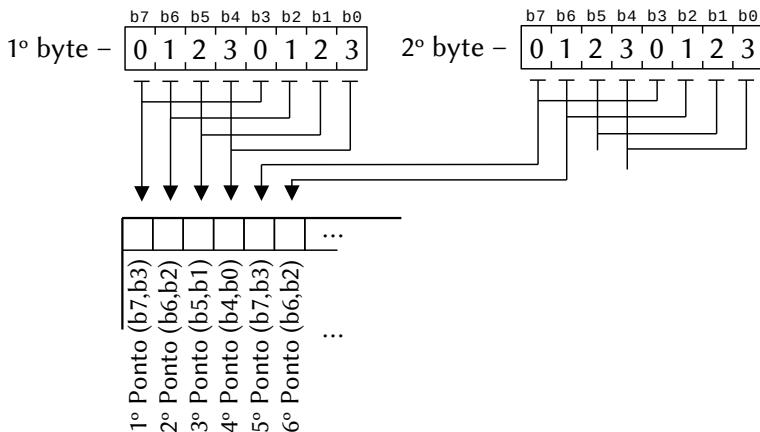
## 5.4 – GRÁFICOS

### 5.4.1 – Gráficos padrão

Um gráfico padrão SymbOS tem 4 cores e pode ter um tamanho máximo de 255x255. Cada objeto gráfico começa com um cabeçalho de 3 bytes:

- 00 1B bit0–6: Largura do gráfico em bytes.
- bit7: Tipo de codificação (0 – CPC, 1 – MSX)
- 01 1B Largura do gráfico em pontos.
- 02 1B Altura do gráfico em pontos.

Logo após o cabeçalho deve estar o tamanho (largura\_em\_bytes \* altura\_em\_pontos). Cada gráfico é armazenado em linhas como os sprites. Os pontos devem ser codificados no formato CPC (Modo 1). Gráficos em um sistema MSX serão automaticamente convertidos para o formato MSX, quando forem exibidos a primeira vez. O bit 7 do byte 0 do cabeçalho contém o formato de codificação atual. Observe que não é permitido armazenar um gráfico original no formato MSX, uma vez que o sistema CPC não é capaz de lidar com esses gráficos! A seguir está uma descrição do formato gráfico do CPC. Cada byte contém 4 pontos:



Apenas aplicativos, que precisam modificar um gráfico após ele ter sido exibido na primeira vez devem se preocupar com o tipo de codificação e com o formato MSX.

### 5.4.2 – Gráficos com cabeçalho estendido

Como a largura de um gráfico é limitada a 255 pontos, não seria possível armazenar uma tela completa (como 320 x 200 no modo CPC 1) em um único bloco. A tela precisa ser dividida em duas partes (por exemplo, 2 x 160 x 200), o que a torna muito difícil escrever rotinas de modificação gráfica.

Os gráficos estendidos não têm essa limitação e também permitem mais de 4 cores. Eles só podem ser usados para o ID de controle 10, "graphic\_extended". Um gráfico pode ser armazenado em uma única peça com largura de até 1020 pontos. O controle "graphic\_extended" então é capaz de exibir uma parte de um grande armazenamento linear gráfico.

A estrutura do cabeçalho estendido é a seguinte:

- 00 1B Largura do gráfico completo em bytes (deve ser um valor par!)
- 01 1B Largura da área gráfica, que deve ser exibida, em pixel
- 02 1B Altura da área gráfica, que deve ser exibida, em pixel
- 03 1W Endereço dos dados gráficos, incluindo o deslocamento da área
- 05 1W Endereço do byte de informação de codificação (este byte deve ser SEMPRE colocado imediatamente antes dos dados gráficos).
- 07 1W Tamanho do gráfico completo
- ?? 1B Informação de codificação
  - bit0–1: Codificação de cor (0 → CPC, 1 → MSX)
  - bit2–3: Profundidade de cor (0 → 4 cores, 1 → 16 cores)
  - Apenas os dois valores iniciais a seguir são permitidos:
    - 0 → 4 cores, formato CPC; um sistema MSX irá converter o gráfico para o formato MSX, quando for mostrado pela primeira vez.
    - 5 → 16 cores, formato MSX; um sistema CPC e PCW irá processar o gráfico completo para 4 cores (formato CPC), quando for exibido pela primeira vez
- ?+1 x dados gráficos

O gráfico em si ("graphic\_data") é armazenado em um único bloco na memória (sem cabeçalho). Então temos dois cabeçalhos ("graphic\_

header\_for\_area\_1" e "graphic\_header\_for\_area\_2") que estão apon-tando para duas áreas diferentes do gráfico completo. Como pode ser visto, ainda são necessários dois controles para exibir o gráfico, mas os dados em si não precisam ser divididos.

### 5.4.3 – Fontes

Uma fonte define a aparência dos caracteres usados para imprimir textos no SymbOS. Uma fonte começa com um cabeçalho simples de 2 bytes, seguido pela máscara de bits dos caracteres. Seu formato é o seguinte:

- 00 1B Altura de cada caractere em pontos (1~15, padrão: 8).
- 00 1B Primeiro caractere da fonte (0~255).
- 00 1B Largura do primeiro caractere em pontos.
- 01 1B Máscara de bits da 1<sup>a</sup> linha do primeiro caractere.
- 02 1B Máscara de bits da 2<sup>a</sup> linha do primeiro caractere.
- ⋮
- 15 1B Máscara de bits da 15<sup>a</sup> linha do primeiro caractere.
- 16 1B Largura do segundo caractere em pontos.
- 17 1B Máscara de bits da 1<sup>a</sup> linha do segundo caractere.
- ⋮

## 5.5 – GERENCIADOR DE SISTEMA

Os comandos do gerenciador de sistema são acionados por meio de uma mensagem, que deve ser enviada com RST 10H (MSG SND) para o processo do gerenciador do sistema. O processo do gerenciador de sistema sempre tem o ID 3.

### 5.5.1 – Gerenciamento de Aplicativos

#### ID: 016 (MSC\_SYS\_PRGRUN) – Program\_Run\_Command

Descrição: Carrega e inicia um aplicativo ou abre um documento com um tipo conhecido carregando primeiro o aplicativo associado. Se o Bit 7 de P3 for “0”, o sistema abrirá uma caixa de mensagem caso ocorra algum erro durante o processo de carregamento.

Biblioteca: SySystem\_PRGRUN

Mensagem: 00 1B 016

01 1W Endereço do caminho e nome do arquivo.

03 1B bit0–3: Caminho do arquivo e número do banco RAM (0~15)

bit7: Sinalizador, se a mensagem de erro do sistema deve ser suprimida.

Resposta: Ver MSR\_SYS\_PRGRUN.

#### **ID: 144 (MSR\_SYS\_PRGRUN) – Program\_Run\_Response**

Descrição: O gerenciador do sistema envia esta mensagem após carregar um aplicativo ou depois de abrir um documento associado. Se a operação for bem-sucedida, o ID do aplicativo e o ID do processo serão colocados em P8 e P9. Se falhar, P8 conterá o código de erro do gerenciador de arquivos.

Mensagem: 00 1B 144.

01 1B Estado:

0 – OK.

1 – Arquivo não existe.

2 – O arquivo não é um executável e seu tipo não é associado a um aplicativo.

3 – Erro ao carregar (P8 contém código de erro).

4 – Memória cheia.

→ Se o estado for 0:

08 1B ID do aplicativo.

09 1B ID do processo (o processo principal de aplicativos).

→ Se o estado for 3:

08 1B Código de erro do gerenciador de arquivos.

#### **ID: 017 (MSC\_SYS\_PRGEND) – Program\_End\_Command**

Descrição: Para um aplicativo e libera todos os recursos usados pelo mesmo. Este comando não libera memória ou interrompe processos, temporizadores ou janelas fechadas que não estão registradas para o aplicativo. Esses recursos devem, primeiro, ser liberados pelo próprio aplicativo.

Biblioteca: SySystem\_PRGEND.

Mensagem: 00 1B 017

01 1B ID do aplicativo

Resposta: O gerenciador do sistema não envia mensagem de resposta.

**ID: 020 (MSC\_SYS\_PRGSTA)** – Program\_Run\_Dialogue\_Command

Descrição: Abre a caixa de diálogo "Executar". O usuário pode então selecionar uma aplicação ou documento.

Mensagem: 00 1B 020.

Resposta: O gerenciador do sistema não envia mensagem de resposta.

**ID: 024 (MSC\_SYS\_PRGSET)** – Program\_Run\_ControlPanel\_Command

Descrição: Inicia o aplicativo do painel de controle ou um de seus dois submódulos.

Mensagem: 00 1B 024.

01 1B Submódulo do painel de controle:

0 – Janela principal.

1 – Configurações de exibição.

2 – Configurações de hora e data.

Resposta: O gerenciador do sistema não envia mensagem de resposta.

**ID: 025 (MSC\_SYS\_PRGTSK)** – Program\_Run\_TaskManager\_Command

Descrição: Inicia o aplicativo gerenciador de tarefas.

Mensagem: 00 1B 025.

Resposta: O gerenciador do sistema não envia mensagem de resposta.

**ID: 030 (MSC\_SYS\_PRGSRV)** – Program\_SharedService\_Command

Descrição: Encontra, inicia e libera serviços compartilhados.

Mensagem: 00 1B 030.

04 1B Tipo de comando

0 – Aplicativo de pesquisa ou serviço compartilhado.

1 – Pesquisar, iniciar e usar serviço compartilhado.

2 – Liberar serviço compartilhado.

→ Se P4 for 0 ou 1:

01 1W Endereço da string de 12 bytes do ID do aplicativo.

→ Se P4 for 0 ou 1:

03 1B Banco RAM (0~15) da string de 12 bytes de ID do aplicativo.

→ Se P4 for 2:

03 1B ID do aplicativo de serviço compartilhado.

Resposta: Consulte MSR\_SYS\_PRGSRV.

**ID: 158 (MSR\_SYS\_PRGSRV) – Program\_SharedService\_Response**

**Descrição:** O tipo de comando 0 (“pesquisar”) retornará 5 (não encontrado) ou 0 (OK). No último caso, você encontrará o aplicativo e o ID do processo em P8 e P9. O tipo de comando 1 (“pesquisar, iniciar e usar”) retornará 0 (OK) se os serviços compartilhados foram encontrados ou carregados com sucesso. Caso contrário, retornará um código de erro de carregamento de 1, 2, 3 ou 4, que é idêntico ao de MSR\_SYS\_PRGRUN. O tipo de comando 2 (“liberação”) sempre retornará 0 (OK).

Mensagem: 00 1B 158.

01 1B Estado:

0 – OK.

5 – Aplicativo ou serviço compartilhado não encontrado (só ocorre no tipo de comando 0).

1-4 – Erro ao iniciar o serviço compartilhado; mesmos códigos de MSR\_SYS\_PRGRUN.

→ Se o tipo de comando for 0 ou 1 e o estado for 0:

08 1B ID do aplicativo de serviço compartilhado.

09 1B ID do processo (processo principal de aplicativos).

→ Se o estado do resultado for 3:

08 1B Código de erro do gerenciador de arquivos.

### 5.5.2 – Comandos do Gerenciador de Sistema

O gerenciador do sistema não enviará mensagens de resposta após o processamento dos comandos abaixo.

**ID: 018 (MSC\_SYS\_SYSWNX) –**

– System\_Dialogue\_NextWindow\_Command

**Descrição:** Abre a caixa de diálogo para alterar a janela atual. Nas próximas a janela será pré-selecionada. ESTE COMANDO AINDA NÃO FOI IMPLEMENTADO.

Mensagem: 00 1B 018.

**ID: 019 (MSC\_SYS\_SYSWPR) –**

– System\_Dialogue\_PreviousWindow\_Command

**Descrição:** Abre a caixa de diálogo para alterar a janela atual. A janela anterior é pré-selecionada. ESTE COMANDO AINDA NÃO FOI IMPLEMENTADO.

Mensagem: 00 1B 019.

**ID: 021 (MSC\_SYS\_SYSSEC) –**

System\_Dialogue\_SystemSecurity\_Command

Descrição: Abre a caixa de diálogo "Segurança SymbOS".

Mensagem: 00 1B 021.

**ID: 022 (MSC\_SYS\_SYSQIT) – System\_Dialogue\_ShutDown\_Command**

Descrição: Abre a caixa de diálogo “desligar”.

Mensagem: 00 1B 022.

**ID: 023 (MSC\_SYS\_SYSOFF) – System\_ShutDown\_Command**

Descrição: Reinicia o computador.

Mensagem: 00 1B 023.

**ID: 028 (MSC\_SYS\_SYSCFG) – System\_Configuration\_Command**

Descrição: Carrega ou salva a configuração ou reinicializa o fundo da área de trabalho ou a proteção de tela.

Mensagem: 00 1B 028.

01 1B Tipo de ação

0 → Recarregar configuração.

1 → Salvar configuração atual.

2 → Recarregar ou reiniciar a imagem de fundo da área de trabalho.

3 → Recarregar ou reiniciar a proteção de tela.

### 5.5.3 – Serviços de Diálogo

**ID: 029 (MSC\_SYS\_SYSWRN) – Dialogue\_Infobox\_Command**

Descrição: Abre uma caixa de informação, aviso ou confirmação e exibe três linhas de texto e até três botões de clique. Se o bit 7 de P4 for “1”, pode ser especificado um símbolo próprio, que será mostrado à esquerda do texto. Se este bit for “0”, será exibido um “!” (símbolo de advertência). Se o bit 6 de P4 for “1”, a janela será aberta como um modal, e você receberá uma mensagem com o número da janela (consulte MSR\_SYS\_SYSWRN). Os dados de conteúdo devem sempre ser colocados na área de transferência de RAM (C000H~FFFFH).

Biblioteca: SySystem\_SYSWRN.

Mensagem: 00 1B 029.

- 01 1W Endereço de dados de conteúdo  
 03 1B Banco de dados de conteúdo (0~15).  
 04 1B bit0~2: Número de botões (1~3).
  - 1 → Botão “OK”.
  - 2 → Botões “Sim”, “Não”.
  - 3 → Botões “Sim”, “Não”, “Cancelar”.
 bit3~5: Texto do título
  - 0 → Padrão (bit7 = 0 → “Erro!”)
  - 1 → “Informação”).
  - 1 → “Erro!”.
  - 2 → “Informação”.
  - 3 → “Aviso”.
  - 4 → “Confirmação”.
 bit6: =1, se a janela deve ser janela modal.  
 bit7: Tipo de caixa:
  - 0 → Padrão (símbolo de aviso [!]).
  - 1 → Informação (o próprio símbolo será usado).

Dados de conteúdo: 00 1W Endereço da 1<sup>a</sup> linha de texto  
 02 1W 4 \* [caneta da 1<sup>a</sup> linha de texto] + 2  
 04 1W Endereço da 2<sup>a</sup> linha de texto  
 06 1W 4 \* [caneta da 2<sup>a</sup> linha de texto] + 2  
 08 1W Endereço da 3<sup>a</sup> linha de texto  
 10 1W 4 \* [caneta da 3<sup>a</sup> linha de texto] + 2  
 → Se o bit 7 de P4 bit for 1:  
 12 1W Endereço de do símbolo (formato gráfico do SymbOS: 24x24px 4col )

Resposta: Consulte MSR\_SYS\_SYSWRN

#### **ID: 157 (MSR\_SYS\_SYSWRN) – Dialogue\_Infobox\_Response**

Descrição: O gerenciador do sistema envia de volta esta mensagem para o aplicativo, quando uma infobox deve ser aberta ou se o usuário clicou em um dos botões.

Mensagem: 00 1B 157.  
 01 1B Tipo de mensagem:
 

- 0 → A infobox está sendo usada atualmente por outro aplicativo. Só pode ser aberto uma por vez, se não for uma mensagem de informação (um botão, não uma janela modal). A

outra infobox deve ser fechada antes que possa ser aberta novamente pelo aplicativo.

- 1 → A infobox foi aberta com sucesso como janela modal. Esta mensagem não será enviada para infoboxes não modais.
- 2 → O usuário clicou em “OK”.
- 3 → O usuário clicou em “Sim”.
- 4 → O usuário clicou em “Não”.
- 5 → O usuário clicou em “Cancelar” ou no botão “Fechar”.

→ Se P1 for 1:

- 02 1B Número da janela da caixa de informações + 1. O aplicativo deve armazenar este número como o ID da janela modal da sua própria janela, para que a infobox seja tratada como janela modal da janela do aplicativo. Quando aberta, a janela do aplicativo não obtém posição de foco.

#### **ID: 031 (MSC\_SYS\_SELOPN) – Dialogue\_FileSelector\_Command**

**Descrição:** Abre a caixa de diálogo de seleção de arquivo. O usuário pode então se mover através da estrutura de diretório, mudar a unidade e pesquisar/selecionar um arquivo ou diretório para abrir ou salvar. A máscara de arquivo/caminho/string de nome (260 bytes) deve ser sempre colocado na área de transferência (C000H~FFFFH).

**Biblioteca:** SySystem\_SELOPN

**Mensagem:** 00 1B 031.

- 06 1B bit0~3: Máscara de arquivo, caminho e nome do banco de RAM (0~15).  
 bit6 = 1, se “abrir” (0) ou “salvar” (1).  
 bit7 = 1, se seleção de arquivo (0) ou diretório (1).

07 1B Filtro de Atributo:

- bit0 = 1 → Não mostrar arquivos somente leitura.
- bit1 = 1 → Não mostra arquivos ocultos.
- bit2 = 1 → Não mostra arquivos de sistema.
- bit3 = 1 → Não mostra entradas de ID de volume.
- bit4 = 1 → Não mostra diretórios.
- bit5 = 1 → Não mostrar arquivos de arquivo.

- 08 1W Máscara de arquivo, caminho e endereço de nome (C000H- FFFFH).

- 00 3B Filtro de extensão de arquivo (por exemplo, “\*”)  
 03 1B 0.  
 04 256B Caminho e nome de arquivo.  
 10 1W Número máximo de entradas de diretório.  
 12 1W Tamanho máximo do buffer de dados do diretório.

Resposta: Ver MSR\_SYS\_SELOPN.

**ID: 159 (MSR\_SYS\_SELOPN) – Dialogue\_FileSelector\_Response**

Descrição: O gerenciador do sistema envia esta mensagem para o aplicativo quando uma caixa de diálogo de seleção de arquivo deve ser aberta. Se a abertura for bem-sucedida, o aplicativo receberá primeiro uma mensagem tipo “-1” e, após o usuário escolher o arquivo ou abortar, uma mensagem do tipo 0 ou 1. Se a abertura falhar, o aplicativo receberá diretamente mensagens do tipo 2, 3 ou 4.

Mensagem: 00 1B 159.

01 1B Tipo de mensagem

0 → O usuário escolheu um arquivo ou diretório e fechou o diálogo com "OK". O nome e o caminho completo do arquivo e pode ser encontrado no buffer de caminho de arquivo da aplicação.

1 → O usuário abortou a seleção do arquivo. O conteúdo do buffer do caminho de arquivo do aplicativo permanece inalterado.

2 → A caixa de diálogo de seleção de arquivo está sendo usada por outro aplicativo. O usuário deve fechar a caixa de diálogo antes de poder ser aberta novamente pela aplicação.

3 → Memória cheia. Não havia memória suficiente disponível para o buffer de diretório e/ou lista de estrutura de dados.

4 → Sem janela disponível. O gerenciador de área de trabalho não conseguiu abrir uma nova janela para o diálogo, pois o número máximo de janelas (32) já foi alcançado.

-1 → O diálogo foi aberto com sucesso e o usuário está fazendo sua seleção de arquivo agora.

→ Se P1 for -1:

02 1B Número da janela da caixa de informações + 1. O aplicativo deve armazenar este número como o ID da janela modal da sua própria janela, para que a infobox seja tratada como janela modal da janela do aplicativo. Quando aberta, a janela do aplicativo não obtém posição de foco.

#### **5.5.4 – Funções do Gerenciador de Sistema**

As funções do gerenciador do sistema devem ser chamadas com RST 28H (BNKFCL).

##### **SYSINF (8103H) – System\_Information**

**Descrição:** Esta função é usada principalmente pelo gerenciador de tarefas e pelo aplicativo do painel de controle. Os tipos de solicitação 0-2 ainda não estão documentados.

**Como chamar:** Id hl, 8103H : rst 28H

**Entrada:** E – Tipo de solicitação:

0 → Obter informações gerais.

1 → Obter informações do aplicativo.

2 → Obter informações da tarefa.

3 → Carregar configuração do dispositivo de armazenamento de massa ( $8 * 16$  bytes). Para uma descrição da estrutura de dados, consulte “Dados de configuração / Área central / armazenamento de massa”.

IX – Endereço de destino (deve ser colocado dentro da área de transferência da RAM).

**Saída:** –

4 → Salvar configuração do dispositivo de armazenamento de massa ( $8 * 16$  bytes). Para uma descrição da estrutura de dados, consulte “Dados de configuração / Área central / armazenamento de massa”.

IX – Endereço de origem (deve ser colocado dentro da área de transferência da RAM).

**Saída:** –

5 → Carregar uma parte da configuração da área central para a memória de aplicativos. Para uma descrição da estrutura de dados, consulte “Dados de Configuração / Área Central”.

D – Número de bytes

IX – Endereço de destino (área de transferência)

IY – Deslocamento de origem (a partir do byte 163 [= caminho do sistema] na área central).

Saída: –

6 → Salvar uma parte da configuração da área central a partir da memória dos aplicativos.

D – Número de bytes

IX – Endereço de origem (área de transferência de memória RAM)

IY – Deslocamento de destino (a partir do byte 163 [= caminho do sistema] na área central).

7 → Obter endereço de memória de configuração.

Saída: DE – Endereço da área central (incluindo cabeçalho de 6 bytes; Banco de RAM “0”).

IX – Endereço da área de dados.

IYI – Banco de memória RAM da área de dados (0~8).

8 → Obter endereço de memória de string de fonte e versão. O endereço da fonte é sempre colocado no banco de RAM “0” e a string da versão é colocada no mesmo banco RAM da área de dados.

Saída: DE – Endereço da fonte (Banco RAM 0).

IX – Comprimento da fonte (= cabeçalho de 2 bytes + 98 \* 16 bytes de bitmaps de caracteres).

IY – Endereço da informação da versão, que tem 32 bytes:

00 1B Versão principal.

01 1B Versão menor.

02 30B String de versão (terminada em 0).

Registradores: AF, BC, HL.

## 5.6 – GERENCIADOR DE ARQUIVOS E DISPOSITIVOS

O processo do gerenciador de sistema sempre tem o ID 3. Observe que no SymbOS todos os textos devem terminar com 0 byte.

### 5.6.1 – Mensagens do Gerenciador de Arquivos

#### **ID: 026 (MSC\_SYS\_SYSFIL)** – System\_Filemanager\_Command

Descrição: Um aplicativo deve enviar esta mensagem ao gerenciador do sistema (ID de processo 3) para chamar uma função gerenciador de arquivos.

Mensagem: 00 1B 026.

- 01 1B ID da função de gerenciador de arquivos.
- 02 1W Entrada para AF.
- 04 1W Entrada para BC.
- 06 1W Entrada para DE.
- 08 1W Entrada para HL.
- 10 1W Entrada para IX.
- 12 1W Entrada para IY.

#### **ID: 154 (MSR\_SYS\_SYSFIL)** – System\_Filemanager\_Response

Descrição: O gerenciador do sistema envia esta mensagem de volta para o aplicativo após a função de gerenciador de arquivos ter sido chamada.

Mensagem: 00 1B 154.

- 01 1B ID da função de gerenciador de arquivos.
- 02 1W Saída para AF.
- 04 1W Saída para BC.
- 06 1W Saída para DE.
- 08 1W Saída para HL.
- 10 1W Saída para IX.
- 12 1W Saída para IY.

### 5.6.2 – Códigos de Erro

- 000 – Dispositivo não existe.
- 001 – OK.
- 002 – Dispositivo não inicializado.
- 003 – A mídia está danificada.
- 004 – Partição não existe.

- 005 – Mídia ou partição não suportada.
- 006 – Erro durante a leitura / gravação do setor.
- 007 – Erro ao posicionar.
- 008 – Abortar durante o acesso ao volume.
- 009 – Erro de volume desconhecido.
- 010 – Nenhum manipulador de arquivos gratuito.
- 011 – Dispositivo não existe.
- 012 – Caminho não existe.
- 013 – Arquivo não existe.
- 014 – Acesso proibido.
- 015 – Caminho ou nome de arquivo inválido.
- 016 – Manipulador de arquivo não existe.
- 017 – Slot do dispositivo já ocupado.
- 018 – Erro na organização do arquivo.
- 019 – Nome de destino inválido.
- 020 – Arquivo / caminho já existe.
- 021 – Código de subcomando errado.
- 022 – Atributo errado.
- 023 – Diretório cheio.
- 024 – Mídia completa.
- 025 – A mídia está protegida contra gravação.
- 026 – O dispositivo não está pronto.
- 027 – O diretório não está vazio.
- 028 – Dispositivo de destino inválido.
- 029 – Não compatível com o sistema de arquivos.
- 030 – Dispositivo não compatível.
- 031 – O arquivo é somente leitura.
- 032 – Canal do dispositivo não disponível.
- 033 – O destino não é um diretório.
- 034 – O destino não é um arquivo.
- 255 – Erro Indefinido.

### 5.6.3 – Funções do Dispositivo de Armazenamento de Massa

**ID: 000 (STOINI) – Storage\_Init**

Descrição: Remove todos os dispositivos de armazenamento de massa.

Entrada: –

Saída: –

Registradores: BC, DE, HL.

**ID: 001 (STONew) – Storage\_New**

Descrição: Adiciona um novo dispositivo de armazenamento de massa.

Entrada: A – Dispositivo (0~7)  
 C – Subdrive  
 DE – Endereço do driver  
 L – Sinalizador de mídia removível (1 → Removível)  
 B – Letra da unidade (A~Z)  
 IX – Nome do dispositivo (11 caracteres)

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro)

Registradores: AF, BC, DE, HL, IX, IY

**ID: 002 (STORLd) – Storage\_Reload**

Descrição: Recarrega um dispositivo de armazenamento de massa, se seu estado de “mídia removível” está ativado. O formato e o tipo de sistema de arquivos serão carregados novamente.

Entrada: A – Dispositivo (0~7).

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

**ID: 003 (STODEl) – Storage\_Delete**

Descrição: Remove um dispositivo de armazenamento de massa existente.

Entrada: A – Dispositivo (0~7).

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL.

**ID: 004 (STOINP) – Storage\_ReadSector**

Descrição: Lê um setor de um dispositivo de armazenamento de massa (sem banco de memória).

Entrada: A – Dispositivo (0~7).

IY, IX – Número do primeiro setor.

B – Número de setores.

DE – Endereço de destino.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

**ID: 005 (STOOUT) – Storage\_WriteSector**

Descrição: Grava um setor em um dispositivo de armazenamento de massa (sem banco de memória).

Entrada: A – Dispositivo (0~7).  
 IY, IX – Número do primeiro setor.  
 B – Número de setores.  
 DE – Endereço de origem.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

**ID: 006 (STOACT) – Storage\_Activate**

Descrição: Carrega o formato e o tipo de sistema de arquivos de um armazenamento de massa dispositivo.

Entrada: A – Dispositivo (0~7).  
 Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).  
 Registradores: AF, BC, DE, HL, IX, IY.

**ID: 007 (STOINF) – Storage\_Information**

Descrição: Retorna informações sobre um dispositivo de armazenamento de massa.

Entrada: A – Dispositivo (0~7).  
 Saída:  
 A – Tipo:  
 00 → Dispositivo não existe.  
 01 → Dispositivo está pronto.  
 02 → Dispositivo não inicializado.  
 03 → Dispositivo está corrompido.  
 B – Mídia:  
 01 → Disquete lado único (Amsdos, PCW).  
 02 → Disco flexível lado duplo (FAT12).  
 08 → RAMDISK (\* ainda não suportado \*).  
 16 → HD IDE ou cartão CF (FAT12, 16 ou 32).  
 C – Sistema de arquivos:  
 01 → Dados Amsdos.  
 02 → Sistema Amsdos.  
 03 → PCW 180K.  
 16 → FAT12.  
 17 → FAT16.  
 18 → FAT32.

D – Setores por cluster.  
 IY, IX – Número total de clusters.

Registradores: E, HL

#### **ID: 008 (STOTRN) – Storage\_DataTransfer**

Descrição: Lê ou grava vários setores (512 bytes) de / para o dispositivo de armazenamento de massa. O primeiro setor da partição do dispositivo é o “0”.

Entrada: A – Dispositivo (0~7).  
 IY, IX – Número do primeiro setor.  
 B – Número de setores.  
 C – Direção (0 – leitura, 1 – gravação).  
 HL – Endereço de origem / destino.  
 E – Banco de memória RAM de origem / destino (0~15).  
 Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).  
 Registradores: AF, BC, DE, HL, IX, IY.

#### **5.6.4 – Funções de Gerenciamento de Arquivos**

##### **ID: 016 (FILINI) – File\_Init**

Descrição: Inicializa todo o gerenciador de arquivos.

Entrada: –

Saída: –

Registradores: AF, BC, DE, HL.

##### **ID: 017 (FILNEW) – File\_New**

Descrição: Cria um novo arquivo e o abre para acesso de leitura / gravação. Se o arquivo já existia, ele será esvaziado primeiro. A operação será abortada, se o arquivo existente for somente leitura ou um subdiretório. Para obter informações adicionais, consulte 018 (FIOPN).

Biblioteca: SyFile\_FILNEW.

Entrada: IXh – Caminho do arquivo e nome do Banco RAM (0~15).

HL – Caminho do arquivo e endereço do nome.

A – Atributos:

bit0 = 1 → Somente leitura.

bit1 = 1 → Oculto.

bit2 = 1 → Sistema.

bit5 = 1 → Arquivo.

Saída: A – ID do manipulador de arquivos.  
CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).  
Registradores: F, BC, DE, HL, IX, IY.

#### **ID: 018 (FILOPN) – File\_Open**

Descrição: Abre um arquivo existente para leitura / gravação. Podem ser abertos até 7 arquivos diferentes ao mesmo tempo.  
Biblioteca: SyFile\_FILOPN.  
Entrada: IXh – Caminho do arquivo e nome do Banco RAM (0~15).  
HL – Caminho do arquivo e endereço do nome.  
Saída: A – ID do manipulador de arquivos.  
CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).  
Registradores: F, BC, DE, HL, IX, IY.

#### **ID: 019 (FILCLO) – File\_Close**

Descrição: Fecha um arquivo aberto. Se houver dados não escritos no buffer, eles serão gravados no disco imediatamente. Este comando fecha um arquivo em qualquer caso, mesmo que tenha ocorrido erro.  
Biblioteca: SyFile\_FILCLO.  
Entrada: A – ID do manipulador de arquivos.  
Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).  
Registradores: AF, BC, DE, HL, IX, IY.

#### **ID: 020 (FILINP) – File\_Input**

Descrição: Lê uma quantidade especificada de bytes de um arquivo aberto. Em seguida, o apontador do arquivo será movido para o byte seguinte ao último byte lido. Assim, é possível ler vários blocos com tamanhos diferentes de um arquivo aberto.  
Biblioteca: SyFile\_FILINP.  
Entrada: A – ID do manipulador de arquivos.  
HL – Endereço de destino.  
E – Banco de memória RAM de destino (0~15).  
BC – Número de bytes a ler.  
Saída BC – Número de bytes efetivamente lidos.  
Z = 1 → Todos os bytes solicitados foram lidos.  
0 → O fim do arquivo foi alcançado, e menos bytes que os solicitados foram lidos (verif. BC).

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, DE, HL, IX, IY.

### **ID: 021 (FILOUT) – File\_Output**

Descrição: Grava uma quantidade especificada de bytes em um arquivo aberto. Em seguida, o apontador do arquivo será movido para o byte seguinte ao último byte escrito. Se o apontador do arquivo estiver em algum lugar no meio do arquivo antes desta operação, os dados neste local serão sobreescritos.

Biblioteca: SyFile\_FILOUT.

Entrada: A – ID do manipulador de arquivos.

HL – Endereço de origem.

E – Banco de RAM de origem (0~15).

BC – Número de bytes a escrever.

Saída: BC – Número de bytes efetivamente escritos.

A = 0 → Todos os bytes foram gravados.

1 → O dispositivo está cheio, e menos bytes foram escritos (verifique BC).

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro)

Registradores: AF, DE, HL, IX, IY.

### **ID: 022 (FILPOI) – File\_Pointer**

Descrição: Move o apontador do arquivo para outra posição. O valor relativo é especificado em IY e IX. IY é o valor mais significativo e IX é o menos (deslocamento = 65 536 \* IY + IX).

Biblioteca: SyFile\_FILPOI

Entrada: A – ID do manipulador de arquivos.

IY, IX – Deslocamento.

C – Ponto de referência:

0 → Início do arquivo (deslocamento sem sinal).

1 → Posição atual do apontador (o deslocamento é sinalizado).

2 → Fim do arquivo (o deslocamento é sinalizado).

Saída: IY, IX – Nova posição absoluta do apontador.

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL.

**ID: 023 (FILF2T) – File\_Decode\_Timestamp**

Descrição: Decodifica o carimbo de data / hora do arquivo, que é usado pelo sistema de arquivos.

Biblioteca: SyFile\_FILF2T.

Entrada: BC – Código de tempo:  
 bit 0–4 → Segundo / 2.  
 bit 5–10 → Minuto.  
 bit 11–15 → Hora.

DE – Código de data:  
 bit 0–4 → Dia (começando em 1).  
 bit 5–8 → Mês (começando em 1).  
 bit 9–15 → Ano – 1980.

Saída: A – Segundo.  
 B – Minuto.  
 C – Hora.  
 D – Dia (começando em 1).  
 E – Mês (começando em 1).  
 HL – Ano.

Registradores: F.

**ID: 024 (FILT2F) – File\_Encode\_Timestamp**

Descrição: Codifica o carimbo de data / hora do arquivo, que é usado pelo sistema de arquivos.

Biblioteca: SyFile\_FILT2F

Entrada: A – Segundo.  
 B – Minuto.  
 C – Hora.  
 D – Dia (começando em 1).  
 E – Mês (começando em 1).  
 HL – Ano.

Saída: BC – Código de hora (ver FILF2T).  
 DE – Código de data (ver FILF2T).

Registradores: AF, HL, IX, IY.

**ID: 025 (FILLIN) – File\_LineInput**

Descrição: Lê uma linha de texto de um arquivo aberto. Uma linha de texto é terminada por um único byte 13, um único byte 10, uma combinação de 13+10, uma combinação de 10+13 ou por um único byte 26 (código de “fim de arquivo”).

Biblioteca: SyFile\_FILLIN.

Entrada: A – ID do manipulador de arquivos.

HL – Endereço do buffer de destino (o tamanho deve ser de 255 bytes).

E – Banco de memória RAM do buffer de destino (0~15).

Saída: C – Número de bytes lidos (0-254; sem terminador).

B – =1, se o final da linha / arquivo for alcançado; ou 0, caso contrário.

Z = 0 → 1 ou mais bytes foram carregados.

1 → Fim de arquivo atingido, nada foi carregado.

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, DE, HL, IX, IY.

### **5.6.5 – Funções de Gerenciamento de Diretório**

As funções de gerenciamento de diretório permitem mostrar e editar o conteúdo de um diretório. A barra normal ("/", Unix, Linux) pode ser usada no lugar da barra invertida ("\", Microsoft).

#### **ID: 032 (DIRDEV) – Directory\_Device**

Descrição: Seleciona a unidade atual. Como os aplicativos são executados em um ambiente multitarefa, outros aplicativos podem alterar a unidade novamente.

Biblioteca: SyFile\_DIRDEV

Entrada: A – Letra do drive ("A"~"Z").

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

#### **ID: 033 (DIRPTH) – Directory\_Path**

Descrição: Seleciona o caminho para a unidade atual. Como os aplicativos são executados em um ambiente multitarefa, outros aplicativos podem alterar o caminho novamente.

Biblioteca: SyFile\_DIRPTH.

Entrada: IXh – Banco de RAM do caminho do arquivo (0~15).

HL – Endereço do caminho do arquivo.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

**ID: 034 (DIRPRS) – Directory\_Property\_Set**

**Descrição:** Altera uma propriedade de um arquivo ou diretório.  
Podem ser alterados os atributos, a hora de criação e a hora de modificação.

**Biblioteca:** SyFile\_DIRPRS

**Entrada:** IXh – Caminho do arquivo e nome do Banco RAM (0~15)

HL – Caminho do arquivo e endereço do nome

A – Tipo de propriedade:

0 – Atributo.

C – Código de atributo:

bit0=1 → Somente leitura.

bit1=1 → Oculto.

bit2=1 → Sistema.

bit5=1 → Arquivo.

1 – Data/hora de modificação.

BC – Código de hora, DE – Código de data.

2 – Data/hora de criação.

BC – Código de tempo, DE – Código de data.

BC, DE – Veja acima.

**Saída:** CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

**Registradores:** AF, BC, DE, HL, IX, IY

**ID: 035 (DIRPRR) – Directory\_Property\_Get**

**Descrição:** Lê uma propriedade de um arquivo ou diretório. Para obter mais informações sobre o código de hora e data, consulte 023 (FILF2T).

**Biblioteca:** SyFile\_DIRPRR

**Entrada:** IXh – Caminho do arquivo e nome do Banco RAM (0~15)

HL – Caminho do arquivo e endereço do nome

A – Tipo de propriedade:

0 → Atributo.

1 → Data/hora de modificação.

2 → Data/hora de criação.

**Saída:** C – Atributos (se solicitado):

bit0=1 → Somente leitura.

bit1=1 → Oculto.

bit2=1 → Sistema.

bit3=1 → ID do Volume.

bit4=1 → Diretório.

bit5=1 → Arquivo.

BC, DE – Códigos de hora e data (se solicitado).

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, HL, IX, IY.

#### **ID: 036 (DIRREN) – Directory\_Rename**

Descrição: Renomeia um arquivo ou diretório. O novo nome do arquivo não deve incluir um caminho. A função retornará erro, se um arquivo ou diretório com o novo nome já existir.

Biblioteca: SyFile\_DIRREN.

Entrada: IXh – Banco RAM (0~15) do antigo e do novo nome de arquivo.

HL – Endereço do caminho / nome do arquivo antigo.

DE – Endereço do novo nome de arquivo.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

#### **ID: 037 (DIRNEW) – Directory\_New**

Descrição: Cria um novo diretório. A função retornará erro, se um arquivo ou diretório com o novo nome já existir.

Biblioteca: SyFile\_DIRNEW

Entrada: IXh – Caminho do diretório e nome do Banco RAM (0~15)  
HL – Caminho do diretório e endereço do nome

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro)

Registradores: AF, BC, DE, HL, IX, IY

#### **ID: 038 (DIRINP) – Directory\_Input**

Descrição: Lê o conteúdo de um diretório. Podem ser especificadas máscaras de arquivo no nome (\* e ? são permitidos) e um filtro de atributo. É recomendável sempre setar o bit 3 (ID do volume) do byte de filtro de atributo.

Biblioteca: SyFile\_DIRINP

Entrada: IXh – Banco de RAM do caminho do diretório (0~15)  
HL – Endereço do caminho do diretório (pode incluir uma máscara de pesquisa).

IXI – Filtro de atributo:

bit0 = 1 → Não mostrar arquivos somente leitura.

bit1 = 1 → Não mostra arquivos ocultos.

bit2 = 1 → Não mostra arquivos de sistema.

bit3 = 1 → Não mostra entradas de ID de volume.

bit4 = 1 → Não mostra diretórios.

bit5 = 1 → Não mostrar arquivos de arquivo.

A – Banco de memória RAM do buffer de destino (0~15).

DE – Endereço do buffer de destino.

BC – Comprimento do buffer de destino.

IY – Número de entradas, que deve ser ignorado.

Saída: HL – Número de entradas lidas.

BC – Espaço não utilizado restante no buffer de destino.

CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, DE, IX, IY.

Estrutura 00 4B Comprimento do arquivo (32 bits).

de dados: 04 1W Código de data, consulte 023 (FILF2T).

06 1W Código de tempo, consulte 023 (FILF2T).

08 1B Atributos, consulte 035 (DIRPRR).

09 ?B Nome do arquivo ou subdiretório.

?? 1B Terminador 0.

#### **ID: 039 (DIRDEL) – Directory\_DeleteFile**

Descrição: Exclui um ou mais arquivos. Podem ser excluídos vários arquivos usando uma máscara de arquivo (\* e ? são permitidos). Arquivos somente leitura não podem ser excluídos. Esta função também não exclui diretórios. Usar 040 (DIRRMD), para excluir diretórios.

Biblioteca: SyFile\_DIRDEL

Entrada: IXh – Caminho do arquivo e nome/máscara do banco de memória RAM (0~15)

HL – Caminho do arquivo e nome/endereço da máscara.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

#### **ID: 040 (DIRRMD) – Directory\_DeleteDirectory**

Descrição: Exclui um subdiretório. O subdiretório deve estar vazio e sem o atributo de somente leitura, caso contrário, a operação será abortada. O diretório pode ser especificado com ou sem "/" no final.

Biblioteca: SyFile\_DIRRMD

Entrada: IXh – Caminho do diretório e nome do Banco RAM (0~15)

HL – Caminho do diretório e endereço do nome

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro)  
 Registradores: AF, BC, DE, HL, IX, IY

**ID: 041 (DIRMOV) – Directory\_Move**

Descrição: Move um arquivo ou subdiretório para outro diretório na mesma unidade.

Biblioteca: SyFile\_DIRMOV.

Entrada: IXh – Arquivo/diretório antigo e novo banco de RAM do caminho (0~15).

HL – Caminho de origem do arquivo / diretório e endereço do nome.

DE – Endereço do caminho destino do arquivo/diretório.

Saída: CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: AF, BC, DE, HL, IX, IY.

**ID: 042 (DIRINF) – Directory\_DriveInformation**

Descrição: Retorna informações sobre uma unidade.

Biblioteca: SyFile\_DIRINF

Entrada: A – Letra do drive (A~Z).

C – Tipo de informação:

0 → Informações gerais da unidade.

1 → Quantidade de memória livre e total.

Saída: → Informação tipo 0:

A – Tipo:

00 → Dispositivo não existe.

01 → Dispositivo está pronto.

02 → Dispositivo não inicializado.

03 → Dispositivo está corrompido.

B – Mídia:

01 → Disquete face simples (Amsdos, PCW).

02 → Disquete face dupla (FAT 12).

08 → RAM disk.

16 → Disco rígido IDE ou cartão CF (Fat 16, Fat 32).

C – Sistema de arquivos:

01 → Dados Amsdos.

02 → Sistema Amsdos.

03 → PCW 180K.

16 → FAT 12.

17 → FAT 16.

18 → FAT 32.

D – Setores por cluster.  
 IY, IX – Número total de clusters.  
 → Tipo de informação 1:  
 HL, DE – Número de setores de 512 bytes livres.  
 IY, IX – Número total de clusters.  
 C – Setores por cluster.  
 → Tipo de informação 0 e 1:  
 CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro).

Registradores: F.

### **ID: 013 (DEVDIR) – Directory\_Input\_Extended**

Descrição: Lê o conteúdo de um diretório e converte-o em dados de controle de lista prontos para uso. Uma área deve ser reservada na RAM com tamanho mínimo recomendável de 4000 bytes para os dados. Uma segunda área, no mesmo banco RAM, deve ser reservada para a estrutura de dados de controle da lista. Seu tamanho é calculado assim: Tamanho = Número\_máximo\_de\_entradas \* (4 + Colunas\_adicionais \* 2). Para obter mais informações sobre a leitura de diretórios, consulte 038 (DIRINP).

Biblioteca: SyFile\_DEVDIR.

Entrada: A – bit0~3: Banco de memória RAM do buffer de destino (0~15).

bit4~7: Banco de RAM do caminho do diretório (0~15).

HL – Endereço do caminho do diretório (pode incluir uma máscara de pesquisa).

DE – Endereço do buffer de destino, cujos quatro primeiros bytes devem conter o seguinte:

00 1W Endereço da tabela de controle da lista.

02 1W Número máximo de entradas.

A função irá sobrescrever essas informações e preencher o buffer com os dados do diretório.

BC – Tamanho máximo do buffer de destino.

IXI – Filtro de atributo.

bit0 = 1 → Não mostrar arquivos somente leitura.

bit1 = 1 → Não mostra arquivos ocultos.

bit2 = 1 → Não mostra arquivos de sistema.

bit3 = 1 → Não mostra entradas de ID de volume.

bit4 = 1 → Não mostra diretórios.  
 bit5 = 1 → Não mostrar arquivos de arquivo.  
 IY – Número de entradas, que deve ser ignorado.  
 IXh – Colunas adicionais  
     bit0 = 1 → Tamanho do arquivo.  
     bit1 = 1 → Data e hora (última modificação).  
     bit2 = 1 → Atributos.  
 Saída:     HL – Número de entradas lidas  
 CY – Estado de erro (0 – Ok, 1 – Erro; A – Código de erro)  
 Registradores: AF, BC, DE, IX, IY.

### **5.6.6 – Funções do Gerenciador de Dispositivos**

As funções do gerenciador de dispositivos devem ser chamadas com RST 20H (BNKSCL).

**TIMGET (810CH) – Device\_TimeGet**  
 Descrição: Retorna a hora atual.  
 Como chamar: rst 20H : dw 810CH  
 Entrada: –  
 Saída:     A – Segundo (0 ~ 59).  
           B – Minuto (0 ~ 59).  
           C – Hora (0 ~ 23).  
           D – Dia (1 ~ 31).  
           E – Mês (1 ~ 12)  
           HL – Ano (1900 ~ 2100).  
           IXI – Fuso horário (-12 ~ +13).  
 Registradores: F, IY.

**TIMSET (810FH) – Device\_TimeSet**  
 Descrição: define a hora atual.  
 Como chamar: rst 20H : dw 810FH  
 Entrada:   A – Segundo (0 ~ 59).  
           B – Minuto (0 ~ 59).  
           C – Hora (0 ~ 23).  
           D – Dia (1 ~ 31).  
           E – Mês (1 ~ 12)  
           HL – Ano (1900 ~ 2100).  
           IXI – Fuso horário (-12 ~ +13).

Saída: -

Registradores: AF, BC, DE, HL, IY.

### **SCRSET (8136H) – Device\_ScreenModeCPCSet**

Descrição: Define o modo de tela dos micros CPC. Esta função é específica do CPC e apenas deve ser chamada por aplicativos que usam a tela inteira.

Como chamar: ld hl, 8136H : rst 28H

Entrada: E – Modo de tela do CPC (0,1,2).

Saída: -

Registradores: -

### **SCRGET (8139H) – Device\_ScreenMode**

Descrição: Retorna o modo de tela, número de cores e resolução.

Como chamar: ld hl, 8139H : rst 28H

Entrada: -

Saída: E – Modo de tela:

CPC/EP: 1, 2                    MSX: 5, 6, 7

PCW: 0                        G9K: 8, 9, 10, 11

D – Número de cores (2~16).

IX – Resolução horizontal.

IY – Resolução vertical.

Registradores: -

### **MOSGET (813CH) – Device\_MousePosition**

Descrição: Retorna a posição atual do ponteiro do mouse.

Como chamar: rst 20H : dw 813CH

Entrada: -

Saída: DE – Posição X.

HL – Posição Y.

Registradores: -

### **MOSKEY (813FH) – Device\_MouseKeyStatus**

Descrição: Retorna o estado atual das teclas do mouse.

Como chamar: rst 20H : dw 813FH

Entrada: -

Saída: A – Estado da tecla:

Bit0 = 1 → Botão esquerdo do mouse pressionado

Bit1 = 1 → Botão direito do mouse pressionado  
 Bit2 = 1 → Botão do meio do mouse pressionado

Registradores: F.

#### **KEYTST (8145H) – Device\_KeyTest**

Descrição: Retorna o estado atual de uma tecla.

Como chamar: Id hl, 8145H : rst 28H

Entrada: E – Código de verificação do teclado.

Resultado: E – Estado da tecla:

0 → A tecla não está pressionada no momento;  
 1 → A tecla está sendo pressionada.

Registradores: AF, BC, D, HL, IX, IY.

#### **KEYSTA (8148H) – Device\_KeyStatus**

Descrição: Retorna o estado das teclas shift / control / alt / capslock.

Como chamar: Id hl, 8148H : rst 28H

Entrada: –

Saída: E – bit0: SHIFT (1 → Pressionada).

bit1: CTRL (1 → Pressionada).

bit2: ALT (1 → Pressionada).

D – Estado da Caps Lock (1 – Travada).

Registradores: AF, BC, HL, IX, IY.

#### **KEYPUT (814BH) – Device\_KeyPut**

Descrição: Coloca um caractere de volta no buffer do teclado.

Como chamar: rst 20H : dw 814BH

Entrada: A – Código ASCII do caractere.

Saída: CY – Estado do buffer de teclado (1 → cheio).

Registradores: AF, BC, HL.

#### **IOMINP (8157H) – Device\_IO\_Multiln [somente CPC]**

Descrição: Lê vários bytes de uma porta de hardware de maneira rápida e os grava em um endereço de destino na memória.  
 Esta função está disponível apenas no SymbOS CPC.

Como chamar: rst 20H : dw 8157H

Entrada: DE – Endereço de destino.

IY – bit0–11: Comprimento em bytes.

bit12–15: Banco RAM de destino (0~15).

IX – Endereço da porta.

Registradores: AF, BC, DE, HL

**IOMOUT (815AH) – Device\_IO\_MultiOut [somente CPC]**

Descrição: Grava vários bytes em uma porta de I/O de maneira rápida a partir de um endereço de origem na memória.  
Esta função está disponível apenas no SymbOS CPC.

Como chamar: `rst 20H : dw 8157H`

Entrada: DE – Endereço de origem.

IY – bit0–11: Comprimento em bytes.

bit12–15: Banco RAM de destino (0~15).

IX – Endereço da porta.

Registradores: AF, BC, DE, HL.

## **5.7 – LINHA DE COMANDO DO SYMSHELL**

Os comandos do SymShell são acionados por meio de uma mensagem, que deve ser enviada com RST 10H (MSG SND) para o processo do SymShell. O SymShell passará seu ID de processo e a resolução da tela de texto para o aplicativo por meio da linha de comando.

### **5.7.1 – Comandos do terminal de texto**

#### **ATTRIB**

Formato: ATTRIB <nome arquivo>

Função: Apresenta atributos atuais do arquivo.

Formato: ATTRIB <nome arquivo>%–R, %+R, %-H, %+H, %-S, %+S

Função: Adiciona (+) ou remove (–) atributos de arquivo

H → arquivo oculto

S → arquivo de sistema

R → somente leitura

#### **CD**

Formato: CD

Função: Apresenta diretório.

Formato: CD [ [d:][diretório] ]

Função: Troca o subdiretório corrente.

Formato: CD [ \ | .. ]

Função: Volta apenas um subdiretório (..) ou volta para o diretório raiz (\).

**CLS**

Formato: CLS

Função: Limpa a tela.

**COPY**

Formato: COPY <arquivo fonte> <arquivo destino>

Função: Copia arquivos. Aceita o caractere coringa ‘ \*’ ( por exemplo, “arquivo.\*”, “\*.ext” ou “\*.\*” )

**DATE**

Formato: DATE [data]

Função: Apresenta ou altera a data do sistema. A data também pode ser alterada clicando duas vezes na hora na barra de tarefas.

**DEL**

Formato: DEL <nome de arquivo>

Função: Deleta um ou mais arquivos. Aceita o caractere coringa ‘ \*’ (por exemplo, “\*.ext” ou “fil\*.ex\*”)

**DIR**

Formato: DIR [%S] [%H] [%F] [%P] [<nome arquivo>]

Função: Apresenta os nomes dos arquivos do disco. Aceita o caractere coringa ‘ \*’ (por exemplo, “\*.ext” ou “fil\*.ex\*”)

[/S] Lista arquivos de sistema.

[/H] Lista arquivos ocultos.

[/F] Calcula o espaço livre no disco.

[/P] Pausa a listagem ao completar uma tela.

**HELP**

Formato: HELP [<comando>]

Função: Apresenta o arquivo de ajuda do <comando> especificado ou lista todos se não houver argumento.

**MKDIR**

Formato: MKDIR [d:] <caminho>

Função: Cria um subdiretório.

**MOVE** (interno, 2)

Formato: MOVE <nome do arquivo> <caminho-destino>

Função: Move arquivos para outra parte do disco. Aceita o caractere coringa '\*' (exemplo: "arquivo.\* destino.\*")

**RMDIR**

Formato: RMDIR <nome do diretório>

Função: Remove um subdiretório vazio.

**REN**

Formato: REN <nome antigo> <nome novo>

Função: Renomeia o arquivo <nome antigo> com <nome novo>. Aceita o caractere coringa '\*' (exemplo: "ren \*.txt \*.bak").

**TIME**

Formato: TIME [<hora>]

Função: Apresenta ou altera a hora do sistema. A hora também pode ser alterada clicando duas vezes na hora na barra de tarefas.

**5.7.1.1 – Aplicativos padrão**

**DIMON.COM** – Monitor de disco de linha de comando para visualizar os discos em hexadecimal.

**NETSTAT.COM** – Exibe as conexões de rede ativas com o respectivo estado.

**TELNET.COM** – Cliente telnet. Digite TELNET <nome de domínio> ou <IP>. Por padrão, a porta TCP 23 é usada. Se o SYMTEL for iniciado sem nome de domínio ou IP, o console é carregado, onde podem ser alteradas as configurações. Para conectar a um host, basta digitar o nome IP ou DNS e pressionar <RETURN>. Para desconectar pressione CTRL-C ou faça logoff no sistema remoto. Pressione CTRL-C também para sair do SYMTEL. Para alternar entre o modo de tela cheia e o modo de janela, pressione GRAPH+RETURN. No modo de tela cheia o desempenho é melhor do que em uma janela.

**UNZIP.COM** – Extrai arquivos .ZIP ou .GZ. Digite UNZIP H para apresentar a ajuda.

**WGET.COM** – Aplicativo de linha de comando para download de arquivos sob o protocolo HTTP, usando o daemon de rede.

### 5.7.2 – Comandos e Respostas do Symshell

Bibliotecas: SyShell\_PARALL  
SyShell\_PARSHL

**ID: 064 (MSC\_SHL\_CHRINP)** – SymShell\_CharInput\_Command

Descrição: Solicita um caractere de uma fonte de entrada. A fonte de entrada pode ser a padrão ou o teclado. Se o teclado for usado, SymShell espera pelo usuário e não enviara resposta se nenhuma tecla for pressionada.

Biblioteca: SyShell\_CHRINP.

Mensagem: 00 1B 064.  
01 1B Canal (0 → Padrão, 1 → Teclado).

Resposta: Veja MSR\_SHL\_CHRINP.

**ID: 192 (MSR\_SHL\_CHRINP)** – SymShell\_CharInput\_Response

Descrição: Se um caractere puder ser recebido do teclado, de um arquivo ou de outra fonte, ele será enviado ao aplicativo por meio desta mensagem. Se o usuário pressionou Control+C ou se o fim do arquivo (EOF) foi alcançado, o sinalizador EOF será ligado.

Mensagem: 00 1B 192.  
01 1B Flag EOF (Se diferente de 0, o fim do arquivo foi alcançado)  
02 1B Caractere.  
03 1B Estado de erro:  
254 → Processo desconhecido (SymShell não reconhece o processo, que enviou o comando, e, por isso, não fornece nenhum serviço).  
253 → Dispositivo de destino cheio.  
252 → Buffer de anel interno cheio.

251 → Muitos processos (SymShell não pode lidar com a quantidade de processos em execução ao mesmo tempo no ambiente de terminal de texto)

Qualquer outro → Consulte “Códigos de erro” no capítulo “Gerenciador de arquivos”.

**ID: 065 (MSC\_SHL\_STRINP)** – SymShell\_StringInput\_Command

Descrição: Solicita uma string de uma fonte de entrada, que pode ser o canal padrão ou o teclado. O comprimento máximo de uma string é de 255 caracteres, então o buffer deve ter um tamanho de 256 bytes (255 + terminador).

Biblioteca: SyShell\_STRINP.

Mensagem: 00 1B 065.

01 1B Canal (0 → Padrão, 1 → Teclado).

02 1B Banco de memória RAM do buffer (0~15).

03 1W Endereço de buffer de destino.

Resposta: Veja MSR\_SHL\_STRINP.

**ID: 193 (MSR\_SHL\_STRINP)** – SymShell\_StringInput\_Response

Descrição: Se uma linha de texto puder ser recebida do teclado, de um arquivo ou de outra fonte (terminada por 13/10), será enviada para o aplicativo por meio desta mensagem de resposta. Se o usuário pressionou Control + C ou se o fim do arquivo for atingido, o sinalizador EOF será setado.

Mensagem: 00 1B 193

01 1B Flag EOF (Se diferente de 0, o fim do arquivo foi alcançado)

03 1B Estado de erro (veja acima “SymShell\_CharInput\_Response”)

**ID: 066 (MSC\_SHL\_CHROUT)** – SymShell\_CharOutput\_Command

Descrição: Envia um caractere para o canal de saída.

Biblioteca: SyShell\_CHROUT.

Mensagem: 00 1B 066.

01 1B Canal (0 – Padrão, 1 – Tela).

02 1B Caractere.

Resposta: Veja MSR\_SHL\_CHROUT

**ID: 194 (MSR\_SHL\_CHROUT) – SymShell\_CharOutput\_Response**

Descrição: Informa a aplicação se o caractere foi enviado corretamente. Um aplicativo não deve enviar mais de um caractere ao mesmo tempo, antes que tal resposta tenha sido recebida.

Mensagem: 00 1B 194.

03 1B Estado de erro (veja acima SymShell\_CharInput\_Response")

**ID: 067 (MSC\_SHL\_STROUT) – SymShell\_StringOutput\_Command**

Descrição: Envia uma string para o destino de saída.

Biblioteca: SyShell\_STROUT.

Mensagem: 00 1B 067.

01 1B Canal (0 – Padrão, 1 – Tela).

02 1B Banco de RAM da coluna (0~15).

03 1W Endereço de string.

05 1B Comprimento da string (sem terminador 0).

Resposta: Veja MSR\_SHL\_STROUT.

**ID: 195 (MSR\_SHL\_STROUT) – SymShell\_StringOutput\_Response**

Descrição: Informa a aplicação se a string foi enviado corretamente. Um aplicativo não deve enviar mais de uma string antes que tal resposta tenha sido recebida.

Mensagem: 00 1B 195.

03 1B Estado de erro (veja acima "SymShell\_CharInput\_Response")

**ID: 068 (MSC\_SHL\_EXIT) – SymShell\_Exit\_Command**

Descrição: O aplicativo informa SymShell sobre um evento de saída. Se um aplicativo fecha sozinho, SymShell deve ser informado sobre isso, para que possa remover o aplicativo de sua área de gestão interna. Neste caso, o tipo de saída deve ser 0 ("sair").

Biblioteca: SyShell\_EXIT

Mensagem: 00 1B 068

01 1B Tipo de saída:

0 → O aplicativo fecha sozinho.

1 → O aplicativo libera o foco e entra no modo desfocado.

Resposta: O SymShell não envia uma mensagem de resposta.

**ID: 069 (MSC\_SHL\_PTHADD)** – SymShell\_PathAdd\_Command

Descrição: ...

Biblioteca: SyShell\_PTHADD.

Mensagem: 00 1B 069.

- 01 1W Endereço do caminho de base (0 – Para padrão).
- 03 1W Endereço do componente de caminho adicional.
- 05 1W Endereço do novo caminho completo.
- 07 1B Caminhos do banco RAM (0~15).

Resposta: Veja MSR\_SHL\_PTHADD

**ID: 197 (MSR\_SHL\_PTHADD)** – SymShell\_PathAdd\_Response

Descrição: ...

Mensagem: 00 1B 197.

- 01 1W Posição seguinte ao último caractere no novo caminho.
- 03 1W Posição seguinte ao último “/” no novo caminho.
- 05 1B bit0 = 1 → Novo caminho termina com “/”.  
bit1 = 1 → Novo caminho contém curingas.

### 5.7.3 – Códigos de Controle do Terminal Symshell

#### Parâmetros de descrição de código

- 00 Parar a saída de texto e ignorar a parte restante da linha.
- 01 –
- 02 Desligar o cursor. Isso fará com que o cursor fique invisível.
- 03 Ligar o cursor.
- 04 Salvar a posição atual do cursor.
- 05 Restaurar a última posição salva do cursor.
- 06 Ativar saída de texto (ver também 21).
- 07 –
- 08 Mover o cursor um caractere para a esquerda.
- 09 Mover o cursor um caractere para a direita.

- 10 Mover o cursor um caractere para baixo.
- 11 Mover o cursor um caractere para cima.
- 12 Limpar a tela e coloque o cursor na posição 1/1 (HOME).
- 13 Mover o cursor para o início da linha atual.
- 14 Mover o cursor por vários caracteres. P1 especifica a direção e o número de posições a mover.
  - P1 = 1~80 → O cursor moverá de 1 a 80 caracteres para a direita.
  - P1 = 81~160 → O cursor moverá de 1 a 80 caracteres para a esquerda (parâmetro – 80).
  - P1 = 161~185 → O cursor moverá de 1 a 25 caracteres para baixo (parâmetro – 160).
  - P1 = 186~210 → O cursor moverá de 1 a 25 caracteres para cima (parâmetro – 185).
- Obs.: O cursor não ultrapassará a borda da tela.
- 15 –
- 16 Limpar o caractere na posição do cursor (usando o espaço [32])
- 17 Limpar linha do cursor para a esquerda.
- 18 Limpar linha do cursor para a direita.
- 19 Limpar tela do cursor para cima.
- 20 Limpar tela do cursor para baixo.
- 21 Desativar a saída de texto. Não haverá mais caracteres impressos até que o código 06 apareça.
- 22 Definir uma guia na coluna atual.
- 23 Limpar uma guia na coluna atual.
- 24 Limpar todas as guias.
- 25 Ir para a próxima guia.
- 26 Preencher a área da tela com um caractere especificado
  - P1 – Caractere.
  - P2 – Posição inicial X.
  - P3 – Posição inicial Y.
  - P4 – Posição final X.
  - P5 – Posição final Y.
- Obs.: Este código de controle ainda não está implementado.**
- 27 –
- 28 Definir tamanho da janela do terminal. O tamanho mínimo é 10x4 e o tamanho máximo é 80x25 (MSX: 80x24). Após a janela ser

redimensionada, a tela será limpa e o cursor colocado no canto superior esquerdo (1/1 – HOME).

P1 – Largura.

P2 – Altura.

- 29 Rolar a janela uma linha para cima ou para baixo uma linha. Isso não influenciará a posição atual do cursor.  
P1 = Direção (1 → Para cima, 2 → Para baixo).
- 30 Mova o cursor para o canto superior esquerdo (1/1).
- 31 Mova o cursor para um local de tela especificado:  
P1 – Posição X (1~80).  
P2 – Posição Y (1~25).

#### **5.7.4 – Códigos ASCII estendidos**

136 – Cursor up	154 – Alt + C	172 – Alt + U
137 – Cursor down	155 – Alt + D	173 – Alt + V
138 – Cursor left	156 – Alt + E	174 – Alt + W
139 – Cursor right	157 – Alt + F	175 – Alt + X
140 – F0	158 – Alt + G	176 – Alt + Y
141 – F1	159 – Alt + H	177 – Alt + Z
142 – F2	160 – Alt + I	178 – Alt + 0
143 – F3	161 – Alt + J	179 – Alt + 1
144 – F4	162 – Alt + K	180 – Alt + 2
145 – F5	163 – Alt + L	181 – Alt + 3
146 – F6	164 – Alt + M	182 – Alt + 4
147 – F7	165 – Alt + N	183 – Alt + 5
148 – F8	166 – Alt + O	184 – Alt + 6
149 – F9	167 – Alt + P	185 – Alt + 7
150 – F.	168 – Alt + Q	186 – Alt + 8
151 – Alt + @	169 – Alt + R	187 – Alt + 9
152 – Alt + A	170 – Alt + S	
153 – Alt + B	171 – Alt + T	

### 5.7.5 – Códigos de Varredura do Teclado

O código de verificação é usado na função "Device\_KeyTest". Observe que eles são iguais em todas as plataformas suportadas.

00 – Cursor Up	20 – F4	40 – 8	60 – S
01 – Cursor Right	21 – Shift	41 – 7	61 – D
02 – Cursor Down	22 – \	42 – U	62 – C
03 – F9	23 – Control	43 – Y	63 – X
04 – F6	24 – ^	44 – H	64 – 1
05 – F3	25 – –	45 – J	65 – 2
06 – Enter	26 – @	46 – N	66 – Esc
07 – F.	27 – P	47 – Space	67 – Q
08 – Cursor Left	28 – ;	48 – 6	68 – Tab
09 – Alt	29 – :	49 – 5	69 – A
10 – F7	30 – /	50 – R	70 – Capslock
11 – F8	31 – .	51 – T	71 – Z
12 – F5	32 – 0	52 – G	72 – Joy Up
13 – F1	33 – 9	53 – F	73 – Joy Down
14 – F2	34 – O	54 – B	74 – Joy Left
15 – F0	35 – I	55 – V	75 – Joy Right
16 – Clr	36 – L	56 – 4	76 – Fire 2
17 – [	37 – K	57 – 3	77 – Fire 1
18 – Return	38 – M	58 – E	78 – [not used]
19 – ]	39 – ,	59 – W	79 – Del

### 5.8 – CONFIGURAÇÃO DO SISTEMA

Todas as configurações do usuário no SymbOS são armazenadas na configuração do sistema, que é salva no arquivo "SYMBOS.INI". Ele é dividido em 5 partes:

1. Cabeçalho, que contém o identificador e o comprimento das três partes seguintes.
2. Área central, que contém dados carregados no primeiro Banco RAM.
3. Área de dados, que contém dados adicionais geralmente carregados carregados em um banco RAM diferente.

4. Área de transferência (atualmente não usada).
5. Fonte.

### **5.8.1 – Cabeçalho**

- 0000 2B Identificador que também contém a versão do arquivo de configuração [byte0] = "S", [byte1] = 1 (versão atual).
- 0002 1W Comprimento de do cabeçalho (= 8 bytes) mais a parte da área central de configuração do sistema SymbOS (será sempre carregado no Banco RAM 0).
- 0004 1W Comprimento da parte da área de dados (o Banco RAM depende da plataforma)
- 0006 1W Comprimento da parte da área de transferência (que é 0, pois atualmente não é usada)

### **5.8.2 – Área Central**

#### **5.8.2.1 – Dispositivos de armazenamento de massa.**

- 0000 128B Configuração do dispositivo; isto consiste em 8 registros de dados em 16 bytes para cada dispositivo:
  - 00 1B Letra da unidade (maiúscula) ou 0, se o slot do dispositivo estiver vazio.
  - 01 1B bit0–3: Tipo (0=Disquete, 1=IDE/SCSI) → slot driver.  
bit4–6: Reservado (definido como 0).  
bit7: Sinalizador de mídia removível (1 = Sim).
  - 02 1B Subdrive:  
→ Se o dispositivo for um disquete:  
bit0–1: Unidade.  
bit2: Cabeça.  
bit3: Sinalizador de passo duplo (1 = Sim).  
bit4–7: Reservado (definido como 0).  
→ Se o dispositivo for um dispositivo IDE/SCSI/SD:  
bit0–3: Partição (0 = Não particionado).  
bit4–7: IDE → Canal (0 = mestre, 1 = escravo).  
SCSI → Subdispositivo (0~15).
  - 03 1B Reservado (definido como 0).
  - 04 12B Nome do dispositivo (terminado por 0).

### 5.8.2.2 – Exibição e diversos (1)

- 0128 17W Paleta de cores (a borda é definida pela palavra 17).  
 Para cada entrada: bit0-3: Componente azul.  
 bit4-7: Componente verde.  
 bit8-11: Componente vermelho.
- 0162 1B Modo de tela:  
 0 [PCW] – 768x255x2      7 [MSX] – 512x212x16  
 1 [CPC, EP] – 320x200x4      8 [G9K] – 384x240x16  
 2 [CPC, EP] – 640x200x2      9 [G9K] – 512x212x16  
 5 [MSX] – 256x212x16      10 [G9K] – 768x240x16  
 6 [MSX] – 512x212x4      11 [G9K] – 1024x212x16
- 0163 32B Caminho do sistema.
- 0195 1B Fuso horário (-12 a +12).
- 0196 1B Tipo de fundo de tela (0~15 → Cor simples, -1 → Gráfico).
- 0197 32B Caminho completo com o nome do arquivo do gráfico de fundo de tela, terminado por 0 (apenas, se “tipo de fundo” for igual a -1).

### 5.8.2.3 – Teclado (1) e mouse

- 0229 1B Atraso de teclado (em passos 1/50 ou 1/60 segundos entre o primeiro e o segundo caractere).
- 0230 1B Velocidade de repetição do teclado (atraso entre cada caractere seguinte).
- 0231 1B Atraso do joystick e do mouse (até que o mouse atinja a velocidade máxima).
- 0232 1B Velocidade (resolução) do joystick do mouse (em pontos).
- 0233 1B Fator de velocidade do mouse:  
 (Movimento\_final = Movimento\_original \* Veloc. mouse / 16).
- 0234 1B Retardo do clique duplo do mouse (tempo máximo em 1/50 ou 1/60 segundos, quando um clique duplo é reconhecido).
- 0235 1B Não zero, para troca das teclas esquerda/direita do mouse.
- 0236 1B Velocidade da roda do mouse

### 5.8.2.4 – Diversos (2) e links da área de trabalho

- 0237 1B Drive SYMBOS.INI (“A”, ...)
- 0238 1B Sinalizadores diversos:  
 bit0: Configuração de salvamento automático.

- 0239 1B Sinalizador (1), se o módulo de extensão SymbOS deve ser carregado.
- 0240 1B Sinalizador para hardware estendido (+1 = Mouse, +2 = Relógio em tempo real, +4 = interface IDE / SCSI, +16 = M4Board)
- 0241 1B Área de trabalho virtual (0 → Sem área de trabalho virtual):  
bit0–3: Resolução X (1 = 512, 2 = 1000).  
bit4–7: Resolução Y (Ainda não definido).
- 0242 1B Número de ícones da área de trabalho.
- 0243 1B Número de entradas de menu / programas iniciar.
- 0244 1B Número de entradas de atalho da barra de tarefas (atualmente não suportado).
- 0245 1B Tipo de máquina:  
 0 – CPC 464                    7 – MSX1  
 1 – CPC 664                    8 – MSX2  
 2 – CPC 6128                  9 – MSX2+  
 3 – CPC 464+                 10 – MSX turboR  
 4 – CPC 6128+                12 – PCW8xxx  
 6 – Enterprise                13 – PCW9xxx
- 0246 16W Posições dos ícones na área de trabalho. Para cada um dos 8 ícones são reservados quatro bytes; os dois primeiros contêm a posição X e os dois últimos a posição Y.
- 0278 32B Caminho e nome do arquivo executável de linha de comando do autoexec.
- 0310 1B Igual a “1”, se o arquivo de linha de comando autoexec deve ser executado.

### 5.8.3 – Área de Dados

#### 5.8.3.1 – Links da área de trabalho (2)

- 0000 400B Nomes de entrada de programas do menu Iniciar (20 entradas de 20 bytes cada, terminados por 0).
- 0400 640B Caminhos e nomes de arquivo do Menu Iniciar (20 entradas em 32 bytes cada, terminados por 0).
- 1040 256B Caminhos e nomes de arquivo da área de trabalho (8 entradas de 32 bytes cada, terminado por 0).

- 1296 192B Nomes dos ícones da área de trabalho (8 entradas, sendo que cada uma consiste em 2 linhas de 12 bytes, e cada linha é terminada por 0)
- 1488 1176B Gráficos de ícones da área de trabalho (8 entradas, cada uma consistindo de um cabeçalho gráfico de 3 bytes cabeçalho gráfico e um bitmap de 144 bytes ( $6^*24$ ))
- 2664 768B Associação de extensão de arquivo (16 entradas em 48 bytes)
- 00 3B Extensão 1 (maiúscula; se byte0 = 1, então toda a entrada não está definida)
  - 03 3B Extensão 2 (se byte0 = 1, então esta entrada não está definida)
  - 06 3B Extensão 3 (s.a.)
  - 09 3B Extensão 4 (s.a.)
  - 12 3B Extensão 5 (s.a.)
  - 15 33B Caminho do aplicativo e nome do arquivo que será iniciado, se um arquivo com uma das extensões listadas acima foi aberto.

### **5.8.3.2 – Protetor de tela**

- 3432 1B Igual a “1”, se o protetor de tela estiver presente.
- 3433 1B Duração de inatividade do usuário, após o qual o protetor de tela será iniciado.
- 3434 33B Caminho e nome de arquivo do aplicativo protetor de tela (terminado por 0).
- 3467 64B Dados de configuração específicos do protetor de tela (podem ser armazenados e lidos aqui)

### **5.8.3.3 – Teclado (2)**

- 3531 80B Definição de teclado (normal)
- 3611 80B Definição de teclado (deslocamento)
- 3691 80B Definição de teclado (controle)
- 3771 80B Definição de teclado (alt)

### **5.8.3.4 – Segurança**

- 3851 16B Nome de usuário seguro.
- 3867 16B Senha.
- 3883 1B Sinalizadores de segurança (ainda não usados, sempre 0).

## 5.9 – APLICATIVOS DO PROTETOR DE TELA

### ID: 001 (MSC\_SAV\_INIT) – ScreenSaver\_Init\_Command

Descrição: O processo do chamador, que iniciou o protetor de tela (geralmente o gerenciador de área de trabalho ou o painel de controle) enviou um comando de inicialização. O protetor de tela agora deve armazenar o ID do processo do remetente para poder enviar uma resposta mais tarde (consulte MSR\_SAV\_CONFIG). Em seguida, ele deve copiar os dados de configuração em sua própria área de memória. Esses dados podem ter um tamanho de até 64 bytes e são armazenados no arquivo SYMBOS.INI junto com as outras configurações do sistema.

Biblioteca: ScrSav\_MAIN.

Mensagem: 00 1B 001.

01 1B Banco de dados de configuração (0~7).

02 1W Endereço de dados de configuração de (64 bytes).

Resposta: Nenhuma resposta esperada do protetor de tela.

### ID: 002 (MSC\_SAV\_START) – ScreenSaver\_Start\_Command

Descrição: O processo do chamador pede ao protetor de tela para iniciar sua animação. A animação será exibida, enquanto nenhuma tecla for pressionada e o mouse não for movido.

Biblioteca: ScrSav\_MAIN.

Mensagem: 00 1B 002.

Resposta: Nenhuma resposta esperada do protetor de tela.

### ID: 003 (MSC\_SAV\_CONFIG) – ScreenSaver\_Config\_Command

Descrição: O processo do chamador pede ao protetor de tela para abrir uma janela de diálogo de configuração. Nessa janela, o usuário pode modificar as configurações do protetor de tela. Se não houver nada para configurar, o protetor de tela pode ignorar este comando ou basta abrir uma janela de informações.

Biblioteca: ScrSav\_MAIN

Mensagem: 00 1B 003.

Resposta: Ver MSR\_SAV\_CONFIG.

**ID: 004 (MSR\_SAV\_CONFIG) – ScreenSaver\_Config\_Response**

**Descrição:** O usuário terminou de modificar as configurações e clicou no botão "Ok" da caixa de diálogo de configuração. O protetor de tela agora deve enviar de volta os dados atualizados para o processo do chamador, para que eles possam ser gravados novamente no arquivo SYMBOS.INI.

**Biblioteca:** ScrSav\_CFGSAV

**Mensagem:** 00 1B 001

01 1B Banco de dados de configuração (0~7).

02 1W Endereço de dados de configuração de (64 bytes).

## 5.10 – MAPA DE MEMÓRIA

### 5.10.1 – Mapa geral da memória

O diagrama a seguir mostra de que forma os diferentes bancos e blocos de memória são usados no SymbOS.

	Banco 0	Banco 1	Banco n
FFFFH	Dados e gerenciador de sistema	Livre	Livre
C000H	Buffers Subrotinas Ger. Disp. Ger. Tela	Livre	Livre
BFFFH	Gerenciador da Área de Trabalho	Livre	Livre
8000H	Ger. área trab., sistema e arquivo – LL	Kernel jumps	Kernel jumps
7FFFH			
4000H			
3FFFH			
0000H			

### 5.10.2 – Mapa da memória do aplicativo

A memória dentro de um banco de RAM de aplicativo (1 – n) é usada da seguinte maneira:

1. 0000–03FF Kernel jumps, multitarefa do Kernel e rotinas de gerenciamento de bancos.
2. 0400–FFFF Código e dados internos do aplicativo.
3. 0400–3FFF Dados do aplicativo usados pelo gerenc. de tela.  
4000–7FFF Um objeto deve estar dentro de um bloco de 16K.  
8000–BFFF  
C000–FFFF
4. Dados de “transferência” do aplicativo C000–FFFF, usados pelo gerenciador de área de trabalho, buffer de mensagens e pilha.

### 5.10.3 – Configurações de memória

O diagrama a seguir mostra como a memória é configurada durante a atividade de um dos módulos do SymbOS.

	DesktopManager (C1)	ScreenManager (C4-7)	FileManager-HL (C4)
FFFFH	Banco n Bloco 3 Transfer RAM	Banco 0 Bloco 3	Banco 0 Bloco 3
C000H			
BFFFH	Banco 0 Bloco 2	Banco 0 Bloco 2 ScreenManager	Banco 0 Bloco 2
8000H			
7FFFH	Banco 0 Bloco 1	Banco n Bloco m RAM de dados	Banco 1 Bloco 0
4000H	DesktopManager		FileManager-HL
3FFFH	Banco 0 Bloco 0	Banco 0 Bloco 0	Banco 0 Bloco 0
0000H			

	FileManager-LL (**)	Application (C2)
FFFFH	Banco 0 Bloco 3	Banco n Bloco 3 Trnf, Code, Data
C000H		
BFFFH	Slot x,y Disk-ROM	Banco n Bloco 2 Code, Data
8000H		
7FFFH	Banco n Bloco m	Banco n Bloco 1 Code, Data
4000H	RAM de dados	
3FFFH	Banco 0 Bloco 0	Banco n Bloco 0 Code, Data
0000H	FileManager-LL	

## 5.11 – GERENCIADOR DE TELA

O gerenciador de tela contém todas as rotinas para o acesso direto ao vídeo por hardware. Há apenas uma função, que também pode ser usada por aplicativos. Deve ser chamada com RST 20H (BNKSCL).

### TXTLEN (815DH) – Screen\_TextLength

Descrição: Retorna a largura e a altura de uma linha de texto em pontos, caso seja impresso na tela. O comprimento do texto (número de caracteres) pode ser definido em IY. Se o texto terminar em 0 ou 13, deve ser usado -1 para o comprimento. Observe, que esta função sempre usa a fonte do sistema para calcular a largura e a altura.

Como chamar:   rst 20H : dw 815DH

Entrada:   HL – Endereço de texto.

              A – Banco de memória do texto (1~15).

              IY – Número máximo de caracteres (comprimento).

Saída:    DE – Largura do texto em pontos.

              A – Altura do texto em pontos.

Registradores:   F, BC, HL, IX.

## 5.12 – DAEMON DE REDE

O daemon de rede SymbOS oferece todos os serviços para acesso total à rede, exatamente como um processo de serviço compartilhado.

### 5.12.1 – Configuração

Config\_Get CFGGET 001 130 A – Tipo; E,HL – Buffer de dados.  
 → (buffer foi preenchido).  
 Config\_Set CFGSET 002 131 A – Tipo; E,HL – Dados de config.  
 → (a configuração foi definida).

### 5.12.2 – Serviços de Camada de Transporte

TCP_Open	TCPOPN	016	144	A – Modo; HL – Porta local (IX, IY – IP remoto; DE – Porta rem.) CY=0 → Ok; A – Handle
TCP_Close	TCPCLC	017	145	A – Handle CY=0 → Ok; A – Handle
TCP_Status	TCPSTA	018	146	A – Handle CY=0 → Ok; A – Handle, L – Estado (BC – Bytes recebidos; IX, IY – IP remoto; DE – Porta remota)
TCP_Receive	TCPRCV	019	147	A – Handle; BC – Comprimento; E,HL – Memória CY=0 → Ok; A – Handle; BC – Número de bytes restantes; Z=1 → todos os bytes foram receb.
TCP_Send	TCPSND	020	148	A – Handle; BC – Comprimento; E,HL – Memória; CY=0 → Ok; A – Handle; BC – Número de bytes enviados; HL – Número de bytes restantes; Z=1 → todos bytes foram enviados.
TCP_Skip	TCPSKP	021	149	A – Handle, BC – Comprimento; CY=0 → Ok; A – Handle.
TCP_Flush	TCPFLS	022	150	A – Handle; CY=0 → Ok; A – Handle.

TCP_Disconnect	TCPDIS	023	151	A – Handle; CY=0 → Ok; A – Handle.
TCP_Event	TCPEVT	-	159	A – Handle; L – Estado; ( BC – Bytes recebidos; IX, IY – IP remoto; DE – Porta remota).
UDP_Open	UDPOPN	032	160	HL – Porta local, E – Banco mem. CY=0 → Ok; A – Handle.
UDP_Close	UDPCLO	033	161	A – Handle; CY=0 → Ok; A – Handle.
UDP_Status	UDPSTA	034	162	A – Handle CY=0 → Ok; A – Handle, L – Estado (BC – Bytes recebidos; IX,IY – IP remoto; DE – Porta rem.)
UDP_Receive	UDPRCV	035	163	A – Handle, HL – Memória; CY=0 → Ok; A – Handle.
UDP_Send	UDPSND	036	164	A – Handle; BC – Comprimento; HL – Memória; IX,IY – IP remoto, DE – Porta remota; CY=0 → Ok; A – Handle.
UDP_Skip	UDPSKP	037	165	A – Handle; CY=0 → Ok; A – Handle.
UDP_Event	UDPEVT	-	175	A – Handle; L – Estado; (BC – Bytes recebidos; IX, IY – IP remoto; DE – Porta remota).

### 5.12.3 – Serviços de Camada de Aplicação

DNS_Resolve	DNSRSV	112	240	E,HL – Endereço CY=0 → Ok, IX,IY – IP.
DNS_Verify	DNSVFY	113	241	E,HL – Endereço A – Tipo de endereço (0 – Nenhum endereço válido, 1 – Endereço IP, 2 – Endereço de domínio).

## 5.13 – CONSTANTES SYMBOS

### 5.13.1 – IDs de processos

PRC\_ID\_KERNEL equ 1      Processo do Kernel.

PRC_ID_DESKTOP	equ 2	Processo do gerenciador de área de trabalho.
PRC_ID_SYSTEM	equ 3	Processo do gerenciador do sistema.

### 5.13.2 – Mensagens

MSC_GEN_QUIT	equ 0	O aplicativo está sendo solicitado a encerrar-se.
MSC_GEN_FOCUS	equ 255	O aplicativo está sendo solicitado a focar sua janela.

### 5.13.3 – Comandos do Kernel

MSC_KRL_MTADDP	equ 1	Adicionar processo (P1/2 – Pilha, P3 – Prioridade (7–alto, 1–baixo), P4 – Banco RAM (0~8)).
MSC_KRL_MTDELP	equ 2	Processo de exclusão (P1 – ID).
MSC_KRL_MTADDT	equ 3	Adicionar temporizador (P1/2 – Pilha, P4 – Banco RAM (0~8)).
MSC_KRL_MTDELT	equ 4	Excluir temporizador (P1 – ID).
MSC_KRL_MTSLPP	equ 5	Definir processo para modo de hibernação.
MSC_KRL_MTWAKP	equ 6	Processo de despertar.
MSC_KRL_TMADDT	equ 7	Adicionar serviço de contador (P1/2 – Endereço, P3 – Banco RAM, P4 – Processo, P5 – Frequência).
MSC_KRL_TMDELT	equ 8	Excluir serviço do contador (P1/2 – Endereço, P3 – Banco RAM).
MSC_KRL_TMDELP	equ 9	Excluir todos os serviços de contador de um processo (P1 – ID do processo).

### 5.13.4 – Respostas do Kernel

MSR_KRL_MTADDP	equ 129	O processo foi adicionado (P1 = 0/1 → Ok / falhou, P2 – ID).
MSR_KRL_MTDELP	equ 130	O processo foi excluído.
MSR_KRL_MTADDT	equ 131	Processo de cronômetro foi excluído (P1 = 0/1 → Ok / falhou, P2 – ID).
MSR_KRL_MTDELT	equ 132	O cronômetro foi removido.
MSR_KRL_MTSLPP	equ 133	O processo está suspenso agora.

MSR_KRL_MTWAKP	equ 134	O processo foi ativado.
MSR_KRL_TMAADDT	equ 135	serviço de contador foi adicionado (P1 = 0 / 1 → Ok / falhou).
MSR_KRL_TMDELT	equ 136	Serviço de contador foi excluído.
MSR_KRL_TMDelp	equ 137	Todos os serviços de contador de um processo foram excluídos.

### 5.13.5 – Comandos do sistema

MSC_SYS_PRGRUN	equ 16	Carregar aplicativo ou documento (P1/2 – Nome do endereço do arquivo, P3 – Nome do arquivo do banco RAM).
MSC_SYS_PRGEND	equ 17	Sair do aplicativo (P1 – ID).
MSC_SYS_SYSWNX	equ 18	Abrir o diálogo para alterar a janela atual (próximo) (-).
MSC_SYS_SYSWPR	equ 19	Abrir o diálogo para alterar a janela atual (anterior) (-).
MSC_SYS_PRGSTA	equ 20	Abrir diálogo para carregar o aplicativo ou documento (-).
MSC_SYS_SYSSEC	equ 21	Diálogo de segurança do sistema aberto (-).
MSC_SYS_SYSQIT	equ 22	Abrir caixa de saída do sistema (-).
MSC_SYS_SYSOFF	equ 23	Desligar (-).
MSC_SYS_PRGSET	equ 24	Iniciar painel de controle (P1 – Submódulo → 0 – Janela principal, 1 – Configurações de exibição, 2 – data / hora).
MSC_SYS_PRGTSK	equ 25	Iniciar o gerenciador de tarefas (-).
MSC_SYS_SYSFIL	equ 26	Chamar função de gerenciador de arquivos (P1 – Número, P2/13 – AF, BC, DE, HL, IX, IY).
MSC_SYS_SYSHLP	equ 27	Iniciar ajuda (-).
MSC_SYS_SYSCFG	equ 28	Chamar função de configuração (P1 – Número, 0 – Carregar, 1 – Salvar, 2 – Recarregar fundo).
MSC_SYS_SYSWRN	equ 29	Abrir mensagem / janela de confirmação (P1/2 – Endereço, P3 – Banco RAM, P4 – Número de botões)

MSC_SYS_PRGSRV	equ 30	Função de serviço compartilhado (P4 – Tipo [0 – Pesquisar, 1 – iniciar, 2 – liberar], P1/2 – Endereço do ID de 12 caracteres, P3 – Endereço de 12 caracteres do Banco RAM ou ID do programa ID, se tipo = 2).
MSC_SYS_SELOPN	equ 31	Abrir o diálogo de seleção de arquivos (P6 – Banco de memória do nome do arquivo, P8/9 – Endereço do nome do arquivo, P7 – Atributos de proibição, P10 – Entradas máximas, P12 – Tamanho máximo do buffer).

### 5.13.6 – Respostas do Sistema

MSR_SYS_PRGRUN	equ 144	O aplicativo foi iniciado (P1 – Resultado → 0 – Ok, 1 – Arquivo não existe, 2 – Arquivo não é executável, 3 – Erro ao carregar [P8 – Código de erro do gerenciador de arquivos], 4 – Memória cheia, P8 – ID do aplicativo, P9 – ID do processo).
MSR_SYS_SYSFIL	equ 154	Função de gerenciador de arquivos retornada (P1 – Número, P2/13 – AF, BC, DE, HL, IX, IY)
MSR_SYS_SYSWRN	equ 157	Resposta da janela de mensagem / confirmação (P1 → 0 – Já em uso, 1 – Aberto [P2 – Número], 2 – Ok, 3 – Sim, 4 – Não, 5 – Cancelar/fechar)
MSR_SYS_PRGSRV	equ 158	Resposta da função de serviço compartilhado (P1 – Estado [5 – Não encontrado, outros códigos: consulte MSR_SYS_PRGRUN], P8 – ID do aplicativo, P9 – ID do processo)
MSR_SYS_SELOPN	equ 159	Mensagem do diálogo de seleção de arquivos (P1 → 0 – Ok, 1 – Cancelar, 2 – Já em uso, 3 – Sem memória livre, 4 – Sem janela livre, -1 = Aberto ok, janela modal foi aberta [P2–Número]).

### 5.13.7 – Comandos da área de trabalho

MSC_DSK_WINOPN	equ 32	Janela aberta (P1 – Banco RAM, P2/3 – Registro de dados de endereço).
MSC_DSK_WINMEN	equ 33	Redesenhar a barra de menus (P1 – ID da janela) [somente se em foco].
MSC_DSK_WININH	equ 34	Redesenhar o conteúdo da janela (P1 – ID da janela, P2 = -1 / -Num / Objeto, P3 – Objeto) [somente se em foco].
MSC_DSK_WINTOL	equ 35	Redesenhar a barra de ferramentas da janela (P1 – ID da janela) [somente se em foco].
MSC_DSK_WINTIT	equ 36	Redesenhar o título da janela (P1 – ID da janela) [somente se em foco].
MSC_DSK_WINSTA	equ 37	Redesenhar garantia de estado da janela (P1 – ID da janela) [somente se em foco].
MSC_DSK_WINMVX	equ 38	Definir conteúdo x deslocamento (P1 – ID da janela, P2/3 – Xpos) [somente se em foco].
MSC_DSK_WINMVY	equ 39	Definir deslocamento de conteúdo Y (P1 – ID da janela, P2/3 – Xpos) [somente se em foco].
MSC_DSK_WINTOP	equ 40	Leva a janela para a frente (P1 – ID da janela) [sempre].
MSC_DSK_WINMAX	equ 41	Maximizar janela (P1 – ID da janela) [sempre].
MSC_DSK_WINMIN	equ 42	Minimizar janela (P1 – ID da janela) [sempre].
MSC_DSK_WINMID	equ 43	Restaurar o tamanho da janela (P1 – ID da janela) [sempre]
MSC_DSK_WINMOV	equ 44	Move a janela para uma nova posição (P1 – ID da janela, P2/3 – Xpos, P4/5 – YPos) [sempre].
MSC_DSK_WINSIZ	equ 45	Redimensionar a janela (P1 – ID da janela, P2/3 – Xpos, P4/5 – YPos) [sempre].

MSC_DSK_WINCLS	equ 46	Fecha e remove janela (P1 – ID da janela) [sempre].
MSC_DSK_WINDIN	equ 47	Redesenhar o conteúdo da janela, mesmo que não tenha foco (P1 – ID da janela, P2 = -1 / -Num / Objeto, P3 – Objeto) [sempre].
MSC_DSK_DSKSrv	equ 48	Solicitação de serviço de área de tra- balho (P1 – Tipo, P2/P5 – Parâmetros).
MSC_DSK_WINSLD	equ 49	Redesenhar as barras de rolagem da janela (P1 – ID da janela) [somente se em foco].
MSC_DSK_WINPIN	equ 50	Redesenhar parte do conteúdo da janela (P1 – ID da janela, P2 = -1 / -Num / Objeto, P3 – Objeto, P4/5 – Xini, P6/7 – Yini, P8/9 – Xlen, P10/11 – Ylen) [sempre].
MSC_DSK_WINSIN	equ 51	Redesenhar o conteúdo de um super- controle (P1 – ID da janela, P2 – ID do supercontrole, P3–SubObjeto) [sempre].

### 5.13.8 – Respostas da área de trabalho

MSR_DSK_WOPNER	equ 160	A janela aberta falhou; o máximo de 32 janelas foi atingido.
MSR_DSK_WOPNOK	equ 161	Janela aberta com sucesso (P4 – Número).
MSR_DSK_WCLICK	equ 162	A janela foi clicada (P1 – Número da janela, P2 – Ação, P3 – Subespecifica- ção, P4/5, P6/7, P8/9 – Parâmetros).
MSR_DSK_DSKSrv	equ 163	Resposta do serviço de área de tra- balho (P1 – Tipo, P2/P5 – Parâmetros).
MSR_DSK_WFOCUS	equ 164	A janela obteve / perdeu o foco (P1 – Número da janela, P2 – Tipo [0 – desfoque, 1 – Foco]).
MSR_DSK_CFOCUS	equ 165	Foco de controle alterado (P1 – Número da janela, P2 – Número do controle, P3 – Razão [0 – Clique do mouse / roda, 1 – Tecla tab])

MSR_DSK_WRESIZ	equ 166	A janela foi redimensionada (P1 – Número da janela).
MSR_DSK_WSCROL	equ 167	O conteúdo da janela foi rolado (P1 – Número da janela).
MSR_DSK_EXTDSK	equ 168	Comando para área de trabalho estendida (usado internamente; P1 – Comando, P2/x – Parâmetros).
FNC_DXT_DSKBGR	equ 001	O plano de fundo foi atualizado.
FNC_DXT_FILRUN	equ 002	O arquivo foi aberto via prgrun (P2/3 – Endereço, P4 – Banco)
FNC_DXT_FILBRW	equ 003	O arquivo foi selecionado através do navegador de arquivos (P2/3 – Endereço, P4 – Banco).
FNC_DXT_MENCLK	equ 004	O menu inicial foi clicado (P2/3 – Valor).
FNC_DXT_DSKCLK	equ 005	A janela da área de trabalho foi clicada (P2 – Ação, P3 – Subespecificação, P4/5, P6/7, P8/9 – Parâmetros).

### 5.13.9 – Comandos Shell

MSC_SHL_CHRINP	equ 64	Caractere é solicitado (P1 – Canal [0 – Padrão, 1 – Teclado]).
MSC_SHL_STRINP	equ 65	A linha é solicitada (P1 – Canal [0 – Padrão, 1 – Teclado], P2 – Banco RAM, P3/4 – Endereço).
MSC_SHL_CHROUT	equ 66	Caracteres devem ser escritos (P1 – Canal [0 – Padrão, 1 – Tela] P2 – Caractere).
MSC_SHL_STROUT	equ 67	A linha deve ser escrita (P1 – Canal [0 – Padrão, 1 – Tela], P2 – Banco RAM, P3/4 – Endereço, P5 – Comprimento).
MSC_SHL_EXIT	equ 68	O aplicativo liberou o foco ou saiu sozinho (P1 → 0 – Saiu, 1 – desfocou).

### 5.13.10 – Respostas do Shell

MSR_SHL_CHRINP	equ 192	Caractere foi recebido (P1 – EOF-flag [0 – Sem EOF], P2 – caractere, P3 – Estado de erro).
----------------	---------	--

MSR_SHL_STRINP	equ 193	A linha foi recebida (P1 – EOF-flag [0 – Sem EOF], P3 – Estado de erro).
MSR_SHL_CHROUT	equ 194	Caractere foi escrito (P3 – Estado de erro)
MSR_SHL_STROUT	equ 195	Linha foi escrita (P3 – Estado de erro).

### 5.13.11 – Mensagens do protetor de tela

MSC_SAV_INIT	equ 1	Inicializa o protetor de tela (P1 – Banco de dados de configuração, P2/3 – Endereço de dados de configuração [64bytes]).
MSC_SAV_START	equ 2	Iniciar proteção de tela.
MSC_SAV_CONFIG	equ 3	Abre a janela de configuração do protetor de tela (no final, o protetor de tela deve enviar o resultado de volta ao remetente).
MSR_SAV_CONFIG	equ 4	Retorna dados de configuração de proteção de tela ajustados pelo usuário (P1 – Banco de dados de configuração, P2/3 – Endereço de dados de configuração [64bytes])

### 5.13.12 – Ações da Área de Trabalho

DSK_ACT_CLOSE	equ 5	O botão Fechar foi clicado ou ALT + F4 foi pressionado.
DSK_ACT_MENU	equ 6	A entrada do menu foi clicada (P8/9 – Valor da entrada do menu).
DSK_ACT_CONTENT	equ 14	Um controle do conteúdo foi clicado (P3 – Subespecificação [ver dsk_sub...], P4 – Chave ou P4/5 – Xpos dentro da janela, P6/7 – Ypos, P8/9 – Valor de controle )
DSK_ACT_TOOLBAR	equ 15	Um controle da barra de ferramentas foi clicado (ver DSK_ACT_CONTENT).
DSK_ACT_KEY	equ 16	A tecla foi pressionada sem tocar / modificar um controle (P4 – Código ASCII).

DSK_SUB_MLCLICK	equ 0	O botão esquerdo do mouse foi clicado.
DSK_SUB_MRCLICK	equ 1	O botão direito do mouse foi clicado.
DSK_SUB_MDCLICK	equ 2	Clique duas vezes com o botão esquerdo do mouse.
DSK_SUB_MMCLICK	equ 3	O botão do meio do mouse foi clicado.
DSK_SUB_KEY	equ 7	O teclado foi clicado e modificou / clicou em um controle (P4 – Código ASCII).
DSK_SUB_MWHEEL	equ 8	A roda do mouse foi movida (P4 – Offset).

### 5.13.13 – Serviços da Área de Trabalho

DSK_SRV_MODGET	equ 1	Obter modo de tela (Saída: P2 – Modo, P3 – Área de trabalho virtual)
DSK_SRV_MODSET	equ 2	Definir modo de tela (Entrada P2 – Modo, P3 – Área de trabalho virtual).
DSK_SRV_COLGET	equ 3	Obter cor (Entrada: P2 – Número, Saída: P2 – Número, P3/4 – Valor RGB).
DSK_SRV_COLSET	equ 4	Definir cor (Entrada: P2 – Número, P3/4 – Valor RGB)
DSK_SRV_DSKSTP	equ 5	Congelar área de trabalho (Entrada: P2 – Tipo [0 – Caneta0, 1 – Raster, 2 – Plano de fundo, 255 – Sem modificação de tela, desligue o mouse])
DSK_SRV_DSKCNT	equ 6	Continuar área de trabalho.
DSK_SRV_DSKPNT	equ 7	Limpar a área de trabalho (Entrada: P2 – Tipo [0 – Pen0, 1 – Raster, 2 – Plano de fundo]).
DSK_SRV_DSKBGR	equ 8	Inicializar e redesenhar o plano de fundo da área de trabalho
DSK_SRV_DSKPLT	equ 9	Redesenhar a área de trabalho completa.

### 5.13.14 – Jumps

jmp_memsum	equ 8100H	;MEMSUM
jmp_sysinf	equ 8103H	;SYSINF

jmp_clcnum	equ 8106H	;CLCNUM
jmp_mtgcnt	equ 8109H	;MTGCNT
jmp_timget	equ 810CH	;TIMGET
jmp_timset	equ 810FH	;TIMSET
jmp_memget	equ 8118H	;MEMGET
jmp_memfre	equ 811BH	;MEMFRE
jmp_memsiz	equ 811EH	;MEMSIZ
jmp_meminf	equ 8121H	;MEMINF
jmp_bnkrdw	equ 8124H	;BNKRDW
jmp_bnkwwd	equ 8127H	;BNKWWD
jmp_bnkrbt	equ 812AH	;BNKRBT
jmp_bnkwbtt	equ 812DH	;BNKWBTT
jmp_bnkcop	equ 8130H	;BNKCOP
jmp_bnkget	equ 8133H	;BNKGET
*empty*	equ 8136H	; *empty*
jmp_scrget	equ 8139H	;SCRGET
jmp_mosget	equ 813CH	;MOSGET
jmp_moskey	equ 813FH	;MOSKEY
jmp_bnk16c	equ 8142H	;BNK16C
jmp_keytst	equ 8145H	;KEYTST
jmp_keysta	equ 8148H	;KEYSTA
jmp_keyput	equ 814BH	;KEYPUT
jmp_bufput	equ 814EH	;BUFPUT
jmp_bufget	equ 8151H	;BUFGET
jmp_bufsta	equ 8154H	;BUFSTA
jmp_iominp	equ 8157H	;IOMINP (cpc only)
jmp_iomout	equ 815AH	;IOMOUT (cpc only)
jmp_bnkcll	equ FF03H	;BNKCLL
jmp_bnkret	equ FF00H	;BNKRET

#### 5.13.15 – Funções de Gerenciador de Arquivos (chamar via MSC\_SYS\_SYSFIL)

FNC_FIL_STOINI	equ 000
FNC_FIL_STONEW	equ 001
FNC_FIL_STORLD	equ 002
FNC_FIL_STODEL	equ 003
FNC_FIL_STOINP	equ 004

FNC_FIL_STOOUT	equ 005
FNC_FIL_STOACT	equ 006
FNC_FIL_STOINF	equ 007
FNC_FIL_STOTRN	equ 008
FNC_FIL_DEVDIR	equ 013
FNC_FIL_DEVINI	equ 014
FNC_FIL_DEVSET	equ 015
FNC_FIL_FILINI	equ 016
FNC_FIL_FILNEW	equ 017
FNC_FIL_FIOPN	equ 018
FNC_FIL_FILCLO	equ 019
FNC_FIL_FILINP	equ 020
FNC_FIL_FILOUT	equ 021
FNC_FIL_FILPOI	equ 022
FNC_FIL_FILF2T	equ 023
FNC_FIL_FILT2F	equ 024
FNC_FIL_FILLIN	equ 025
FNC_FIL_DIRDEV	equ 032
FNC_FIL_DIRPTH	equ 033
FNC_FIL_DIRPRS	equ 034
FNC_FIL_DIRPRR	equ 035
FNC_FIL_DIRREN	equ 036
FNC_FIL_DIRNEW	equ 037
FNC_FIL_DIRINP	equ 038
FNC_FIL_DIRDEL	equ 039
FNC_FIL_DIRRMD	equ 040
FNC_FIL_DIRMOV	equ 041
FNC_FIL_DIRINF	equ 042

## 6 – UZIX

### 6.1 – COMANDOS

#### 6.1.1 – Convenções Usadas

**NOME DO COMANDO** (tipo do comando)

Formato: Formatos válidos para o comando

Função: Forma de operação do comando

Detalhes: Descreve alguns detalhes sobre o formato

Os comandos do Uzix são todos carregados do disco. Nesse guia estão descritos todos os comandos e utilitários que são instalados por padrão no UZIX 2.0.

#### 6.1.1.1 – Notações de Formato

<nomearq>

Nome de arquivo na forma: dir1/dir2/arquivo

<nomearqs>

Vários nomes de arquivo na forma: dir1/dir2/arquivo

<nomedir>

Nome de diretório na forma: /dir1/dir2/

[ ] Delimita parâmetro opcional.

| Significa que apenas um dos itens pode ser utilizado.

Um <dispositivo> pode ser:

fd0~fd7 Drives de disquete.

null Dispositivo nulo.

lpr Impressora.

tty/tty0~tty2 Monitor.

console Teclado.

mem/kmem Memória.

sga0~sga(n) Partições em disco rígido.

sge(n) Partição em disco rígido onde está o UZIX.

### **6.1.2 – Descrição dos Comandos**

#### **ADDUSER (Utilitário de Administração)**

Formato: adduser

Função: Adiciona um usuário ao sistema.

#### **ALIAS (Utilitário Shell)**

Formato: alias [<nome> [<comando> [<comando> ...]]]

Função: Apresenta ou define um comando alias.

#### **BANNER (Utilitário Uzix)**

Formato: banner <mensagem>

Função: Imprime uma mensagem em caracteres grandes.

#### **BASENAME (Utilitário Shell)**

Formato: basename <nome> [sufixo]

Função: Remove orientação de componentes de um diretório.

#### **BOGOMIPS (Utilitário de Sistema)**

Formato: bogomips

Função: Imprime a velocidade de processamento em BogoMips.

#### **CAL (Utilitário Uzix)**

Formato: cal [mês] ano

Função: Apresenta um calendário.

#### **CAT (Utilitário de Arquivos)**

Formato: cat <nomearqs>

Função: Concatena arquivos e imprime na saída padrão.

#### **CD (Utilitário de Arquivos)**

Formato: cd [<nomedir>]

Função: Troca diretórios.

#### **CDIFF (Utilitário de Texto)**

Formato: cdiff [-c n] <arq1> <arq2>

Função: Imprime a diferença entre dois arquivos com contexto.

Detalhes: [-c] Produz uma saída contendo n linhas de contexto.

**CGREP** (Utilitário de Texto)

Formato: `cgrep [-a n] [-b n] [-f] [-l n] [-n] [-w n] <padrão> [<arqs>...]`

Função: Procura uma string e imprime as linhas onde forem encontradas.

Detalhes: `[-a]` Número de linhas a apresentar após a linha encontrada

`[-b]` Número de linhas a apresentar antes da linha encontrada

`[-f]` Suprime nome de arquivo na saída.

`[-l]` Trunca linhas no tamanho n antes da comparação.

`[-n]` Suprime números de linha na saída.

`[-w]` Define o tamanho da janela (mesmo que `-a` e `-b`)

**CHGRP** (Utilitário de Arquivo)

Formato: `chgrp <gid> <nomearq>`

Função: Troca o usuário proprietário do grupo para cada arquivo.

**CHMOD** (Utilitário de Arquivo)

Formato: `chmod <modo_ascii> | <modo_octal> <nomearqs>`

Função: Troca as permissões de acesso aos arquivos.

Detalhes: O formato simbólico (ASCII) para o modo é o seguinte:

`[ugoा] [+ | -] [rwx]`, onde

`u → usuário    a → todos    x → gravação`

`g → grupo    r → leitura    + → adiciona permissão`

`o → outros    w → escrita    - → remove permissão`

O formato numérico (octal) é o seguinte:

1º dígito octal: 1 – salva imagem texto dos atributos

2 – ID de grupo

4 – ID de usuário

2º dígito octal: 1 – execução

2 – escrita

4 – leitura

**CHOWN** (Utilitário de arquivo)

Formato: `chown <uid> <nomearq>`

Função: Troca o usuário comum e o usuário proprietário do grupo para o arquivo especificado.

**CHROOT** (Utilitário de Arquivo)

Formato: chroot <nomedir>

Função: Troca o diretório raiz.

**CKSUM** (Utilitário de Arquivo)

Formato: cksum [<nomearq> [<nomearq ...]]

Função: Apresenta o checksum e o tamanho do arquivo.

**CLEAR** (Utilitário Shell)

Formato: clear

Função: Limpa a tela.

**CMP** (Utilitário de Arquivo)

Formato: cmp <nomearq1> <nomearq2>

Função: Compara arquivos.

**CRC** (Utilitário de Arquivo)

Formato: crc [<nomearq> [<nomearq ...]]

Função: Apresenta o checksum dos dados do arquivo.

**CP** (Utilitário de Arquivo)

Formato: cp [-pifsmrRvx] <nomearq1> <nomearq2>

          cp [-pifsrRvx] <nomearq1> [<nomearq2>...] <dir>

Função: Copia arquivos.

Detalhes: [-p] Preserva todos os atributos do arquivo original.

          [-i] Verifica se há arquivo com o mesmo nome no destino.

          [-f] Remove arquivos no destino.

          [-s] Copia apenas alguns atributos.

          [-m] Copia vários subdiretórios para apenas um.

          [-r] Copia diretórios recursivamente.

          [-R] Copia diretórios e trata arquivos especiais como  
              ordinários.

          [-v] Apresenta o nome dos arquivos antes de copiar.

          [-x] Pula diretórios que estão em sistemas de arquivo  
              diferentes de onde a cópia começou.

**CPDIR** (Utilitário de Arquivo)

Formato: cpdir [-ifvx] <nomedir1> <nomedir2>

Função: Copia diretórios.

Detalhes: [-i] Verifica se há arquivo com o mesmo nome no destino  
 [-f] Remove arquivos no destino  
 [-v] Apresenta o nome dos arquivos antes de copiar  
 [-x] Pula subdiretórios que estão em sistemas de arquivo diferentes de onde a cópia começou.

## **DATE** (Utilitário Uzix)

Formato: date

Função: Apresenta a data e a hora correntes do sistema.

## **DD** (Utilitário de Arquivo)

Formato: dd [if=<nomearq>] [of=<nomearq>] [ibs=<bytes>]  
 [obs=<bytes>] [bs=<bytes>] [cbs=<bytes>]  
 [files=<número>] [skip=<blocos>] [seek=<blocos>]  
 [count=<blocos>] [conv={ascii | ebcdic  
 ibm | lcase | ucase | swab | noerror | sync}]

Função: Copia arquivo convertendo o mesmo.

Detalhes: [if=<nomearq>] Lê de arquivo  
 [of=<nomearq>] Escreve para arquivo  
 [ibs=<bytes>] Lê <bytes> bytes por vez  
 [obs=<bytes>] Escreve <bytes> bytes por vez  
 [bs=<bytes>] Lê e escreve <bytes> bytes por vez  
 [cbs=<bytes>] Converte <bytes> bytes por vez  
 [files=<núm.>] Copia <núm.> arquivos  
 [skip=<blocos>] Pula <blocos> blocos de tamanho “bs”  
 no início da entrada  
 [seek=<blocos>] Pula <blocos> blocos de tamanho “bs”  
 no início da saída  
 [count=<blocos>] Copia somente <blocos> de tamanho “bs”  
 na entrada  
 conv=conversão[,conversão...]] – converte o arquivo de  
 acordo com os seguintes argumentos:

ascii	Converte de EBCDIC para ASCII.
ebcdic	Converte de ASCII para EBCDIC.
ibm	Converte de ASCII para EBCDIC alternativo.
lcase	Converte todos os caracteres para minúsculos.
ucase	Converte todos os caracteres para maiúsculos.
swab	Permuta um par de bytes entrados.
noerror	Continua após detectar algum erro.
sync	Completa um bloco “bs” com bytes 00H.

**DF** (Utilitário de Arquivo)

Formato: df [-ikn]

Função: Apresenta o espaço livre em disco em unidades de 512 bytes.

Detalhes: [-i] Lista informações usadas pelos inodes.

[-k] Apresenta em unidades de 1 Kbyte.

[-n] Não acessa /etc/mtab para obter informações.

**DHRY** (Utilitário de Sistema)

Formato: dhry

Função: Apresenta a velocidade de processamento em dhystones.

**DIFF** (Utilitário de Texto)

Formato: diff [-c | -e | -C n] [-br] <nomearq1> <nomearq2>

Função: Imprime a diferença entre dois arquivos

Detalhes: [-C n] Produz uma saída contendo n linhas de contexto.

[-b] Ignora espaços em branco na comparação.

[-c] Produz uma saída contendo 3 linhas de contexto.

[-e] Produz um “ed-script” para converter.

[-r] Aplica diff recursivamente.

**DIRNAME** (Utilitário Shell)

Formato: dirname <nomearq>

Função: Imprime o sufixo de um nome de arquivo.

**DOSDEL** (Utilitário Uzix)

Formato: dosdel <drivedos><nomearqdos>

Função: Apaga um arquivo em discos MSXDOS.

**DOSDIR** (Utilitário Uzix)

Formato: dosdir [-lr] <drivedos>

Função: Lista arquivos de um disco MSXDOS.

Detalhes: [-l] Listagem longa.

[-r] Imprime subdiretórios de forma recursiva e descendente.

**DOSREAD** (Utilitário Uzix)

Formato: dosread [-a] <drivedos><nomearqdos> [<nomearquzix>]

Função: Lê um arquivo de um disco MSXDOS.

Detalhes: [-a] Arquivo ASCII.

**DOSWRITE** (Utilitário Uzix)

Formato: doswrite [-a] <drivedos><nomearqdos> [<nomearquizx>]

Função: Escreve um arquivo em um disco MSXDOS.

Detalhes: [-a] Arquivo ASCII.

**DU** (Utilitário Uzix)

Formato: du [-as] [-l n] <nomedir> ...

Função: Apresenta o espaço ocupado por diretórios e subdiretórios

Detalhes: [-a] Apresenta o espaço usado por todos os arquivos.

[-s] Apenas sumário.

[-l] Lista n níveis de subdiretórios.

**ECHO** (Utilitário Shell)

Formato: echo [-ne] [<string> [<string>...]]

Função: Apresenta uma linha de texto

Detalhes: [-n] Não alimenta linha ao final do texto

[-e] Habilita interpretação dos seguintes caracteres:

\a alerta (campainha).

\b backspace.

\c suprime alimentação de linha.

\f avanço de formulário.

\n nova linha.

\r retorno de carro (return).

\t tabulação horizontal.

\v tabulação vertical.

\ \ ignora espaço no texto entre \\ (backslash).

\nnn apresenta caractere de código ASCII nnn (octal).

\xnn apresenta caractere de código ASCII nn (hex).

**ED** (Utilitário de Texto)

Formato: ed [-Ghs] [-p string] [arquivo]

Função: Executa um editor de texto padrão.

-G Força retrocompatibilidade.

-h Apresenta a ajuda do programa.

-s Suprime diagnósticos.

-p Especifica um prompt de comando.

**EXIT** (Utilitário de Administração)

Formato: exit [<status>]

Função: Sai da sessão atual.

**FALSE** (Utilitário Shell)

Formato: false

Função: Não faz nada; simplesmente retorna com estado de erro “1”.

**FGREP** (Utilitário de Texto)

Formato: fgrep [-cfhlnsv] [<arquivo\_string>] [<string>] <nomearq>...

Função: Procura uma string e imprime as linhas onde for econtrada.

Detalhes: [-c] Imprime apenas a quantidade de linhas.

[-f] Procura string no arquivo <nomearq>.

[-h] Omite cabeçalhos de arquivo da saída.

[-l] Lista nomes de arquivo apenas uma vez.

[-n] Imprime números de linha para cada linha.

[-s] Apenas status.

[-v] Imprime apenas linhas sem a <string>.

**FILE** (Utilitário Uzix)

Formato: file <nomearq> [<nomearq>...]

Função: Faz uma suposição sobre qual tipo o arquivo é.

**FLD** (Utilitário de Texto)

Formato: fld -u -z\* -[b t s? i? fm1.n1,m2.n2] {<arq\_entrada> [<arq\_saida>]}

Função: Lê e concatena campos de um arquivo

Detalhes: [-?] Mostra ajuda. Mesmo que [-h].

-u Descompacta tabs

-p] Compacta tabs

-z\* Pula os primeiros \* espaços

-[b] Pula os espaços iniciais do campo

-[t] Remove espaços excessivos do campo

-[s?] Separador de campos na saída será “?”

-[i?] Separador de campos na entrada será “?”

-[fm1.n1,m2.n2] Definição de campo

    m1.n1 = início do campo e m2.n2 = fim do campo,  
    onde m = nº de campos e n = nº de caracteres.

-[f#] Pega o campo da entrada do usuário

**FORTUNE** (Utilitário Uzix)

Formato: fortune

Função: Imprime, aleatoriamente, um provérbio.

**GREP (Utilitário de Texto)**

Formato: grep -cnfv {‐p<padrão>} <nomearqs>

Função: Procura uma string e imprime as linhas onde for encontrada.

Detalhes: [‐c] Imprime apenas a quantidade de linhas.

[‐f] Imprime nomes de arquivos.

[‐n] Imprime números de linha para cada linha

[‐v] Imprime apenas linhas sem a <string>

[‐p] Define a string (padrão). Os seguintes caracteres de controle podem ser usados:

x caractere ordinário.

\ quota qualquer caractere.

^ início de linha.

\$ fim de linha.

. qualquer caractere.

:l minúsculas.

:u maiúsculas.

:a alfabéticos.

:d dígitos (numéricos).

:n alfanuméricos.

:r caracteres russos.

:s espaço.

:t tabulação.

:c caracteres de controle (exceto LF e TAB).

:e inicia sub-expressão.

\* repete zero ou mais.

+ repete um ou mais.

- opcionalmente procura a expressão.

[..] qualquer destes (na faixa DE-PARA).

[^..] qualquer exceto estes.

\nnn valor numérico (estilo C).

**HEAD (Utilitário de Texto)**

Formato: head [‐n] [<nomearqs> ...]

Função: Imprime as primeiras linhas do arquivo.

Detalhes: [‐n] número de linhas a imprimir (o padrão é 10).

**HELP (Utilitário Uzix)**

Formato: help

Função: Imprime alguns comandos com o respectivo formato.

**INIT** (Utilitário de Administração)

Formato: /bin/init

Função: Controle de inicialização de processos..

**KILL** (Utilitário Uzix)

Formato: kill [-sinal] pid [pid...]

Função: Termina processos do sistema.

Detalhes: [-sinal] é um sinal a ser enviado para um processo que está sendo executado (ex. HUP, INT, QUIT, KILL ou 9).

**LOGIN** (Utilitário de Administração)

Formato: login <nomeusuário>

Função: Inicia uma sessão.

**LN** (Utilitário de Texto)

Formato: ln [-ifsSmrRvx] <nomearq1> <nomearq2>

In [-ifsSrRvx] <nomearq> [<nomearq>...] <nomedir>

Função: Adiciona links entre arquivos.

Detalhes: [-i] Avisa antes de remover arquivos destino existentes.

[ -f] Remove arquivos destino existentes.

[ -s] Adiciona link simbólico.

[ -S] Adiciona link simbólico enquanto tenta link normal.

[ -m] Intercala árvores.

[ -r] Adiciona link recursivo para diretórios.

[ -R] Mesmo que [-r].

[ -v] Imprime o nome do arquivo antes de adicionar link.

[ -x] Pula subdiretórios que estão em sistemas de arquivo diferentes de onde a adição de links começou.

**LOGOUT** (Utilitário Uzix)

Formato: logout

Função: Encerra uma sessão.

**LS** (Utilitário de Arquivo)

Formato: ls [-1ACFLRacdfgiklqrstu] [<nomearq> [<nomearq>...]]

Função: Lista o conteúdo de diretórios.

Detalhes: [-1] Usa apenas uma coluna na saída.

[ -A] Lista todos os arquivos, exceto ":" e "..".

- [**-C**] Ordena arquivos na listagem (em colunas).
- [**-F**] Não identifica o tipo de arquivo.
- [**-L**] Lista os arquivos pelos links simbólicos.
- [**-R**] Lista o conteúdo dos diretórios recursivamente.
- [**-a**] Lista todos os arquivos, inclusive “.” e “..”.
- [**-c**] Ordena arquivos de acordo com a data de alteração.
- [**-d**] Lista diretórios como outros arquivos.
- [**-f**] Não ordena arquivos e diretórios.
- [**-g**] Imprime o nome do usuário proprietário do grupo.
- [**-i**] Imprime o número do inode dos arquivos.
- [**-k**] Imprime o tamanho dos arquivos em Kbytes.
- [**-l**] Imprime os atributos dos arquivos.
- [**-q**] Imprime interrogações no lugar de caracteres especiais.
- [**-r**] Ordena arquivos e diretórios em ordem inversa.
- [**-s**] Imprime o tamanho dos arquivos em bytes.
- [**-t**] Ordena arquivos de acordo com a data de criação.
- [**-u**] Ordena arquivos de acordo com a data do último acesso.

### **MAN** (Utilitário de Sistema)

Formato: man –wqv [seção] <nomecomando>

Função: Apresenta o manual on-line.

Detalhes: **-w** Apresenta apenas o manual com seção/nome exatos.

- q** Modo silencioso, para comandos formatadores defeituosos.
- v** Modo de apresentação formatada (verbose).

### **MKDIR** (Utilitário de Arquivo)

Formato: mkdir [**-p**] [**-m** <modo>] <nomedir>

Função: Criar diretórios.

Detalhes: [**-p**] Cria diretórios-pai (parents) de acordo com a máscara.

- [-m]** Define o modo (0666 menos os bits de umask).

### **MKNOD** (Utilitário de Arquivo)

Formato: mknod [**-m** <modo>] <nomearq> {b | c | u} <maior> <menor>

Função: Cria arquivos especiais

Detalhes: [**-m**] Define o modo.

b Arquivo bufferizado (bloco).

c / u Arquivo não bufferizado (caractere).

**MORE** (Utilitário Uzix)

Formato: more <nomearqs>

Função: Utilitário de paginação.

Detalhes: Quando o prompt estiver presente, usar as seguintes teclas:  
 espaço Apresenta a próxima página.  
 return Apresenta a próxima linha.  
 n Vai para o próximo arquivo, se existir.  
 p Vai para o arquivo anterior, se existir.  
 q Abandona o comando more.

**MOUNT** (Utilitário Uzix)

Formato: mount [-r] <dispositivo> <caminho>

Função: Monta o <dispositivo> no <caminho> especificado.

Detalhes: [-r] Monta no modo somente-leitura.

**MV** (Utilitário de Arquivo)

Formato: mv [-isfmvx] <nomearq1> <nomearq2>

mv [-ifsvx] <nomearq> [<nomearq> ...] <nomedir>

Função: Renomeia ou move arquivos.

Detalhes: [-i] Avisa antes de sobrescrever arquivos com mesmo nome.

[-f] Remove arquivos-destino existentes.

[-s] Cria link simbólico e não move o arquivo.

[-m] Intercala diretórios sem procurar diretório alvo.

[-v] Imprime o nome dos arquivos antes de mover.

[-x] Pula subdiretórios que estão em sistemas de arquivo diferentes de onde o movimento de arqs começou.

**PASSWD** (Utilitário de Administração)

Formato: passwd [<login>]

Função: Troca a senha do usuário.

**PROMPT** (Utilitário Shell)

Formato: prompt <string>

Função: Altera o prompt do Uzix.

**PS** (Utilitário Uzix)

Formato ps [-] [lusmahrn]

Função: Imprime um relatório do estado do processo.

Detalhes: [-l] Formato longo.  
 [-u] Formato usuário (nome do usuário e hora inicial).  
 [-s] Formato sinal.  
 [-m] Informação sobre memória.  
 [-a] Apresenta processos de outros usuários também.  
 [-h] Sem cabeçalho.  
 [-r] Somente processos em execução.  
 [-n] Saída numérica para usuário.

### **PWD** (Utilitário Shell)

Formato: pwd

Função: Imprime o caminho do diretório de trabalho atual.

### **QUIT** (Utilitário de Administração)

Formato: quit

Função: Encerra a sessão atual.

### **REBOOT** (Utilitário de Administração)

Formato: reboot

Função: Reseta o computador.

### **RM** (Utilitário de Arquivo)

Formato: rm <nomearq>

Função: Remove arquivos.

### **RMDIR** (Utilitário de Arquivo)

Formato: rmdir [-p] <nomedir>

Função: Remove diretórios.

Detalhes: [-p] Remove diretório-pai se estiver vazio depois da remoção do diretório especificado.

### **SASH** (Utilitário tipo Aplicativo)

Formato: sash

Função: É um tipo de shell com comandos internos.

### **SET** (Utilitário de Administração)

Formato: [<nome> [<valor>]]

Função: Apresenta ou define variáveis de ambiente.

**SLEEP** (Utilitário de Administração)

Formato: sleep [<segundos>]

Função: Faz o sistema “dormir” por <segundos> segundos.

**SU** (Utilitário de Administração)

Formato: su [<nomeusuário>]

Função: Conecta temporariamente como superusuário ou outro usuário.

**SOURCE** (Utilitário Uzix)

Formato: source <nomearq>

Função: Apresenta o “fonte” do arquivo.

**SUM** (Utilitário de Arquivo)

Formato: sum [<nomearq> [<nomearq>...]]

Função: Analiza a checksum e o contador de blocos do arquivo.

**SYNC** (Utilitário de Programação)

Formato: sync

Função: Descarrega os buffers do sistema de arquivos.

**TAIL** (Utilitário de Texto)

Formato: tail [-c n | -n n] [-f] [<nomearq> [<nomearq>]]

Função: Imprime as últimas linhas de um arquivo.

Detalhes: [-c] Imprime n caracteres.

[-f] Em FIFO ou arquivo especial, ler depois de EOF.

[-n] Imprime n linhas.

**TAR** (Utilitário de Arquivo)

Formato: tar [cxt] [voFfpD] <nomearqtape> [<nomearq> [<nomearq>...]]

Função: Concatena/extrai arquivos para armazenagem.

Detalhes: [c] Cria novo arquivo tar.

[x] Extrai arquivos do arquivo tar.

[t] Lista o conteúdo do arquivo tar.

[v] Modo verbose.

[o] Define usuário e proprietário originais na extração.

[F] Ignora erros.

- [f] Próximo argumento é o nome do arquivo tar.
- [p] Restaura modos do arquivo, ignora máscara.
- [D] Não adiciona diretórios recursivamente.

### **TEE** (Utilitário Shell)

Formato: tee <nomearq>

Função: Lê da entrada padrão e escreve em um arquivo.

### **TIME** (Utilitário Uzix)

Formato: time <comando> [<argumento do comando>]

Função: Executa o comando e imprime a hora real, a hora do usuário e a hora do sistema (horas–minutos–segundos).

### **TOP** (Utilitário Uzix)

Formato: top [-d <atraso>] [-q] [-s] [-i]

Função: Lista os processos mais ativos.

Detalhes: [-d] Especifica o tempo para atualização da tela.

[-q] Especifica atualização sem atraso algum.

[-s] Modo seguro (desativa comandos interativos).

[-i] Ignora processos ociosos.

### **TOUCH** (Utilitário de Arquivo)

Formato: touch [-c] [-d <hora/data>] [-m] <nomearq>

Função: Troca a hora e a data dos arquivos.

Detalhes: [-c] Não cria arquivos que não existem.

[-d] Troca conforme <hora/data> ao invés de usar a hora/ data atual. Formato: HH:MM:SS DD:MM:AA.

[-m] Altera apenas a hora/data de modificação do arquivo

### **TR** (Utilitário de Texto)

Formato: tr from to [+<início>] [-<fim>] [<arqentrada> [<arqsaída>]]

Função: Troca os caracteres de um arquivo (translitera).

Detalhes: Seqüências de escape:

:z	Faixa vazia	:	a	Mesmo que a-zA-Z
:l	Mesmo que a-z	:	u	Mesmo que A-Z
:m	Mesmo que á-ô	:	b	Mesmo que Ç-f
:r	Mesmo que á-ôÇ-f	:	d	Mesmo que 0-9
:n	Mesmo que a-zA-Z0-9	:	s	Mesmo que \001-\040
:	Toda a faixa ASCII menos \0			

**TRACE** (Utilitário Uzix)

Formato: trace {on}

Função: Modo trace?

**TRUE** (Utilitário Shell)

Formato: true

Função: Não faz nada, somente retorna com status de erro 0.

**UMOUNT** (Utilitário Uzix)

Formato: umount <dispositivo>

Função: Desmonta sistema de arquivos do dispositivo especificado.

**UMASK** (Utilitário Uzix)

Formato: umask [<máscara>]

Função: Remove máscaras.

**UNALIAS** (Utilitário Shell)

Formato: unalias <nome>

Função: Remove um comando tipo alias.

**UNAME** (Utilitário Shell)

Formato: uname [-snrvma]

Função: Imprime informações sobre o sistema.

Detalhes: [-m] Imprime tipo de máquina.

[ -n ] Imprime nome da máquina cliente na rede.

[ -r ] Imprime distribuição do sistema operacional.

[ -s ] Imprime nome do sistema operacional.

[ -v ] Imprime versão do sistema operacional.

[ -a ] Imprime todos os itens acima.

**UNIQ** (Utilitário de Texto)

Formato: uniq [-cduzN.M+L] [-<campos>] [+<letras>] [<nomearq>]

Função: Remove linhas duplicadas em arquivos ordenados.

Detalhes: [-u] Somente imprime linhas não repetidas.

[ -d ] Somente imprime linhas duplicadas.

[ -c ] Imprime o número de vezes que a linha é repetida.

[ -z ] Mesmo que -c, mas imprime em números octais.

[ -N.M ] Pula N palavras e M letras.

[ +L ] Compara somente L letras.

**WC (Utilitário de Texto)**

Formato: `wc [-bhpw] [<nomearq>]`

Função: Imprime o número de bytes, palavras e linhas de um arquivo.

Detalhes: `[-b]` Abre arquivo no modo binário

`[-h]` Apresenta a ajuda do programa

`[-p]` Contagem de páginas

`[-w]` Encontra a largura máxima de linha

**WHOAMI (Utilitário Shell)**

Formato: `whoami`

Função: Imprime o nome do usuário associado com o ID do usuário atual.

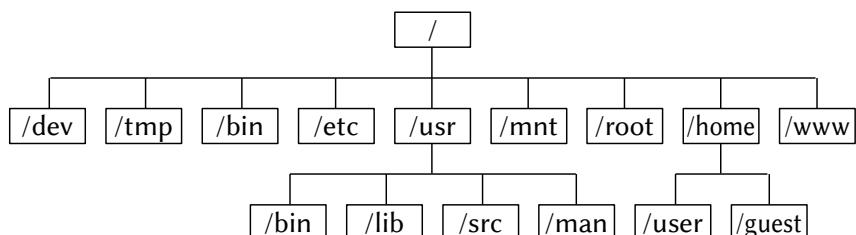
**YES (Utilitário Shell)**

Formato: `yes [<string>]`

Função: Imprime “y” ou `<string>` repetidamente na saída padrão.

## 6.2 – ESTRUTURA HIERÁRQUICA

No Uzix há uma estrutura pré-definida de subdiretórios. Essa estrutura pode ser modificada pelo usuário, mas não é aconselhável fazê-lo porque ela é padrão no mundo Unix. Essa estrutura é a seguinte:



Cada um desses subdiretórios tem um uso específico, mas não obrigatório. A descrição de cada um está abaixo.

/ diretório raiz

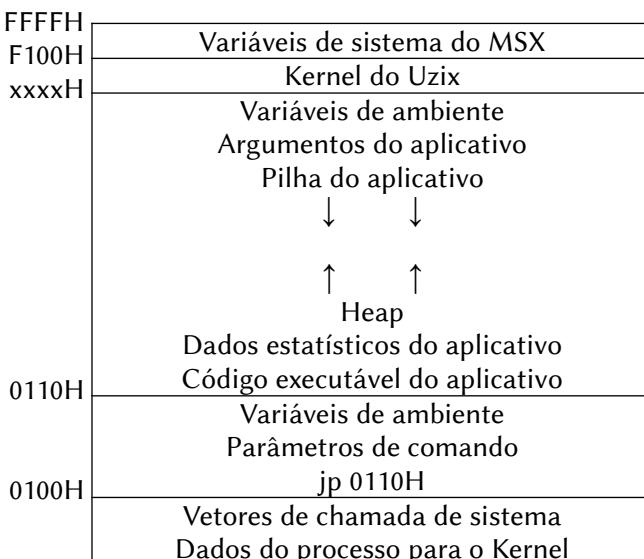
/dev contém os nomes arquivos especiais associados a dispositivos de hardware ou software.

/tmp Usado por todo o sistema para a criação de arquivos temporários.

/bin	Contém as aplicações mais genéricas do sistema.
/etc	Arquivos usados para administrar o sistema.
/usr	Arquivos gerais do sistema. Esse subdiretório contém mais 4 subdiretórios: /bin Aplicações genéricas /lib Bibliotecas /src Códigos-fonte /man Manuais do sistema (arquivos-texto).
/mnt	Usado como ponto de conexão de um sistema de arquivo de outro dispositivo. Também usado para montagem (mount).
/root	Diretório de trabalho do administrador do sistema.
/home	Usado pelos usuários comuns como área de trabalho.
/user	Usuário “user”.
/guest	Usuário “guest”.
/www	Arquivos de internet.

### 6.3 – MAPEAMENTO DE MEMÓRIA

O mapeamento de memória é a maior diferença entre o Uzix 1.0 e 2.0. Ele está ilustrado abaixo, onde xxxxH é 8000H para o Uzix 1.0 e C000H para o 2.0.



O Uzix 1.0 fica inteiramente residente na área alta de memória, a partir do endereço 8000H. Todo processo sempre ocupa 32 Kbytes de memória. Já o Uzix 2.0 tem uma parte residente na página 3 (a partir de C000H) e faz as chamadas adicionais a partir daí. Cada processo pode ter 16K, 32K ou 48K.

## 6.4 – CHAMADAS DE SISTEMA

O Uzix é um sistema operacional para o MSX que implementa as funcionalidades do AT&T Unix Version 7. É um sistema multiusuário e implementa multitarefa preemptiva, oferecendo ainda infra-estrutura de rede (TCP/IP). Entretanto, os seguintes cuidados devem ser tomados:

- NUNCA devem ser usadas as instruções DI e EI;
- NUNCA deve ser feito acesso direto ao hardware;
- NUNCA devem ser acessados dados abaixo de 0100H ou acima da aplicação.

Para fazer uma chamada de sistema é necessário empilhar os parâmetros na ordem inversa da declaração, depois o número da chamada e então fazendo um CALL 08H. É responsabilidade da aplicação desempilhar os parâmetros apás o CALL. O valor de retorno, de 16 bits, é colocado no registrador DE. A única exceção é a chamada lseek, cujo valor de retorno é de 32 bits e é colocado em HL:DE (HL é a palavra mais significativa). A tabela abaixo lista as chamadas diretas, seus parâmetros e número de chamada.

### 6.4.1 – Chamadas Diretas de Sistema

**ACCESS (#00)** – Determina o nível de acesso de um arquivo.

Sintaxe:    err = access (path, mode)

      int     err

      char    \*path

      int     mode

Entrada: path: String apontando para o arquivo a ser analisado.

          mode: 0 – Testa se o arquivo existe e pode ser pesquisado.

          1 – executar

          2 – escrever

          4 – ler

Saída:      err:    0 → Teste bem-sucedido (se mode = 0).  
               -1 → Erro (código de erro em errno).

Assembler: (access = 33.)  
               sys access; name; mode

### **ALARM (#01) – Agenda um sinal após um tempo especificado.**

Sintaxe:    time = alarm (secs)

              int     time

              int     secs

Entrada:    secs:   Tempo em segundos (máximo 32 767).

Saída:       time:   Tempo anterior restante no alarme.

Assembler: (alarm = 27.)  
               (seconds in r0)  
               sys alarm  
               (previous amount in r0)

Nota:       Faz com que o sinal SIGALRM seja enviado para o processo chamador depois de um número de segundos dado pelo argumento. A menos que seja capturado ou ignorado, o sinal termina o processo. O valor de retorno é a quantidade de tempo restante anteriormente.

### **BRK (#02) – Altera a alocação do núcleo.**

Sintaxe:    err = brk (addr)

              int     err

              char   \*addr

Entrada:    addr: Endereço.

Saída:       err:    0 → Comando executado com sucesso.  
               -1 → O programa precisa de mais memória que o limite do sistema ou estouro do número máximo de registros de segmentação.

Assembler: (brk = 17.)  
               sys break; addr

Nota:       Define, para o sistema, o local mais baixo não usado pelo programa (chamado de intervalo) para addr. Normalmente apenas programas com crescimento cujas áreas de dados aumentam precisam usar break.

**CHDIR (#03)** – Altera diretório padrão.

Sintaxe: err = chdir (path)

int err

char \*path

Entrada: path: String do diretório a ser definido.

Saída: err: 0 → Comando executado com sucesso.

-1 → “path” não é um diretório ou não é pesquisável.

Assembler: (chdir = 12.)

sys chdir; dirname

**CHMOD (#04)** – Altera atributos de um arquivo

Sintaxe: err = chmod (path, mode)

int err

char \*path

int mode

Entrada: path: String apontando para o arquivo a ser alterado.

mode: resultante de um OR combinando os seguintes valores:

04 000 Definir ID do usuário na execução.

02 000 Definir ID de grupo na execução.

01 000 Salvar imagem de texto após a execução.

00 400 Pode ser lido pelo proprietário.

00 200 Pode ser escrito pelo proprietário.

00 100 Executar (pesquisar no diretório) pelo proprietário.

00 070 Ler, escrever, executar (pesquisar) por grupo.

00 007 Ler, escrever, executar (pesquisar) por outros.

Saída: err: 0 → Comando executado com sucesso.

-1 → Arquivo não localizado ou usuário sem permissão de acesso.

Assembler: (chmod = 15.)

sys chmod; name; mode

**CHOWN (#05)** – Troca usuário e grupo de um arquivo.

Sintaxe: err = chown (path, owner, group)

int err

char \*path

int owner

int group

Entrada: path: String apontando para o arquivo.  
 owner: Novo usuário do arquivo.  
 group: Novo grupo do arquivo.

Saída: err: 0 → Usuário trocado.  
 -1 → Troca ilegal de usuário.

Assembler: (chown = 16.)  
 sys chown; name; owner; group

### **CLOSE (#06) – Fecha um arquivo.**

Sintaxe: err = close(path)  
 int err  
 char \*path

Entrada: path: String apontando para o arquivo a ser fechado.

Saída: err: 0 → Arquivo fechado com sucesso.  
 -1 → Descritor de arquivo desconhecido.

Assembler: (close = 6.)  
 (file descriptor in r0)

### **GETSET (#07) – Implementa chamadas que leem ou alteram valores de variáveis de sistema.**

Sintaxe: var = getset(operação, ...)  
 int var  
 operação → depende da função chamada.

Entrada: getset(0) → getpid(void)  
 getset(1) → getppid(void)  
 getset(2) → getuid(void)  
 getset(3,uid) → setuid (uid)  
                   int uid  
 getset(4) → geteuid(void)  
 getset(5) → getgid(void)  
 getset(6,gid) → setgid(int gid)  
 getset(7) → getegid(void)  
 getset(8) → getprio(void)  
 getset(9,pid,prio) → setprio(pid, prio)  
                   int pid  
                   char prio  
 getset(10) → umask(mask)  
                   int mask  
 getset(11,onoff) → systrace(onoff)  
                   int onoff

Saída: Depende da função chamada.  
 Assembler: Depende da função chamada.

**DUP (#08)** – Duplica um descriptor de arquivo aberto.

Sintaxe: newd = dup(oldd)  
           int    newd  
           int    oldd

Entrada: oldd: Descritor de arquivo antigo.  
 Saída: newd: Novo descritor de arquivo.  
           –1 se o descritor for inválido ou se já houver  
           muitos arquivos abertos.

Assembler: (dup = 41.)  
           (file descriptor in r0)  
           (new file descriptor in r1)  
           sys dup  
           (file descriptor in r0)

**DUP2 (#09)** – Duplica um descritor de arquivo aberto.

Sintaxe: err = dup2(oldd, newd)  
           int    newd  
           int    oldd

Nota: A entrada dup2 é implementada adicionando 0100 a oldd.

**EXECVE (#10)** – Executa um arquivo.

Sintaxe: err = execve(name, argv, envp)  
           int    err  
           char  \*name  
           char  \*\*argv  
           char  \*\*envp

Entrada: name: Nome do arquivo a ser executado.  
 argv: Matriz de apontadores para os argumentos.  
 envp: Apontador para uma matriz de strings que  
        constituem o ambiente do processo.

**EXIT (#11)** – Encerra um processo.

Sintaxe: param = exit(status)  
           int    param  
           int    status

Entrada: status: O byte mais baixo (LSB) é passado para “param”.  
 Saída: param: Recebe o byte mais baixo de “status”.  
 Assembler: (exit = 1.)  
               (status in r0)  
               sys exit

### **FORK (#12)** – Gera um novo processo.

Sintaxe: newp = fork(void)  
               int     newp  
 Entrada: Nenhuma.  
 Saída: newp: ID do processo criado. Se retornar -1, houve falha  
               na criação do processo.  
 Assembler: (fork = 2.)  
               sys fork  
               (new process return)  
               (old process return, new process ID in r0)

### **FSTAT (#13)** – Obtém informações sobre o arquivo.

Sintaxe: stat = fstat(fd, \*buf)  
               int     stat  
               int     fd  
               void    \*buf  
 Biblioteca: #include <sys/types.h>  
               #include <sys/stat.h>  
 Entrada: fd: Descriptor de arquivo.  
               \*buf: Apontador para um buffer vazio.  
 Saída: stat: Informações obtidas. São as mesmas dos comandos  
               open, creat, dup ou pipe.

### **GETFSYS (#14)** – Obtém informações do sistema.

Sintaxe: stat = getfsys(dev, \*buf)  
               int     stat  
               int     dev  
               void    \*buf  
 Biblioteca: –  
 Entrada: dev: Dispositivo.  
               \*buf: Apontador para um buffer vazio.  
 Saída: stat: Estado obtido.

**IOCTL (#15) – Controle de dispositivos.**

Sintaxe: err = ioctl(fd, req, ...)

int err

int fd

int req

Biblioteca: #include <sgtty.h>

Entrada: fd: Descritor de arquivo.

req: Request.

Saída: err: 0 → Comando bem sucedido.

-1 → O descritor de arquivo não refere ao tipo de arquivo para o qual foi direcionado.

Assembler: (ioctl = 54.)

sys ioctl; fildes; request; argp

**KILL (#16) – Envia o sinal “sig” para o processo especificado em “r0”.**

Sintaxe: err = kill(pid, sig)

int err

int pid

int sig

Entrada: pid: Identificador do processo (ID).

sig: Sinal a ser enviado.

Saída: err: 0 → O processo foi encerrado.

-1 → O processo não existe ou não tem o mesmo ID do usuário atual ou não é superusuário.

Assembler: (kill = 37.)

(process number in r0)

sys kill; sig

Nota: Se o número do processo for 0, o sinal é enviado para todos os outros processos no grupo de processos do remetente.

**LINK (#17) – Link para um arquivo.**

Sintaxe: err = link(oldname, newname)

int err

char \*oldname

char \*newname

Entrada: oldname: Nome de arquivo antigo.

newname: Novo nome de arquivo.

Saída:      err:    0 → Link criado com sucesso.  
               -1 → Falha da criação do link.

Assembler: (link = 9.)  
               sys link; oldname; newname

### **MKNOD (#18) – Cria um diretório ou um arquivo especial.**

Sintaxe:     err = mknod(name, mode, dev)  
               int    err  
               char \*name  
               int    mode  
               int    dev

Entrada:    name: String apontando para o novo arquivo/diretório.  
               mode: Modo do novo arquivo/diretório.  
               dev: Dispositivo.

Saída:       err:    0 → Arquivo/diretório criado com sucesso.  
               -1 → Arquivo/diretório já existe ou usuário não é  
                   superusuário.

Assembler: (mknod = 14.)  
               sys mknod; name; mode; addr

Nota:        Apenas o superusuário pode usar este comando.

### **MOUNT (#19) – Monta o sistema de arquivos.**

Sintaxe:     err = mount(spec, dir, rwflag)  
               int    err  
               char \*spec  
               char \*dir  
               int    rwflag

Biblioteca: #include <sys/mount.h>

Entrada:    spec:    -  
               dir:    -  
               rwflag: -

Saída:       err:    0 → Comando executado com sucesso.  
               -1 → Erro na execução do comando.

### **OPEN (#20) – Abre um arquivo para leitura ou escrita.**

Sintaxe:     err = open(name, flags, mode)  
               int    err  
               char \*name  
               int    flags  
               int    mode

Entrada: name: Nome do arquivo a ser aberto.  
 flags: –  
 mode: 0 → leitura  
       1 → leitura/gravação  
 Saída: err: 0 → Arquivo aberto com sucesso.  
       –1 → Falha na abertura do arquivo.  
 Assembler: (open = 5.)  
           sys open; name; mode  
           (file descriptor in r0)

#### **PAUSE (#21)** – Pausa o sistema.

Sintaxe: err = pause(void)  
           int     err  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Assembler: (pause = 29.)  
           sys pause  
 Nota: Este comando nunca retorna normalmente. É usado para pausar o sistema até receber um sinal de “kill” ou “alarm”.

#### **PIPE (#22)** – Cria um canal interprocesso.

Sintaxe: err = pipe(fd)  
           int     err  
           int     \*fd  
 Entrada: fd: Descriptor de arquivo.  
 Saída: err: 0 → Canal criado com sucesso.  
       –1 → Falha na criação do canal.  
 Assembler: (pipe = 42.)  
           sys pipe  
           (read file descriptor in r0)  
           (write file descriptor in r1)

#### **READ (#23)** – Lê de um arquivo.

Sintaxe: err = read(fd, buf, bytes)  
           int     err  
           int     fd  
           void    \*buf  
           int     bytes

Entrada: fd: Descritor de arquivo.  
buf: Buffer vazio.  
bytes: Número de bytes a ler.

Saída: err: 0 → Fim de arquivo foi atingido.  
-1 → Erro de leitura.

Assembler: (read = 3.)  
(file descriptor in r0)  
sys read; buffer; nbytes  
(byte count in r0)

### **SBRK (#24) – Altera a alocação do núcleo.**

Sintaxe: err = sbrk(int incr)  
Nota: Ver BRK (#02).

### **LSEEK (#25) – Move o ponteiro de leitura / gravação.**

Sintaxe: err = lseek(fd, offset, flag)  
int err  
int fd  
long offset  
int flag

Entrada: fd: Descritor de arquivo.  
offset: Deslocamento.  
Flag: 0 → O ponteiro é setado para “offset” bytes.  
1 → O ponteiro é definido para sua posição atual mais deslocamento (offset).  
2 → O ponteiro é definido para o tamanho do arquivo mais deslocamento (offset)

Saída: err: 0 → Comando executado com sucesso.  
-1 → Erro na execução do comando.

Assembler: (lseek = 19.)  
(file descriptor in r0)  
sys lseek; offset1; offset2; whence  
[Offset1 e offset2 são as palavras alta e baixa de offset;  
r0 e r1 contém o ponteiro ao retornar].

### **SIGNAL (#26) – Pega ou ignora sinais.**

Sintaxe: err = signal(sig\_num, (\*func)(int) )  
int err  
char sig\_num  
void (\*func)(int)

Entrada: sig\_num: Número do sinal.  
 (\*func)(int): –

Saída: O valor (int)–1 é retornado se o sinal fornecido estiver fora da faixa.

Nota: Lista de sinais com nomes como no arquivo “include”.

1	SIGHUP	Desligamento.
2	SIGINT	Interrupção .
3*	SIGQUIT	Sair.
4*	SIGILL	Instrução ilegal (não reinicializado quando detectado).
5*	SIGTRAP	Trace trap (não zerado quando detectado).
6*	SIGIOT	Instrução IOT.
7*	SIGEMT	Instrução EMT.
8*	SIGFPE	Exceção de ponto flutuante.
9	SIGKILL	Kill (não pode ser capturado ou ignorado).
10*	SIGBUS	Erro de barramento.
11*	SIGSEGV	Violação de segmentação.
12*	SIGSYS	Argumento ruim para chamada de sistema.
13	SIGPIPE	Escrever em um tubo ou link sem ninguém para lê-lo.
14	SIGALRM	Despertador .
15	SIGTERM	Sinal de terminação de software.
16		Não atribuídos.

Os sinais com estrela na lista acima causam uma imagem do núcleo se não capturado ou ignorado.

Assembler: (signal = 48.)  
 sys signal; sig; label  
 (old label in r0)

### **STAT (#27) – Obtém o estado do arquivo**

Sintaxe: err = stat(path, buf)  
 int err  
 char \*path  
 void \*buf

Biblioteca: #include <sys/types.h>  
 #include <sys/stat.h>

Entrada: path: Nome/caminho do arquivo.  
 buf: Buffer vazio.

Saída: err: –

**STIME (#28)** – Configura data e hora do sistema.

Sintaxe: err = stime(tvec)

int err

int \*tvec

Entrada: tvec: Tempo em segundos a partir de 01/01/1970.

Saída: err: 0 → Data e hora definidas com sucesso.

-1 → Erro na definição da data e hora.

Assembler: (stime = 25.)

(time in r0-r1)

sys stime

**SYNC (#29)** – Atualiza o superblock.

Sintaxe: err = sync(void)

int err

Entrada: Nenhuma.

Saída: Nenhuma.

Assembler: (sync = 36.)

sys sync

**TIME (#30)** – Obtém a data e a hora.

Sintaxe: void time(tloc)

int \*tloc

Biblioteca: #include <sys/types.h>

#include <sys/timeb.h>

Entrada: Nenhuma.

Saída: tloc: Tempo em segundos a partir de 01/01/1970.

**TIMES (#31)** – Retorna informações de tempo.

Sintaxe: t = times(struct tms \*tvec)

int t

Entrada: Nenhuma.

Saída: Ver nota.

Assembler: (times = 43.)

sys times; buffer

Nota: Retorna informações de tempo para o processo atual e para os processos filho encerrados do processo atual. Todos os tempos estão em 1/Hz segundos, onde Hz = 60 ou Hz = 50. Após a chamada, o buffer aparecerá da seguinte forma:

```
struct tbuffer {
    long proc_user_time;
    long proc_system_time;
    long child_user_time;
    long child_system_time;
};
```

**UMOUNT (#32)** – Desmonta sistemas de arquivos.

Sintaxe: err = umount(spec)  
           int     err  
           char   \*spec

Entrada: –

Saída: –

Assembler: –

**UNLINK (#33)** – Remove entrada de diretório.

Sintaxe: err = unlink(path)  
           int     err  
           char   \*path

Entrada: path: String com o diretório a ser removido.

Saída: err: 0 → Entrada de diretório removido com sucesso.  
           –1 → Erro na remoção do diretório.

Assembler: (unlink = 10.)  
           sys unlink; name

**UTIME (#34)** – Define hora de um arquivo.

Sintaxe: err = utime(path, utimbuf \*buf)  
           int     err  
           char   \*path  
           struct utimbuf \*buf

Biblioteca: #include <sys/types.h>

Entrada: path: Nome/caminho do arquivo.  
           Buf: –

Saída: –

Assembler: (utime = 30.)  
           sys utime; file; timep

**WAITPID (#35)** – Espera que o processo mude de estado.

Sintaxe: err = waitpid(pid, statloc, options)

```

int    err
int    pid
int    *statloc
int    options

```

Biblioteca: #include <sys/types.h>  
#include <sys/wait.h>

Entrada: pid: ID do processo.  
statloc: Estado de espera.

options: <-1: Espera por qualquer processo filho cujo ID  
do grupo de processo seja igual ao valor  
absoluto de pid.  
-1: Espera por qualquer processo filho.  
0: Esperar por qualquer processo filho cujo ID  
do grupo de processos seja igual ao do  
processo de chamada.  
>0: Espera pelo processo filho cujo ID do  
é igual ao valor do pid.

### **WRITE (#36) – Escreve em um arquivo**

Sintaxe: err = write(fd, buf, nbytes)  
int fd  
void \*buf  
int nbytes

Entrada: fd: Descritor de arquivo.  
buf: Buffer com nbytes contíguos que serão escritos  
no arquivo.  
nbytes: Número de bytes a serem escritos.

Saída: err: 0 → Escrita bem-sucedida.  
-1 → Erro durante a escrita.

Assembler: (write = 4.)  
(file descriptor in r0)  
sys write; buffer; nbytes  
(byte count in r0)

### **REBOOT (#37) – Reinicia o sistema**

Sintaxe: err = reboot(p1, p2)  
int err  
char p1  
char p2

Biblioteca: #include <sys/reboot.h>  
               #include <unistd.h>  
 Entrada:   p1:    –  
               p2:    –  
 Saída:     err:    0 → Não aplicável.  
               –1 → Erro na reinicialização.

### **SYMLINK (#38)** – Cria um novo nome para um arquivo.

Sintaxe:   err = symlink(oldname, newname)  
               int      err  
               char    \*oldname  
               char    \*newname  
 Biblioteca: #include <fcntl.h>  
               #include <unistd.h>  
 Entrada:   oldname:  Nome antigo.  
               newname: Novo nome.  
 Saída:     err:    0 → Comando executado com sucesso.  
               –1 → Erro na criação do nome.

### **CHROOT (#39)** – Altera o diretório raiz.

Sintaxe:   err = chroot(path)  
               int      err  
               char    \*path  
 Biblioteca: #include <unistd.h>  
 Entrada:   path:   Novo caminho para o diretório raiz.  
 Saída:     err:    0 → Comando executado com sucesso.  
               –1 → Erro na alteração.

### **MOD\_REG (#40)**

Sintaxe:   err = mod\_reg (sig, (func)())  
               int      err  
               int      sig  
               int    (\*func)()

### **MOD\_DEREG (#41)**

Sintaxe:   err = mod\_dereg (sig)  
               int      err  
               int      sig

**MOD\_CALL (#42)**

Sintaxe:    err = mod\_call (sig, fnc, args, argsz)  
               int     err  
               int     sig  
               int     fnc  
               char   \*args  
               int     argsz

**MOD\_SENDREPLY (#43)**

Sintaxe:    err = mod\_sendreply (pid, fnc, r, rsz)  
               int     err  
               int     pid  
               int     fnc  
               char   \*r  
               int     rsz

**MOD\_REPLY (#42)**

Sintaxe:    err = mod\_reply (sig, fcn, r)  
               int     err  
               int     sig  
               int     fcn  
               char   \*r

**6.4.2 – Chamada Indireta de Sistema****CREAT – Cria um novo arquivo**

Chamada: creat(path, mode)  
               char   \*path  
               int     mode

Sintaxe:    err = open(path, 0x301, mode)

Entrada:    path: Nome do arquivo.  
               0x301: O\_CREAT | O\_TRUNC | O\_WRONLY  
               mode: Modo.

Saída:      err:    0 → Arquivo criado com sucesso.  
                   -1 → Erro na criação do arquivo.

Assembler: (creat = 8.)  
               sys creat; name; mode  
                   (file descriptor in r0)

### 6.4.3 – Chamadas via GETSET

**GETPID** – Obtém o ID do processo.

Chamada: getpid(void)

Sintaxe: id = getset(0)

Biblioteca: #include <unistd.h>

Entrada: Nenhuma.

Saída: id: Identificador do processo.

Assembler: (getpid = 20.)

sys getpid

(pid in r0)

**GETPPID** – Obtém o ID do processo chamador.

Chamada: getppid(void)

Sintaxe: id = getset(1)

Biblioteca: #include <unistd.h>

Entrada: Nenhuma.

Saída: id: Identificador do processo.

**GETUID** – Retorna a identidade do usuário real do processo atual.

Chamada: getuid(void)

Sintaxe: id = getset(2)

Biblioteca: #include <unistd.h>

Entrada: Nenhuma.

Saída: id: Identificador do processo.

Assembler: (getuid = 24.)

sys getuid

(real user ID in r0, effective user ID in r1)

**SETUID** – Define a identidade do usuário e do grupo.

Chamada: setuid(uid)

Sintaxe: err = getset(3, uid)

int err

int uid

Biblioteca: #include <unistd.h>

Entrada: uid: Identificador do processo do usuário.

Saída: err: 0 → ID definido com sucesso.

-1 → Erro na definição do ID.

Assembler: (setuid = 23.)  
 (user ID in r0)  
 sys setuid  
  
 (setgid = 46.)  
 (group ID in r0)  
 sys setgid

**GETEUID** – Retorna o ID do usuário efetivo do processo chamador.

Chamada: geteuid(void)  
 Sintaxe: id = getset(4)  
 Biblioteca: #include <unistd.h>  
 Entrada: Nenhuma.  
 Saída: id: Identificador do processo.

**GETGID** – Retorna o ID do usuário real do processo atual.

Chamada: getgid(void)  
 Sintaxe: id = getset(5)  
 Biblioteca: #include <unistd.h>  
 Entrada: Nenhuma.  
 Saída: id: Identificador do usuário.  
 Assembler: (getgid = 47.)  
 sys getgid  
 (real group ID in r0, effective group ID in r1)

**SETGID** – Define o ID do grupo.

Chamada: setgid(gid)  
 Sintaxe: err = getset(6, gid)  
 int err  
 int gid  
 Biblioteca: #include <unistd.h>  
 Entrada: gid: Identificador do grupo.  
 Saída: err: 0 → ID definido com sucesso.  
 -1 → Erro na definição do ID.  
 Assembler: (setgid = 46.)  
 (group ID in r0)  
 sys setgid

**GETEGID** – Retorna o ID do grupo efetivo do processo chamador.

Chamada: getegid(void)

Sintaxe: id = getset(7)

Biblioteca: #include <unistd.h>

Entrada: Nenhuma.

Saída: id: Identificador do grupo.

**GETPRIO** – Retorna a prioridade do processo atual.

Chamada: getprio(void)

Sintaxe: prd = getset(8)

Biblioteca: #include <sched.h>

Entrada: Nenhuma

Saída: prd: Prioridade.

**SETPRIO** – Define o prioridade de um processo.

Chamada: setprio(pid, prio)

Sintaxe: err = getset(9, pid, prio)

int err

int pid

char prio

Biblioteca: #include <unistd.h>

Entrada: pid: Identificador do processo.

prio: Prioridade do processo.

Saída: err: 0 → Prioridade definida com sucesso.

-1 → Erro na definição da prioridade.

**UMASK** – Define máscara de criação de um arquivo.

Chamada: umask(mask)

Sintaxe: oldm = getset(10, mask)

int oldm

int mask

Entrada: mask: Modo. Apenas os 9 bits mais baixos são válidos.

Saída: oldm: Valor antigo da máscara.

Assembler: (umask = 60.)

sys umask; complmode

**SYSTRACE** – Gerar e aplicar protocolos para as chamadas de sistema.

Chamada: systrace(onoff)

Sintaxe: err = getset(11, onoff)

void err

int onoff

Entrada: onoff: Novo protocolo.

Saída: -

#### 6.4.4 – Módulo TCP/IP

O módulo TCP/IP implementa um subconjunto de IPv4 e permite que o Uzix se comunique com outros sistemas que suportem o protocolo. A assinatura do módulo TCP/IP é 04950H, e ele provê as funções listadas abaixo.

Chamada Protótipo C

FNC#

ipconnect	int ipconnect(char mode, ip struct t *ipstruct)	1
ipgetc	int ipgetc(uchar socknum)	2
ipputc	int ipputc(uchar socknum, uchar byte)	3
ipwrite	int ipwrite(uchar socknum, uchar *bytes, int len)	4
ipread	int ipread(uchar socknum, uchar *bytes, int len)	5
ipcclose	int ipcclose(uchar socknum)	6
iplisten	int iplisten(int aport, uchar protocol)	7
ipaccept	int ipaccept(ip struct t *ipstruct, int aport, uchar block)	8
ping	int ping(uchar *IP, unsigned long *unused, uint len)	9
setsockopt	int setsockopt(uchar socknum, uint timeout)	10
ipunlisten	int ipunlisten(int aport)	11
ipgetpingreply	icmpdata t *ipgetpingreply(void)	12
gettcpinfo	tcpinfo t *gettcpinfo(void)	13
getsockinfo	sockinfo t *getsockinfo(uchar socknum)	14

Os tipos de dados usados são:

```
// números de protocolo (protocolo para iplisten)
ICMP_PROTOCOL = 1
TCP_PROTOCOL = 6
UDP_PROTOCOL = 17

// modos de abertura
TCP_ACTIVE_OPEN = 255
TCP_PASSIVE_OPEN = 0
```

```
// protocolos (modo do ipconnect)
IPV4_TCP    = 1
IPV4_UDP    = 2
IPV4_ICMP   = 3

// modos UDP
UDPMODE_ASC  = 1
UDPMODE_CKSUM = 2

// codigos de erro
ECONTIMEOUT = 080H
ECONREFUSED = 081H
ENOPERM      = 082H
ENOPORT      = 083H
ENOROUTE     = 084H
ENOSOCK      = 085H
ENOTIMP      = 086H
EPROT        = 087H
EPORTINUSE   = 088H

// estados permitidos para sockstatus em sockinfo_t
TCP_CLOSED      = 000H
TCP_LISTEN       = 001H
TCP_SYN_SENT     = 042H
TCP_SYN_RECEIVED = 043H
TCP_ESTABLISHED  = 0C4H
TCP_FIN_WAIT1    = 045H
TCP_FIN_WAIT2    = 046H
TCP_CLOSE_WAIT   = 087H
TCP_CLOSING      = 008H
TCP_LAST_ACK     = 009H
TCP_TIMEWAIT     = 00AH
UDP_LISTEN       = 091H
UDP_ESTABLISHED  = 094H

ip_struct_t = { uchar remote_ip[4],
                 uint remote_port,
                 uint local_port }

icmpdata_t = { uchar type,
                uchar icmpcode,
                unsigned long unused,
```

```

        uchar data[28], /* pad para 64 bytes */
        uint len;
        uchar sourceIP[4],
        uchar ttl }

tcpinfo_t = { uchar IP[4],
               uchar dns1ip[4],
               uchar dns2ip[4],
               char datalink[5],
               char domainname[DOMSIZE=128],
               int used_sockets,
               int avail_sockets,
               int used_buffers,
               int avail_buffers,
               int IP_chksum_errors }

sockinfo_t = { int localport,
               int remoteport,
               uchar remote_ip[4],
               char socketstatus, /* bit 7: permissao
                                   de escrita
                                   bit 6: estado de
                                   listen
                                   bits 3-0: estado
                                   */
               char sockettype, /* TCP=1, UDP=2 */
               char sockerr, /* codigo de erro */
               int pid }

```

#### 6.4.5 – Códigos de erro

As chamadas de sistema do Uzix retornam um valor maior que 0 em caso de sucesso e menor que 0 em caso de erro. O código de erro é colocado na variável global (definida no stub dos programas Uzix) **errno**. Abaixo estão relacionados os possíveis códigos de erro.

EPERM	1 Operation not permitted */
ENOENT	2 No such file or directory
ESRCH	3 No such process
EINTR	4 Interrupted system call
EIO	5 I/O error

ENXIO	6 No such device or address
E2BIG	7 Arg list too long
ENOEXEC	8 Exec format error
EBADF	9 Bad file number
ECHILD	10 No child processes
EAGAIN	11 Try again
ENOMEM	12 Out of memory
EACCES	13 Permission denied
EFAULT	14 Bad address
ENOTBLK	15 Block device required
EBUSY	16 Device or resource busy
EEXIST	17 File exists
EXDEV	18 Cross-device link
ENODEV	19 No such device
ENOTDIR	20 Not a directory
EISDIR	21 Is a directory
EINVAL	22 Invalid argument
ENFILE	23 File table overflow
EMFILE	24 Too many open files
ENOTTY	25 Not a typewriter
ETXTBSY	26 Text file busy
EFBIG	27 File too large
ENOSPC	28 No space left on device
ESPIPE	29 Illegal seek
EROFS	30 Read-only file system
EMLINK	31 Too many links
EPIPE	32 Broken pipe
EDOM	33 Math argument out of domain of func
ERANGE	34 Math result not representable
EDEADLK	35 Resource deadlock would occur
ENAMETOOLONG	36 File name too long
ENOLCK	37 No record locks available
EINVFNC	38 Function not implemented
ENOTEMPTY	39 Directory not empty
ELOOP	40 Too many symbolic links encountered
ESHELL	41 It's a shell script
ENOSYS	EINVFNC

## 6.5 – CÓDIGOS DO TERMINAL VT-52

Ctrl G	07H	Beep.
Ctrl H	08H	Backspace.
Ctrl I	09H	TAB.
Ctrl J	0AH	Avança uma linha.
Ctrl K	0BH	Move cursor para a origem.
Ctrl L	0CH	Limpa a tela e move cursor para a origem.
Ctrl M	0DH	Retorno de carro
Ctrl \	1CH	Avança cursor uma posição.
Ctrl ]	1DH	Retrocede cursor uma posição.
Ctrl ^	1EH	Move cursor para cima.
Ctrl _	1FH	Move cursor para baixo.
	7FH	Deleta caractere e move cursor à esquerda.
Esc A	1BH,41H	Move cursor para cima.
Esc B	1BH,42H	Move cursor para baixo.
Esc C	1BH,43H	Move cursor para a direita.
Esc D	1BH,44H	Move cursor para a esquerda.
Esc E	1BH,45H	Limpa a tela e coloca cursor na origem.
Esc H	1BH,48H	Coloca cursor na origem.
Esc J	1BH,4AH	Apaga até o fim da tela, não move cursor.
Esc j	1BH,6AH	Limpa a tela e coloca cursor na origem.
Esc K	1BH,4BH	Apaga até o fim da linha, não move cursor.
Esc L	1BH,4CH	Insere linha acima do cursor, move o resto da tela para baixo, deixa cursor no início da nova linha.
Esc I	1BH,6CH	Apaga até o fim da linha, não move cursor.
Esc M	1BH,4DH	Apaga linha do cursor, move o resto da tela para linha e coloca cursor no início da próxima linha.
Esc x 4	1BH,78H,34H	Seleciona cursor em bloco.
Esc x 5	1BH,78H,35H	Desliga cursor.
Esc Y n m	1BH,59H,m,n	Move cursor para coluna m-32 e linha y-32.
Esc y 4	1BH,79H,34H	Seleciona cursor sublinhado.
Esc y 5	1BH,79H,35H	Liga cursor.

## 7 – WiOS

### 7.1 – DRIVER DE ARQUIVO DE SISTEMA

Descrição: Driver de arquivo de sistema  
 Nome Driver: “File-System”  
 GDA: \_hfsdrv  
 Header-File: FSFNC.H

Todas as funções de disco têm os mesmos números e nomes que seus correspondentes DOS 2.

**\_SELDSK** (Função ‘0x0e’)

Descrição: Define a unidade atual.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_CURDRV** (Função ‘0x19’)

Descrição: Obtém unidade atual.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_RDABS** (Função ‘0x2f’)

Descrição: Leitura absoluta de setores.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_WRABS** (Função ‘0x30’)

Descrição: Gravação absoluta de setores.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_FFIRST** (Função ‘0x40’)

Descrição: Encontra a primeira entrada.  
 Argumentos: char \* ponteiro para o nome do arquivo / bloco de informações do arquivo.  
                  char \* ponteiro para o novo bloco de informações do arquivo.  
                  char busca atributos.  
 Retorno: Nenhum.

**\_FNEXT** (Função '0x41')

Descrição: Encontra a próxima entrada.  
 Argumentos: char \* ponteiro para o bloco de informações do arquivo  
 Retorno: Nenhum.

**\_OPEN** (Função '0x43')

Descrição: Abre arquivo handle.  
 Argumentos: char \* Ponteiro para o nome do arquivo.  
               int      Modo.  
 Retorno: 0 → Erro.  
               1~255 → Número do arquivo handle.

**\_CREATE** (Função '0x44')

Descrição: Cria arquivo e abre handle.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_CLOSE** (Função '0x45')

Descrição: Fecha o arquivo handle.  
 Argumentos: int      Identificador do arquivo handle.  
 Retorno: 0 → Feito.  
               1~255 → Erro.

**\_READ** (Função '0x48')

Descrição: Lê do identificador de arquivo.  
 Argumentos: int      Identificador do arquivo handle.  
               char \*    endereço de destino (4000H ~ BFFFH).  
               uint     tamanho (0 ~ 16 384).  
               T\_SEG   Segmento de destino (0 ~ 3071).  
               char     Dummy (deve ser 0) modo (CRC, CRYPT).  
 Retorno: 0 → Feito.  
               1~255 → Erro.

**\_WRITE** (Função '0x49')

Descrição: Escreve no arquivo handle.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_SEEK** (Função '0x4a')

Descrição: Dusca (posição do ponteiro do arquivo)  
 Argumentos: int Arquivo handle.  
               uint deslocamento (0 ~ 65 535).  
               char modo: 0 → do início do arquivo;  
                         1 → do final do arquivo;  
                         2 → em relação à posição atual.

Retorno: 0 → Concluído.  
 1~255 → Erro.

**\_DELETE** (Função '0x4d')

Descrição: Excluir arquivo ou subdiretório.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_RENAME** (Função '0x4e')

Descrição: Renomear arquivo ou subdiretório.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_MOVE** (Função '0x4f')

Descrição: Mover arquivo ou subdiretório.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_ATTR** (Função '0x50')

Descrição: Alterar atributos de arquivo ou subdiretório.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_FTIME** (Função '0x51')

Descrição: Obter/definir data e hora do arquivo.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_VERIFY** (Função '0x58')

Descrição: Obter configuração de sinalizador de verificação.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_GETCD** (Função ‘0x59’)

Descrição: Obter o diretório atual.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_CHDIR** (Função ‘0x5a’)

Descrição: Alterar o diretório atual.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_PARSE** (Função ‘0x5b’)

Descrição: Analisar o nome do caminho.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_FORMAT** (Função ‘0x67’)

Descrição: Formatar o disco.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**\_D2V** (Função ‘0x80’)

Descrição: Lê diretamente do arquivo handle para porta de I/O  
                   (por exemplo, VRAM)  
 Argumentos: int     Arquivo handle.  
               Uint    Tamanho.  
               Char    Número da porta.  
 Retorno:    0 → Feito.  
               1~255 → Erro.

**7.2 – DRIVER EXTERNO**

Descrição: Driver externo  
 Nome Driver: “External driver”  
 GDA:        \_hexdrv  
 Header file: EXTFNC.H

**NONAME** (Função ‘1’)

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**RESKEY** (Função '2')

Descrição: Limpa o buffer de teclado da tarefa de chamada.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**GETKEY** (Função '3')

Descrição: Obtém o próximo caractere do teclado no buffer da tarefa de chamada.  
 Argumentos: Nenhum.  
 Retorno: 0 → Nenhuma tecla foi pressionada desde a última GETKEY  
           1~255 → Código ASCII do caractere.

**7.3 – DRIVER GRÁFICO DE E/S**

Descrição: Driver gráfico de E/S  
 Nome driver: "Graphic-IO Driver"  
 GDA: \_hgiodrv  
 Header-File: GIOFNC.H

**NONAME0** (Função '1')

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**NONAME1** (Função '2')

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**NONAME2** (Função '3')

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**GET\_VHANDLE** (Função '4')

Descrição: Obtém o número handle de um arquivo no VideoRAM-Directory (usado para encontrar alças de fontes e ícones)

Argumentos: char ICON (para ícone)  
                   FONT (para fonte)  
                   char \* Nome do arquivo  
 Retorno: 0 ~ 65 534 → Número handle do ícone/fonte.  
                   -1 → Arquivo não encontrado.

#### **ADFONTLIB** (Função '5')

Descrição: Carrega uma fonte do disco.  
 Argumentos: char \* Nome do arquivo.  
 Retorno: 0 ~ 65 534 → Número handle da fonte carregada.  
                   -1 → Arquivo não encontrado.  
 Nota: Cuidado para não sobrecarregar a memória, pois nenhuma verificação é feita.

#### **NONAME2** (Função '6')

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

## 7.4 – DRIVER GRÁFICO

Descrição: Driver gráfico  
 Nome driver: “Graphic Driver”  
 GDA: \_hgrpdrv  
 Header-File: GRPFNC.H

#### **MOUSEBOX** (Função '1')

Descrição: Define a caixa onde o mouse pode se mover. As coordenadas são absolutas (os cantos devem estar na ordem superior-esquerda/inferior-direita).  
 Argumentos: int x (coordenadas do canto superior esquerdo)  
                   int y  
                   int x (coordenadas do canto inferior direito)  
                   int y  
 Retorno: Nenhum.

#### **PPOINTER** (Função '2')

Descrição: Coloca o ponteiro do mouse em coordenadas absolutas.  
 Argumentos: int x (coordenadas do ponteiro do mouse)  
                   int y

Retorno: Nenhum.  
 Nota: Se as coordenadas estiverem fora da caixa do mouse, o ponteiro irá para dentro dela na próxima interrupção.

#### **MOUSEMOVE** (Função '3')

Descrição: Move o ponteiro do mouse de acordo com o movimento do usuário (não necessário).  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

#### **STOREBOX** (Função '4')

Descrição: Armazena as coordenadas VDA atuais.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.  
 Nota: Existe apenas um conjunto de coordenadas VDA a serem armazenadas. Uma vez que esta função também é usada pelo WiOS, a função RESTOBOX pode não restaurar as coordenadas armazenadas após uma pesquisa ou chamando o driver da janela.

#### **RESTOBOX** (Função '5')

Descrição: Restaura o último conjunto de coordenadas VDA armazenadas com STOREBOX  
 Argumentos: Nenhum.  
 Retorno: Nenhum.  
 Nota: Ver nota em STOREBOX.

#### **SETBOX** (Função '6')

Descrição: Definir a área de exibição válida (VDA – valid display area). As coordenadas são absolutas.  
 Argumentos: int x (coordenadas do primeiro canto)  
               int y  
               int x (coordenadas do canto oposto)  
               int y  
 Retorno: Nenhum.  
 Nota: Os cantos das notas podem ser definidos livremente. Eles serão ajustados para o canto superior esquerdo / inferior direito.

**COPY** (Função '7')

**Descrição:** Copia a área gráfica correspondente para a VDA.

**Argumentos:** int sx (coordenadas do canto superior esquerdo)  
 int sy da área de origem)  
 int nx (tamanho da área a ser copiada)  
 int ny  
 int dx (coordenadas de destino)  
 int dy

**Retorno:** 0 → A área foi totalmente copiada.  
 1 → Nada foi copiado (o destino está totalmente fora da VDA).  
 2 → A área foi cortada (o destino está parcialmente fora da VDA).

**Nota:** A direção de cópia é corrigida automaticamente se a origem e o destino se cruzarem.

**PUTICON** (Função '8')

**Descrição:** Coloca ícone na área correspondente à VDA.

**Argumentos:** int 0 ~ 32767 → identificador do ícone padrão da VRAM.

-1 → os dados gráficos vem da memória e '*\*\_adrblk*' deve conter as seguintes informações:

offset tam.

+0	2	endereço,
+2	2	segmento dados gráficos do ícone (modo de cor 32768)
+4	2	largura do ícone
+6	2	altura do ícone

int x coordenadas de destino

int y

**Retorno:** 0 → O ícone foi totalmente colocado na VDA.  
 1 → Nada foi colocado (o destino está totalmente fora da VDA)  
 2 → O ícone foi cortado (o destino está parcialmente fora da VDA)

**EXPICON** (Função '9')

**Descrição:** Expande o ícone para o tamanho total da VDA.

Argumentos: int 0 ~ 32 767 → Identificador do ícone padrão da VRAM.

-1 → Os dados do ícone vem da memória  
(consulte '\*'\_adrblk' em PUTICON)

int x (coordenadas)

int y

Retorno: Nenhum.

Nota: A coordenada x,y define o destino do ícone 'master'. A área em torno deste ícone está lado a lado com o mesmo ícone.

#### **SETFONT** (Função '10')

Descrição: Define a fonte atual

Argumentos: int Número handle da fonte (font-handle)

Retorno: Altura da fonte.

Nota: Nenhuma verificação de erro para o identificador de fonte é feita. Se um identificador de fonte inexistente é enviado, a altura da fonte atual é retornada e nada é alterado.

#### **SETCOL** (Função '11')

Descrição: Define a cor da fonte.

Argumentos: int Cor do primeiro plano.

int Cor de fundo.

Retorno: Nenhum.

#### **SETPOS** (Função '12')

Descrição: Define a posição atual do texto.

Argumentos: int x Coordenadas da posição do texto.

int y

int Distância em pixels entre caracteres.

Retorno: Nenhum.

#### **WRTXT** (Função '13')

Descrição: Escreve texto na tela de acordo com a VDA.

Argumentos: char \* Endereço da string

T\_SEG Segmento da string

int 0 → Escreve todos os caracteres até o byte 0.

1 ~ 32 767 → Escreve o número especificado de caracteres.

Retorno: int Tamanho total do texto em pixels (não o comprimento do texto exibido)  
 Nota: Cada sublinhado sublinhará o próximo caractere. A distância entre os caracteres é definida com a função 'SET-POS'.

#### **GETTXTLEN** (Função '14')

Descrição: Obtém o tamanho em pixels de uma string terminada em 0.  
 Argumentos: char \* Endereço da string.  
               T\_SEG Segmento da string.  
 Retorno: Nenhum.  
 Nota: Cada sublinhado sublinhará o próximo caractere. A distância entre os caracteres é definida com a função 'SET-POS'.

#### **GETCHARS** (Função '15')

Descrição: Retorna o número de caracteres que cabem na largura especificada.  
 Argumentos: char \* endereço da string  
               T\_SEG Segmento da string  
               int Largura da área em pixels  
               char Direção: 0 → para frente (incrementar ponteiro de texto).  
                                 1~255 → para trás.  
 Retorno: int Número de caracteres.

#### **BOX** (Função '16')

Descrição: Desenhar caixa pintada na tela de acordo com a VDA.  
 Argumentos: int x Coordenadas do canto  
               int y superior esquerdo.  
               int nx Largura  
               int ny da caixa.  
               int 0~32767 → Ccor da caixa.  
               Int operação lógica (consulte o manual técnico do chip V9990).  
 Retorno: Nenhum.

**GETBOX** (Função '17')

Descrição: Retorna as coordenadas VDA atuais

Argumentos: Nenhum.

Retorno: Endereço do bloco de dados:

offset tam.

+0 2 coordenada x do canto superior esquerdo.

+2 2 coordenada y do canto superior esquerdo.

+4 2 coordenada x do canto inferior direito.

+6 2 coordenada y do canto inferior direito.

Nota: Use esta função para armazenar as coordenadas VDA atuais permanentemente em sua própria área de memória (melhor que STOREBOX ao pesquisar ou chamar o driver de janela).

**7.5 – DRIVER DE MEMÓRIA****Driver de memória**

Descrição: Driver de memória

Nome driver: -

GDA: \_hmemdrv

Header-File: MEMFNC.H

Este é um driver interno do WiOS e não tem nome.

**ALLSEG** (Função '1')

Descrição: Aloca um novo segmento

Argumentos: T\_TASK 0~252 Número handle da tarefa.

Retorno: -1 → não há segmentos livres.

0~32767 → número do segmento alocado.

**ENASEG** (Função '2')

Descrição: Habilita segmento na página 1 / 2

Argumentos: T\_SEG 0..32767 Número do segmento

char 1 → página 1 (4000H-7FFFH)

2 → página 2 (8000H-BFFFH)

Retorno: 0 → Segmento foi ativado.

-1 → Segmento não pôde ser ativado (fora do intervalo).

-2 → Segmento não foi alocado.

**FRESEG** (Função '3')

Descrição: Libera segmento alocado.  
 Argumentos: T\_SEG 0..32767 Número do segmento.  
 Retorno: 0 → Segmento foi liberado.  
           -1 → Segmento não pôde ser liberado (fora do intervalo).  
           -2 → Segmento não estava alocado para esta tarefa.

**GETSEG** (Função '4')

Descrição: Retorna o estado do segmento.  
 Argumentos: T\_SEG 0..32767 Número do segmento.  
 Retorno: 0~252 Identificadores de tarefa/driver que alocaram o segmento.  
           253 Usado por aplicativo não especificado (foi alocado durante a inicialização do WiOS)  
           254 Memória muito lenta para CPU (por exemplo, mapeadores da Sony no modo turbo).  
           255 O segmento está livre.

**NONAME** (Função '5')

Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**SEGCOPY** (Função '6')

Descrição: Copia um determinado número de bytes de um segmento para outro.  
 Argumentos: Endereços relativos nos segmentos (0000H ~ 3FFFH):  
           T\_SEG 0 ~ 32 767 Segmento de origem.  
           T\_SEG 0 ~ 32 767 Segmento de destino.  
           uint 0 ~16 383 Endereço da origem (0 ~16 383).  
           uint 0 ~16 383 Endereço de destino (0 ~16 383).  
           uint 0 ~16 384 Número de bytes a serem copiados.  
 Retorno: 0 → Os dados foram copiados.  
           -1 → Dados fora do intervalo.  
 Nota: Nenhuma verificação é feita para ver se os segmentos estão alocados ou para qual tarefa eles pertencem.

**SEGNCPY** (Função '7')

**Descrição:** Copia bytes de um segmento para outro até que um determinado byte seja encontrado.

**Argumentos:** Endereços relativos nos segmentos (0000H ~ 3FFFH):

T\_SEG 0 ~ 32 767 Segmento de origem.

T\_SEG 0 ~ 32 767 Segmento de destino.

uint 0 ~16 383 Endereço da origem (0 ~16 383).

uint 0 ~16 383 Endereço de destino (0 ~16 383).

uint 0 ~16 384 Número de bytes a serem copiados.

Char 0 ~255 Byte terminador.

**Retorno:** -1 → Dados fora do intervalo.

0 ~16 384 Número de bytes efetivamente copiados.

**Nota:** Nenhuma verificação é feita se o byte existe. SEGNCOPY copiará os dados até o byte ser encontrado, portanto deve-se ter certeza de que o byte existe no intervalo.

**7.6 – DRIVER PADRÃO**

**Descrição:** Driver padrão

**Nome driver:** –

**GDA:** \_hstddrv

**Header-File:** STDFNC.H

Este é um driver interno do WiOS e não tem nome.

**ADDINT** (Função '1')

**Descrição:** Adiciona uma interrupção.

**Argumentos:** T\_TASK 0~252 Identificador de tarefa/driver.

T\_SEG 0..32767 Segmento.

uint 4000H~7FFFH Endereço da rotina de interrupção.

**Retorno:** 0~MAXINT Identificador de interrupção ('MAXINT' está em DEF.H)

255 Sem mais interrupções.

**Nota:** A rotina de interrupção pode não ser EI.

**DELINT** (Função '2')

**Descrição:** Remove uma interrupção.

**Argumentos:** T\_INT Identificador de interrupção.

Retorno: 0 → A interrupção foi removida.  
 255 → A interrupção não pôde ser removida.

**DRAG** (Função ‘3’)  
 Descrição: Inicia uma operação de arrastar.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.  
 Nota Em construção. Não use ainda.

## 7.7 – DRIVER DE TAREFA

Descrição: Driver de tarefa  
 Nome driver: –  
 GDA: \_htaskdrv  
 Header-File: TASKFNC.H

Este é um driver interno do WiOS e não tem nome.

**NONAME** (Função ‘1’)  
 Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**LOADTASK** (Função ‘2’)  
 Descrição: Carrega uma tarefa do disco  
 Argumentos: char \* Ponteiro para string drive+path+filename.  
 Retorno: –1 → A tarefa não pôde ser carregada.  
 0~252 → Identificador de tarefa.

**NONAME** (Função ‘3’)  
 Descrição: Usado internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**KILLTASK** (Função ‘4’)  
 Descrição: Remove uma tarefa da memória e libera todos os segmentos e interrupções.  
 Argumentos: T\_TASK Identificador da tarefa.

Retorno: 0 → O evento de retorno foi enviado.

-1 → A tarefa não existe.

Nota: Função ainda não suportada. Não usar.

### **S\_NAME** (Função ‘5’)

Descrição: Pesquisa por nome se uma tarefa/driver estiver presente (carregada).

Argumentos: char \* Ponteiro para string com o nome da tarefa a ser pesquisada. Deve estar entre 4000H e 7FFFH.

Retorno: 0 → Não encontrado.

#0 → Endereço do bloco de dados:

offset tam.

+0 2 Identificador da tarefa/driver.

+2 2 Número da versão.

### **ADD\_EVENT** (Função ‘127’)

Descrição: Envia um evento para uma tarefa.

Argumentos: uint Endereço do bloco de eventos  
 membro função  
 task Identificador da tarefa de destino.  
 event Evento.  
 array[ ] 12 bytes de informação para a tarefa  
 recebedora.

char Número handle da tarefa de envio (preenchida  
 pelo WiOS)

Retorno: Nenhum.

Nota: Observe que o bloco de eventos deve estar fora da página 1 e 2 para que a tarefa de recebimento possa acessar o bloco sem comutação de segmento. Recomenda-se usar ‘eventblk’ do GDA para armazenar eventos.

## **7.8 – DRIVER DE JANELA**

Descrição: Driver de janela

Nome driver: “Window Driver”

GDA: \_hwindrv

Header-File: WINFNC.H

A função explicação-argumento requer a definição de ‘struct WINSTR windat;’ como uma variável global no código-fonte C para ter acesso à estrutura da janela por meio de nomes.

Se houver problemas com algumas palavras específicas da janela na lista a seguir, consulte a ilustração do capítulo sobre os ícones.

Como este é o driver mais complexo, existem algumas funções que ainda não foram implementadas. Ainda há funcionalidade suficiente para escrever aplicativos normais e a maioria das funções ausentes podem ser simuladas com algumas soluções alternativas.

### **7.8.1 – A Estrutura da Janela**

Tipo	Nome	Descrição
int	handle	Identificador da janela.
int	x	Posição x absoluta na tela.
int	y	Posição y absoluta na tela.
int	nx	Largura da área de trabalho visível.
int	ny	Altura da área de trabalho visível. A área de trabalho visível está sempre disponível para uso da tarefa. Ícones de janela não usam esta área já que o WiOS os coloca ao redor do janela, exceto para os seguintes casos, onde os ícones são copiados para a área do usuário: <ul style="list-style-type: none"> <li>• Ícone Voltar, Fechar ou Alternar Tamanho, se não houver barra de título</li> <li>• Ícone Redimensionar, se não houver barra de rolagem.</li> </ul>
int	vx	Largura virtual da janela.
int	vy	Altura virtual da janela. Só são necessários para o cálculo das barras de rolagem e pode ser 0 se não houver nenhuma.
int	scrx	Deslocamento de rolagem horizontal.
int	scry	Deslocamento de rolagem vertical. Estes parâmetros sempre contém o deslocamento absoluto em pontos para o canto superior esquerdo da janela (como SET SCROLL em BASIC) e não é necessário se não houver barras de rolagem.

int	minx	Largura mínima da janela.
int	miny	Altura mínima da janela.
		Estes parâmetros contêm o tamanho mínimo da janela, que pode ser dimensionada pelo usuário, bloqueando apenas a criação de janelas menores. Não devem ser feitas alterações manualmente a partir de uma tarefa.
int	maxx	Largura máxima da janela.
int	maxy	Altura máxima da janela.
int	behind	Número handle da janela na frente (veja abaixo). Pode ser usado para verificar o nível da janela
T_TASK		Tarefa que criou a janela atual. Usado internamente para determinar qual tarefa deve ser informada se o usuário arrasta uma janela e para fechar todas as janelas relacionadas se a janela de uma tarefa for fechada.
char	winflag	sinalizadores bitmap da área da janela. bit 7: Não utilizados (deve ser 0). bit 6: Ainda não implementado (deve ser 0). bit 5: Ainda não implementado (deve ser 0). bit 4: Ainda não implementado (deve ser 0). bit 3: Auto-repetição em ícones de seta. bit 2: É janela-filho (veja 'parent' abaixo). bit 1: Janela pode ser arrastada (via barra de título). bit 0: 0 → Redesenho da janela necessário. 1 → O WiOS pode redesenhar toda a janela (ainda não implementado).
char	iconflag	Sinalizadores de ícones bitmap. bit 7: Não utilizado (deve ser 0). bit 6: Barra de rolagem horizontal. bit 5: Ícone de ajuste de tamanho. bit 4: Barra de rolagem vertical. bit 3: Ícone de alternância de tamanho. bit 2: Barra de título. bit 1: Ícone Fechar. bit 0: Ícone Voltar.
char	workflag	Sinalizadores de área de trabalho da janela bitmap. bit 7: Não utilizado (deve ser 0). bit 6: Não utilizado (deve ser 0). bit 5: Clique duplo notifica a tarefa.

	bit 4: Notificação de clique liberado sobre a área de trabalho da tarefa (para arrastar e soltar).
	bit 3: Clique notifica a tarefa (uma vez).
	bit 2: Clique notifica a tarefa (sempre).
	bit 1: Notificar a tarefa continuamente enquanto o ponteiro estiver sobre a área de trabalho.
	bit 0: Ignorar todos os cliques.
int parent	Identificador da janela pai. -1: É uma janela pai. 0~255: Identificador da janela pai se a janela for uma janela filho. As janelas filho não são movidas junto com a janela pai se esta for.
char statflag	Sinalizadores de estado de janela bitmap bits 7~2: Não utilizados (devem ser 0). bit 1: A janela está minimizada (ainda não implementada). bit 0: A janela está maximizada (ainda não implementada).
int realnx	Largura real da janela na tela.
int realny	Altura real da janela na tela.
char dummy[25]	Bytes são para preencher o bloco de dados da janela e são reservados para uso futuro. Devem ser preenchidos com 0.

## 7.8.2 – Funções do Window Driver

### CREATE\_WIN (Função ‘1’)

Descrição: Adicionar janela à lista e enviar evento aberto

Argumentos: uint Endereço dos argumentos do bloco de dados da janela:

x, y

nx, ny

vx, vy

scrx, scry

minx, miny

maxx, maxy

behind: -2 → atrás.

-1 → frente.

0~255 janela especificada

Se a janela for uma janela filho, ‘frente’ sempre significa no topo do bloco da

janela filho, e ‘atrás’ significa diretamente acima da janela pai (na parte de trás do bloco da janela filho).

winflag

iconflag

workflag

parent: -1 → É uma janela pai.

0~255 → Identificador da janela pai.

statflag: deve ser 0

dummy[25]: todos os 25 bytes devem ser 0.

Retorno: -1 → A janela não pôde ser criada.

0~255 → Identificador da nova janela.

Nota: Os parâmetros handle, task, realnx e realny são preenchidos pelo WiOS, não importa se eles têm dados válidos ou não. Esta função sempre cria uma nova janela após ‘CREATE\_WIN’ mas a janela NÃO é desenhada na tela. É apenas registrada e um evento ‘OPEN’ é enviado para a tarefa após o próximo ‘polling’.

### **OPEN\_WIN** (Função ‘2’)

Descrição: Altera os dados da janela existente na lista, calcula a nova posição e envia eventos de redesenho para todas as tarefas afetadas.

Argumentos: uint Ponteiro para dados da janela (veja acima).

Retorno: -1 → A janela não pôde ser aberta

1~255 → Número de janelas abertas (incluindo janelas filho).

Nota: Os dados da janela devem ser válidos e estar completos. Se o valor de trás for alterado, todas as janelas filho (se a janela for pai) também serão para que elas estejam sempre diretamente acima da janela pai.

### **CLOSE\_WIN** (Função ‘3’)

Descrição: Remove a janela e todas as janelas filho conectadas da lista e envia eventos de redesenho para as tarefas da janela.

Argumentos: uint 0~255 Identificador da janela a ser fechada.

Retorno: -1 → A janela não pôde ser excluída.

1~255 → Número de janelas fechadas (incluindo janelas filho)

Nota: Certifique-se de usar esta função apenas em janelas pertencentes à sua tarefa.

#### **GET\_WIN\_STATE** (Função ‘4’)

Descrição: Copia os dados da janela da lista para a memória da tarefa.

Argumentos: uint Endereço para onde os dados devem ser copiados. O parâmetro ‘handle’ deve conter o identificador da janela.

Retorno: 0000H~FFFEH → Endereço dos dados da janela (igual ao endereço enviado).  
-1 (FFFFH) → Janela não encontrada.

#### **DRAWFRAME** (Função ‘5’)

Descrição: Desenha a moldura da janela.

Argumentos: int Coordenada x de destino.  
int Coordenada y de destino.  
uint Ponteiro para o endereço da estrutura da janela.  
O parâmetro ‘handle’ deve conter o identificador da janela.  
char deve ser ‘0’.

Retorno: Nenhum.

Nota: Nenhuma verificação de validade é feita para o identificador de janela.

### **7.8.3 – Redesenhando Janelas**

#### **GETWIB** (Função ‘6’)

Descrição: Obtém o bloco de informações da janela a redesenhar.

Argumentos: uint 0~255 Identificador da janela.

Retorno: 0000H~FFFEH → Endereço da WIB.  
-1 (FFFFH) → Janela não encontrada.

Nota: Para a estrutura da WIB (window information block), consulte o capítulo sobre redesenho de janelas.

#### **COPYWIN** (Função ‘7’)

Descrição: Copia a área gráfica para a tela de exibição usando a pilha de redesenho.

Argumentos: uint Endereço da WIB.

Retorno: Nenhum.  
 Nota A VDA é alterada e a WIB deve estar fora das páginas 1 e 2 ('GETWIB' sempre retorna um valor válido)

**NONAME** (Função '8')  
 Descrição: Usada internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**DRW\_MENU** (Função '9')  
 Descrição: Desenha um menu correspondente aos dados da tarefa.  
 Argumentos: Ver seção "Referência ao Menu".  
 Retorno: Ver seção "Referência ao Menu".

**CHK\_MENU** (Função '10')  
 Descrição: Verifica as coordenadas e retorna o número do item.  
 Argumentos: –  
 Retorno: –1 → Nenhum item nestas coordenadas  
           0~32767 → Número do item.  
 Nota: Para informações detalhadas sobre menus e valores de retorno, consulte o capítulo "Referência ao Menu".

**NONAME** (Função '11')  
 Descrição: Usada internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**NONAME** (Função '12')  
 Descrição: Usada internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

**NONAME** (Função '13')  
 Descrição: Usada internamente.  
 Argumentos: Nenhum.  
 Retorno: Nenhum.

A rotina de redesenho precisa saber mais do que apenas os dados na estrutura da janela. Estes dados podem ser ordenados pelo driver

da janela e são chamados de bloco de informações da janela (window-information-block – WIB).

Possui a seguinte estrutura:

T_SEG	stackseg	Segmento da pilha de redesenho.
uint	elements	Número de áreas retangulares na pilha de redesenho
struct WINSTR	*windat	Ponteiro para dados da janela (fora da página 1 ou 2). Os elementos de janela necessários são: x,y nx, ny realx, realy
uint	sx,sy	coordenadas de início de cópia de janela (sempre 0, 212).
uint	offx, offyoffset	para área de trabalho.
int	moffx, moffy	Deslocamento de movimentação em relação à última posição.
uint	toffx, toffy	Deslocamento de título.
uint	t_nx, t Ny	Largura do título.

A pilha de redesenho tem a seguinte estrutura:

uint	x1,y1	Coordenada superior esquerda (absoluta).
uint	x2,y2	Coordenada inferior direita (absoluta).

A rotina de redesenho da tarefa deve ficar assim:

- Chamar ‘GETWIB’ e armazenar o endereço de retorno
- Se os elementos forem igual a 0, nada a fazer
- Desenhar os próprios dados nas coordenadas (sx+offx, sy+offy) que vêm com a função ‘GETWIB’.
- chamar ‘COPYWIN’.

## 7.9 – DEFINIÇÃO DE EVENTOS

Os nomes dos eventos estão em ‘EVENTFNC.H’.

**E\_NULL** (Função ‘0’)

Descrição: Evento nulo.

Argumentos: Nenhum.

**E\_REDRAW** (Função '1')

Descrição: Solicitação de redesenho de janela.

Argumentos: array[0] Identificador da janela a ser redesenhada.

**E\_SCROLL** (Função '2')

Descrição: Solicitação de rolagem da janela.

Argumentos: task Remetente da tarefa (esta tarefa).

array[0] Identificador da janela a ser rolada.

array[1] Desloc. horizontal de rolagem (absoluto).

array[2] Deslocamento vertical de rolagem (absoluto).

Nota: Um evento 'E\_WOPEN' é enviado imediatamente após o evento de rolagem para a tarefa. Ela só precisa determinar se os novos deslocamentos de rolagem são aceitos ou não e, se sim, atualizar os dados da janela. Tarefas que enviam uma solicitação de rolagem para janelas de outras tarefas devem sempre enviar o evento 'E\_WOPEN' imediatamente após a solicitação.

**E\_WOPEN** (Função '3')

Descrição: Solicitação de abertura de janela.

Argumentos: task Remetente da tarefa (esta tarefa).

array[0] Identificador da janela a ser aberta.

array[1] valor de trás.

array[2] Coord x do canto superior esquerdo da janela.

array[3] Coord y do canto superior esquerdo da janela.

array[4] Largura da área de trabalho.

array[5] Altura da área de trabalho.

Nota: Largura e altura são as dimensões da área de trabalho, não o tamanho real da janela.

**E\_WCLOSE** (Função '4')

Descrição: Solicitação de fechamento de janela.

Argumentos: task Remetente da tarefa (esta tarefa)

array[0] Identificador da janela a ser fechada.

**E\_PNTOUT** (Função '5')

Descrição: O ponteiro do mouse deixou a janela.

Argumentos: array[0] Identificador da janela.

**E\_PNTIN** (Função '6')

Descrição: O ponteiro entrou na janela.

Argumentos: array[0] Identificador da janela.

**E\_MCLICK** (Função '7')

Descrição: Clique do mouse.

Argumentos: array[0] Identificador da janela que foi clicada.

array[1] Coordenada x do clique

array[2] Coordenada y do clique

array[3] Tipo de clique de bitmap (=1 se for clicado):

0 → botão esquerdo é pressionado.

1 → botão esquerdo é pressionado.

2 → botão direito é pressionado.

3 → botão direito está segurado.

array[4] Tempo (contador de 16 bits) em que o clique foi realizado.

Nota: As coordenadas fornecidas são de onde o ponteiro estava na hora do clique.

**E\_EDRAG** (Função '8')

Descrição: Fim de uma operação de arrastar.

Argumentos: task Remetente da tarefa (esta tarefa)

array[0] Tipo de dados

array[1] Informações detalhadas sobre dados

Nota: Para o tipo de dados, consulte a seção “Definição dos tipos de dados”.

**E\_WKAREA\*** (Função '9')

Descrição: O ponteiro está sobre a área de trabalho (somente se o bit 1 do sinalizador da área de trabalho for '1')

Argumentos: array[0] Identificador da janela abaixo do ponteiro

**E\_MENU\*** (Função '10')

Descrição: Ainda não suportada.

Argumentos: Nenhum.

**E\_SHUTDOWN** (Função '11')

Descrição: Solicita que a tarefa seja encerrada.

Argumentos: Nenhum.

Nota: Não precisa ser manipulada.

**E\_USERMSG** (Função ‘16’)

Descrição: Definível pelo usuário.

Argumentos: task Remetente da tarefa (esta tarefa).

array[0] Código da mensagem do usuário.

array[1..5] Definíveis pelo usuário.

Nota: Deve ser manipulada mesmo que a tarefa não receba dados externos.

**7.9.1 – Dados enviados com o evento ‘E\_EDRAC’**

Os nomes dos tipos de dados entre colchetes são definidos em ‘DTDEF.H’ e devem ser enviados em array[0], o especificador em array[1].

Tipo de dados: 0 (‘TEXT’)

Especificador: 0 (‘STRING’)

Descrição: Uma ou mais strings de texto sem função especial.

Tipo de dados: 0 (‘TEXT’)

Especificador: 1 (‘FILENAME’)

Descrição: Uma ou mais strings de texto com nomes de arquivos a serem manipulados/editados/abertos.

Tipo de dados: 1 (GRÁFICO)

Especificador: 0~255 bits por pixel.

Descrição: Uma imagem de bitmap retangular (a resolução é definida nos próprios dados).

**7.9.2 – Dados enviados com o evento ‘E\_USERMSG’**

Sempre que enviar mensagens de usuário, array[0] mantém o código da mensagem. Ou seja, neste caso, código ‘DATA’, que é definido em ‘DTDEF.H’.

Dado: **STRING**

Argumentos: array[1] endereço do bloco de dados

array[2] segmento do bloco de dados

Bloco de dados

offset tam. Faixa Descrição

+0 2 0~65 535 Número de strings.

+2 2 1~3071 Número de segmentos usados para dados.

+4,6,... 2 0~3071 Nº de segmentos com strings.

Depois dos números do segmento:

2 0 Strings terminadas em nulo.

1~255 Número de campos.

- se as strings forem terminadas em nulo, elas virão imediatamente após o '0'.

- se as strings estiverem em campos de comprimento fixo, elas vêm após o comprimento do campo – 2 bytes de cada campo. Após os dados de comprimento vem o campo formatado de dados da string.

Nota: Se as strings excederem 16K, cabe ao programa trocar os segmentos.

Dado: **FILENAME**

array[1] endereço do bloco de dados

array[2] segmento do bloco de dados

Bloco de dados

offset tam. Faixa Descrição

+0 2 0~65535 Número de nomes de arquivos

Depois do número de nomes de arquivos:

Strings terminadas em nulo

Nota A estrutura de dados do nome do arquivo é igual à das strings terminadas em nulo, com a diferença de que a tarefa de destino não precisa inserir dados em algum lugar, mas sim abrir os arquivos. Também cabe à tarefa de destino analisar as extensões de nome de arquivo ou os arquivos e verificar se eles são válidos para a tarefa.

Dado: **GRÁFICO**

array[1] Endereço do bloco de dados.

array[2] Segmento do bloco de dados.

Bloco de dados

offset tam. Faixa Descrição

+0 2 Não utilizado.

+2 2 1~3071 Número de segmentos usados para a imagem.

+4,6,... 2 0~3071 Número de segmentos com dados de imagem.

Depois dos números do segmento:

2 0~65535 Largura da imagem.

2 0~65535 Altura da imagem.

Dados de imagem mapeados de 15 bits (consulte VDP-spec).

**Nota** Se uma imagem exceder 16k, cabe ao programa trocar os segmentos.

## 7.10 – REFERÊNCIAS GDA

Esta seção descreve a lista de variáveis GDA necessárias. A área GDA começa em C030H. Não há marcação de final, pois a GDA pode ser expandida no futuro. Variáveis reservadas não devem ser alteradas.

**IMPORTANTE:** Todas as variáveis GDA são ponteiros para o endereço da variável ou campo. As definições de tipo estão em 'DEF.H'.

Tipo	Nome	Offset	tam.	Descrição
T_SEG	* _tot_seg	+0	2	Número total de segmentos.
T_SEG	* _free_seg	+2	2	Número atual de segmentos livres.
T_SEG	* _tmp_seg	+4	2	Número do segmento temporário.
<não usado>		+6	2	
<uso interno>		+8	2	
T_TASK	* _p1_task	+10	2	Identif. da tarefa atual na página 1.
T_SEG	* _p1_seg	+12	2	Segmento atual na página 1.
char	* _p1rseg	+14	2	Nº do segmento atual na página 1.
char	* _p1_slt	+16	2	Slot do segmento atual na página 1.
<não usado>		+18	2	
T_SEG	* _p2_seg	+20	2	Nº do segmento atual na página 2.
char	* _p2rseg	+22	2	número do segmento atual do mapeador na página 2
char	* _p2_slt	+24	2	Slot do segmento atual na página 2.
<uso interno>		+26	2	
uint	(* _caldrv)(.)	+28	2	Endereço de entrada de chamada da função WiOS.
T_TASK	* _hmemdrv	+30	2	Identificador do driver de memória.
T_TASK	* _hstddrv	+32	2	Identificador do driver padrão.

T_TASK	* _htaskdrv	+34	2	Identificador do driver de tarefa.
T_TASK	* _hfsdrv	+36	2	Identificador do driver do sistema de arquivos.
T_TASK	* _hgiodrv	+38	2	Identificador do driver gráfico de E/S.
T_TASK	* _hgrpdrv	+40	2	Identificador do driver gráfico.
T_TASK	* _hwindrv	+42	2	Identificador do driver da janela.
T_TASK	* _hextdrv	+44	2	Identificador do driver externo.
uint	* _wiosver	+46	2	Versão atual da parte interna do WiOS.
VOID	(* _wiosend)()	+48	2	Chamada de desligamento do WiOS (somente na versão Alpha).
uint	(*_poll)()	+50	2	Endereço de entrada de “polling”.
<uso interno>		+52 até +106		
VOID	(* _dump)()	+108	2	Entrada para fazer um dump de memória no V9958 (argumento é endereço).
struct	* _eventblk	+110	2	Endereço do bloco de eventos EBSTR que contém os dados após o “polling”.
struct				Endereço do bloco de informações da janela.
WIBSTR	* _wibblk	+112	2	Endereço do bloco de dados da janela.
struct				Identificador da fonte atual
WINSTR	* _winblk	+114	2	Identif. da fonte padrão ('system')
uint	* _act_font	+116	2	
uint	* _std_font	+118	2	
<uso interno>		+120	2	
char	* _busyflag	+122	2	Estado do VDP (gravação direta)
uint	* _adrblk	+124	2	Endereço de chave de segmento global (bloco de endereço independente).
char	* _mb	+126	2	Estado atual dos botões do mouse: bit 0 – estado do botão esquerdo (1=pressionado). bit 1 – estado do botão direito (1=pressionado).
char	* _cltype	+128	2	Tipo de último clique do mouse: 0 – clique único esquerdo (1 = clicado). 1 – duplo clique esquerdo (1=clicado). 2 – clique único com o botão direito (1=clicado). 3 – clique duplo com o botão direito (1=clicado).

uint	* _cltime	+130	2	Tempo de clique (matriz de 2): offset descrição +0 Hora do último clique com o botão esquerdo +2 Hora do último clique com o botão direito.
int	* _coord	+132	2	Coordenadas atuais do mouse (matriz de 2).
uint	(* _cal_seg)()	+134	2	Ponto de entrada para as tarefas chamarem funções multipartes
<uso interno>		+136	2	
uint	(* _dcal_seg)()	+138	2	Ponto de entrada para os drivers chamarem funções multipartes
<uso interno>		+140	2	
<uso interno>		+142	2	

Esta lista pode ser expandida no futuro.

## 7.11 – ESTRUTURA DO BLOCO DE MENU

O bloco de menu é um grande campo de dados onde um ou mais tipos de menu podem ser definidos.

Os seguintes símbolos são usados na definição da estrutura:

### prefixo

- b Byte (1 byte).
- w Palavra (2 bytes).
- t Texto (comprimento de byte variável).
- # Definição de valores válidos para ‘D’ (ver sufixo).

### sufixo

- N Qualquer número.
- D Número de uma lista predefinida.
- S Cadeia de caracteres (string).
- 0 Deve ser este valor.

Estilos de borda válidos:

- 0 Sem borda desenhada.
- 1 Caixa pintada com cor de fundo.

## Definição

bN Número de blocos de itens

tipo

bD

#0 → Opção

Uma vez para cada bloco de opções:

wN Font-handle.

wN Deslocamento x.

wN Deslocamento y.

w0 Largura do bloco de opções  
(preenchido pelo WiOS).

w0 Altura do bloco de opções  
(preenchido pelo WiOS).

bD Estilo de borda.

bN Distância da borda em pixels.

bD Direção.

#0 De cima para baixo.

#1 Da esquerda para a direita.

Para cada entrada:

bD Sinalizadores de entrada (bitmap)

bit (=0) (=1)

selecões válidas para entradas de estado	0	Fim da lista	Entrada válida
	1	Desabilitado	Habilitado (selecionável)
	2	Não selec.	Selecionada
	3~7	Não utilizados (deve ser reiniciado)	
tS String terminada em nulo		continua em Entry Flags (até que o bit 0 seja resetado).	

#1 → Rádio

Uma vez para cada bloco de rádio:

wN Font-handle.

wN Deslocamento x.

wN Deslocamento y.

w0 Largura do bloco de opções  
(preenchido pelo WiOS).

w0 Altura do bloco de opções  
(preenchido pelo WiOS).

bD	Estilo de borda.
bN	Distância da borda em pixels.
bD	Direção.
#0	De cima para baixo.
#1	Da esquerda para a direita.

Para cada entrada:

bD	Sinalizadores de entrada (bitmap)
	bit            (=0)                    (=1)

selecções	0	fim da lista	entrada válida
válidas para	1	desabilitado	habilitado (selecionável)
entradas de	2	não selecionada	selecionada
estado	3~7	Não utilizados (deve ser reiniciado).	

tS	String terminada em nulo
----	--------------------------

Continua em Entry Flags (até que o bit 0 seja resetado).

#2	→	Lista
----	---	-------

Uma vez para cada bloco de lista

wN	font-handle
wN	x-offset
wN	deslocamento y
w0	largura do bloco de opções (preenchido por WiOS)
w0	altura do bloco de opções (preenchido por WiOS)
bD	estilo de borda
bN	distância da borda em pixels

Para cada entrada:

bD	Sinalizadores de entrada (bitmap)
	bit            (=0)                    (=1)

selecções	0	Fim da lista	Entrada válida
válidas para	1	Desabilitado	Habilitado (selecionável)
entradas de	2	Sem sub-lista	Sub-lista(seta à direita)
sub-lista	3~7	Não utilizados (deve ser reiniciado)	

tS	String terminada em nulo
----	--------------------------

Continua em Entry Flags (até que o bit 0 seja resetado).

#3	→	Barra de rolagem
----	---	------------------

bD	Sinalizadores de entrada (bitmap)
	bit            (=0)                    (=1)

--	--

Estado de entrada:	0	inválido	válido
	wN	Deslocamento x	
	wN	Deslocamento y	
	wN	Tamanho visível do campo em pixels	
	wN	Tamanho virtual do campo em pixels	
	wN	Tamanho real do campo em pixels	
	wN	Deslocamento de rolagem da barra de rolagem em pixels	
	bD	Direção	
	#0	De cima para baixo	
	#1	Da esquerda para a direita	
#4 → Caixa			
	bD	Sinalizadores de entrada (bitmap)	
	bit	(=0)	(=1)
Estado de entrada:	0	inválido	válido
	wN	x-offset	
	wN	deslocamento y	
	wN	largura	
	wN	altura	
	bD	estilo de borda	
	bN	distância da borda em pixels	
	bN	distância da borda em pixels	

## 8 – VARIÁVEIS DE SISTEMA

### 8.1 – ÁREA DE SISTEMA PARA O MSXDOS1

**F1C1H, 1** – Contador regressivo para os drives. Colocando este contador em 0, os motores dos drives são parados.

**F1C2H, 1** – Subcontador do contador regressivo para o drive.

**F1C3H, 1** – Subcontador do contador regressivo para o drive.

**F1C4H, 1** – Número do drive atualmente ativo.

**F1C5H, 1** – Número da trilha onde a cabeça do drive A: está.

**F1C6H, 1** – Número da trilha onde a cabeça do drive B: está.

**F1C7H, 1** – Drive lógico ativo.

**F1C8H, 1** – Número de drives físicos presentes.

**F1C9H, 24** – Rotina para impressão na tela de uma string terminada por “\$”. DE – Endereço inicial da string.

**F1CAH~F1E1H – ?**

**F1E2H, 6** – Rotina para abortar o programa em caso de erro.

**F1E8H, 12** – Chama o endereço apontado por (HL) na RAM e retorna com a página do DOS Kernel (BDOS) ativa.

**F1F4H, 3** – Jump para a rotina de checagem do nome de arquivo.  
HL – Endereço do primeiro caractere do nome de arquivo.

**F1F7H, 4** – Nome de dispositivo “PRN”.

**F1FBH, 4** – Nome de dispositivo “LST”.

**F1FFH, 4** – Nome de dispositivo “NUL”.

**F203H, 4** – Nome de dispositivo “AUX”.

**F207H, 4** – Nome de dispositivo “CON”.

**F20BH, 11** – Reservado para novos nomes de dispositivos ou arquivos.

**F216H, 1** – Número do dispositivo atual:

–5 → PRN; –4 → LST; –3 → NUL; –2 → AUX; –1 → CON.

**F217H~F220H – ?**

**F221H, 2** – Data do FCB do arquivo atual.

**F223H, 2** – Hora do FCB do arquivo atual.

**F22BH, 12** – Tabela contendo o número de dias dos meses do ano.

F22BH [31] Janeiro	F231H [31] Julho
F22CH [28] Fevereiro	F232H [31] Agosto
F22DH [31] Março	F233H [30] Setembro
F22EH [30] Abril	F234H [31] Outubro
F22FH [31] Maio	F235H [30] Novembro
F230H [30] Junho	F236H [31] Dezembro

**F237H, 4** – Usada internamente pela função 10 do BDOS.

**F23BH, 1** – Flag para indicar se os caracteres devem ir para a impressora. (0=não; outro valor, sim)

**F23CH, 2** – Endereço atual da DTA.

**F23EH, 1** – ?

**F23FH, 4** – Número do setor atual do disco.

**F243H, 2** – Apontador para o endereço do DPB do drive atual.

**F245H, 1** – Setor atual relativo do diretório a partir do primeiro (0).

**F246H, 1** – Drive que contém o setor atual do diretório (0=A:, 1=B:, etc.)

**F247H, 1** – Drive corrente (0=A:, 1=B:, etc)

**F248H, 1** – Dia

**F249H, 1** – Mês

**F24AH, 1** – Ano-1980 (somar 1980 para obter o ano correto)

**F24BH, 1** – ?

**F24CH, 2** – Hora e minutos

**F24EH, 1** – Dia da semana (0=domingo, 1=segunda, etc.)

### **8.1.1 – Hooks chamados pelas rotinas de disco**

**F24FH, 3** – Rotina que apresenta a mensagem “Insert disk for drive”.

A – Número do drive (41H=A:, 42H=B:, etc)

- F252H, 3** – Obtém o conteúdo da FAT.
- F255H, 3** – Rotina de reparação do nome de arquivo.
- F258H, 3** – Rotina de procura de diretório.
- F25BH, 3** – Incrementa a entrada do diretório (última entrada em A).
- F25EH, 3** – Rotina que calcula o próximo setor do diretório.
- F261H, 3** – Rotina de reparação do nome de arquivo.
- F264H, 3** – Rotina da função 'OPEN'.
- F267H, 3** – Retorna a última FAT.
- F26AH, 3** – Rotina 'GETDPB' da interface de disco (SFIRST).
- F26DH, 3** – Rotina da função 'CLOSE' (escreve FAT).
- F270H, 3** – Rotina da função 'RDABS - 2FH' (HL=DMA, DE=setor, B=nº setores). H.DISKREAD.
- F273H, 3** – Rotina de manipulação de erro no acesso ao disco.
- F276H, 3** – Rotina da função 'WRABS' (grava setor).
- F279H, 3** – Rotina da função 'WRABS' (HL=DMA, DE=setor, B=nº setores).
- F27CH, 3** – Rotina de multiplicação (HL = DE \* BC).
- F27FH, 3** – Rotina de divisão (BC = BC / DE; HL = resto).
- F282H, 3** – Retorna o cluster absoluto.
- F285H, 3** – Retorna o próximo cluster absoluto.
- F288H, 3** – Leitura de setor do disco.
- F28BH, 3** – Escrita de setor no disco.
- F28EH, 3** – Inicia a operação de leitura de blocos (records) do disco.
- F291H, 3** – Finaliza a operação de leitura de blocos (records) do disco.
- F294H, 3** – Fim da operação de leitura de blocos (records) do disco.
- F297H, 3** – Erro na operação com blocos (records).
- F29AH, 3** – Inicia a operação de escrita de blocos (records) do disco.
- F29DH, 3** – Finaliza a operação de escrita de blocos (records) do disco.

**F2A0H, 3** – Calcula setores sequenciais.

**F2A3H, 3** – Obtém o número de setores de um cluster.

**F2A6H, 3** – Aloca uma sequência de FAT's.

**F2A9H, 3** – Libera uma sequência de FAT's.

**F2ACH, 3** – Função 'BUFIN' (adiciona dados no buffer).

**F2AFH, 3** – Função 'CONOUT' (BDOS 02H).

**F2B2H, 3** – Obtém a hora e a data do arquivo.

**F2B5H, 3** – Rotina de identificação do mês de fevereiro (28/29 dias).

### 8.1.2 – Outros dados do DOS

**F2B8H, 1** – Número da entrada atual do diretório.

**F2B9H, 11** – Nome de arquivo/extensão do arquivo atual.

**F2C4H, 1** – Byte de atributos do arquivo da última entrada do diretório lida. Se o bit 7 estiver setado, os arquivos com um atributo NOT para 0 podem ser abertos. Isso pode ser feito setando o bit 7 do byte do FCB-drive, ao chamar a rotina BDOS OPEN. (FCB+0).

**F2C5H~F2CEH** – ?

**F2CFH, 2** – Hora do arquivo atual.

**F2D1H, 2** – Data do arquivo atual.

**F2D3H, 2** – Cluster inicial do arquivo atual.

**F2D5H, 4** – Tamanho do arquivo atual.

**F2D9H~F2DBH** – ?

**F2DCH, 1** – Arquivos com atributos diferentes de F2DCH também são aceitos (O bit-7 de F2C4H tem prioridade).

**F2DDH~F2E0H** – ?

**F2E1H, 1** – Drive atual para escrita e leitura absoluta de setores.

**F2E2H~F2FDH** – ?

**F2FEH, 2** – Subcontador do contador regressivo para o drive.

**F301H, 1** – ?

**F302H, 2** – Apontador para o manipulador da rotina de abortagem para o MSXDOS.

**F304H, 2** – Armazena o valor do registrador SP (Stack Pointer).

**F306H, 1** – Drive corrente para o MSXDOS (0=A:, 1=B:, etc).

**F307H, 2** – Armazena o valor do registrador DE (Endereço do FCB).

**F309H, 2** – Usado pelo DPB para procura (primeiro/próximo).

**F30BH, 2** – Setor atual do diretório.

**F30DH, 1** – Flag de verificação (0=desligada; outro valor, ligada).

**F30EH, 1** – Formato da data (0- aammdd; 1- mmddaa; 2- ddmmaa).

**F30FH, 4** – Área usada pelo modo Kanji.

**F313H, 1** – Versão do MSXDOS:

0 = MSXDOS 1.x; 20H = MSXDOS 2.0; 21H = MSXDOS 2.1; etc.

Obs.: o NEXTOR retorna 99H.

**F314H~F322H** – ?

**F323H, 2** – Endereço do manipulador de erro de disco.

**F325H, 2** – Endereço do manipulador das teclas CTRL+C.

### 8.1.3 – Hooks para a porta 'COM':

**F327H, 5** – Rotina 'AUXINP' (A=byte lido do dispositivo AUX).

**F32CH, 5** – Rotina 'AUXOUT' (A=byte a ser enviado ao disp. AUX).

**F331H, 5** – Rotina de manipulação das funções do BDOS.

### 8.1.4 – Teclado

**F336H, 1** – Sinalizador de tecla pressionada. Contém FFH se alguma tecla estiver pressionada e 03H para CTRL+STOP.

**F337H, 5** – Contém o código ASCII da tecla pressionada e 03H para CTRL+STOP pressionadas juntas.

### 8.1.5 – Variáveis do MSXDOS

**F338H, 1** – Flag para indicar a presença de relógio interno (0=não; outro valor, sim).

**F339H, 7** – Rotina usada pelo relógio interno.

**F340H, 1 – REBOOT**

Se for 0, o DOS reinicializará todas as variáveis novamente.

**F341H, 1 – RAMAD0**

Slot da página 0 da RAM (formato igual a RDSL – 000CH/BIOS).

**F342H, 1 – RAMAD1**

Slot da página 1 da RAM (formato igual a RDSL – 000CH/BIOS).

**F343H, 1 – RAMAD2**

Slot da página 2 da RAM (formato igual a RDSL – 000CH/BIOS).

**F344H, 1 – RAMAD3**

Slot da página 3 da RAM (formato igual a RDSL – 000CH/BIOS).

**F345H, 1 – Número de buffers livres (025H)**

**F346H, 1 – Flag para indicar se o sistema foi inicializado a partir do MSXDOS em disquete. (0=não; outro valor, sim)**

**F347H, 1 – NMBDRV**

Número total de drives lógicos no sistema.

**F348H, 1 – MASTER**

ID do slot do DOS Kernel (formato igual a RDSL – 000CH/BIOS).

**F349H, 2 – HIMSAV**

Apontador para uma cópia da FAT do último drive lógico conectado (1,5 Kbytes) seguida de uma cópia da FAT do penúltimo drive lógico conectado (1,5 Kbytes) e assim sucessivamente, até o drive A:. Também indica a área mais alta de memória disponível para o DOS.

**F34BH, 2 – Endereço final do Kernel do MSXDOS (início para o COMMAND.COM).** O endereço inicial do Kernel do MSXDOS é armazenado em 0006H/0007H.

**F34DH, 2 – SECBUF**

Apontador para uma cópia da FAT do drive corrente (1,5K).

**F34FH, 2 – BUFFER**

Apontador do buffer de 512 bytes usado como DTA do Disk-BASIC.

**F351H, 2 – DIRBUF**

Apontador para um buffer de 512 bytes usado para transferência de setores do disco (usado por DSKI\$ e DSKO\$ do BASIC).

### **8.1.6 – Endereços do DPB**

#### **F353H, 2 – DPBBASE**

Apontador para o DPB do arquivo atual.

#### **F355H, 16 – DPBLIST**

F355H, 2 – Endereço do DPB do drive A:  
 F357H, 2 – Endereço do DPB do drive B:  
 F359H, 2 – Endereço do DPB do drive C:  
 F35BH, 2 – Endereço do DPB do drive D:  
 F35DH, 2 – Endereço do DPB do drive E:  
 F35FH, 2 – Endereço do DPB do drive F:  
 F361H, 2 – Endereço do DPB do drive G:  
 F363H, 2 – Endereço do DPB do drive H:

### **8.1.7 – Rotinas usadas pelo MSXDOS**

#### **F365H, 3 – Jump da rotina de leitura de slots primários.**

(A ← estado do slot primário)

#### **F368H, 3 – SETROM**

Jump para a rotina que coloca o DOS Kernel (BDOS) na página 1  
 (não disponível a partir do Disk BASIC)

#### **F36BH, 3 – SETRAM**

Jump para a rotina que coloca a RAM na página 1 (não disponível a partir do Disk BASIC).

### **8.1.8 – Rotinas de movimento inter-slot**

#### **F36EH, 3 – SLTMOV**

Jump para LDIR da RAM na página 1 (não disponível a partir do Disk BASIC).

#### **F371H, 3 – AUXINP**

Jump para a rotina de entrada do dispositivo auxiliar.  
 Saída: A ← valor lido (1AH quando CTRL+Z).

#### **F374H, 3 – AUXOUT**

Jump para a rotina de saída do dispositivo auxiliar.  
 Entrada: A ← valor a enviar.

**F377H, 3 – BLDCCHK**

Jump para a rotina do comando 'BLOAD'. O endereço apontado por F378H/F379H é o endereço mais alto de RAM disponível para o Disk BASIC. Contém JP 0000H sob MSXDOS.

**F37AH, 3 – BSVCHK**

Jump para a rotina do comando 'BSAVE' ( Contém JP 0000H sob MSXDOS). Entrada: c ← número da rotina a chamar.

**F37DH, 3 – ROMBDOS**

Jump para manipulador dos comandos do BDOS.

\*\*\* Consulte também os endereços F85FH a F87EH e FB20H a FB34H.

**8.2 – ÁREA DE SISTEMA PARA O MSXDOS2****8.2.1 – Informações físicas dos discos**

**F1C1H, 1** – Contador regressivo para os drives. Setando esse contador em 0, os motores dos drives são parados.

**F1C2H, 1** – Subcontador do contador regressivo para o drive.

**F1C3H, 1** – Subcontador do contador regressivo para o drive.

**F1C4H, 1** – Número do drive atualmente ativo.

**F1C5H, 1** – Número da trilha onde a cabeça do drive A: está.

**F1C6H, 1** – Número da trilha onde a cabeça do drive B: está.

**F1C7H, 1** – Drive lógico ativo.

**F1C8H, 1** – Número de drives físicos presentes.

**8.2.2 – Hooks chamados pelas rotinas de disco (1)**

**F1C9H, 24** – Rotina para impressão na tela de uma string terminada por "\$". DE ← endereço inicial da string.

**F1E2H~F1E4H – ?**

**F1E5H, 3** – Jump para o manipulador de interrupção (somente durante o processamento das funções do BDOS).

**F1E8H, 3** – Jump para a rotina do BIOS 'RDSLت-000CH' (somente durante o processamento das funções do BDOS).

**F1EBH, 3** – Jump para a rotina do BIOS ‘WRSLT-0014H’ (somente durante o processamento das funções do BDOS).

**F1EEH, 3** – Jump para a rotina do BIOS ‘CALSLT-001CH’ (somente durante o processamento das funções do BDOS).

**F1F1H, 3** – Jump para a rotina do BIOS ‘ENASLT-0024H’ (somente durante o processamento das funções do BDOS).

**F1F4H, 3** – Jump para a rotina do BIOS ‘CALLF-0030H’ (somente durante o processamento das funções do BDOS).

**F1F7H, 3** – Jump para a rotina de troca para o “Modo DOS” (páginas 0 e 2 para os segmentos do sistema).

**F1FAH, 3** – Jump para a rotina de troca para o “Modo Usuário”.

**F1FDH, 3** – Jump para a rotina que seleciona os segmentos do DOS Kernel na página 1.

**F200H, 3** – Jump para a rotina que aloca um segmento de 16 Kbytes de RAM.

**F203H, 3** – Jump para a rotina que libera um segmento de 16 Kbytes de RAM.

**F206H, 3** – Jump para a rotina do BIOS ‘RDSLت-000CH’.

**F209H, 3** – Jump para a rotina do BIOS ‘WRSLT-0014H’.

**F20CH, 3** – Jump para a rotina do BIOS ‘CALSLT-001CH’.

**F20FH, 3** – Jump para a rotina do BIOS ‘CALLF-0030H’.

**F212H, 3** – Jump para a rotina que coloca segmento de 16 Kbytes na página indicada por HL.

**F215H, 3** – Jump para a rotina que lê página do segmento de 16 Kbytes atual. HL ← página lida.

**F218H, 3** – Jump para a rotina que habilita segmento de 16 Kbytes da memória mapeada na página 0.

**F21BH, 3** – Jump para a rotina que lê segmento atual de 16 Kbytes da memória mapeada na página 0.

**F21EH, 3** – Jump para a rotina que habilita segmento de 16 Kbytes da memória mapeada na página 1.

**F221H, 3** – Jump para a rotina que lê segmento atual de 16 Kbytes da memória mapeada na página 1.

**F224H, 3** – Jump para a rotina que habilita segmento de 16 Kbytes da memória mapeada na página 2.

**F227H, 3** – Jump para a rotina que lê segmento atual de 16 Kbytes da memória mapeada na página 2.

**F22AH, 3** – A página 3 não suporta mudança de segmento.

**F22DH, 3** – Jump para a rotina que lê segmento atual de 16 Kbytes da memória mapeada na página 3.

**F230H~F23BH** – ?

### 8.2.3 – Informações lógicas dos discos

**F23CH, 1** – Drive lógico atual (0=A:, 1=B:, etc.).

**F23DH, 2** – Endereço atual da DTA.

**F23FH, 4** – Número do setor atual para acesso.

**F243H, 2** – Endereço do DPB do drive atual.

**F245H, 1** – Número relativo do setor atual da área do diretório.

**F246H, 1** – Número do drive do diretório atual (0=A:, 1=B:, etc.).

**F247H, 1** – Número do drive corrente (0=A:, 1=B:, etc.).

**F248H, 3** – Dia / F248H+1 = Mês / F248H+2 = Ano-1980 (Somar 1980 para obter o ano correto)

**F24BH, 1** – ?

**F24CH, 2** – Hora

**F24EH, 1** – Dia da semana

### 8.2.4 – Hooks chamados pelas rotinas de disco (2)

**F24FH, 3** – H.PROM

Jump para a rotina que apresenta a mensagem “Insert disk for drive”. A ← número do drive (41H=A; 42H=B; etc)

**F252H, 3** – Hook chamado antes da execução de uma função do BDOS. Página 0 ← mapa do bloco (F2D0H). Página 2 ← mapa do bloco (F2CFH).

**F255H, 3** – Hook da rotina de reparação de nome de arquivo.

**F258H, 3** – Hook da rotina de manipulação de subdiretórios do Disk BASIC. Usado por várias outras rotinas.

**F25BH, 3** – Hook da rotina que incrementa a entrada de diretório. A nova entrada é armazenada em AF.

**F25EH, 3** – Hook da rotina que carrega o próximo setor do diretório.

**F261H, 3** – Hook da função 02H do BDOS.

**F264H, 3** – Rotina OPEN

**F267H, 3** – Retorna a última FAT

**F26AH, 3** – Procura pelo primeiro FCB (SFIRST)

**F26DH, 3** – Escreve a FAT.

**F270H, 3** – Hook da rotina de leitura direta de setores (função 2FH do BDOS). HL ← DMA, DE ← setor inicial, B ← nº de setores.

**F273H, 3** – Erro de disco.

**F276H, 3** – Hook de escrita no setor de subdiretório (pasta)

**F279H, 3** – Hook da rotina de escrita direta de setores (função 30H do BDOS). HL ← DMA, DE ← setor inicial, B ← nº de setores.

**F27CH, 3** – Hook da rotina de multiplicação ( $HL = DE * BC$ ).

**F27FH, 3** – Hook da rotina de divisão ( $BC = BC / DE$ ;  $HL = \text{resto}$ ).

**F282H~F282H** – ?

## 8.2.5 – Variáveis do MSXDOS2

**F2B3H, 2** – Endereço da TPA definido pelo usuário. Os 32 bytes iniciais da TPA são usados para funções especiais:

*Off set*      *Descrição*

00H~02H      Reservados

03H      Usado pelo VDP speed (bit 3 de F2B6H)

04H~1FH      Reservados

20H      Expansão do BDOS e rotinas de interrupção

**F2B5H, 1** – ?

**F2B6H, 1 – Byte de flags:**

- b0~b2 – Reservados
- b3 – VDP rápido (0=sim; 1=não)
- b4 – Endereço TPA usuário (0=sim; 1=não)
- b5 – Reset (0=não; 1=sim)
- b6 – BusReset (0=sim; 1=não)
- b7 – Reboot (0=não; 1=sim)

**F2B7H, 1 – Número da versão (normalmente 10H = v1.0).**

**F2B8H, 1 – Número da entrada atual do diretório.**

**F2B9H~F2BFH – ?**

**F2C0H, 5 – Segundo hook da rotina de interrupção (usado pela Disk-ROM).**

**F2C5H, 2 – Endereço da tabela de mapeamento.**

**F2C7H, 1 – Página lógica atual da mapper na página física 0.**

**F2C8H, 1 – Página lógica atual da mapper na página física 1.**

**F2C9H, 1 – Página lógica atual da mapper na página física 2.**

**F2CAH, 1 – Página lógica atual da mapper na página física 3 (não pode ser alterado).**

**F2CBH, 1 – Cópia de F2C7H durante a execução das rotinas do BDOS.**

**F2CCH, 1 – Cópia de F2C8H durante a execução das rotinas do BDOS.**

**F2CDH, 1 – Cópia de F2C9H durante a execução das rotinas do BDOS.**

**F2CEH, 1 – Cópia de F2CAH durante a execução das rotinas do BDOS.**

**F2CFH, 1 – Número do último bloco de 16K disponível da memória mapeada. Durante a execução das rotinas do BDOS, os blocos são trocados na página 2 (segmento de buffer).**

**F2D0H, 1 – Número do último bloco de 16K disponível da memória mapeada. Durante a execução das rotinas do BDOS, os blocos são trocados na página 0 (segmento de código).**

**F2D1H~F2D4H – ?**

**F2D5H, 5 – Segundo hook EXTBIO (rotina do hook FCALL [FFCAH]).**

**F2DAH, 4 – Endereço da segunda ROM do BDOS para manipulação de funções.**

**F2DEH, 4** – Endereço da ROM do BDOS para manipulação de funções.

**F2E2H~F2E5H** – ?

**F2E6H, 2** – Buffer usado para armazenamento temporário do registrador IX.

**F2E8H, 2** – Buffer usado para armazenamento temporário do registrador SP.

**F2EAH, 1** – Estado dos slots primários após a execução de uma função do BDOS.

**F2EBH, 1** – Mesmo que F2EAH, mas para slots secundários

**F2ECH, 1** – Flag para checagem do status do disco. (00H=off, FFH=on).

**F2EDH~F2FAH** – ?

**F2FBH, 2** – Apontador para um buffer temporário durante a interpretação de um código de erro.

**F2FDH, 1** – Drive do qual o MSXDOS2.SYS deverá ser carregado.  
(01H=A:, 02H=B:, etc).

**F2FEH, 2** – Endereço do topo da pilha do buffer do DOS.

**F300H, 1** – Flag de verificação (00H=off, FFH=on).

**F301H~F30CH** – ?

**F30DH, 1** – Flag de verificação do disco (00H=off, FFH=on).

**F30EH~F312H** – ?

**F313H, 1** – Versão do DOS2 (ex.: 22H = v2.2). O NEXTOR retorna 99H.

**F314H~F322H** – ?

**F323H, 2** – DISKVE

Endereço do manipulador de erro de disco.

**F325H, 2** – BREAKV

Endereço do manipulador das teclas CTRL+C.

**F327H~F33CH** – ?

**F33DH, 3** – Jump para o comando BASIC ‘LEN’ (acesso aleatório a arquivos).

**F341H, 1** – RAMAD0

Slot da página 0 da RAM (formato igual a ‘RDSL’ – 000CH/Main).

**F342H, 1 – RAMAD1**

Slot da página 1 da RAM (formato igual a 'RDSLT' - 000CH/Main).

**F343H, 1 – RAMAD2**

Slot da página 2 da RAM (formato igual a 'RDSLT' – 000CH/Main).

**F344H, 1 – RAMAD3**

Slot da página 3 da RAM (formato igual a 'RDSLT' - 000CH/Main).

**F345H, 1 – ?****F346H, 1 – MSXDOS**

Flag para indicar se o sistema foi inicializado a partir do MSXDOS em disquete. (0=não; outro valor, sim)

**F347H, 1 – ?****F348H, 1 – MASTER**

ID do slot do DOS Kernel primário (master). No caso do DOS2 é a interface primária que contenha a ROM do DOS2. O formato é igual a RDSLT – 000CH/BIOS).

**8.2.6 – Apontadores e buffers (FAT, DTA, FCB, DPB)****F349H, 2 – HIMSAV**

Apontador para uma cópia da FAT do último drive lógico conectado (1,5 Kbytes) seguida de uma cópia da FAT do penúltimo drive lógico conectado (1,5 Kbytes) e assim sucessivamente, até o drive A:. Também indica a área mais alta de memória disponível para o usuário.

**F34BH~F34CH – ?****F34DH, 2 – SECBUF**

Apontador para uma cópia da FAT do drive corrente (1,5 Kbytes).

**F34FH, 2 – BUFFER**

Apontador para uma área de 512 bytes usada como DTA do Disk-BASIC.

**F351H, 2 – DIRBUF**

Apontador para um buffer de 512 bytes usado para transferência de setores do disco.

**F353H, 2 – FCBBASE**

Apontador para o FCB do arquivo atual.

**F355H, 16 – DPBLIST**

Lista de apontadores para os DPB's de todos os oito drives possíveis, reservando dois bytes para cada um.

F355H, 2 ← drive A:	F35DH, 2 ← drive E:
F357H, 2 ← drive B	F35FH, 2 ← drive F:
F359H, 2 ← drive C:	F361H, 2 ← drive G:
F35BH, 2 ← drive D:	F363H, 2 ← drive H:

**F364H~F377H – ?****8.2.7 – Jumps do sistema****F378H – BLDCHK+1**

Endereço da rotina do manipulador do comando 'BLOAD'.

**F37AH, 3 – Jump secundário para o segmento de sistema na pág. 0.****F37DH – BDOS**

Jump para o manipulador de funções do BDOS.

\*\*\* Consulte também os endereços F85FH a F87EH e FB20H a FB34H.

**8.3 – SUBROTINAS INTER-SLOT****RDPRIM (F380H)**

Função: Lê um byte de qualquer endereço de qualquer slot.

Entrada: A – Slot primário a ser lido.

D – Slot atual para retorno.

Saída: E – Byte lido.

Código: F380H RDPRIM: OUT (0A8H), A  
                   F382H           LD     E, (HL)  
                   F383H           JR     WRPRM1

**WRPRIM (F385H)**

Função: Escreve um byte em qualquer endereço de qualquer slot.

Entrada: A – Slot primário a ser lido.

D – Slot atual para retorno.

E – Byte a ser escrito.

Saída: Nenhuma

Código: F385H WRPRIM: OUT (0A8H), A  
                   F387H           LD     (HL), E  
                   F388H   WRPRM1: LD     A, D  
                   F389H           OUT    (0A8H), A

**CLPRIM** (F38CH)

Função: Chama um endereço em qualquer slot.  
 Entrada: A – Slot primário que contém a rotina.  
 IX – Endereço a ser chamados.  
 PUSH AF – Slot atual para retorno (em A).  
 Saída: Depende da rotina chamada.

Código:

F38CH	CLPRIM:	OUT	(0A8H), A
F38EH		EX	AF, AF'
F38FH		CALL	CLPRM1
F392H		EX	AF, AF'
F393H		POP	AF
F394H		OUT	(0A8H), A
F396H		EX	AF, AF'
F397H		RET	
F398H	CLPRM1:	JP	(IX)

**8.4 – FUNÇÃO USR E MODOS TEXTO****USR TAB** (F39AH, 20)

Valor inicial: FCERR

Conteúdo: São dez variáveis de sistema de dois bytes cada que apontam para o endereço de execução de uma rotina assembly a ser chamada pela função USR. A primeira posição aponta para USR0, a segunda para USR1 e assim por diante. O valor inicial aponta para a rotina do gerador de erro.

**8.5 – ÁREA USADA PELA TELA****LINL40** (F3AEH, 1)

Valor inicial: 39

Conteúdo: Largura da tela no modo texto Screen 0.

**LINL32** (F3AFH, 1)

Valor inicial: 29

Conteúdo: Largura da tela no modo texto Screen 1.

**LINLEN** (F3B0H, 1)

Valor inicial: 39

Conteúdo: Largura atual da tela de texto.

**CRTCNT** (F3B1H, 1)

Valor inicial: 24

Conteúdo: Número de linhas nos modos de texto.

**CLMSLT** (F3B2H, 1)

Valor inicial: 14

Conteúdo: Localização horizontal no caso de itens divididos por vírgula no comando PRINT.

### 8.5.1 – Screen 0

**TXTNAM** (F3B3H, 2)

Valor inicial: 0000H

Conteúdo: Endereço na VRAM da tabela de nomes dos padrões.

**TXTCOL** (F3B5H, 2)

Valor inicial: 0000H

Conteúdo: Variável não usada

**TXTCGP** (F3B7H, 2)

Valor inicial: 0800H

Conteúdo: Endereço na VRAM da tabela de padrões dos caracteres.

Observação: Nessa variável reside o único bug, ou erro, encontrado

nos micros MSX2. Quando na Screen 0 for dado o comando WIDTH até 40, o valor estará correto. Porém, se o comando WIDTH for de 41 até 80, o valor correto será de 1000H, mas essa variável continuará marcando 0800H. Nesse caso, ao trabalhar com um programa assembly a partir do BASIC, deve ser usada uma instrução ADD HL,HL para corrigir o valor. Nos modelos MSX2+ e MSX turbo R, o valor correto desta variável é 0000H, de modo que a instrução mostrada não afeta a compatibilidade, a despeito desse bug não existir nesses modelos.

**TXTATTR** (F3B9H, 2)

Valor inicial: 0000H

Conteúdo: Variável não usada

**TXTPAT** (F3BBH, 2)

Valor inicial: 0000H

Conteúdo: Variável não usada

### 8.5.2 – Screen 1

**T32NAM** (F3BDH, 2)

Valor inicial: 1800H

Conteúdo: Endereço da tabela de nomes dos padrões.

**T32COL** (F3BFH, 2)

Valor inicial: 2000H

Conteúdo: Endereço na VRAM da tabela de cores.

**T32CGP** (F3C1H, 2)

Valor inicial: 0000H

Conteúdo: Endereço na VRAM da tabela de padrões.

**T32ATR** (F3C3H, 2)

Valor inicial: 1B00H

Conteúdo: Endereço na VRAM da tabela de atributos dos sprites.

**T32PAT** (F3C5H, 2)

Valor inicial: 3800H

Conteúdo: Endereço na VRAM da tabela de padrões dos sprites.

### 8.5.3 – Screen 2

**GRPNAM** (F3C7H, 2)

Valor inicial: 1800H

Conteúdo: Endereço na VRAM da tabela de nomes dos padrões.

**GRPCOL** (F3C9H, 2)

Valor inicial: 2000H

Conteúdo: Endereço na VRAM da tabela de cores.

**GRPCGP** (F3CBH, 2)

Valor inicial: 0000H

Conteúdo: Endereço na VRAM da tabela de padrões.

**GRPATR** (F3CDH, 2)

Valor inicial: 1B00H

Conteúdo: Endereço na VRAM da tabela de atributos dos sprites.

**GRPPAT** (F3CFH, 2)

Valor inicial: 3800H

Conteúdo: Endereço na VRAM da tabela de padrões dos sprites.

### 8.5.4 – Screen 3

**MLTNAM** (F3D1H, 2)

Valor inicial: 0800H

Conteúdo: Endereço da tabela de nomes dos padrões.

**MLTCOL** (F3D3H, 2) – Variável não usada.

**MLTCGP** (F3D5H, 2)

Valor inicial: 0000H

Conteúdo: Endereço na VRAM da tabela de padrões.

**MLTATR** (F3D7H, 2)

Valor inicial: 1B00H

Conteúdo: Endereço na VRAM da tabela de atributos dos sprites.

**MLTPAT** (F3D9H, 2)

Valor inicial: 3800H

Conteúdo: Endereço na VRAM da tabela de padrões dos sprites.

### 8.5.5 – Outros valores para a tela

**CLIKSW** (F3DBH, 1)

Valor inicial: 1

Conteúdo: Liga/desliga click das teclas (0=desliga; outro valor, liga). Pode ser alterada pelo comando SCREEN.

**CSRY** (F3DCH, 1)

Valor inicial: 1

Conteúdo: Coordenada Y (vertical) do cursor nos modos texto.

**CSRX** (F3DDH, 1)

Valor inicial: 1

Conteúdo: Coordenada X (horizontal) do cursor nos modos texto.

**CNSDFG** (F3DEH, 1)  
 Valor inicial: 0  
 Conteúdo: Liga/desliga a apresentação das teclas de função (0=liga, outro valor, desliga). Pode ser alterada pelo comando KEY ON/OFF.

## 8.6 – ÁREA DOS REGISTRADORES DO VDP

**RG0SAV** (F3DFH, 1)  
 Conteúdo: Cópia do registrador R#0 do VDP.

**RG1SAV** (F3E0H, 1)  
 Conteúdo: Cópia do registrador R#1 do VDP.

**RG2SAV** (F3E1H, 1)  
 Conteúdo: Cópia do registrador R#2 do VDP.

**RG3SAV** (F3E2H, 1)  
 Conteúdo: Cópia do registrador R#3 do VDP.

**RG4SAV** (F3E3H, 1)  
 Conteúdo: Cópia do registrador R#4 do VDP.

**RG5SAV** (F3E4H, 1)  
 Conteúdo: Cópia do registrador R#5 do VDP.

**RG6SAV** (F3E5H, 1)  
 Conteúdo: Cópia do registrador R#6 do VDP.

**RG7SAV** (F3E6H, 1)  
 Conteúdo: Cópia do registrador R#7 do VDP.

**STATFL** (F3E7H, 1)  
 Valor inicial: 00H  
 Conteúdo: Cópia do registrador de status do VDP. No MSX2 ou superior, armazena o conteúdo do registrador S#0.

### 8.6.1 – Área para o VDP V9938

**RG8SAV** (FFE7H, 1)  
 Conteúdo: Cópia do registrador 8 do VDP.

- RG9SAV** (FFE8H, 1)  
Conteúdo: Cópia do registrador 9 do VDP.
- R10SAV** (FFE9H, 1)  
Conteúdo: Cópia do registrador 10 do VDP.
- R11SAV** (FFEAH, 1)  
Valor inicial: 00H  
Conteúdo: Cópia do registrador 11 do VDP.
- R12SAV** (FFEBH, 1)  
Conteúdo: Cópia do registrador R#12 do VDP.
- R13SAV** (FFECH, 1)  
Conteúdo: Cópia do registrador R#13 do VDP.
- R14SAV** (FFEDH, 1)  
Conteúdo: Cópia do registrador R#14 do VDP.
- R15SAV** (FFEEH, 1)  
Conteúdo: Cópia do registrador R#15 do VDP.
- R16SAV** (FFEFH, 1)  
Conteúdo: Cópia do registrador R#16 do VDP.
- R17SAV** (FFF0H, 1)  
Conteúdo: Cópia do registrador R#17 do VDP.
- R18SAV** (FFF1H, 1)  
Conteúdo: Cópia do registrador R#18 do VDP.
- R19SAV** (FFF2H, 1)  
Conteúdo: Cópia do registrador R#19 do VDP.
- R20SAV** (FFF3H, 1)  
Conteúdo: Cópia do registrador R#20 do VDP.
- R21SAV** (FFF4H, 1)  
Conteúdo: Cópia do registrador R#21 do VDP.
- R22SAV** (FFF5H, 1)  
Conteúdo: Cópia do registrador R#22 do VDP.

**R23SAV** (FFF6H, 1)  
 Conteúdo: Cópia do registrador R#23 do VDP.

### 8.6.2 – Área para o VDP V9958

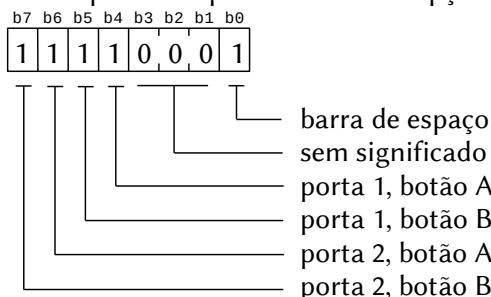
**R25SAV** (FFFAH, 1)  
 Conteúdo: Cópia do registrador R#25 do VDP (V9958).

**R26SAV** (FFFFBH, 1)  
 Conteúdo: Cópia do registrador R#26 do VDP (V9958).

**R27SAV** (FFFCH, 1)  
 Conteúdo: Cópia do registrador R#27 do VDP (V9958).

## 7.7 – MISCELÂNEA

**TRGFLG** (F3E8H, 1)  
 Valor inicial: 11110001B  
 Conteúdo: Estado dos botões do joystick. (0=pressionado, 1=não pressionado). Essa variável é constantemente atualizada pelo manipulador de interrupção.



**FORCLR** (F3E9H, 1)  
 Valor inicial: 15  
 Conteúdo: Cor de frente e dos caracteres. Pode ser alterada pelo comando COLOR.

**BAKCLR** (F3EAH, 1)  
 Valor inicial: 4  
 Conteúdo: Cor de fundo. Pode ser alterada pelo comando COLOR.

**BDRCLR** (F3EBH, 1)

Valor inicial: 7

Conteúdo: Cor da borda. Pode ser alterada pelo comando COLOR.

**MAXUPD** (F3ECH, 3)

Valor inicial: JP 0000H (C3H, 00H, 00H)

Conteúdo: Usada internamente pelo comando CIRCLE.

**MINUPD** (F3EFH, 3)

Valor inicial: JP 0000H (C3H, 00H, 00H)

Conteúdo: Usada internamente pelo comando CIRCLE.

**ATRBYT** (F3F2H, 1)

Valor inicial: 15

Conteúdo: Código de cor usada para gráficos.

**8.8 – ÁREA USADA PELO COMANDO PLAY****QUEUES** (F3F3H, 2)

Valor inicial: QUETAB (F959H)

Conteúdo: Apontador para a fila de execução do comando PLAY.

**FRCNEW** (F3F5H, 1)

Valor inicial: 255

Conteúdo: Usada internamente pelo interpretador BASIC.

**MCLTAB** (F956H, 2)

Conteúdo: Endereço da tabela de comando a ser usada pelos macro-comandos PLAY e DRAW.

**MCLFLG** (F958H, 1)

Conteúdo: Flag para indicar qual comando está sendo processado (0=DRAW; não zero, PLAY).

**QUETAB** (F959H, 24)

Conteúdo: Esta tabela contém os dados para as três filas musicais e para a fila RS232C, reservando seis bytes para cada uma, como ilustrado abaixo:

+0: posição para colocar.  
+1: posição para pegar.  
+2: indicação de devolução.  
+3: tamanho do buffer na fila.  
+4: endereço do buffer na fila (high).  
+5: endereço do buffer na fila (low).

F959H = voz A.

F95FH = voz B.

F965H = voz C.

F96AH = RS232C.

**QUEBAK** (F971H, 4)

Conteúdo: Usada para substituição de caracteres nas filas.

F971H +0 – Voz A

+1 – Voz B

+2 – Voz C

+3 – RS232C (apenas MSX1)

**PRSCNT** (FB35H, 1)

Conteúdo: Usada internamente pelo comando PLAY para contar o número de operandos completados. O bit 7 será ligado após cada um dos três operandos serem analizados.

**SAVSP** (FB36H, 2)

Conteúdo: Salva o valor do registrador SP antes da execução do comando PLAY.

**VOICEN** (FB38H, 1)

Conteúdo: Número da voz que está sendo atualmente processada (0, 1 ou 2).

**SAVVOL** (FB39H, 2)

Valor inicial: 00H

Conteúdo: Salva o volume durante a geração de uma pausa.

**MCLLEN** (FB3BH, 1)

Valor inicial: 00H

Conteúdo: Comprimento da string que está sendo analisada.

**MCLPTR** (FB3CH, 2)

Valor inicial: 0000H

Conteúdo: Endereço do operando que está sendo analisado.

**QUEUEN** (FB3EH, 1)

Conteúdo: Utilizada pelo manipulador de interrupção para conter o número da fila musical que está sendo atualmente processada.

**MUSICF** (FB3FH, 1)

Conteúdo: Flag para indicar quais filas musicais serão utilizadas.

**PLYCNT** (FB40H, 1)

Conteúdo: Número de sequências do comando PLAY armazenados na fila.

### 8.8.1 – Filas para o comando PLAY

**VOICAQ** (F975H, 128)

Valor inicial: DEFS 128 (00H .... 00H)

Conteúdo: Fila para a voz A.

**VOICBQ** (F9F5H, 128)

Valor inicial: DEFS 128 (00H .... 00H)

Conteúdo: Fila para a voz B.

**VOICCQ** (FA75H, 128)

Valor inicial: DEFS 128 (00H .... 00H)

Conteúdo: Fila para a voz C.

### 8.8.2 – Offset para o buffer de parâmetros do comando PLAY

METREX (+00, 2) Contador de duração

VCXLEN (+02, 1) Comprimento da string

VCXPTR (+03, 2) Endereço da string

VCXSTP (+05, 2) Endereço dos dados na pilha

QLENGX (+07, 1) Tamanho do pacote musical em bytes

NTICSX (+08, 2) Pacote musical

TONPRX (+10, 2) Período do tom

AMPRX (+12, 1) Volume e envelope

ENVPRX (+13, 2) Período do envelope

OCTAVX	(+15, 1)	Oitava
NOTELEX	(+16, 1)	Comprimento do tom
TEMPOX	(+17, 1)	Tempo
VOLUMX	(+18, 1)	Volume
ENVLPX	(+19, 14)	Forma de onda do envelope
MCLSTX	(+33, 3)	Reservado para a pilha
MCLSEX	(+36, 1)	Inicialização da pilha
VCBSIZ	(+37, 1)	Tamanho do buffer de parâmetros

### 8.8.3 – Área de dados para o buffer de parâmetros

**VCBA** (FB41H, 37)

- Conteúdo: Parâmetros para a voz A.
- +00, 2 – Contador de duração
  - +02, 1 – Comprimento da string
  - +03, 2 – Endereço da string
  - +05, 2 – Endereço de dados na pilha
  - +07, 1 – Tamanho do pacote musical
  - +08, 7 – Pacote musical
  - +15, 1 – Oitava
  - +16, 1 – Comprimento
  - +17, 1 – Tempo
  - +18, 1 – Volume
  - +19, 2 – Período da envoltória
  - +21,16 – Espaço para dados da pilha

**VCBB** (FB66H, 37)

- Conteúdo: Parâmetros para a voz B.  
(Estrutura idêntica à da voz A).

**VCBC** (FB8BH, 37)

- Conteúdo: Parâmetros para a voz C.  
(Estrutura idêntica à da voz A).

### 8.9 – ÁREA PARA O TECLADO

**SCNCNT** (F3F6H, 1)

Valor inicial: 1

Conteúdo: Intervalo para a varredura das teclas.

**REPCNT** (F3F7H, 1)

Valor inicial: 50

Conteúdo: Tempo de atraso para o início da autorrepetição das teclas.

**PUTPNT** (F3F8H, 2)

Valor inicial: KEYBUF (FBF0H)

Conteúdo: Aponta para o endereço de escrita do buffer de teclado.

**GETPNT** (F3FAH, 2)

Valor inicial: KEYBUF (FBF0H)

Conteúdo: Aponta para o endereço de leitura do buffer de teclado.

**OLDKEY** (FBDAH, 11)

Conteúdo: Estado anterior da matriz do teclado.

**NEWKEY** (FBE5H, 11)

Conteúdo: Novo estado da matriz do teclado.

**KEYBUF** (FBF0H, 40)

Conteúdo: Buffer circular que contém os caracteres lidos do teclado.

## 8.10 – ÁREA USADA PELO CASSETE

**CS1200** (F3FCH, 5)

Valor inicial: +0 → 53H – primeira metade para o bit 0  
+1 → 5CH – Segunda metade para o bit 0  
+2 → 26H – primeira metade para o bit 1  
+3 → 2DH – Segunda metade para o bit 1  
+4 → 0FH – Contagem dos ciclos para cabeçalho breve [ciclos = (0F400H) \* 2 / 256]

Conteúdo: Parâmetros para o cassete para 1200 bauds.

**CS2400** (F401H, 5)

Valor inicial: +0 → 25H – primeira metade para o bit 0  
+1 → 2DH – Segunda metade para o bit 0  
+2 → 0EH – primeira metade para o bit 1  
+3 → 16H – Segunda metade para o bit 1  
+4 → 1FH – Contagem dos ciclos para cabeçalho breve [ciclos = (0F405H) \* 4 / 256]

Conteúdo: Parâmetros para o cassete para 2400 bauds.

**LOW** (F406H, 2)

Valor inicial: +0 → 53H – primeira metade para o bit 0  
+1 → 5CH – Segunda metade para o bit 0  
Conteúdo: Largura para o bit 0 do baud rate atual.

**HIGH** (F408H, 2)

Valor inicial: +0 → 26H – primeira metade para o bit 1  
+1 → 2DH – Segunda metade para o bit 1  
Conteúdo: Largura para o bit 1 do baud rate atual.

**HEADER** (F40AH, 1)

Valor inicial: 0FH  
Conteúdo: Contagem dos ciclos para cabeçalho breve atual.

**8.11 – ÁREA USADA PELO COMANDO CIRCLE****ASPCT1** (F40BH, 2)

Conteúdo: 256/relação de aspecto. Pode ser alterada pelo comando SCREEN para uso do comando CIRCLE.

**ASPCT2** (F40DH, 2)

Conteúdo: 256\*relação de aspecto. Pode ser alterada pelo comando SCREEN para uso do comando CIRCLE.

**ASPECT** (F931H, 2)

Valor inicial: 0  
Conteúdo: Relação de aspecto.

**CENCNT** (F933H, 2)

Valor inicial: 0  
Conteúdo: Contagem de pontos do ângulo final.

**CLINEF** (F935H, 1)

Valor inicial: 0  
Conteúdo: Flag usada para indicar o desenho de uma linha a partir do centro da circunferência. O bit 0 será ligado de uma linha for requerida a partir do ângulo inicial e o bit 7 será ligado se a linha for requerida a partir do ângulo final.

**CNPNTS** (F936H, 2)

Valor inicial: 0

Conteúdo: Número de pontos dentro de um segmento de 45 graus da circunferência.

**CPLOTF** (F938H, 1)

Conteúdo: Usada internamente pelo comando CIRCLE.

**CPCNT** (F939H, 2)

Conteúdo: Coordenada Y dentro do segmento atual de 45 graus da circunferência.

**CPCNT8** (F93BH, 2)

Valor inicial: 0

Conteúdo: Contagem total de pontos da posição atual.

**CPCSUM** (F93DH, 2)

Valor inicial: 0

Conteúdo: Contador da computação de pontos.

**CSTCNT** (F93FH, 2)

Valor inicial: 0

Conteúdo: Contagem de pontos do ângulo inicial da circunferência.

**CSCLXY** (F941H, 1)

Valor inicial: 0

Conteúdo: Escala entre X e Y. Usada pela instrução CIRCLE.

**CSAVEA** (F942H, 2)

Conteúdo: Área reservada para ADVGRP.

**CSAVEM** (F944H, 1)

Conteúdo: Área reservada para ADVGRP.

**CXOFF** (F945H, 2)

Conteúdo: Coordenada X a partir do centro da circunferência.

**CYOFF** (F947H, 2)

Conteúdo: Coordenada Y a partir do centro da circunferência.

## 8.12 – ÁREA USADA INTERNAMENTE PELO BASIC

### **ENDPRG** (F40FH, 5)

Valor inicial: ":"; 00H; 00H; 00H; 00H.

Conteúdo: Falso fim de linha de programa para os comandos RESUME e NEXT.

### **ERRFLG** (F414H, 1)

Valor inicial: 00H

Conteúdo: Área para salvar o número de erro.

### **LPTPOS** (F415H, 1)

Valor inicial: 00H

Conteúdo: Armazena posição atual da cabeça da impressora.

### **PRTFLG** (F416H, 1)

Valor inicial: 00H

Conteúdo: Flag para selecionar saída para tela ou impressora (0=tela; outro valor, impressora).

### **NTMSXP** (F417H, 1)

Valor inicial: 00H

Conteúdo: Flag para selecionar o tipo de impressora.  
(0=impressora padrão MSX; outro valor, impressora não MSX). Pode ser alterada pelo comando SCREEN.

### **RAWPRT** (F418H, 1)

Valor inicial: 00H

Conteúdo: Flag para determinar se os caracteres gráficos de controle serão modificados ao serem enviados para a impressora (0=modifica; outro valor, não modifica).

### **VLZADR** (F419H, 2)

Valor inicial: 0000H

Conteúdo: Endereço do caractere para a função VAL.

### **VLZDAT** (F41BH, 1)

Valor inicial: 00H

Conteúdo: Caractere que deve ser substituído por 0 pela função VAL.

**CURLIN** (F41CH, 2)

Valor inicial: FFFFH

Conteúdo: Número de linha atual do interpretador BASIC.  
O valor FFFFH indica modo direto.

### 8.12.1 – Buffers de texto BASIC

**KBFMIN** (F41EH, 1)

Valor inicial: ":"

Conteúdo: Esse byte é um prefixo fictício para o texto atomizado contido em KBUF.

**KBUF** (F41FH, 318)

Valor inicial: 00H, 00H, ... 00H

Conteúdo: Esse buffer guarda a linha BASIC atomizada coletada pelo interpretador.

**BUFMIN** (F55DH, 1)

Valor inicial: ":"

Conteúdo: Prefixo fictício para o texto contido em BUF. É usado para sincronizar o manipulador da instrução INPUT quando este começa a analisar o texto coletado.

**BUF** (F55EH, 258)

Valor inicial: 00H, 00H, ... 00H

Conteúdo: Esse buffer guarda, no formato ASCII, os caracteres coletados do teclado pela rotina padrão INLIN.

**ENDBUF** (F660H, 1)

Valor inicial: 00H

Conteúdo: Byte para prevenir overflow em BUF (F55EH).

### 8.12.2 – Dados Gerais

**TTYPOS** (F661H, 1)

Conteúdo: Usada pelo comando PRINT para guardar a posição virtual do cursor.

**DIMFLG** (F662H, 1)

Conteúdo: Usada internamente pelo comando DIM.

<b>VALTYP</b>	(F663H, 1)
Conteúdo:	Guarda o tipo de variável contida em DAC (F3F6H): 2=inteira; 3=string; 4=precisão simples; 8=precisão dupla.
<b>DORES</b>	(F664H, 1)
Conteúdo:	Usada internamente pelo comando DATA para manter o texto no formato ASCII.
<b>DONUM</b>	(F665H, 1)
Conteúdo:	Flag usada internamente pelo BASIC.
<b>CONTXT</b>	(F666H, 2)
Conteúdo:	Armazena o endereço do texto usado pela rotina CHRGTR.
<b>CONSAV</b>	(F668H, 1)
Conteúdo:	Armazena a token de uma constante numérica; usada pela rotina GHRGTR.
<b>CONTYP</b>	(F669H, 1)
Conteúdo:	Armazena o tipo de uma constante numérica encontrada no texto de programa BASIC. É usada pela rotina padrão CHRGTR.
<b>CONLO</b>	(F66AH, 8)
Conteúdo:	Armazena uma constante numérica usada pela rotina padrão CHRGTR.
<b>MEMSIZ</b>	(F672H, 2)
Conteúdo:	Endereço mais alto de memória que pode ser usado pelo BASIC.
<b>STKTOP</b>	(F674H, 2)
Conteúdo:	Endereço do topo da pilha do Z80. Usada internamente pelo BASIC.
<b>TXTTAB</b>	(F676H, 2)
Valor inicial:	8000H
Conteúdo:	Endereço inicial da área de texto BASIC.

**TEMPPT** (F678H, 2)

Valor inicial: TEMPST (F67AH)

Conteúdo: Endereço da próxima posição livre em TEMPST.

**TEMPST** (F67AH, 30)

Valor inicial: DEFS 30 (00H, .... 00H)

Conteúdo: Buffer usado para armazenar descritores de strings.

**DSCTMP** (F698H, 3)

Valor inicial: 00H, 00H, 00H

Conteúdo: Salva o descritor de uma string durante o processamento.

**FRETOP** (F69BH, 2)

Valor inicial: F168H

Conteúdo: Endereço da próxima posição livre na área de strings.

**TEMP3** (F69DH, 2)

Valor inicial: 0000H

Conteúdo: Usada internamente pelo interpretador para armazenamento temporário de várias rotinas.

**ENDFOR** (F6A1H, 2)

Valor inicial: 0000H

Conteúdo: Endereço para o comando FOR.

**DATLIN** (F6A3H, 2)

Valor inicial: 00H

Conteúdo: Número de linha do comando DATA para uso do comando READ.

**SUBFLG** (F6A5H, 1)

Valor inicial: 00H

Conteúdo: Flag usada para controlar o processamento de índices na busca de variáveis tipo matriz.

**FLGINP** (F6A6H, 1)

Valor inicial: 00H

Conteúdo: Flag usada pelos comandos INPUT e READ (0=INPUT; outro valor, READ).

**TEMP** (F6A7H, 2)  
 Conteúdo: Usada internamente pelo interpretador.

### 8.12.3 – Controle de linhas BASIC em tempo de execução

<b>PTRFLG</b>	(F6A9H, 1)
Conteúdo:	Usada internamente pelo interpretador para conversão dos números de linha em apontadores (0=operando não convertido; outro valor, operando convertido).
<b>AUTFLG</b>	(F6AAH, 1)
Conteúdo:	Flag para o comando AUTO (0=comando AUTO inativo; outro valor, comando AUTO ativo).
<b>AUTLIN</b>	(F6ABH, 2)
Conteúdo:	Número da última linha BASIC entrada.
<b>AUTINC</b>	(F6ADH, 2)
Valor inicial:	10
Conteúdo:	Valor de incremento para a função AUTO.
<b>SAVTXT</b>	(F6AFH, 2)
Conteúdo:	Armazena o endereço atual do texto BASIC durante a execução.
<b>SAVSTK</b>	(F6B1H, 2)
Conteúdo:	Armazena o endereço atual da pilha do Z80. Usada pelo manipulador de erro e pela instrução RESUME.
<b>ERRLIN</b>	(F6B3H, 2)
Conteúdo:	Número de linha BASIC onde ocorreu algum erro.
<b>DOT</b>	(F6B5H, 2)
Conteúdo:	Último número de linha durante o processamento. Usada internamente pelo interpretador e pelo manipulador de erro.
<b>ERRTXT</b>	(F6B7H, 2)
Conteúdo:	Endereço do texto BASIC onde ocorreu algum erro. Usada pelo comando RESUME.

<b>ONELIN</b>	(F6B9H, 2)
Conteúdo:	Endereço da linha de programa que deve ser executada ao ocorrer algum erro. Setada por ON ERROR GOTO.
<b>ONEFLG</b>	(F6BBH, 1)
Conteúdo:	Flag para indicar execução de rotina de erro (0=não executando; outro valor, rotina em execução).
<b>TEMP2</b>	(F6BCH, 2)
Conteúdo:	Usada internamente pelo interpretador.
<b>OLDLIN</b>	(F6BEH, 2)
Conteúdo:	Armazena a última linha executada pelo programa. É atualizada pelos comandos END e STOP para ser usada pelo comando CONT.

#### **8.12.4 – Endereços de armazenamento do texto BASIC**

<b>OLDTXT</b>	(F6C0H, 2)
Conteúdo:	Armazena o endereço da última instrução do texto BASIC.
<b>VARTAB</b>	(F6C2H, 2)
Conteúdo:	Endereço do primeiro byte da área de armazenamento das variáveis do BASIC.
<b>ARYTAB</b>	(F6C4H, 2)
Conteúdo:	Endereço do primeiro byte da área de armazenamento das matrizes do BASIC.
<b>STREND</b>	(F6C6H, 2)
Conteúdo:	Endereço do primeiro byte após a área de armazenamento das matrizes, variáveis ou texto BASIC.
<b>DATPTR</b>	(F6C8H, 2)
Conteúdo:	Endereço do comando DATA atual para uso do comando READ.
<b>DEFTBL</b>	(F6CAH, 26)
Conteúdo:	Área de armazenamento do tipo de variável por nomes em ordem alfabética. Podem ser alteradas pelo grupo de comandos "DEF xxx".

### 8.12.5 – Área para as funções do usuário

**PRMSTK** (F6E4H, 2)

Conteúdo: Definição prévia do bloco na pilha do Z80.

**PRMLEN** (F6E6H, 2)

Conteúdo: Comprimento do bloco de parâmetro "FN" atual em PARM1.

**PARM1** (F6E8H, 100)

Conteúdo: Buffer para armazenamento das variáveis da função "FN" que está sendo avaliada.

**PRMPRV** (F74CH, 2)

Valor inicial: PRMSTK (F6E4H)

Conteúdo: Endereço do bloco de parâmetro "FN" anterior.

**PRMLN2** (F74EH, 2)

Conteúdo: Comprimento do bloco de parâmetros "FN" que está sendo montado em PARM2.

**PARM2** (F750H, 100)

Conteúdo: Buffer usado para as variáveis da função "FN" atual.

**PRMFLG** (F7B4H, 1)

Conteúdo: Flag para indicar quando PARM1 está sendo procurada.

**ARYTA2** (F7B5H, 2)

Conteúdo: Último endereço para procura de variável.

**NOFUNS** (F7B7H, 1)

Conteúdo: Flag para indicar à função "FN" a existência de variáveis locais (0=não há variáveis; outro valor, há variáveis).

**TEMP9** (F7B8H, 2)

Conteúdo: Usada internamente pelo interpretador.

**FUNACT** (F7BAH, 2)

Conteúdo: Número de funções "FN" atualmente ativas.

**SWPTMP** (F7BCH, 8)

Conteúdo: Buffer utilizado para conter o primeiro operando de um comando SWAP.

**TRCFLG** (F7C4H, 1)  
 Conteúdo: Flag para o comando TRACE (0=TRACE OFF, outro valor, TRACE ON).

### 8.12.6 – Área de dados do interpretador

<b>FNKSTR</b>	(F87FH, 160)
Conteúdo:	Área reservada para armazenar o conteúdo das teclas de função (16 caracteres x 10 posições).
<b>CGPNT</b>	(F91FH, 3)
Conteúdo:	Endereço da fonte de caracteres. O primeiro byte é o ID do slot e os outros dois o endereço.
<b>NAMBAS</b>	(F922H, 2)
Conteúdo:	Endereço da tabela de nomes no modo texto atual.
<b>CGPBAS</b>	(F924H, 2)
Conteúdo:	Endereço da tabela geradora de padrões no modo texto atual.
<b>PATBAS</b>	(F926H, 2)
Conteúdo:	Endereço atual da tabela geradora de sprites.
<b>ATRBAS</b>	(F928H, 2)
Conteúdo:	Endereço atual da tabela de atributos dos sprites.
<b>CLOC</b>	(F92AH, 2)
Conteúdo:	Usada internamente pelas rotinas gráficas.
<b>CMASK</b>	(F92CH, 1)
Conteúdo:	Usada internamente pelas rotinas gráficas.
<b>MINDEL</b>	(F92DH, 2)
Conteúdo:	Usada internamente pelo comando LINE.
<b>MAXDEL</b>	(F92FH, 2)
Conteúdo:	Usada internamente pelo comando LINE.

## 8.13 – ÁREA PARA O MATH-PACK

- FBUFFR** (F7C5H, 43)  
 Conteúdo: Usado internamente pelo MATH-PACK.
- DECTMP** (F7F0H, 2)  
 Conteúdo: Usado para transformar um número inteiro em um número de ponto flutuante.
- DECTM2** (F7F2H, 2)  
 Conteúdo: Usada internamente pela rotina de divisão.
- DEC\_CNT** (F7F4H, 1)  
 Conteúdo: Usada internamente pela rotina de divisão.
- DAC** (F7F6H, 16)  
 Conteúdo: Acumulador primário que contém um número durante uma operação matemática.
- HOLD8** (F806H, 48)  
 Valor inicial: 00H, 00H ... 00H  
 Conteúdo: Área de armazenamento para a multiplicação decimal.
- HOLD2** (F836H, 8)  
 Valor inicial: 00H, 00H ... 00H  
 Conteúdo: Usada internamente pelo MATH-PACK.
- HOLD** (F83EH, 8)  
 Valor inicial: 00H, 00H ... 00H  
 Conteúdo: Usada internamente pelo MATH-PACK.
- ARG** (F847H, 16)  
 Conteúdo: Acumulador secundário que contém o número a ser calculado com DAC (F7F6H).
- RNDX** (F857H, 8)  
 Conteúdo: Armazena o último número aleatório de dupla precisão. Usada pela função RND.

## 8.14 – ÁREA DE DADOS DO SISTEMA DE DISCO

<b>MAXFIL</b>	(F85FH, 1)
Conteúdo:	Número de buffers de I/O existentes. Pode ser alterada pela instrução MAXFILES.
<b>FILTAB</b>	(F860H, 2)
Conteúdo:	Endereço inicial da área de dados dos arquivos.
<b>NULBUF</b>	(F862H, 2)
Conteúdo:	Aponta para o buffer usado pelos comandos SAVE e LOAD.
<b>PTRFIL</b>	(F864H, 2)
Conteúdo:	Endereço dos dados do arquivo atualmente ativo.
<b>RUNFLG</b>	(F866H,0)
Conteúdo:	Não-zero, se algum programa foi carregado e executado. Usada pelo operando ",R" do comando LOAD.
<b>FILNAM</b>	(F866H, 11)
Conteúdo:	Área para armazenamento de um nome de arquivo.
<b>FILNM2</b>	(F871H, 11)
Conteúdo:	Área para armazenamento de um nome de arquivo para ser comparado com FILNAM.
<b>NONLY</b>	(F87CH, 1)
Conteúdo:	Flag para indicar se um programa está sendo carregado ou não (0=programa não está sendo carregado; outro valor, programa está sendo carregado).
<b>SAVEND</b>	(F87DH, 2)
Conteúdo:	Usada pelo comando BSAVE para conter o endereço final do programa assembly que deve ser salvo.
<b>HOKVLD</b>	(FB20H, 1)
Valor inicial:	01H
Conteúdo:	O bit 0 deste byte indica a presença de uma BIOS estendida. 0 = Sem BIOS, 1 = Há pelo menos uma BIOS que pode ser chamada no endereço 0FFCAH (EXTBIO).

**DRVINV** (FB21H, 9)

Valor inicial: variável

Conteúdo: ID do slot e número de unidades conectadas às interfaces de disco.

DRVINV+0 = Número de drives conectados na interface de disco primária.

+1 = ID do slot da interface do disco mestre.

+2 = Número de unidades conectadas à interface do disco mestre.

+3 = ID do slot da 2<sup>a</sup> interface de disco+4 = Número de unidades conectadas à 2<sup>a</sup> interface de disco+5 = ID do slot da 3<sup>a</sup> interface de disco+6 = Número de unidades conectadas à 3<sup>a</sup> interface de disco+7 = ID do slot da 4<sup>a</sup> interface do disco+8 = Número de unidades conectadas à 4<sup>a</sup> interface do disco**DRVINT** (FB29H, 12)

Valor inicial: variável

Conteúdo: ID do slot e endereço de cada manipulador de interrupção das interfaces de disco. (3 \* 4 bytes)

DRVINT+0 = ID do slot de cada manipulador de interrupções da interface principal.

+1 = Endereço do manipulador de interrupções da interface principal.

+3 = ID do slot de cada manipulador de interrupções da 2<sup>a</sup> interface+4 = Endereço do manipulador de interrupções da 2<sup>a</sup> interface+6 = ID do slot de cada manipulador de interrupções da 3<sup>a</sup> interface+7 = Endereço do manipulador de interrupções da 3<sup>a</sup> interface+9 = ID do slot de cada manipulador de interrupções da 4<sup>a</sup> interface+10 – Endereço do manipulador de interrupções da 4<sup>a</sup> interface

## 8.15 – ÁREA USADA PELO COMANDO PAINT

**LOHMSK** (F949H, 1)

Valor inicial: 0

Conteúdo: Posição mais à esquerda da excursão LH.

**LOHDIR** (F94AH, 1)

Valor inicial: 0

Conteúdo: Direção de pintura requerida pela excursão LH.

**LOHADR** (F94BH, 2)

Valor inicial: 0000H

Conteúdo: Posição mais à esquerda da excursão LH.

**LOHCNT** (F94DH, 2)

Valor inicial: 0

Conteúdo: Tamanho da excursão LH.

**SKPCNT** (F94FH, 2)

Valor inicial: 0

Conteúdo: Contador de salto devolvido por SCANR (012CH).

**MOVCNT** (F951H, 2)

Valor inicial: 0

Conteúdo: Contador de movimento devolvido por SCANR (012CH).

**PDIREC** (F953H, 1)

Conteúdo: Direção de pintura: 40H, para baixo; C0H, para cima; 00H, terminar.

**LFPROG** (F954H, 1)

Conteúdo: Flag usada pelo comando PAINT para indicar se houve progresso à esquerda (0=não houve progresso; outro valor, houve progresso).

**RTPROG** (F955H, 1)

Conteúdo: Flag usada pelo comando PAINT para indicar se houve progresso à direita (0=não houve progresso; outro valor, houve progresso).

## 8.16 – ÁREA ADICIONADA PARA O MSX2

### DPPAGE (FAF5H, 1)

Valor inicial: 0

Conteúdo: Página de vídeo que está atualmente sendo apresentada.

### ACPAGE (FAF6H, 1)

Valor inicial: 0

Conteúdo: Página de vídeo ativa para receber comandos.

### AVCSAV (FAF7H, 1)

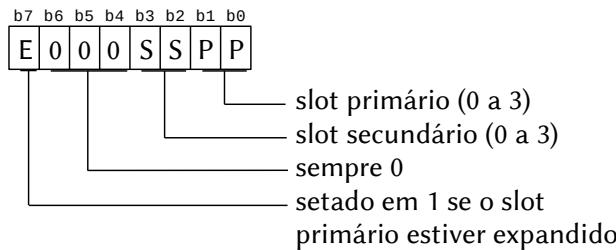
Valor inicial: 0

Conteúdo: Usada pela porta de controle AV.

### EXBRSA (FAF8H, 1)

Valor inicial: 10 000 111B

Conteúdo: Slot da Sub-ROM, no formato abaixo:



### CHRCNT (FAF9H, 1)

Valor inicial: 0

Conteúdo: Contador de caracteres no buffer. Usada para transição Roman-Kana (0, 1 ou 2).

### ROMA (FAFAH, 2)

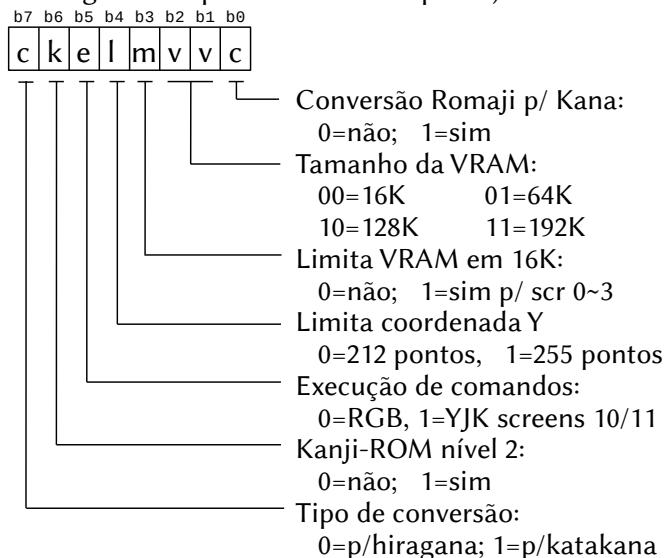
Valor inicial: 0

Conteúdo: Armazena o caractere do buffer para a transição Roman-Kana (somente na versão japonesa).

### MODE (FAFCH, 1)

Valor inicial: 10 001 001B

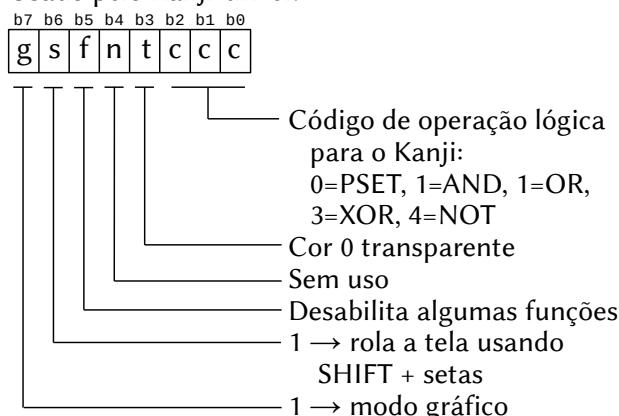
Conteúdo: Flag de modo e tamanho da VRAM (os bits 4, 5, 6 e 7 só têm significado para MSX2+ ou superior)



### **NORUSE** (FAFDH, 1)

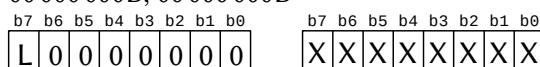
Valor inicial: 00H

Conteúdo: Usado pelo Kanji-driver.



### **XSAVE** (FAFEH, 2)

Valor inicial: 00 000 000B, 00 000 000B



**YSAVE** (FB00H, 2)

Valor inicial: 00 000 000B, 00 000 000B

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	
L	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y

L=1 → requisição de interrupção da caneta ótica

0000 000 = sem significado

XXXXXXXXX = coordenada X

YYYYYYYYY = coordenada Y

**LOGOPR** (FB02H, 1)

Valor inicial: 00H

Conteúdo: Código de operação lógica para o VDP.

**8.17 – ÁREA USADA PELA RS232C****RSTMP** (FB03H, 1)

Valor inicial: 00H

Conteúdo: Armazenamento temporário para a RS232C.

Obs.: mesmo endereço de TOCNT.

**TOCNT** (FB03H, 1)

Valor inicial: 00H

Conteúdo: Contador usado pela interface RS232C.

Obs.: mesmo endereço de RSTMP.

**RSFCB** (FB04H, 2)

Valor inicial: 0000H

Conteúdo: Endereço do FCB da RS232C.

**RSIQLN** (FB06H, 1)

Valor inicial: 00H

Conteúdo: Usada internamente pela RS232C.

**MEXBIH** (FB07H, 5)

Valor inicial: C9H, C9H, C9H, C9H, C9H

Conteúdo: FB07H +0: RST 030H

+1: Byte de ID do slot

+2: Endereço (low)

+3: Endereço (high)

+4: RET

Usada internamente pela RS232C.

<b>OLDSTT</b>	(FB0CH, 5)
Valor inicial:	C9H, C9H, C9H, C9H, C9H
Conteúdo:	FB0CH+0: RST 030H +1: Byte de ID do slot +2: Endereço (low) +3: Endereço (high) +4: RET
	Usada internamente pela RS232C.
<b>OLDINT</b>	(FB12H, 5)
Valor inicial:	C9H, C9H, C9H, C9H, C9H
Conteúdo:	FB12H +0: RST 030H +1: Byte de ID do slot +2: Endereço (low) +3: Endereço (high) +4: RET
	Usada internamente pela RS232C.
<b>DEVNUM</b>	(FB17H, 1)
Conteúdo:	Byte offset.
<b>DATCNT</b>	(FB18H, 3)
Conteúdo:	FB18H +0: ID de slot +1: Apontador +2: Apontador
<b>ERRORS</b>	(FB1BH, 1)
Valor inicial:	00H
Conteúdo:	Código de erro da RS232C.
<b>FLAGS</b>	(FB1CH, 1)
Valor inicial:	00 000 011B
Conteúdo:	Flags usadas pela RS232C.
<b>ESTBLS</b>	(FB1DH, 1)
Valor inicial:	FFH
Conteúdo:	Bit booleano para uso da RS232C.
<b>COMMSK</b>	(FB1EH, 1)
Valor inicial:	C1H
Conteúdo:	Máscara da RS232C.

**LSTCOM** (FB1FH, 1)

Valor inicial: E8H

Conteúdo: Usada internamente pela RS232C.

## 8.18 – ÁREA DE DADOS GERAIS

**ENSTOP** (FBB0H, 1)

Conteúdo: Flag para habilitar uma saída forçada para o interpretador ao detectar as teclas CTRL+ SHIFT+GRAPH+CODE pressionadas juntas (0=desab.; outro valor, habilitada).

**BASROM** (FBB1H, 1)

Valor inicial: 00H

Conteúdo: Localização do texto BASIC (0=RAM; outro valor, ROM).

**LINTTB** (FBB2H, 24)

Conteúdo: São 24 flags para indicar se cada uma das linhas de tela de texto avançou para a linha seguinte (0=avançou; outro valor, não avançou).

**FSTPOS** (FBCAH, 2)

Conteúdo: Primeira localização do caracter coletado pela rotina INLIN (00B1H) do BIOS.

**CODSAV** (FBCCH, 1)

Valor inicial: 00H

Conteúdo: Caractere substituído pelo cursor nas telas de texto.

**FNKSW1** (FBCEH, 1)

Valor inicial: 01H

Conteúdo: Flag para indicar quais teclas de função são mostradas quando habilitadas por KEY ON (1=F1 a F5; 0=F6 a F10).

**FNKFLG** (FBCEH, 10)

Conteúdo: Flags para habilitar, inibir ou paralizar a execução de uma linha definida pelo comando ON KEY GOSUB. São modificadas por KEY(n) ON/OFF/STOP (0=KEY(n) OFF/STOP; 1=KEY(n) ON).

**ONGSBF** (FBD8H, 1)

Conteúdo: Flag para indicar se algum dispositivo requereu uma interrupção de programa (0=normal; outro valor indica interrupção ativa).

<b>CLIKFL</b>	(FBD9H, 1)																		
Conteúdo:	Flag de click das teclas. Usada pelo manipulador de interrupção.																		
<b>LINWRK</b>	(FC18H, 40)																		
Conteúdo:	Buffer usado pelo BIOS para conter uma linha completa de caracteres da tela.																		
<b>PATWRK</b>	(FC40H, 8)																		
Conteúdo:	Buffer usado pelo BIOS para conter um padrão de caractere 8x8.																		
<b>BOTTOM</b>	(FC48H, 2)																		
Conteúdo:	Endereço mais baixo usado pelo interpretador, normalmente 8000H.																		
<b>HIMEM</b>	(FC4AH, 2)																		
Conteúdo:	Endereço mais alto de RAM disponível. Pode ser modificado pelo comando CLEAR.																		
<b>TRPTBL</b>	(FC4CH, 78)																		
Conteúdo:	<p>Esta tabela contém o estado atual dos dispositivos de interrupção. Cada dispositivo aloca três bytes na tabela. O primeiro byte contém o estado do dispositivo (bit 0=ligado; bit 1=parado; bit 2=ativo). Os outros dois bytes contêm o endereço da linha de programa a ser executada caso ocorra uma interrupção.</p> <table> <tr> <td>FC4CH/FC69H</td> <td>(3 x 10 bytes)</td> <td>ON KEY GOSUB</td> </tr> <tr> <td>FC6AH/FC6CH</td> <td>(3 x 1 byte)</td> <td>ON STOP GOSUB</td> </tr> <tr> <td>FC6DH/FC6FH</td> <td>(3 x 1 byte)</td> <td>ON SPRITE GOSUB</td> </tr> <tr> <td>FC70H/FC7EH</td> <td>(3 x 5 bytes)</td> <td>ON STRIG GOSUB</td> </tr> <tr> <td>FC7FH/FC81H</td> <td>(3 x 1 byte)</td> <td>ON INTERVAL GOSUB</td> </tr> <tr> <td>FC82H/FC99H</td> <td></td> <td>Reservado para expansão</td> </tr> </table>	FC4CH/FC69H	(3 x 10 bytes)	ON KEY GOSUB	FC6AH/FC6CH	(3 x 1 byte)	ON STOP GOSUB	FC6DH/FC6FH	(3 x 1 byte)	ON SPRITE GOSUB	FC70H/FC7EH	(3 x 5 bytes)	ON STRIG GOSUB	FC7FH/FC81H	(3 x 1 byte)	ON INTERVAL GOSUB	FC82H/FC99H		Reservado para expansão
FC4CH/FC69H	(3 x 10 bytes)	ON KEY GOSUB																	
FC6AH/FC6CH	(3 x 1 byte)	ON STOP GOSUB																	
FC6DH/FC6FH	(3 x 1 byte)	ON SPRITE GOSUB																	
FC70H/FC7EH	(3 x 5 bytes)	ON STRIG GOSUB																	
FC7FH/FC81H	(3 x 1 byte)	ON INTERVAL GOSUB																	
FC82H/FC99H		Reservado para expansão																	
<b>RTYCNT</b>	(FC9AH, 1)																		
Conteúdo:	Controle de interrupção.																		
<b>INTFLG</b>	(FC9BH, 1)																		
Conteúdo:	Se CTRL+STOP são pressionadas, esta variável é colocada em 03H e o processamento interrompido; se STOP for pressionada, o valor é 04H; caso contrário, é mantida em 00H.																		

<b>PADY</b>	(FC9CH, 1)
Conteúdo:	Coordenada Y do paddle.
<b>PADX</b>	(FC9DH, 1)
Conteúdo:	Coordenada X do paddle.
<b>JIFFY</b>	(FC9EH, 2)
Conteúdo:	Esta variável é continuamente incrementada pelo manipulador de interrupção. Seu valor pode ser lido ou atribuído pela função TIME. Também é utilizada internamente pelo comando PLAY.
<b>INTVAL</b>	(FCA0H, 2)
Valor inicial:	0000H
Conteúdo:	Duração do intervalo usado por ON INTERVAL GOSUB.
<b>INTCNT</b>	(FCA2H, 2)
Valor inicial:	0000H
Conteúdo:	Contador para a instrução ON INTERVAL GOSUB.
<b>LOWLIM</b>	(FCA4H, 1)
Valor inicial:	31H
Conteúdo:	Duração mínima para o bit de partida durante a leitura do cassete.
<b>WINWID</b>	(FCA5H, 1)
Valor inicial:	22H
Conteúdo:	Duração da discriminação do ciclo alto/baixo durante a leitura do cassete.
<b>GRPHED</b>	(FCA6H, 1)
Conteúdo:	Flag para o envio de um caractere gráfico (0=normal; 1=caractere gráfico).
<b>ESCCNT</b>	(FCA7H, 1)
Conteúdo:	Área de contagem dos códigos de escape.
<b>INSFLG</b>	(FCA8H, 1)
Conteúdo:	Flag para indicar o modo de inserção (0=normal; outro valor, modo de inserção)

<b>CSRSW</b>	(FCA9H, 1)
Conteúdo:	Flag para indicar se o cursor será mostrado (0=não; outro valor, sim). Pode ser modificada pelo comando LOCATE.
<b>CSTYLE</b>	(FCAAH, 1)
Conteúdo:	Forma do cursor (0=bloco; outro valor, sub-alinhado).
<b>CAPST</b>	(FCABH, 1)
Conteúdo:	Estado da tecla CAPS LOCK (0=desligada; outro valor, ligada).
<b>KANAST</b>	(FCACH, 1)
Conteúdo:	Estado da tecla KANA (0=desligada; outro valor, ligada).
<b>KANAMD</b>	(FCADH, 1)
Conteúdo:	Tipo de teclado (0=KANA, outro valor, JIS). Flag usada apenas em máquinas japonesas.
<b>FLBMEM</b>	(FCAEH, 1)
Conteúdo:	Flag para indicar carregamento de programa em BASIC (0=está carregando; outro valor, não).
<b>SCRMOD</b>	(FCAFH, 1)
Conteúdo:	Número do modo de tela atual.
<b>OLDSCR</b>	(FCB0H, 1)
Conteúdo:	Modo de tela do último modo texto.
<b>CASPRV</b>	(FCB1H, 1)
Valor inicial:	00H
Conteúdo:	Usada pelo cassete nos MSX1,MSX2 e MSX2+. No MSX turbo R, guarda o valor da porta A7H.
<b>BDRATR</b>	(FCB2H, 1)
Conteúdo:	Código de cor da borda. Usado por PAINT.
<b>GXPOS</b>	(FCB3H, 2)
Conteúdo:	Coordenada X gráfica.
<b>GYPOS</b>	(FCB5H, 2)
Conteúdo:	Coordenada Y gráfica.

<b>GRPACX</b>	(FCB7H, 2)
Conteúdo:	Acumulador gráfico para a coordenada X.
<b>GRPACY</b>	(FCB9H, 2)
Conteúdo:	Acumulador gráfico para a coordenada Y.
<b>DRWFGL</b>	(FCBBH, 1)
Conteúdo:	Flag usada pelo comando DRAW.
<b>DRWSCL</b>	(FCBCH, 1)
Conteúdo:	Fator de escala para o comando DRAW. O valor 0 indica que não será usada a escala.
<b>DRWANG</b>	(FCBDH, 1)
Conteúdo:	Ângulo para o comando DRAW.
<b>RUNBNF</b>	(FCBEH, 1)
Conteúdo:	Flag para indicar se o comando BLOAD ou BSAVE está em execução (somente para o sistema de disco)
<b>SAVENT</b>	(FCBFH, 2)
Valor inicial:	0000H
Conteúdo:	Endereço inicial para os comandos BSAVE e BLOAD (somente para o sistema de disco)

## 8.19 – ROTINAS DE EXPANSÃO DA BIOS

### **EXTBIO** (FFCAH)

Objetivo:	Expandir diretamente a BIOS do sistema.
Entrada:	A – Sempre 0. D – Identificador do dispositivo (o número de dispositivo 0 é usado para obter as extensões instaladas). E – Função a ser chamada.
<b>Nota:</b>	Consulte a seção “ROGINAS DA BIOS ESTENDIDA” para mais detalhes.

### **DISINT** (FFCFH)

Objetivo:	Chamada pela função 2 do “broadcast”.
Entrada:	Nenhuma.

**ENAINT** (FFD4H)

Objetivo: Chamada pela função 2 do “broadcast”.

Entrada: Nenhuma.

FFD9H~FFE6H → Contém o código das rotinas DISINT e ENAINT.

## 8.20 – ÁREA DE DADOS PARA OS SLOTS E PÁGINAS

**EXPTBL** (FCC1H, 4)

Valor inicial: Variável.

Conteúdo: Tabela de flags para indicar se os slots primários estão expandidos:

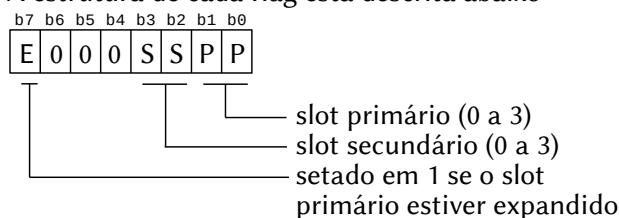
FCC1H → slot primário 0 (slot da Main-ROM).

FCC2H → slot primário 0 (slot da Main-ROM).

FCC3H → slot primário 0 (slot da Main-ROM).

FCC4H → slot primário 0 (slot da Main-ROM).

A estrutura de cada flag está descrita abaixo:

**SLTTBL** (FCC5H, 4)

Conteúdo: Estes quatro bytes contêm o estado possível dos quatro registradores de slot primário, no caso do slot estar expandido.

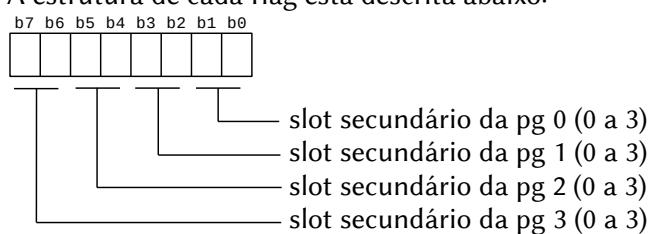
FCC5H → estado para slot primário 0

FCC6H → estado para slot primário 1

FCC7H → estado para slot primário 2

FCC8H → estado para slot primário 3

A estrutura de cada flag está descrita abaixo:



**SLTATTR** (FCC9H, 64)

Conteúdo: Tabela de atributos para cada página de cada slot.

	b7	b6	b5	b4	b3	b2	b1	b0
FCC9H -	B	D	I	-	-	-	-	-
FCCA9H -	B	D	I	-	-	-	-	-
FCCBH -	B	D	I	-	-	-	-	-
FCCCH -	B	D	I	-	-	-	-	-
FCCDH -	B	D	I	-	-	-	-	-
FCCEH -	B	D	I	-	-	-	-	-
FCCFH -	B	D	I	-	-	-	-	-
FD06H -	B	D	I	-	-	-	-	-
FD07H -	B	D	I	-	-	-	-	-
FD08H -	B	D	I	-	-	-	-	-

+-----+  
|      |  
|      |  
|      |  
+-----+

Quando 1, manip.  
 de instrução  
 Quando 1, manip.  
 de dispositivo  
 Quando 1,  
 programa BASIC

**SLTWRK** (FD09H, 128)

Conteúdo: Esta tabela aloca dois bytes como área de trabalho para cada página de cada slot.

FD09H -			Área de trabalho página 0 slot 0-0	
FD0AH -				
FD0BH -			Área de trabalho página 1 slot 0-0	
FD0CH -				
FD0DH -			Área de trabalho página 1 slot 0-0	
...				
FD86H -			Área de trabalho página 3 slot 3-3	
FD87H -				
FD88H -				

**PROCNM** (FD89H, 16)

Conteúdo: Armazena o nome de uma instrução expandida (comando CALL) ou expansão de dispositivo (comando OPEN). Um byte 0 indica o fim do nome.

**DEVICE** (FD99H, 1)

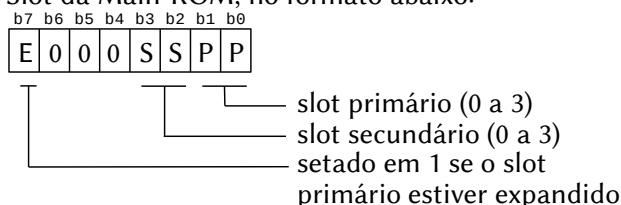
Conteúdo: Armazena o ID de um dispositivo em cartucho (0 a 3).

FD9AH~FFC9H → Área dos hooks (listados mais à frente)

### 8.20.1 – Slot da Main-ROM

**MINROM** (FFF7H, 1)

Conteúdo: Slot da Main-ROM, no formato abaixo:



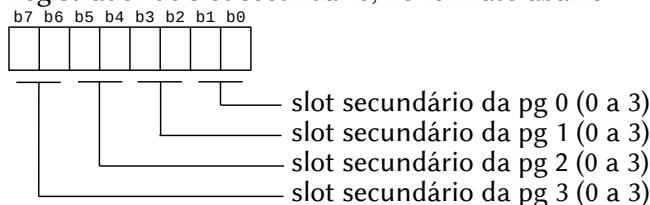
FFF8H~FFF9H → Não usados

FFFDH~FFFEH → Não usados

### 8.20.2 – Registrador de slot secundário

**SLTSL** (FFFFH, 1)

Conteúdo: Registrador de slot secundário, no formato abaixo:



## 8.21 – DESCRIÇÃO DOS HOOKS

**HKEYI** (FD9AH)

Chamada: Início do manipulador de interrupção (KEYINT, 0038H)

Objetivo: Adicionar rotinas de manipulação de interrupção.

Também pode se usar para testar quando a interrupção for causada por outro dispositivo que não o VDP.

<b>HTIMI</b>	(FD9FH)
Chamada:	Chamada pela rotina de interrupção (KEYINT, 0038H) logo após a leitura do registro de status 0 do VDP.
Objetivo:	Adicionar rotinas de manipulação de interrupção. Também pode ser usado para sincronizar a exibição gráfica, adicionando gráficos durante o Vblank.
<b>HCHPU</b>	(FDA4H)
Chamada:	Início da rotina CHPUT (00A2H).
Objetivo:	Conectar outros dispositivos de console além da tela. O registrador A contém o código do caractere quando este hook é chamado.
<b>HDSPC</b>	(FDA9H)
Chamada:	Início da rotina DSPSCR (apresenta cursor).
Objetivo:	Conectar outros dispositivos de console além da tela.
<b>HERAC</b>	(FDAEH)
Chamada:	Início da rotina ERASCR (apaga cursor)
Objetivo:	Conectar outros dispositivos de console além da tela.
<b>HDSPF</b>	(FDB3H)
Chamada:	Início da rotina DSPFNK (apresenta teclas de função).
Objetivo:	Conectar outros dispositivos de console além da tela.
<b>HERAF</b>	(FDB8H)
Chamada:	Início da rotina ERAFNK (apaga teclas de função)
Objetivo:	Conectar outros dispositivos de console além da tela.
<b>HTOTE</b>	(FDBDH)
Chamada:	Início da rotina TOTEXT (força tela para modo texto)
Objetivo:	Conectar outros dispositivos de console além da tela.
<b>HCHGE</b>	(FDC2H)
Chamada:	Início da rotina CHGET (pega um caractere).
Objetivo:	Conectar outros dispositivos de console além do teclado.
<b>HINIP</b>	(FDC7H)
Chamada:	Início da rotina INIPAT (inicialização dos padrões dos caracteres).
Objetivo:	Usar outra tabela de caracteres.

**HKEYC** (FDCCH)

**Chamada:** Início da rotina KEYCOD (decodificador de caracteres do teclado).

**Objetivo:** Mudar a configuração do teclado. Quando este hook é chamado, o registrador A contém: (nº da linha) × 8 + nº da coluna da tecla pressionada na matriz do teclado.

**HKEYA** (FDD1H)

**Chamada:** Início de MSXIO NMI (KEY EASY)

**Objetivo:** Mudar a maneira que uma tecla é interpretada.

**HNMI** (FDD6H)

**Chamada:** Início do manipulador de interrupção não mascarável (NMI, 0066H).

**Objetivo:** A NMI é desabilitada em um MSX padrão; logo, este hook não tem uso.

**HPINL** (FDDBH)

**Chamada:** Início da rotina PINLIN (pega uma linha)

**Objetivo:** Usar outros dispositivos e/ou métodos de entrada, como 80 colunas de texto ou outros dispositivos de entrada além do teclado.

**HQINL** (FDE0H)

**Chamada:** Início rotina QINLIN (pega uma linha apresentando "?").

**Objetivo:** Usar outros dispositivos e/ou métodos de entrada, como 80 colunas de texto ou outros dispositivos de entrada além do teclado.

**HINLI** (FDE5H)

**Chamada:** Início da rotina INLIN.

**Objetivo:** Usar outros dispositivos e/ou métodos de entrada, como 80 colunas de texto ou outros dispositivos de entrada além do teclado.

**HONGO** (FDEAH)

**Chamada:** Início do manipulador do comando ON GOTO e ON GOSUB.

**Objetivo:** Desviar o acesso a estas instruções do BASIC.

**HDSKO** (FDEFH)

Chamada: Início do comando BASIC "DSKO\$".

Objetivo: Usado pela Disk-ROM para gravar um setor no disco.

**HSETS** (FDF4H)

Chamada: Início do comando BASIC "SET".

Objetivo: Adicionar novas funcionalidades ao comando SET. No MSX1, a instrução SET apenas chama este hook e retorna um erro. No MSX2 ou superior, instruções como SET SCREEN, SET ADJUST, etc podem ser manipuladas.

**HNAME** (FDF9H)

Chamada: Início do comando BASIC "NAME".

Objetivo: Conectar dispositivos de disco.

**HKILL** (FDFEH)

Chamada: Início do comando BASIC "KILL".

Objetivo: Conectar dispositivos de disco.

**H IPL** (FE03H)

Chamada: Início do comando BASIC "IPL" (Initial Program Loading).

Objetivo: Reservado. Não há uso conhecido para esta instrução, mas este hook pode ser usado para adicionar funções à instrução IPL.

**HCOPY** (FE08H)

Chamada: Início do comando BASIC "COPY".

Objetivo: conectar dispositivos de disco.

**HCMD** (FE0DH)

Chamada: Início do comando BASIC "CMD" (Comandos Expandidos).

Objetivo: Reservado. Não há uso conhecido para esta instrução, mas este hook pode ser usado para adicionar funções à instrução CMD.

**HDSKF** (FE12H)

Chamada: Início do comando BASIC "DSKF".

Objetivo: Conectar dispositivos de disco.

**HDSKI** (FE17H)

Chamada: Início do comando BASIC "DSKI\$".  
Objetivo: Conectar dispositivos de disco.

**HATTR** (FE1CH)

Chamada: Início do manipulador do comando BASIC "ATTR\$".  
Objetivo: Conectar dispositivos de disco.

**HLSET** (FE21H)

Chamada: Início do manipulador do comando BASIC "LSET".  
Objetivo: Conectar dispositivos de disco.

**HRSET** (FE26H)

Chamada: Início do manipulador do comando BASIC "RSET".  
Objetivo: Conectar dispositivos de disco.

**HFIEL** (FE2BH)

Chamada: Início do manipulador do comando FIELD.  
Objetivo: Conectar dispositivos de disco.

**HMKI\$** (FE30H)

Chamada: Início do manipulador do comando MKI\$.  
Objetivo: Conectar dispositivos de disco.

**HMKS\$** (FE35H)

Chamada: Início do manipulador do comando MKS\$.  
Objetivo: Conectar dispositivos de disco.

**HMKD\$** (FE3AH)

Chamada: Início do manipulador do comando MKD\$.  
Objetivo: Conectar dispositivos de disco.

**HCVI** (FE3FH)

Chamada: Início do manipulador do comando CVI.  
Objetivo: Conectar dispositivos de disco.

**HCVS** (FE44H)

Chamada: Início do manipulador do comando CVS.  
Objetivo: Conectar dispositivos de disco.

**HCVD** (FE49H)

Chamada: Início do manipulador do comando CVD.  
Objetivo: Conectar dispositivos de disco.

**HGETP** (FE4EH)

Chamada: Localizar FCB (pegar apontador de arquivo).  
Objetivo: Conectar dispositivos de disco.

**HSETP** (FE53H)

Chamada: Localizar FCB (setar apontador de arquivo).  
Objetivo: Conectar dispositivos de disco.

**HNOFO** (FE58H)

Chamada: Manipulador do comando OPEN (OPEN sem FOR).  
Objetivo: Conectar dispositivos de disco.

**HNULO** (FE5DH)

Chamada: Manipulador do comando OPEN (abrir arq. não usado).  
Objetivo: Conectar dispositivos de disco.

**HNTFL** (FE62H)

Chamada: Fecha buffer 0 de I/O.  
Objetivo: Conectar dispositivos de disco.

**HMERG** (FE67H)

Chamada: Início do manipulador dos comandos MERGE e LOAD.  
Objetivo: Conectar dispositivos de disco.

**HSAVE** (FE6CH)

Chamada: Início do manipulador do comando SAVE.  
Objetivo: Conectar dispositivos de disco.

**HBINS** (FE71H)

Chamada: Início do manipulador do comando SAVE (em binário).  
Objetivo: Conectar dispositivos de disco.

**HBINL** (FE76H)

Chamada: Início do manipulador do comando LOAD (em binário).  
Objetivo: Conectar dispositivos de disco.

**HFILE** (FE7BH)

Chamada: Início do manipulador do comando FILES.  
Objetivo: Conectar dispositivos de disco.

**HDGET** (FE80H)

Chamada: Início do manipulador dos comandos GET e PUT.  
Objetivo: Conectar dispositivos de disco.

**HFILO** (FE85H)

Chamada: Manipulador de saída sequencial.  
Objetivo: Conectar dispositivos de disco.

**HINDS** (FE8AH)

Chamada: Manipulador de entrada sequencial.  
Objetivo: Conectar dispositivos de disco.

**HRSLF** (FE8FH)

Chamada: Manipulador de seleção prévia de drive.  
Objetivo: Conectar dispositivos de disco.

**HSAVD** (FE94H)

Chamada: Reservar disco atual (comandos LOC e LOF).  
Objetivo: Conectar dispositivos de disco.

**HLOC** (FE99H)

Chamada: Início do manipulador da função LOC.  
Objetivo: Conectar dispositivos de disco.

**HLOF** (FE9EH)

Chamada: Início do manipulador da função LOF.  
Objetivo: Conectar dispositivos de disco.

**HEOF** (FEA3H)

Chamada: Início do manipulador da função EOF.  
Objetivo: Conectar dispositivos de disco.

**HFPOS** (FEA8H)

Chamada: Início do manipulador da função FPOS.  
Objetivo: Conectar dispositivos de disco.

**HBAKU** (FEADH)

Chamada: Início do manipulador da instrução LINEINPUT#.  
Objetivo: Conectar dispositivos de disco.

**HPARD** (FEB2H)

Chamada: Início da rotina que analisa o nome do dispositivo.  
Objetivo: Expandir ou adiconar nomes de dispositivos.

**HNODE** (FEB7H)

Chamada: Início da rotina NODEVN, que é chamada quando  
nenhum nome foi encontrado na tabela de nomes de  
dispositivos.  
Objetivo: Atribuir o nome do dispositivo padrão para outro  
dispositivo.

**HPOSD** (FEBCH)

Chamada: Analisar nome de dispositivo (SPCDEV POSDSK).  
Objetivo: Conectar dispositivos de disco.

**HDEVN** (FEC1H)

Chamada: Processar nome de dispositivo.  
Objetivo: Expandir nome lógico de dispositivo.

**HGEND** (FEC6H)

Chamada: Início da rotina que atribui nome de dispositivo.  
Objetivo: Expandir nome lógico de dispositivo.

**HRUNC** (FECBH)

Chamada: Início da rotina que Inicializa as variáveis do  
interpretador para os comandos RUN e NEW.  
Objetivo: Permite atribuir novas funções para os comandos.

**HCLEA** (FED0H)

Chamada: Inicializar variáveis do interpretador para comando  
CLEAR.  
Objetivo: Permite atribuir novas funções para o comando ou  
prevenir apagamento acidental de variáveis.

**HLOPD** (FED5H)

- Chamada: Inicializar variáveis do interpretador (geral).  
 Objetivo: Usar outros valores-padrão para variáveis.

**HSTKE** (FEDAH)

- Chamada: Início da rotina STKERR (erro de pilha), usada pela instrução CLEAR do Basic.  
 Objetivo: Este hook é chamado após a verificação de ROMs executáveis em cada slot na inicialização o MSX, imediatamente antes do sistema iniciar o ambiente BASIC ou DOS. Portanto permite reexecutar automaticamente a ROM após a instalação dos discos.

**HISFL** (FEDFH)

- Chamada: Início da rotina ISFLIO, que testa se o arquivo deve ser gravado ou lido.

**HOUTD** (FEE4H)

- Chamada: Início da rotina OUTDO, que enviar um caractere para a tela ou para a impressora.

**HCRDO** (FEE9H)

- Chamada: Início da rotina que envia CR+LF para a rotina OUTDO.  
 Objetivo: Permite usar uma impressora com alimentação automática de linha, por exemplo.

**HDSKC** (FEEEH)

- Chamada: Entrada de atributo de disco.

**HDOGR** (FEF3H)

- Chamada: Início iníco da rotina interna DOGRPH, usada pelas instruções gráficas do BASIC (LINE, CIRCLE, etc.)  
 Objetivo: Alterar ou expandir as instruções gráficas.

**HPRGE** (FEF8H)

- Chamada: Final da execução de um programa BASIC.  
 Objetivo: Adicionar rotina a ser executada após o término do programa BASIC.

**HERRP** (FEFDH)

- Chamada: Início da rotina de apresentação de mensagens de erro.  
 Objetivo: Adicionar ou alterar mensagens de erro.

**HERRF** (FF02H)

- Chamada: Final da rotina de apresentação de mensagens de erro.  
 Objetivo: Adicionar rotina a ser executada após a apresentação da mensagem de erro.

**HREAD** (FF07H)

- Chamada: “Ok” do loop principal (interpretador pronto).  
 Objetivo: Adicionar rotina a ser executada após a apresentação do prompt (“Ok”).

**HMAIN** (FF0CH)

- Chamada: Início do loop principal de execução de texto BASIC do interpretador.  
 Objetivo: Adicionar rotina a ser executada sempre que o interpretador BASIC for acessado.

**HDIRD** (FF11H)

- Chamada: Início da execução de comando direto (declaração direta).  
 Objetivo: Adicionar rotina ou prevenir execuções.

**HFINI** (FF16H)

- Chamada: Início da rotina FININT, que inicia a interpretação de uma instrução BASIC.  
 Objetivo: Alterar o processamento das instruções BASIC.

**HFINE** (FF1BH)

- Chamada: Fim da rotina FININT, que inicializa a interpretação de uma instrução BASIC.

**HCRUN** (FF20H)

- Chamada: Início da rotina CRUNCH (42B9H), que converte um texto BASIC da forma ASCII para a forma atomizada.

**HCRUS** (FF25H)

- Chamada: Início da rotina CRUSH (4353H), que procura uma palavra reservada na lista alfabética da ROM.

- HISRE** (FF2AH)  
 Chamada: Início da rotina ISRESV (437CH), quando uma palavra reservada é encontrada pela rotina CRUSH.
- HNTFN** (FF2FH)  
 Chamada: Início da rotina NTFN2 (43A4H), quando uma palavra reservada é seguida por um número de linha.
- HNOTR** (FF34H)  
 Chamada: Início da rotina NOTRSV (44EBH), quando a sequência de caracteres examinada pela rotina CRUNCH não é uma palavra reservada.
- HSNGF** (FF39H)  
 Chamada: Início do manipulador do comando FOR.
- HNEWS** (FF3EH)  
 Chamada: Início da rotina NEWSTT (4601H) do interpretador, que executa um texto BASIC atomizado.
- HGONE** (FF43H)  
 Chamada: Início da rotina GONE2, usada pelas instruções de salto (GOTO, THEN, etc).
- HCHRG** (FF48H)  
 Chamada: Início da rotina CHRGET (entrada de caractere pelo teclado).  
 Objetivo: Usar outro teclado.
- HRETU** (FF4DH)  
 Chamada: Início do manipulador do comando RETURN.
- HPTRF** (FF52H)  
 Chamada: Início do manipulador do comando PRINT.
- HCOMP** (FF57H)  
 Chamada: Início da rotina interna COMPRT (4A94H), usada pelo manipulador do comando PRINT.

**HFINP** (FF5CH)

- Chamada: Início da rotina que zera PRTFLG e PRTFIL para finalização do comando PRINT.  
 Objetivo: Adicionar rotina a ser executada após o comando PRINT.

**HTRMN** (FF61H)

- Chamada: Início do manipulador de erro dos comandos READ e INPUT.  
 Objetivo: Processamento do erro.

**HFRME** (FF66H)

- Chamada: Rotina FRMEVL (4C64H) – Avaliador de Expressões.  
 Objetivo: Permite adicionar novas funções matemáticas.  
 Entrada: HL = apontador para o texto BASIC  
 Saída: HL = apontador para a expressão encontrada  
 VALTYP (F663H) – Tipo de valor da expressão  
 DAC (F7F6H) = valor encontrado

**HNTPL** (FF6BH)

- Chamada: Rotina FRMEVL (4CA6H) – Avaliador de Expressões.  
 Objetivo: Permite adicionar novas funções matemáticas.

**HEVAL** (FF70H)

- Chamada: Avaliador de Fatores (4DD9H)  
 Objetivo: Permite adicionar novas funções matemáticas.

**HOKNO** (FF75H)

- Chamada: Início da rotina de função transcendental do interpretador BASIC (hook removido no MSX turbo R. Foi substituído por HMDIN).  
 Objetivo: Permite adicionar novas funções matemáticas.

**HMDIN** (FF75H)

- Chamada: Início da rotina de manipulação das interrupções da interface MIDI (Somente no MSX turbo R com MIDI interna).  
 Objetivo: Adicionar ou alterar funcionalidades da interface MIDI.

**HFING** (FF7AH)

- Chamada: Avaliador de fatores.

**HISMI** (FF7FH)

Chamada: Início do manipulador do comando MID\$.

**HWIDT** (FF84H)

Chamada: Início do manipulador do comando WIDTH.

**HLIST** (FF89H)

Chamada: Início do manipulador do comando LIST.

**HBUFL** (FF8EH)

Chamada: De-simbolizar para comando LIST (532DH).

**HFRQI** (FF93H)

Chamada: Converte para inteiro (543FH). Hook removido no MSX turbo R. Substituído por HMDTM.

**HMDTM** (FF93H)

Chamada: Início da rotina de manipulação do timer da interface MIDI (Somente no MSX turbo R com MIDI interna).

Objetivo: Adicionar ou alterar funcionalidades da interface MIDI.

**HSCNE** (FF98H)

Chamada: Início da rotina SCNEX2 (5514H) do interpretador BASIC (conversão de um número de linha em um endereço de memória e vice-versa)

**HFRET** (FF9DH)

Chamada: Procura um local livre para armazenar o próximo descritor de uma variável alfanumérica (string).

**HPTRG** (FFA2H)

Chamada: Início da rotina PTRGET (5EA9H) do interpretador BASIC, que obtém o ponteiro de uma variável.

Objetivo: Usar outro valor padrão para as variáveis.

**HPHYD** (FFA7H)

Chamada: Início da rotina PHYDIO (physical disk input-output).

Objetivo: Conectar dispositivos de disco.

**HFORM** (FFACH)

Chamada: Início da rotina FORMAT (format disk).

Objetivo: Conectar dispositivos de disco.

**HERRO** (FFB1H)

Chamada: Início do manipulador de erro.

Objetivo: Manipulação de erros por programas aplicativos.

**HLPTO** (FFB6H)

Chamada: Início da rotina LPTOUT (00A5H).

Objetivo: Usar outros modelos de impressoras.

**HLPTS** (FFBBH)

Chamada: Início da rotina LPTSTT (00A5H).

Objetivo: Usar outros modelos de impressoras.

**HSCRE** (FFC0H)

Chamada: Início do manipulador do comando SCREEN.

Objetivo: Expandir o comando SCREEN.

**HPLAY** (FFC5H)

Chamada: Início do manipulador do comando PLAY.

Objetivo: Expandir o comando PLAY.

## 9 – ROTINAS DA BIOS

Este apêndice fornece a descrição das rotinas da BIOS disponíveis para o usuário.

Existem vários tipos de rotinas da BIOS, as que estão na Main-ROM, as que estão na Sub-ROM, as rotinas do Math-Pack e as rotinas de extensão acessadas por EXTBIO na área de trabalho, além de várias outras disponibilizadas por cartuchos de expansão e das rotinas do interpretador BASIC.

A Notação para as rotinas é a seguinte:

**LABEL** (Endereço da rotina / localização)

Função: descreve a função da rotina.

Entrada: descreve os parâmetros para a chamada da rotina.

Saída: descreve os parâmetros re retorno da rotina.

Registradores: lista os registradores modificados pela rotina.

### 9.1 – ROTINAS DA MainROM

#### 9.1.1 – Rotinas RST

**CHKRAM** (0000H/Main)

Função: Testa a RAM e inicializa as variáveis de sistema. Uma chamada a esta rotina provocará um reset por software.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: todos

**SYNCHR** (0008H/Main)

Função: Testa se o caractere apontado por (HL) é o especificado. Se não for, gera “Syntax error”; caso contrário chama CHRGTR (0010H).

Entrada: O caractere a ser testado deve estar em (HL) e o caractere para comparação após a instrução RST (parâmetro em linha), conforme o exemplo abaixo:

```

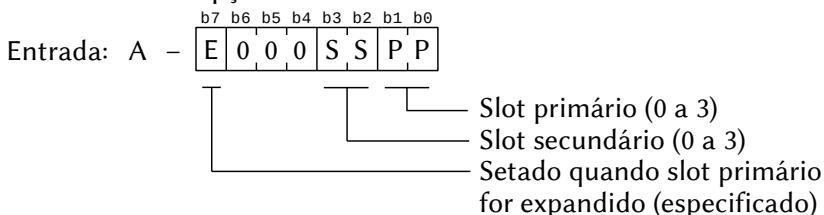
LD   HL,CARACT
RST 008H
DEFB 'A'
|
CARACT: DEFB 'B'
```

Saída: HL é incrementado em 1 e A recebe (HL). Quando o caractere testado for numérico, a flag CY é setada; o fim de declaração (00H ou 3AH) seta a flag Z.

Registradores: AF, HL.

#### **RDSL T (000CH/Main)**

Função: Lê um byte de memória no slot especificado em A. As interrupções são desabilitadas durante a leitura.



HL – Endereço de memória a ser lido.

Saída: A – Contém o valor do byte lido.

Registradores: AF, BC, DE.

#### **CHRGTR (0010H/Main)**

Função: Pega um caractere (token) do texto BASIC.

Entrada: HL – Endereço do caractere a ser lido.

Saída: HL é incrementado em 1 e A recebe (HL). Quando o caractere for numérico, a flag CY é setada; o fim de declaração (00H ou 3AH) seta a flag Z.

Registradores: AF, HL.

#### **WRSL T (0014H/Main)**

Função: Escreve um byte de memória no slot especificado em A. As interrupções são desabilitadas durante a escrita.

Entrada: A – Indicador de slot (igual a RDSL T – 000CH).

HL – Endereço para a escrita do byte.

E – Byte a ser escrito.

Saída: Nenhuma

Registradores: AF, BC, D.

#### **OUTDO (0018H/Main)**

Função: Envia um byte para o dispositivo atual.

Entrada: A – Byte a ser enviado. Se PRTFLG (F416H) for diferente de 0, o byte é enviado para a impressora; se PTRFIL (F864H) for diferente de 0, o byte é enviado ao arquivo especificado por PTRFIL.

Saída: Nenhuma.

Registradores: Nenhum.

#### **CALSLT (001CH)**

Função: Chama uma rotina em qualquer slot (chamada inter-slot).

Entrada: IY – O ID de slot deve ser especificado nos 8 bits mais altos no mesmo formato de RDSLت (000CH).

IX – Endereço da rotina a ser chamada.

Saída: Depende da rotina chamada.

Registradores: Depende da rotina chamada.

#### **DCOMPR (0020H)**

Função: Compara HL com DE.

Entrada: HL, DE.

Saída: Seta a flag Z se HL = DE; seta a flag CY se HL < DE.

Registradores: AF.

#### **ENASLT (0024H)**

Função: Habilita uma página em qualquer slot. Somente as páginas 1 e 2 podem ser habilitadas por esta rotina; a 0 e a 3 não. As interrupções são desativadas durante a habilitação.

Entrada: A – Indicador de slot (igual a RDSLت – 000CH).

Saída: Nenhuma.

Registradores: Todos.

#### **GETYPR (0028H/Main)**

Função: Obtém o tipo de operando contido em DAC.

Entrada: Nenhuma

Saída: Flags CY, S, Z e P/V, conforme tabela abaixo:

Inteiro:	C=1	S=1*	Z=0	P/V=1
----------	-----	------	-----	-------

Precisão simples:	C=1	S=0	Z=0	P/V=0*
-------------------	-----	-----	-----	--------

Precisão dupla:	C=0*	S=0	Z=0	P/V=1
-----------------	------	-----	-----	-------

String:	C=1	S=0	Z=1*	P/V=1
---------	-----	-----	------	-------

Obs.: Os tipos podem ser reconhecidos unicamente pelas flags marcadas com “\*”.

Registradores: AF.

**CALLF** (0030H/Main)

Função: Chama uma rotina em qualquer slot usando parâmetros em linha. Muito útil para chamar rotinas através dos hooks. A sequência de chamada é a seguinte:

```
RST 030H ;chama CALLF
DEFB n      ;n é ID de slot (igual a RDSLT)
DEFW nn     ;nn é o endereço a ser chamado
RET         ;retorno ao sistema
```

Entrada: Pelo método descrito.

Saída: Depende da rotina chamada.

Registradores: Depende da rotina chamada (mais AF).

**KEYINT** (0038H/Main)

Função: Executa a rotina de interrupção e varredura do teclado.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**9.1.2 – Rotinas para inicialização I/O****INITIO** (0000H/Main)

Função: Inicializa os dispositivos de entrada e saída.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**INIFNK** (003EH/Main)

Função: Inicializa o conteúdo das teclas de função.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**9.1.3 – Rotinas para acesso ao VDP****DISSCR** (0041H/Main)

Função: Desabilita a apresentação de tela.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: AF, BC.

**ENASCR** (0044H/Main)

Função: Habilita a apresentação de tela.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF, BC.

**WRTVDP** (0047H/Main)

Função: Escreve um byte de dados em um registrador do VDP.  
 Entrada: C – Registrador que receberá o dado. Pode variar de 0 a 7 para MSX1, de 0 a 23 / 32 a 46 para MSX2 e de 0 a 23 / 25 a 27 / 32 a 46 para MSX2+ ou superior.  
 B – Byte de dados  
 Saída: Nenhuma.  
 Registradores: AF, BC.

**RDVRM** (004AH/Main)

Função: Lê um byte da VRAM. Esta rotina lê apenas os 14 bits mais baixos de endereço (16K para o TMS9918 do MSX1). Para acessar toda a VRAM é necessário usar a rotina NRDVRM (0174H).  
 Entrada: HL – Endereço da VRAM a ser lido.  
 Saída: A – Byte lido.  
 Registradores: AF.

**WRTVRM** (004DH/Main)

Função: Escreve um byte da VRAM. Esta rotina acessa apenas os 14 bits mais baixos de endereço (16K para o TMS9918 do MSX1). Para acessar toda a VRAM é necessário usar a rotina NWRVRM (0177H).  
 Entrada: HL – Endereço da VRAM a ser escrito.  
 A – Byte a ser escrito.  
 Saída: Nenhuma.  
 Registradores: AF.

**SETRD** (0050H/Main)

Função: Prepara a VRAM para leitura sequencial usando a função de autoincremento de endereço do VDP. É um meio de leitura mais rápido que o uso de um loop com a rotina RDVRM (004AH). Esta rotina acessa apenas os 14 bits mais baixos de endereço (16K para o TMS9918 do MSX1). Para acessar toda a VRAM é necessário usar a rotina NSETRD (016EH).

Entrada: HL – Endereço na VRAM para início da leitura

Saída: Nenhuma.

Registradores: AF.

#### **SETWRT (0053H/Main)**

Função: Prepara a VRAM para escrita sequencial usando a função de autoincremento de endereço do VDP. As características são as mesmas de SETRD (0050H). Para acessar toda a VRAM é necessário usar a rotina NSTWRT (0171H).

Entrada: HL – Endereço da VRAM para início da leitura.

Saída: Nenhuma.

Registradores: AF.

#### **FILVRM (0056H/Main)**

Função: Preenche uma área da VRAM com um único byte de dados. Esta rotina acessa apenas os 14 bits mais baixos de endereço (16K para o TMS9918 do MSX1). Para acessar toda a VRAM é necessário usar a rotina BIGFIL (016BH).

Entrada: HL – Endereço da VRAM para início da escrita.

BC – Quantidade bytes a serem escritos.

A – Byte a ser escrito.

Saída: Nenhuma.

Registradores: AF, BC.

#### **LDIRMV (0059H/Main)**

Função: Copia um bloco de dados da VRAM para a RAM.

Entrada: HL – Endereço fonte na VRAM.

DE – Endereço destino na RAM.

BC – Tamanho do bloco (comprimento).

Obs.: todos os 16 bits de endereço são válidos.

Saída: Nenhuma.

Registradores: Todos.

#### **LDIRVM (005CH/Main)**

Função: Copia um bloco de dados da RAM para a VRAM.

Entrada: HL – Endereço fonte na RAM.

DE – Endereço destino na VRAM.

BC – Tamanho do bloco (comprimento).

Obs.: todos os 16 bits de endereço são válidos.

Saída: Nenhuma.

Registradores: Todos.

**CHGMOD** (005FH/Main)

Função: Troca os modos de tela. Esta rotina não inicializa a paleta de cores. Para isso, é necessário usar a rotina CHGMDP (01B5H/Sub-ROM).

Entrada: A – 0 a 3 para MSX1, 0 a 8 para MSX2 ou 0 a 12 para MSX2+ ou superior (Obs.: o modo 9 só é válido para micros coreanos).

Saída: Nenhuma.

Registradores: Todos.

**CHGCLR** (0062H/Main)

Função: Troca as cores da tela.

Entrada: FORCLR (F3E9H) – Cor de frente.  
BAKCLR (F3EAH) – Cor de fundo.  
BDRCLR (F3EBH) – Cor da borda.

Saída: Nenhuma.

Registradores: Todos.

**NMI** (0066H/Main)

Função: Executa a rotina NMI (Non-Maskable Interrupt – Interrupção não marcarável). Em uma máquina MSX padrão, apenas faz uma chamada ao hook HNMI (FDD6H) e retorna sem nenhum processamento.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**CLRSPR** (0069H/Main)

Função: Inicializa todos os sprites. A tabela de padrões dos sprites é limpa (preenchida com zeros), os números dos sprites são inicializados com a série 0~31 e a cor dos sprites é igualada à cor de fundo. A localização vertical dos sprites é colocada em 209 (para as Screens 0 a 3) ou em 217 (para as Screens 4 a 9 ou 10 a 12).

Entrada: SCRMOD (FCAFH) – Modo screen.

Saída: Nenhuma.

Registradores: Todos.

**INITXT** (006CH/Main)

- Função:** Inicializa a tela no modo texto (Screen 0). A paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/Sub-ROM).
- Entrada:** TXTNAM (F3B3H) – Endereço da tabela de nomes.  
 TXTCGP (F3B7H) – Endereço da tabela de padrões.  
 LINL40 (F3AEH) – Número de caracteres por linha.
- Saída:** Nenhuma.
- Registradores:** Todos.

**INIT32** (006FH/Main)

- Função:** Inicializa a tela no modo gráfico 1 (Screen 1). A paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/Sub-ROM).
- Entrada:** T32NAM (F3BDH) – End. da tabela de nomes dos caracteres.  
 T32COL (F3BFH) – End. da tabela de cores dos caracteres.  
 T32CGP (F3C1H) – End. da tabela de padrões dos caracteres.  
 T32ATR (F3C3H) – End. da tabela de atributos dos sprites.  
 T32PAT (F3C5H) – End. da tabela de padrões dos sprites.
- Saída:** Nenhuma.
- Registradores:** Todos.

**INIGRP** (0072H/Main)

- Função:** Inicializa a tela no modo gráfico de alta resolução do MSX1 (Screen 2). A paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/Sub-ROM).
- Entrada:** GRPNAM (F3C7H) – endereço da tabela de nomes dos padrões.  
 GRPCOL (F3C9H) – Endereço da tabela de cores.  
 GRPCGP (F3CBH) – Endereço da tabela geradora de padrões.  
 GRPATR (F3CDH) – Endereço da tabela de atributos dos sprites.  
 GRPPAT (F3CFH) – Endereço da tabela de padrões dos sprites.
- Saída:** Nenhuma.
- Registradores:** Todos.

**INIMLT** (0075H/Main)

- Função:** Inicializa a tela no modo multicor do MSX1 (Screen 3). A paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/Sub-ROM).

Entrada: MLTNAM (F3D1H) – Endereço da tabela de nomes dos padrões.  
 MLTCOL (F3D3H) – Endereço da tabela de cores.  
 MLTCGP (F3D5H) – Endereço da tabela geradora de padrões.  
 MLTATTR (F3D7H) – Endereço da tabela de atributos dos sprites.  
 MLTPAT (F3D9H) – Endereço da tabela de padrões dos sprites.  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **SETTXT** (0078H/Main)

Função: Coloca apenas o VDP no modo texto (Screen 0).  
 Entrada: Igual a INITXT (006CH).  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **SETT32** (007BH/Main)

Função: Coloca apenas o VDP no modo gráfico 1 (Screen 1).  
 Entrada: Igual a INIT32 (006FH).  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **SETGRP** (007EH/Main)

Função: Coloca apenas o VDP no modo gráfico 2 (Screen 2).  
 Entrada: Igual a INIGRP (0072H).  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **SETMLT** (0081H/Main)

Função: Coloca apenas o VDP no modo multicor (Screen 3).  
 Entrada: Igual a INIMLT (0075H).  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **CALPAT** (0084H/Main)

Função: Retorna o endereço da tabela geradora do padrão de um sprite.  
 Entrada: A – Número do sprite.  
 Saída: HL – Endereço na VRAM.  
 Registradores: AF, DE, HL.

**CALATR** (0087H/Main)

Função: Retorna o endereço da tabela de atributos de um sprite.

Entrada: A – Número do sprite.

Saída: HL – Endereço na VRAM.

Registradores: AF, DE, HL.

**GSPSIZ** (008AH/Main)

Função: Retorna o tamanho atual dos sprites.

Entrada: Nenhuma.

Saída: A – Tamanho do sprite em bytes. A flag CY é setada se o tamanho for 16 x 16 e resetada caso contrário.

Registradores: AF.

**GRPPRT** (008DH/Main)

Função: Apresenta um caractere em uma tela gráfica.

Entrada: A – Código ASCII do caractere. Quando a screen for 5 a 12 especifique código de oper. lógica em LOGOPR (FB02H).

Saída: Nenhuma.

Registradores: Nenhum.

**9.1.4 – Rotinas para acesso ao PSG****GICINI** (0090H/Main)

Função: Inicializa o PSG e seta os valores iniciais para o comando PLAY.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**WRTPSG** (0093H/Main)

Função: Escreve um byte de dados em um registrador do PSG.

Entrada: A – Número do registrador do PSG.

E – Byte de dados a ser escrito.

Saída: Nenhuma.

Registradores: Nenhum.

**RDPSC** (0096H/Main)

Função: Lê o conteúdo de um registrador do PSG.

Entrada: A – Número do registrador do PSG.

Saída: A – Byte lido.

Registradores: Nenhum.

**STRTMS** (0099H/Main)

Função: Testa se o comando PLAY está sendo executado. Se não estiver, inicia a execução.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**9.1.5 – Rotinas para acesso ao teclado, tela e impressora****CHSNS** (009CH/Main)

Função: Verifica o buffer de teclado.

Entrada: Nenhuma.

Saída: Se a flag Z estiver setada, o buffer está vazio; caso contrário a flag Z será resetada.

Registradores: AF.

**CHGET** (009FH/Main)

Função: Entrada de um caractere pelo teclado com espera.

Entrada: Nenhuma.

Saída: A – Código ASCII do caractere.

Registradores: AF.

**CHPUT** (00A2H/Main)

Função: Apresenta um caractere na tela de texto.

Entrada: A – Código ASCII do caractere a ser apresentado.

Saída: Nenhuma.

Registradores: Nenhum.

**LPTOUT** (00A5H/Main)

Função: Envia um caractere para a impressora.

Entrada: Código ASCII do caractere a ser enviado.

Saída: Se falhar, CY retorna setada.

Registradores: F.

**LPTSTT** (00A8H/Main)

Função: Testa o status da impressora.

Entrada: Nenhuma.

Saída: A = 0 (e flag Z = 1) → Impressora não está pronta.  
255 (e flag Z = 0) → Impressora pronta.

Registradores: AF.

**CNVCHR** (00A8H/Main)

Função: Testa o cabeçalho gráfico e converte se necessário.  
 Entrada: A – Código ASCII do caractere.  
 Saída: CY=0 – Não há cabeçalho gráfico.  
          CY=1 e Z=1 – O código convertido é colocado em A.  
          CY=1 e Z=0 – O código não convertido retorna em A.  
 Registradores: AF.

**PINLIN** (00AEH/Main)

Função: Coleta uma linha de texto e armazena em um buffer até que  
          que a tecla RETURN ou STOP seja pressionada.  
 Entrada: Nenhuma.  
 Saída: HL – Endereço inicial do buffer menos 1.  
          CY – Setada se a tecla STOP foi pressionada.  
 Registradores: Todos.

**INLIN** (00B1H/Main)

Função: Mesma que PINLIN (00AEH), mas setando AUTFLG (F6AAH).  
 Entrada: Nenhuma.  
 Saída: HL – Endereço inicial do buffer menos 1.  
          CY – Setada se a tecla STOP foi pressionada.  
 Registradores: Todos.

**QINLIN** (00B4H/Main)

Função: Executa INLIN (00B1H) apresentando “?” e um espaço.  
 Entrada: Nenhuma.  
 Saída: HL – Endereço inicial do buffer menos 1.  
          CY – Setada se a tecla STOP foi pressionada.  
 Registradores: Todos.

**BREAKX** (00B7H/Main)

Função: Testa se CTRL+STOP são pressionadas juntas. Durante a  
          verificação as interrupções são desabilitadas.  
 Entrada: Nenhuma.  
 Saída: CY – Setada se CTRL+STOP estão pressionadas.  
 Registradores: AF.

**BEEP** (00C0H/Main)

Função: Gera um beep.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: Todos.

**CLS** (00C3H/Main)

Função: Limpa a tela.

Entrada: A flag Z deve estar setada.

Saída: Nenhuma.

Registradores: AF, BC, DE.

**POSIT** (00C6H/Main)

Função: Move o cursor para uma coordenada específica.

Entrada: H – coordenada X (horizontal)

L – coordenada Y (vertical)

Saída: Nenhuma.

Registradores: AF.

**FNKSB** (00C9H/Main)

Função: Testa se os comandos associados às teclas de função estão sendo apresentados na tela verificando a flag FNKFLG (FBCEH) e inverte o estado de apresentação (se a flag estiver ligada, desliga e se estiver desligada, liga).

Entrada: FNKFLG (FBCEH).

Saída: Nenhuma.

Registradores: Todos.

**ERAFNK** (00CCH/Main)

Função: Desliga a apresentação das teclas de função.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**DSPFNK** (00CFH/Main)

Função: Liga a apresentação das teclas de função.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**TOTEXT** (00D2H/Main)

Função: Força a tela para o modo texto (Screen 0 ou 1).

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

### 9.1.6 – Rotinas de acesso I/O para jogos

#### **GTSTCK** (00D5H/Main)

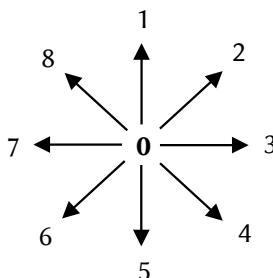
Função: Retorna o estado do joystick ou das teclas do cursor.

Entrada: A = 0 → Teclas do cursor.

1 → Joystick na porta 1.

2 → Joystick na porta 2.

Saída: A – Direção do joystick ou das teclas de função conforme a ilustração abaixo:



Registradores: Todos.

#### **GTTRIG** (00D8H/Main)

Função: Retorna o estado dos botões do mouse, joystick ou barra do teclado.

Entrada: A = 0 → barra de espaço.

1 → joystick na porta 1, botão A.

2 → joystick na porta 2, botão A.

3 → joystick na porta 1, botão B.

4 → joystick na porta 2, botão B.

Saída: A – 0 → botão testado não está pressionado.

255 → botão testado está pressionado.

Registradores: AF, BC.

#### **GTPAD** (00DBH/Main)

Função: Retorna o estado de um touch pad, de um trackball ou de um mouse ligado a um dos conectores de joystick.

Entrada: A – código de função:

0 – Checa touch pad na porta 1 (255 se conectado)

1 – Retorna a coordenada X (horizontal).

- 2 – Retorna a coordenada Y (vertical).
- 3 – Retorna o estado de tecla (255 se pressionada).
- 4 – Checa touch pad na porta 2 (255 se conectado).
- 5 – Retorna a coordenada X (horizontal).
- 6 – Retorna a coordenada Y (vertical).
- 7 – Retorna o estado de tecla (255 se pressionada).
- 8 – Checa caneta ótica (255 se conectada / tocando tela).
- 9 – Retorna a coordenada X (horizontal).
- 10 – Retorna a coordenada Y (vertical).
- 11 – Retorna o estado de chave (255 se pressionada).
- 12 – Checa mouse na porta 1 (255 se conectado).
- 13 – Retorna offset da coordenada X (horizontal).
- 14 – Retorna offset da coordenada Y (vertical).
- 15 – Sempre 0.
- 16 – Checa mouse na porta 2 (255 se conectado).
- 17 – Retorna offset da coordenada X (horizontal).
- 18 – Retorna offset da coordenada Y (vertical).
- 19 – Sempre 0.
- 20 – Checa 2<sup>a</sup> caneta ótica (255 se conectada ou tocando a tela).
- 21 – Retorna a coordenada X (horizontal).
- 22 – Retorna a coordenada Y (vertical).
- 23 – Retorna o estado de tecla (255 se pressionada).

Saída: A – estado ou valor, conforme descrito acima.

Registradores: Todos.

Obs.: Para os códigos de função 8 a 23, chama NEWPAD (01ADH) na SubROM. Para o MSX turbo R, as funções de caneta ótica (8 a 11) foram eliminadas.

#### **GTPDL (00DEH/Main)**

Função: Retorna os valores de paddles ligados aos conectores de joystick.

Entrada: A – identificação do paddle (1 a 12).

1, 3, 5, 7, 9, 11 – paddles ligados na porta 1.

2, 4, 6, 8, 10, 12 – paddles ligados na porta 2.

Saída: A – valor lido (0 a 255).

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

### **9.1.7 – Rotinas de acesso I/O para gravador cassette**

#### **TAPION (00E1H/Main)**

Função: Lê o header da fita após ligar o motor do cassete.

Entrada: Nenhuma.

Saída: Se falhar, a flag CY retorna setada.

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **TAPIN (00E4H/Main)**

Função: Lê dados da fita.

Entrada: Nenhuma.

Saída: A – Byte lido.

CY – Setada se a leitura falhar.

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **TAPIOF (00E7H/Main)**

Função: Para a leitura da fita.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **TAPOON (00EAH/Main)**

Função: Escreve o header na fita após ligar o motor do cassete.

Entrada: A = 0 → Header curto; outro valor → Header longo.

Saída: Se falhar, a flag CY retorna setada.

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **TAPOUT (00EDH/Main)**

Função: Escreve dados na fita.

Entrada: A – Byte a ser escrito.

Saída: Se falhar, a flag CY retorna setada.

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **TAPOOF (00F0H/Main)**

Função: Para a escrita na fita.

Entrada: Nenhuma.

Saída: Se falhar, a flag CY retorna setada.

Registradores: Todos.

Obs.: Esta rotina foi eliminada no MSX turbo R.

#### **STMOTR** (00F3H/Main)

Função: Liga ou desliga o motor do cassete.

Entrada: A = 0 → Liga o motor

1 → Desliga o motor

255 → inverte o estado do motor

Saída: Nenhuma.

Registradores: AF.

Obs.: Esta rotina foi eliminada no MSX turbo R.

### **9.1.8 – Rotinas para a fila do PSG**

#### **LFTQ** (00F6H/Main)

Função: Retorna o número de bytes livres em uma fila musical do PSG.

Entrada: A – Número da fila (0, 1 ou 2).

Saída: HL – Espaço livre deixado na fila.

Registradores: AF, BC, HL.

#### **PUTQ** (00F9H/Main)

Função: Coloca um byte em uma das filas musicais do PSG.

Entrada: A – número da fila (0, 1 ou 2).

E – byte de dados.

Saída: Flag Z setada se a fila estiver cheia.

Registradores: AF, BC, HL.

#### **GETVCP** (0150H/Main)

Função: Retorna o endereço do byte 2 no buffer de voz do PSG.

Entrada: A – Número da voz (0, 1 ou 2)

Saída: HL – Endereço no buffer de voz.

Registradores: AF, HL.

#### **GETVC2** (0153H/Main)

Função: Retorna o endereço de qualquer byte no buffer de voz do PSG.

Entrada: VOICEN (FB38H) – Número da voz (0, 1 ou 2).

L – Número do byte (0 a 36).

Saída: HL – Endereço no buffer de voz.

Registradores: AF, HL.

### 9.1.9 – Rotinas para as telas gráficas do MSX1

#### **RIGHTC** (00FCH/Main)

Função: Desloca o pixel atual uma posição para a direita.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF.

#### **LEFTC** (00FFH/Main)

Função: Desloca o pixel atual uma posição para a esquerda.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF.

#### **UPC** (0102H/Main)

Função: Desloca o pixel atual uma posição para cima.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF.

#### **TUPC** (0105H/Main)

Função: Testa a posição do pixel atual e, se possível, desloca o mesmo uma posição para cima.  
 Entrada: Nenhuma.  
 Saída: CY = 1 se o pixel não pôde ser movido por exceder o limite superior da tela.  
 Registradores: AF.

#### **DOWNC** (0108H/Main)

Função: Desloca o pixel atual uma posição para baixo.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF.

#### **TDOWNC** (010BH/Main)

Função: Testa a posição do pixel atual e, se possível, desloca o mesmo uma posição para baixo.  
 Entrada: Nenhuma.  
 Saída: CY = 1 se o pixel não pôde ser movido por exceder o limite inferior da tela.  
 Registradores: AF.

**SCALXY** (010EH/Main)

Função: Limita as coordenadas do pixel para a área visível da tela.

Entrada: BC – Coordenada X (horizontal).

DE – Coordenada Y (vertical).

Saída: BC – Coordenada X limitada à borda

DE – Coordenada Y limitada à borda

CY = 1 se houver limitação das coordenadas.

Registradores: AF.

**MAPXYC** (0111H/Main)

Função: Converte um par de coordenadas gráficas no endereço físico do pixel atual (coloca o “cursor” na coordenada).

Entrada: BC – Coordenada X (horizontal).

DE – Coordenada Y (vertical).

Saída: Nenhuma.

Registradores: AF, D, HL.

**FETCHC** (0114H/Main)

Função: Retorna o endereço físico do pixel atual.

Entrada: Nenhuma.

Saída: A recebe o conteúdo de CMASK (F92CH).

HL recebe o conteúdo de CLOC (F92AH).

Registradores: A, HL.

**STOREC** (0117H/Main)

Função: Estabelece o endereço físico do pixel atual.

Entrada: A é copiado para CMASK (F92CH).

HL é copiado para CLOC (F92AH).

Saída: Nenhuma.

Registradores: Nenhum.

**SETATTR** (011AH/Main)

Função: Estabelece a cor de frente para as rotinas SETC (0120H) e NSETCX (0123H).

Entrada: A – Código de cor (0 a 15).

Saída: CY – Setada se o código de cor for inválido.

Registradores: F.

**READC** (011DH/Main)

Função: Retorna o código de cor do pixel atual.

Entrada: Nenhuma.

Saída: A – Código de cor do pixel atual (0 a 15).  
 Registradores: AF, EI.

**SETC** (0120H/Main)

Função: Estabelece a cor do pixel atual.  
 Entrada: ATRBYT (F3F2H) – Código de cor (0 a 15), estabelecida por SETATTR (011AH).  
 Saída: Nenhuma.  
 Registradores: AF, EI.

**NSETCX** (0123H/Main)

Função: Estabelece a cor de múltiplos pixels horizontais a partir do pixel atual, para a direita.  
 Entrada: ATRBYT (F3F2H) – Código de cor (0 a 15), estabelecida por SETATTR (011AH).  
           HL – Número de pixels a colorir.  
 Saída: Nenhuma.  
 Registradores: AF, EI.

**GTASPC** (0126H/Main)

Função: Retorna as razões de aspecto da instrução CIRCLE.  
 Entrada: Nenhuma.  
 Saída: DE recebe o conteúdo de ASPCT1 (F40BH).  
           HL recebe o conteúdo de ASPCT2 (F40DH).  
 Registradores: DE, HL.

**PNTINI** (0129H/Main)

Função: Estabelece a cor de contorno para a instrução PAINT.  
 Entrada: A – Código de cor do contorno (0 a 15).  
 Saída: CY – Setada se o código de cor for inválido.  
 Registradores: AF.

**SCANR** (012CH/Main)

Função: Usada pelo manipulador da instrução PAINT para percorrer uma área, da esquerda para a direita, partindo do pixel atual até que um código de cor igual a BDRATTR (FCB2H) seja encontrado ou a borda da tela seja atingida.  
 Entrada: B = 0 → Não preenche a área percorrida.  
           255 → Preenche a área percorrida.  
           DE – Número de pulsos (pixels da mesma cor ignorados).

Saída: HL – Número de pixels percorridos.  
 DE – Número de pulsos restantes.  
 Registradores: AF, BC, DE, HL, EI.

#### **SCANL** (012FH/Main)

Função: Mesma que SCANR (012CH), exceto que o percurso será da direita para a esquerda e a área será sempre preenchida.

Entrada: Nenhuma.

Saída: HL – Número de pixels percorridos.

Registradores: AF, BC, DE, HL, EI.

### **9.1.10 – Miscelânea**

#### **CHGCAP** (0132H/Main)

Função: Altera o estado lo LED do Caps Lock.

Entrada: A = 0 apaga o LED; outro valor, acende o LED.

Saída: Nenhuma.

Registradores: AF.

#### **CHGSND** (0135H/Main)

Função: Altera o estado da porta de 1 bit geradora de som.

Entrada: A = 0 desliga o bit, outro valor liga o bit.

Saída: Nenhuma.

Registradores: AF.

#### **RSLREG** (0138H/Main)

Função: Lê o conteúdo do registrador de slot primário.

Entrada: Nenhuma.

Saída: A – Valor lido.

Registradores: A.

#### **WSLREG** (013BH/Main)

Função: Escreve no registrador de slot primário.

Entrada: A – Valor a ser escrito.

Saída: Nenhuma.

Registradores: Nenhum.

#### **RDVDP** (013EH/Main)

Função: Lê o registrador de status do VDP.

Entrada: Nenhuma.

Saída: A – Valor lido.

Registradores: A.

**SNSMAT** (0141H/Main)

Função: Lê o valor de uma linha da matriz de teclado.

Entrada: A – Linha a ser lida.

Saída: A – Valor lido (o bit correspondente a uma tecla pressionada será 0).

Registradores: AF, C.

**ISFLIO** (014AH/Main)

Função: Testa se está ocorrendo uma operação de I/O de dispositivo.

Entrada: Nenhuma.

Saída: A = 0 se o dispositivo estiver ativo (está ocorrendo operação de I/O); outro valor o dispositivo está inativo.

Registradores: AF.

**OUTDLP** (014DH/Main)

Função: Saída formatada para a impressora. Difere de LPTOUT nos seguintes pontos:

- 1 – Se o caractere enviado for um TAB (09H) serão enviados espaços até atingir um múltiplo de 8;
- 2 – Para impressoras não-MSX, hiraganas são convertidos para katakanas e caracteres gráficos são convertidos para caracteres de 1 byte;
- 3 – Se houver falha, ocorrerá um erro de I/O.

Entrada: A – Caractere a ser enviado.

Saída: Nenhuma.

Registradores: F.

**KILBUF** (0156H/Main)

Função: Limpa o buffer de teclado.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: HL.

**CALBAS** (0159H/Main)

Função: Executa uma chamada inter-slot para qualquer rotina do interpretador BASIC.

Entrada: IX – Endereço a ser chamado.

Saída: Depende da rotina chamada.

Registradores: Depende da rotina chamada.

### 9.1.11 – Rotinas para acesso ao sistema de disco

#### **PHYDIO** (0144H/Main)

Função: Ler ou gravar um ou mais setores no drive especificado.

Entrada: CY = 0 → leitura.

1 → gravação.

A – Número do drive (0 = A; 1 = B; etc).

B – Número de setores a ler ou gravar.

C – ID de formatação do disco:

F0H – 63 setores por trilha (para HD's)

F8H – 80 trilhas, 9 setores por trilha, face simples.

F9H – 80 trilhas, 9 setores por trilha, face dupla.

FAH – 80 trilhas, 8 setores por trilha, face simples.

FBH – 80 trilhas, 8 setores por trilha, face dupla.

FCH – 40 trilhas, 9 setores por trilha, face simples.

FDH – 40 trilhas, 9 setores por trilha, face dupla.

DE – Número do primeiro setor a ser lido ou gravado.

HL – Endereço da RAM a partir do qual serão gravados os setores a ler do disco ou retirados os setores a gravar no disco.

Saída: CY – Setada se houve erro de leitura ou gravação.

A – código de erro se CY=1:

0 – protegido contra escrita.

2 – Não pronto.

4 – Erro de dados.

6 – Erro de busca.

8 – Setor não encontrado.

10 – Erro de escrita.

12 – Parâmetros inválidos.

14 – Memória insuficiente.

16 – Erro indefinido.

B – Número de setores efetivamente lidos ou escritos.

Registradores: Todos.

Obs.: Em algumas interfaces de HD, quando o bit 7 do registrador C é setado, será usado um esquema de endereçamento de 23 bits e os bits 0-6 do registrador C devem conter os bits 23-16 do número do setor.

**FORMAT** (0147H/Main)

Função: Formatar um disquete. Ao ser chamada, serão apresentadas uma série de perguntas que deverão ser respondidas para iniciar a formatação. Não há padrão para essas perguntas; elas podem ser diferentes para cada interface de drive.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**9.1.12 – Rotinas adicionadas para o MSX2****SUBROM** (015CH/Main)

Função: Executa uma chamada inter-slot para a SubROM.

Entrada: IX – Endereço a ser chamado (ao mesmo tempo coloca IX na pilha).

Saída: Depende da rotina chamada.

Registradores: IY, AF', BC', DE', HL', mais os registradores modificados pela rotina chamada.

**EXTROM** (015FH/Main)

Função: Executa uma chamada inter-slot para a SubROM.

Entrada: IX – Endereço a ser chamado.

Saída: Depende da rotina chamada.

Registradores: IY, AF', BC', DE', HL', mais os registradores modificados pela rotina chamada.

**CHKSLZ** (0162H/Main)

Função: Procura slots para a SubROM.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**CHKNEW** (0165H/Main)

Função: Testa o modo Screen.

Entrada: Nenhuma.

Saída: CY = 1 se Screen for 5, 6, 7 ou 8.

Registradores: AF.

**EOL** (0168H/Main)

Função: Apaga até o fim da linha.  
 Entrada: H – Coordenada X do cursor.  
           L – Coordenada Y do cursor.  
 Saída: Nenhuma.  
 Registradores: Todos.

**BIGFIL** (016BH/Main)

Função: Mesma que FILVRM (0056H): preenche uma área da VRAM com um único byte de dados, com a diferença que as Screens 0 a 3 não são testadas e o preenchimento pode ultrapassar o limite de 16K dessas screens.  
 Entrada: HL – endereço da VRAM para início da escrita.  
           BC – quantidade bytes a serem escritos.  
           A – byte a ser escrito.  
 Saída: Nenhuma.  
 Registradores: AF, BC.

**NSETRD** (016EH/Main)

Função: Prepada a VRAM para leitura sequencial usando a função de autoincremento de endereço do VDP.  
 Entrada: HL – Endereço da VRAM a partir do qual os dados serão lidos. Todos os bits são válidos.  
 Saída: Nenhuma.  
 Registradores: AF.

**NSTWRT** (0171H/Main)

Função: Prepada a VRAM para escrita sequencial usando a função de autoincremento de endereço do VDP.  
 Entrada: HL – Endereço da VRAM a partir do qual os dados serão escritos. Todos os bits são válidos.  
 Saída: Nenhuma.  
 Registradores: AF.

**NRDVRM** (0174H/Main)

Função: Lê o conteúdo de um byte da VRAM.  
 Entrada: HL – Endereço da VRAM a ser lido.  
 Saída: A – Byte lido.  
 Registradores: AF.

**NWRVRM** (0177H/Main)

Função: Escreve um byte de dados na VRAM.  
 Entrada: HL – Endereço da VRAM a ser escrito.  
           A – Byte a ser escrito.  
 Saída: Nenhuma.  
 Registradores: AF.

**9.1.13 – Rotinas adicionadas para o MSX2+****RDRES** (017AH/Main)

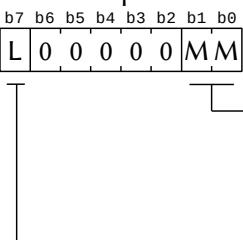
Função: Retorna o estado do reset.  
 Entrada: Nenhuma.  
 Saída: A – b7=0 indica reset total (por hardware)  
           b7=1 indica reset parcial (por software)  
 Registradores: A.  
 Obs.: No reset total (por hardware) o conteúdo da RAM é apagado e aparece o logo “MSX” na inicialização. No reset parcial (por software) apenas a área de trabalho é inicializada e não aparece o logo “MSX” na inicialização.

**WRRES** (017DH/Main)

Função: Modifica o estado do reset.  
 Entrada: A – b7=0 para reset total (por hardware)  
           b7=1 para reset parcial (por software)  
 Saída: Nenhuma.  
 Registradores: Nenhum.

**9.1.14 – Rotinas adicionadas para o MSX turbo R****CHGCPU** (0180H/Main)

Função: Trocar de microprocessador (modo de operação).

Entrada: A – 

Modo de operação:  
 00 – Z80  
 01 – R800 ROM  
 10 – R800 DRAM  
 LED de modo no painel  
 0 – apagado, 1 – aceso

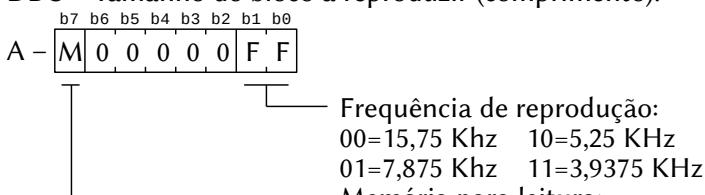
Saída: Nenhuma.  
 Registradores: AF.

### **GETCPU (0183H/Main)**

Função: Retorna em qual modo o computador está operando.  
 Entrada: Nenhuma.  
 Saída: A = 0→Z80; 1→R800 ROM; 2→R800 DRAM.  
 Registradores: AF.

### **PCMPLY (0186H/Main)**

Função: Reproduzir sons através do PCM.  
 Entrada: EHL – Endereço para início da leitura.  
 DBC – Tamanho do bloco a reproduzir (comprimento).



Obs.: Usar 15,75 KHz apenas no modo R800 DRAM.  
 Saída: CY = 0 → Reprodução OK; 1→Erro na reprodução.

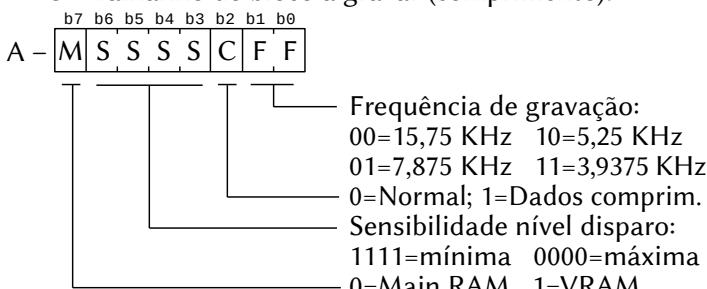
A = 0 → Erro na especificação da frequência.  
 A = 1 → Interrupção por CTRL+STOP.

EHL – Endereço até onde efetivamente reproduziu.

Registradores: Todos.

### **PCMREC (0189H/Main)**

Função: Digitalizar sons através do PCM.  
 Entrada: EHL – Endereço para início da gravação.  
 DBC – Tamanho do bloco a gravar (comprimento).



Obs.: A frequência de 15,75 KHz só pode ser usada no modo R800 DRAM.

Saída: CY = 0 → Gravação OK; 1 → Erro na gravação.  
 A = 0 → Erro na especificação da frequência.  
 1 → Interrupção por CTRL+STOP.  
 EHL – Endereço até onde efetivamente gravou.

Registradores: Todos.

### 9.1.15 – Rotinas inter-slot da área de trabalho

#### **RDPRIM** (F380H/Work Area)

Função: Lê um byte de qualquer endereço de qualquer slot.

Entrada: A – Slot primário a ser lido.  
 D – Slot atual para retorno.

Saída: E – Byte lido.

#### **WRPRIM** (F385H/Work Area)

Função: Escreve um byte em qualquer endereço de qualquer slot.

Entrada: A – Slot primário a ser lido.  
 D – Slot atual para retorno.  
 E – Byte a ser escrito.

Saída: Nenhuma.

#### **CLPRIM** (F38CH/Work Area)

Função: Chama um endereço em qualquer slot.

Entrada: A – slot primário que contém a rotina.  
 IX – endereço a ser chamados.  
 PUSH AF – Slot atual para retorno (em A).

Saída: Depende da rotina chamada.

## 9.2 – ROTINAS DA SubROM

### 9.2.1 – Rotinas para funções gráficas do BASIC

#### **PAINT** (0069H/SubROM) – Comando BASIC

Função: Pinta uma área em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC (parâmetros do comando PAINT).

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**PSET** (006DH/SubROM) – Comando BASIC

Função: Desenha um ponto em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC (parâmetros do comando PSET).

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**ATRSCN** (0071H/SubROM) – Comando BASIC

Função: Retorna atributos de cor.

Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**GLINE** (0075H/SubROM) – Comando BASIC

Função: Desenha uma linha em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**DOBOXF** (0079H/SubROM) – Comando BASIC

Função: Desenha um retângulo preenchido em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**DOLINE** (007DH/SubROM) – Comando BASIC

Função: Desenha uma linha em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC (parâmetros do comando PSET).

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**BOXLIN** (0081H/SubROM) – Comando BASIC

Função: Desenha um retângulo em uma tela gráfica.

Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

**PUTSPR** (0151H/SubROM) – Comando BASIC

Função: Apresenta um sprite em uma tela gráfica.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**COLOR** (0155H/SubROM) – Comando BASIC

Função: Altera as cores da tela, dos sprites ou da paleta.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**SCREEN** (0159H/SubROM) – Comando BASIC

Função: Troca os modos de tela.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**WIDTH** (015DH/SubROM) – Comando BASIC

Função: Altera o número de caracteres por linha no modo texto.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**VDP** (0161H/SubROM) – Comando BASIC

Função: Escreve dados em um registrador do VDP.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**VDPF** (0165H/SubROM) – Comando BASIC

Função: Lê dados de um registrador do VDP.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**BASE** (0169H/SubROM) – Comando BASIC

Função: Escreve dados no registrador de base do VDP.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**BASEF** (0169H/SubROM) – Comando BASIC

Função: Lê dados do registrador de base do VDP.

Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.

Registradores: Todos.

### 9.2.2 – Rotinas para funções gráficas

**DOGRPH** (0085H/SubROM)

Função: Desenha uma linha em uma tela gráfica.

Entrada: BC – Coordenada X inicial.

HL – Coordenada Y inicial.

GXPOS (FCB3H) – Coordenada X final.

GYPOS (FCB5H) – Coordenada Y de final.

ATRBYT (F3F2H) – Atributos.

LOGOPR (FB02H) – Código de operação lógica.

Saída: Nenhuma.

Registradores: AF.

**GRPPRT** (0089H/SubROM)

Função: Imprime um caractere em uma tela gráfica do MSX2.

Entrada: A – Código ASCII do caractere.

ATRBYT (F3F2H) – Atributos.

LOGOPR (FB02H) – Código de operação lógica.

Saída: Nenhuma.

Registradores: Todos.

**SCALXY** (008DH/SubROM)

Função: Limita as coordenadas do pixel para a área visível da tela.

Entrada: BC – Coordenada X (horizontal).

DE – Coordenada Y (vertical).

Saída: BC – Coordenada X limitada à borda.

DE – Coordenada Y limitada à borda.

CY = 1 se houver limitação das coordenadas.

Registradores: AF.

**MAPXYC** (0091H/SubROM)

Função: Converte um par de coordenadas gráficas no endereço físico do pixel atual (coloca o “cursor” na coordenada).

Entrada: BC – Coordenada X (horizontal).  
DE – Coordenada Y (vertical).

Saída: Screen 3: HL, CLOC(F92AH) – Endereço na VRAM.  
A, CMASK(F92CH) – Máscara.  
Screen 5,12: HL, CLOC(F92AH) – Coordenada X.  
A, CMASK(F92CH) – Coordenada Y.

Registradores: F.

#### **READC** (0095H/SubROM)

Função: Lê os atributos de um pixel.

Entrada: CLOC(F92AH) – Coordenada X.  
CMASK(F92CH) – Coordenada Y.

Saída: A – Atributo.

Registradores: AF.

#### **SETATTR** (0099H/SubROM)

Função: Seta atributo em ATRBYT (F3F2H).

Entrada: A – Atributo.

Saída: CY = 1 se houver erro no atributo.

Registradores: F.

#### **SETC** (009DH/SubROM)

Função: Seta atributo do pixel.

Entrada: CLOC(F92AH) – Coordenada X.  
CMASK(F92CH) – Coordenada Y.  
ATRBYT (F3F2H) – Atributo.

Saída: Nenhuma.

Registradores: AF.

#### **TRIGHT** (00A1H/SubROM)

Função: Move um pixel para a direita.

Entrada: CLOC(F92AH) – Coordenada X.  
CMASK(F92CH) – Coordenada Y.

Saída: CLOC(F92AH) – Nova coordenada X.  
CMASK(F92CH) – Nova coordenada Y.  
CY = 1 se a borda da tela for atingida.

Registradores: AF.

Obs.: Somente para Screen 3.

**RIGHTC** (00A5H/SubROM)

Função: Move um pixel para a direita.

Entrada: CLOC(F92AH) – Coordenada X.  
CMASK(F92CH) – Coordenada Y.

Saída: CLOC(F92AH) – Nova coordenada X.  
CMASK(F92CH) – Nova coordenada Y.

Registradores: AF.

Obs.: Somente para Screen 3. Esta rotina é igual a TRIGHT (00A1H) exceto pela ausência do retorno da flag CY.

**TLEFTC** (00A9H/SubROM)

Função: Move um pixel para a esquerda.

Entrada: Igual a TRIGHT (00A1H/SubROM).

Saída: Igual a TRIGHT (00A1H/SubROM).

Registradores: AF.

Obs.: Somente para Screen 3.

**LEFTC** (00ADH/SubROM)

Função: Move um pixel para a esquerda.

Entrada: Igual a RIGHTC (00A5H/SubROM).

Saída: Igual a RIGHTC (00A5H/SubROM).

Registradores: AF.

Obs.: Somente para Screen 3. Esta rotina é igual a TLEFTC (00A9H) exceto pela ausência do retorno da flag CY.

**TDOWNC** (00B1H/SubROM)

Função: Move um pixel para baixo.

Entrada: Igual a TRIGHT (00A1H/SubROM).

Saída: Igual a TRIGHT (00A1H/SubROM).

Registradores: AF.

Obs.: Somente para Screen 3.

**DOWNC** (00B5H/SubROM)

Função: Move um pixel para baixo.

Entrada: Igual a RIGHTC (00A5H/SubROM).

Saída: Igual a RIGHTC (00A5H/SubROM).

Registradores: AF.

Obs.: Somente para Screen 3. Esta rotina é igual a TDOWNC (00A9H) exceto pela ausência do retorno da flag CY.

**TUPC** (00B9H/SubROM)

Função: Move um pixel para cima.  
 Entrada: Igual a TRIGHT (00A1H/SubROM).  
 Saída: Igual a TRIGHT (00A1H/SubROM).  
 Registradores: AF.  
 Obs.: Somente para Screen 3.

**UPC** (00BDH/SubROM)

Função: Move um pixel para cima.  
 Entrada: Igual a RIGHTC (00A5H/SubROM).  
 Saída: Igual a RIGHTC (00A5H/SubROM).  
 Registradores: AF.  
 Obs.: Somente para Screen 3. Esta rotina é igual a TUPC (00B9H) exceto pela ausência da flag CY.

**SCANR** (00C1H/SubROM)

Função: Percorre uma área, da esquerda para a direita, partindo do pixel atual até que um código de cor igual a BDRATR (FCB2H) seja encontrado ou a borda da tela seja atingida.  
 Entrada: B = 0 → Não preenche a área percorrida.  
           255 → Preenche a área percorrida.  
           C – Contador até a borda.  
 Saída: DE – Contador até a borda.  
           C – Flag de pixel modificado.  
 Registradores: Todos.

**SCANL** (00C5H/SubROM)

Função: Percorre uma área, da direita para a esquerda, partindo do pixel atual até que um código de cor igual a BDRATR (FCB2H) seja encontrado ou a borda da tela seja atingida.  
 Entrada: DE – Contador até a borda.  
 Saída: DE – Contador até a borda.  
           C – Flag de pixel modificado.  
 Registradores: Todos.

**NVBXLN** (00C9H/SubROM)

Função: Desenha um retângulo.  
 Entrada: BC – Coordenada X inicial.  
           HL – Coordenada Y inicial.  
           GXPOS (FCB3H) – Coordenada X final.

**GYPOS** (FCB5H) – Coordenada Y de final.

**ATRBYT** (F3F2H) – Atributos.

**LOGOPR** (FB02H) – Código de operação lógica.

Saída: Nenhuma.

Registradores: Todos.

#### **NVBXFL** (00CDH/SubROM)

Função: Desenha um retângulo preenchido.

Entrada: Igual a NVBXLN (00C9H/SubROM).

Saída: Nenhuma.

Registradores: Todos.

### 9.2.3 – Rotinas duplicadas (iguais às da MainROM)

#### **CHGMOD** (00D1H/SubROM)

Função: Troca os modos de tela.

Entrada: A – 0 a 3 para MSX1, 0 a 8 para MSX2 ou 0 a 12 para MSX2+ ou superior (Obs.: o modo 9 só é válido para micros coreanos).

Saída: Nenhuma.

Registradores: Todos.

#### **INITXT** (00D5H/SubROM)

Função: Inicializa a tela no modo texto (Screen 0).

Entrada: TXTNAM (F3B3H) – Endereço da tabela de nomes.

TXTCGP (F3B7H) – Endereço da tabela de padrões.

LINL40 (F3AEH) – Número de caracteres por linha.

Saída: Nenhuma.

Registradores: Todos.

#### **INIT32** (00D9H/SubROM)

Função: Inicializa a tela no modo Screen 1.

Entrada: T32NAM (F3BDH) – Endereço da tabela de nomes dos caracteres.

T32COL (F3BFH) – Endereço da tabela de cores dos caracteres.

T32CGP (F3C1H) – Endereço da tabela de padrões dos caracteres.

T32ATTR (F3C3H) – Endereço da tabela de atributos dos sprites.

T32PAT (F3C5H) – Endereço da tabela de padrões dos sprites.

Saída: Nenhuma.

Registradores: Todos.

**INIGRP** (00DDH/SubROM)

Função: Inicializa a tela no modo Screen 2.  
 Entrada: GRPNAM (F3C7H) – Endereço da tabela de nomes dos padrões.  
 GRPCOL (F3C9H) – Endereço da tabela de cores.  
 GRPCGP (F3CBH) – Endereço da tabela geradora de padrões.  
 GRPATR (F3CDH) – Endereço da tabela de atributos dos sprites.  
 GRPPAT (F3CFH) – Endereço da tabela de padrões dos sprites.  
 Saída: Nenhuma.  
 Registradores: Todos.

**INIMLT** (00E1H/SubROM)

Função: Inicializa a tela no modo multicor do MSX1 (Screen 3).  
 Entrada: MLTNAM (F3D1H) – Endereço da tabela de nomes dos padrões.  
 MLTCOL (F3D3H) – Endereço da tabela de cores.  
 MLTCGP (F3D5H) – Endereço da tabela geradora de padrões.  
 MLTATTR (F3D7H) – Endereço da tabela de atributos dos sprites.  
 MLTPAT (F3D9H) – Endereço da tabela de padrões dos sprites.  
 Saída: Nenhuma.  
 Registradores: Todos.

**SETTXT** (00E5H/SubROM)

Função: Coloca apenas o VDP no modo texto (Screen 0).  
 Entrada: Igual a INITXT (00D5H/SubROM).  
 Saída: Nenhuma.  
 Registradores: Todos.

**SETT32** (00E9H/SubROM)

Função: Coloca apenas o VDP no modo gráfico 1 (Screen 1).  
 Entrada: Igual a INIT32 (00D9H/SubROM).  
 Saída: Nenhuma.  
 Registradores: Todos.

**SETGRP** (00EDH/SubROM)

Função: Coloca apenas o VDP no modo gráfico 2 (Screen 2).  
 Entrada: Igual a INIGRP (00E1H/SubROM).  
 Saída: Nenhuma.  
 Registradores: Todos.

**SETMLT** (00F1H/SubROM)

Função: Coloca apenas o VDP no modo multicor (Screen 3).  
 Entrada: Igual a INIMLT (0075H).

Saída: Nenhuma.

Registradores: Todos.

#### **CLRSPR** (00F5H/SubROM)

Função: Inicializa todos os sprites. A tabela de padrões dos sprites é limpa (preenchida com zeros), os números dos sprites são inicializados com a série 0~31 e a cor dos sprites é igualada à cor de fundo. A localização vertical dos sprites é colocada em 209 (para as Screens 0 a 3) ou em 217 (para as Screens 4 a 9 ou 10 a 12).

Entrada: SCRMOD (FCAFH) – Modo screen.

Saída: Nenhuma.

Registradores: Todos.

#### **CALPAT** (00F9H/SubROM)

Função: Retorna o endereço da tabela geradora do padrão de um sprite.

Entrada: A – Número do sprite.

Saída: HL – Endereço na VRAM.

Registradores: AF, DE, HL.

#### **CALATR** (00FDH/SubROM)

Função: Retorna o endereço da tabela de atributos de um sprite.

Entrada: A – Número do sprite.

Saída: HL – Endereço na VRAM.

Registradores: AF, DE, HL.

#### **GSPSIZ** (0101H/SubROM)

Função: Retorna o tamanho atual dos sprites.

Entrada: Nenhuma.

Saída: A – Tamanho do sprite em bytes. A flag CY é setada se o tamanho for 16 x 16 e resetada caso contrário.

Registradores: AF.

### **9.2.4 – Rotinas diversas para o MSX2 ou superior**

#### **GETPAT** (0105H/SubROM)

Função: Retorna o padrão de um caractere.

Entrada: A – Código ASCII do caractere.

Saída: PATWRK (FC40H) – padrão do caractere.

Registradores: Todos.

**WRTVRM** (0109H/SubROM)

Função: Escreve um byte de dados na VRAM.

Entrada: HL – Endereço da VRAM.

A – Byte a ser escrito.

Saída: Nenhuma.

Registradores: AF.

**RDVRM** (010DH/SubROM)

Função: Lê o conteúdo de um byte da VRAM.

Entrada: HL – Endereço da VRAM a ser lido.

Saída: A – Byte lido.

Registradores: AF.

**CHGCLR** (0111H/SubROM)

Função: Troca as cores da tela.

Entrada: FORCLR (F3E9H) – Cor de frente

BAKCLR (F3EAH) – Cor de fundo

BDRCLR (F3EBH) – Cor da borda

Saída: Nenhuma.

Registradores: Todos.

**CLSSUB** (0115H/SubROM)

Função: Limpar a tela.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**CLRTXT** (0119H/SubROM)

Função: Limpar tela de texto.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**DSPFNK** (011DH/SubROM)

Função: Apresenta o conteúdo das teclas de função.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**DELLNO** (0121H/SubROM)

Função: Apaga uma linha no modo texto.  
 Entrada: L – Número da linha a ser apagada.  
 Saída: Nenhuma.  
 Registradores: Todos.

**INSLNO** (0125H/SubROM)

Função: Adiciona uma linha no modo texto.  
 Entrada: L – Número da linha a ser adicionada.  
 Saída: Nenhuma.  
 Registradores: Todos.

**PUTVRM** (0129H/SubROM)

Função: Coloca um caractere em uma tela de texto.  
 Entrada: H – Coordenada Y.  
           L – Coordenada X.  
 Saída: Nenhuma.  
 Registradores: AF.

**WRTVDP** (012DH/SubROM)

Função: Escreve um byte de dados em um registrador do VDP.  
 Entrada: C – Número do registrador que receberá o dado.  
           B – Byte de dados.  
 Saída: Nenhuma.  
 Registradores: AF, BC.

**VDPSTA** (0131H/SubROM)

Função: Lê o conteúdo de um registrador do VDP.  
 Entrada: A – Número do registrador a ser lido (0 a 9).  
 Saída: A – Valor lido.  
 Registradores: F.

**KYKLOK** (0135H/SubROM)

Função: Controle da tecla KANA e do LED KANA em micros japoneses.  
 Entrada: ?  
 Saída: ?  
 Registradores: ?

**PUTCHR** (0139H/SubROM)

Função: Pega um código de tecla, converte para KANA e o coloca em um buffer (em micros japoneses).

Entrada: CY = 0 → Faz conversão; 1 → Não faz conversão.

Saída: ?

Registradores: Todos.

**SETPAG** (013DH/SubROM)

Função: Define as páginas de vídeo.

Entrada: DPPAGE (FAF5H) – página apresentada na tela.

ACPAGE (FAF6H) – página ativa para receber comandos.

Saída: Nenhuma.

Registradores: AF.

**NEWPAD** (01ADH/SubROM)

Função: Retorna o estado do mouse ou da caneta ótica.

Entrada: A – Código de função:

0 a 7 – Sem efeito.

8 – Checa caneta ótica (255 se conectada/tocando a tela).

9 – Retorna a coordenada X (horizontal).

10 – Retorna a coordenada Y (vertical).

11 – Retorna o estado de chave (255 se pressionada).

12 – Checa mouse na porta 1 (255 se conectado).

13 – Retorna offset da coordenada X (horizontal).

14 – Retorna offset da coordenada Y (vertical).

15 – Sempre 0.

16 – Checa mouse na porta 2 (255 se conectado).

17 – Retorna offset da coordenada X (horizontal).

18 – Retorna offset da coordenada Y (vertical).

19 – Sempre 0.

20 – Checa 2ª caneta ótica (255 se conectada ou tocando a tela).

21 – Retorna a coordenada X (horizontal).

22 – Retorna a coordenada Y (vertical).

23 – Retorna o estado de tecla (255 se pressionada).

Saída: A – Estado ou valor, conforme descrito acima.

Registradores: Todos.

**CHGMDP** (01B5H/SubROM)

Função: Troca os modos de tela e inicializa a paleta de cores.

Entrada: A – 0 a 3 para MSX1, 0 a 8 para MSX2 ou 0 a 12 para MSX2+ ou superior (O modo 9 só é válido para micros coreanos).

Saída: Nenhuma.

Registradores: Todos.

#### **KNJPRT** (01BDH/SubROM)

Função: Escreve um caractere Kanji em uma tela gráfica (Screens 5 a 8 ou 10 a 12). Esta rotina está presente apenas em micros com Kanji ROM.

Entrada: BC – Código JIS do caractere Kanji.

A – Modo de apresentação:

0 – Todas as linhas da tela.

1 – Linhas pares.

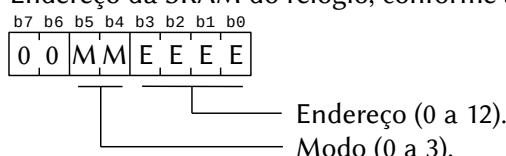
2 – Linhas ímpares.

Registradores: AF.

#### **REDCLK** (01F5H/SubROM)

Função: Lê um nibble de dados da memória do relógio (Clock-IC).

Entrada: C – Endereço da SRAM do relógio, conforme abaixo:



Saída: A – Nibble lido (4 bits mais baixos).

Registradores: AF.

#### **WRTCLK** (01F9H/SubROM)

Função: Escreve um nibble de dados na memória do relógio.

Entrada: C – Endereço da SRAM do relógio (igual a REDCLK).

A – Nibble a ser escrito (4 bits mais baixos).

Saída: Nenhuma.

Registradores: F.

### **8.2.5 – Rotinas de manipulação da paleta de cores**

#### **INIPLT** (0141H/SubROM)

Função: Inicializa a paleta de cores (a paleta atual é salva na VRAM).

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: AF, BC, DE.

**RSTPLT** (0145H/SubROM)

Função: Recupera a paleta de cores salva na VRAM.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: AF, BC, DE.

**GETPLT** (0149H/SubROM)

Função: Retorna os níveis de cores da paleta.  
 Entrada: A – Número de cor na paleta (0 a 15).  
 Saída: B – 4 bits altos para o nível de vermelho.  
           B – 4 bits baixos para o nível de azul.  
           C – 4 bits baixos para o nível de verde.

Registradores: AF, DE.

**SETPLT** (014DH/SubROM)

Função: Modifica os níveis de cores da paleta.  
 Entrada: D – Número de cor na paleta (0 a 15).  
           A – 4 bits altos para o nível de vermelho.  
           A – 4 bits baixos para o nível de azul.  
           E – 4 bits baixos para o nível de verde.  
 Saída: Nenhuma.  
 Registradores: AF.

**9.2.6 – Rotinas diversas usadas pelo BASIC****VPOKE** (0171H/SubROM) – Comando BASIC

Função: Escreve um byte de dados na VRAM.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**VPEEK** (0175H/SubROM) – Comando BASIC

Função: Lê um byte de dados da VRAM.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**SETS** (0179H/SubROM) – Comando BASIC

Função: Executa os parâmetros dos comandos BEEP, ADJUST, TIME e DATE.  
 Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**BEEP** (017DH/SubROM) – Comando BASIC

Função: Gera um beep.  
 Entrada: HL – Apontador para o início do texto BASIC.

Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**PROMPT** (0181H/SubROM) – Comando BASIC

Função: Apresenta o prompt do BASIC (“Ok” por padrão).  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**SDFSCR** (0185H/SubROM) – Comando BASIC

Função: Recupera os parâmetros de tela do Clock-IC. Quando CY=1, o conteúdo das teclas de função será apresentado.  
 Entrada: CY = 0 após chamar o MSXDOS.  
 Saída: ?  
 Registradores: Todos.

**SETSCR** (0189H/SubROM) – Comando BASIC

Função: Recupera os parâmetros de tela do Clock-IC e apresenta uma mensagem de boas vindas.  
 Entrada: ?  
 Saída: ?  
 Registradores: Todos.

**SCOPY** (018DH/SubROM) – Comando BASIC

Função: Executa cópias entre a VRAM, matrizes do BASIC e arquivos em disco.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.  
 Registradores: Todos.

**GETPUT** (01B1H/SubROM) – Comando BASIC

Função: Executa os parâmetros dos comandos GET TIME, GET DATE e PUT KANJI.  
 Entrada: HL – Apontador para o início do texto BASIC.  
 Saída: HL – Aponta para o final dos parâmetros do comando.

1060

Registradores: Todos.

### 9.2.7 – Rotinas de transferência de bloco (bit-blit)

#### **BLTVV** (0191H/SubROM)

Função: Transfere dados de uma área da VRAM para outra.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – SX – Coordenada X da fonte.

(F564H,2) – SY – Coordenada Y da fonte.

(F566H,2) – DX – Coordenada X de destino.

(F568H,2) – DY – Coordenada Y de destino.

(F56AH,2) – NX – Número de pixels na direção X.

(F56CH,2) – NY – Número de pixels na direção Y.

(F56EH,1) – CDUMMY – (não requer dados).

(F56FH,1) – ARGT – Seleciona a direção e a VRAM expandida (igual a R#45 do VDP).

(F570H,1) – LOGOP – Código de operação lógica (igual aos códigos do VDP).

Saída: CY = 0.

Registradores: Todos.

#### **BLTVM** (0195H/SubROM)

Função: Transfere dados da Main RAM para a VRAM.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – DPTR – Endereço fonte na RAM.

(F564H,2) – DUMMY – (não requer dados).

(F566H,2) – DX – Coordenada X de destino.

(F568H,2) – DY – Coordenada Y de destino.

(F56AH,2) – NX – Número de pixels na direção X  
(não requer dados; já preenchida).

(F56CH,2) – NY – Número de pixels na direção Y  
(não requer dados; já preenchida).

(F56EH,1) – CDUMMY – (não requer dados).

(F56FH,1) – ARGT – Seleciona a direção e a VRAM expandida (igual a R#45 do VDP).

(F570H,1) – LOGOP – Código de operação lógica (igual aos códigos do VDP).

Saída: CY = 0 → Transferência bem-sucedida.

CY = 1 → Erro na transferência.

Registradores: Todos.

Obs.: O espaço de memória a ser alocado, em bytes, deve obedecer às seguintes fórmulas:

Screen 6:  $(NX * NY) / 4 + 4$

Screens 5 e 7:  $(NX * NY) / 2 + 4$

Screens 8, 10, 11 e 12:  $(NX * NY) + 4$

### **BLTMV** (0199H/SubROM)

Função: Transfere dados da VRAM para a Main RAM.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – SX – Coordenada X da fonte.

(F564H,2) – SY – Coordenada Y da fonte.

(F566H,2) – DPTR – Endereço destino na RAM.

(F568H,2) – DUMMY – (não requer dados).

(F56AH,2) – NX – Número de pixels na direção X.

(F56CH,2) – NY – Número de pixels na direção Y.

(F56EH,1) – CDUMMY – (não requer dados).

(F56FH,1) – ARGT – Seleciona a direção e a VRAM expandida (igual a R#45 do VDP).

Saída: CY = 0.

Registradores: Todos.

Obs.: O espaço de memória a ser alocado, em bytes, deve obedecer às seguintes fórmulas:

Screen 6:  $(NX * NY) / 4 + 4$

Screens 5 e 7:  $(NX * NY) / 2 + 4$

Screens 8, 10, 11 e 12:  $(NX * NY) + 4$

### **BLTVD** (019DH/SubROM)

Função: Transfere dados do disco para a VRAM.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – FN PTR – Endereço do nome do arquivo.

(F564H,2) – DUMMY – (não requer dados).

(F566H,2) – DX – Coordenada X de destino.

(F568H,2) – DY – Coordenada Y de destino.

(F56AH,2) – NX – Número de pixels na direção X  
(não requer dados; já preenchida).

(F56CH,2) – NY – Número de pixels na direção Y  
(não requer dados; já preenchida).

(F56EH,1) – CDUMMY – (não requer dados).

(F56FH,1) – ARGT – Seleciona a direção e a VRAM expandida (igual a R#45 do VDP).

(F570H,1) – LOGOP – Código de operação lógica (igual aos códigos do VDP).

Saída: CY = 0 → Transferência bem-sucedida.

CY = 1 → Erro na transferência ou nos parâmetros.

Registradores: Todos.

#### **BLTDV** (01A1H/SubROM)

Função: Transfere dados da VRAM para o disco.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – SX – Coordenada X da fonte.

(F564H,2) – SY – Coordenada Y da fonte.

(F566H,2) – FN PTR – Endereço do nome do arquivo.

(F568H,2) – DUMMY – (não requer dados).

(F56AH,2) – NX – Número de pixels na direção X.

(F56CH,2) – NY – Número de pixels na direção Y.

(F56EH,1) – CDUMMY – Dummy (não requer dados).

Saída: CY = 0.

Registradores: Todos.

#### **BLTMD** (01A5H/SubROM)

Função: Transfere dados do disco para a Main RAM.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – FN PTR – Endereço do nome do arquivo.

(F564H,2) – DUMMY – (não requer dados).

(F566H,2) – S PTR – Endereço inicial dos dados.

(F568H,2) – E PTR – Endereço final dos dados.

Saída: CY = 0

Registradores: Todos.

#### **BLTDM** (01A9H/SubROM)

Função: Transfere dados da Main RAM para o disco.

Entrada: HL – Deve conter o valor F562H.

(F562H,2) – S PTR – Endereço inicial dos dados.

(F564H,2) – E PTR – Endereço final dos dados.

(F566H,2) – FN PTR – Endereço do nome do arquivo.

Saída: CY = 0

Registradores: Todos.

## 9.3 – ROTINAS DO MATH-PACK

### 9.3.1 – Funções matemáticas em ponto flutuante

<b>DECSUB</b>	268CH	DAC $\leftarrow$ DAC - ARG	(precisão dupla)
<b>DECADD</b>	269AH	DAC $\leftarrow$ DAC + ARG	
<b>DECMUL</b>	27E6H	DAC $\leftarrow$ DAC * ARG	
<b>DECDIV</b>	289FH	DAC $\leftarrow$ DAC / ARG	
<b>COS</b>	2993H	DAC $\leftarrow$ COS (DAC)	
<b>SIN</b>	29A8H	DAC $\leftarrow$ SIN (DAC)	
<b>TAN</b>	29FBH	DAC $\leftarrow$ TAN (DAC)	
<b>ATN</b>	2A14H	DAC $\leftarrow$ ATN (DAC)	
<b>LOG</b>	2A72H	DAC $\leftarrow$ LOG (DAC)	
<b>SQR</b>	2AFFH	DAC $\leftarrow$ SQR (DAC)	
<b>EXP</b>	2B4AH	DAC $\leftarrow$ EXP (DAC)	
<b>SGNEXP</b>	37C8H	DAC $\leftarrow$ DAC $\wedge$ ARG	(precisão simples)
<b>DBLEXP</b>	37D7H	DAC $\leftarrow$ DAC $\wedge$ ARG	(precisão dupla)

### 9.3.2 – Operações com números inteiros

<b>UMULT</b>	314AH	DE $\leftarrow$ BC * DE	(multiplicação sem sinal)
<b>ISUB</b>	3167H	HL $\leftarrow$ DE - HL	
<b>IADD</b>	3172H	HL $\leftarrow$ DE + HL	
<b>IMULT</b>	3193H	HL $\leftarrow$ DE * HL	
<b>IDIV</b>	31E6H	HL $\leftarrow$ DE / HL	
<b>IMOD</b>	323AH	HL $\leftarrow$ DE mod HL DE $\leftarrow$ DE / HL	
<b>INTEXP</b>	383FH	DAC $\leftarrow$ DE $\wedge$ HL	

### 9.3.3 – Funções especiais

<b>DECNRM</b>	26FAH	Normaliza DAC, removendo zeros excessivos da mantissa. (Ex. 0.00123 $\rightarrow$ 0.123E-2).
<b>DECROU</b>	273CH	Arredonda DAC.
<b>RND</b>	2BDFH	Gera um número aleatório a partir do número contido em DAC e o retorna em DAC.
<b>SIGN</b>	2E71H	A $\leftarrow$ sinal da mantissa em DAC.

<b>ABSFN</b>	2E82H	Extrai o valor absoluto (módulo) do número contido em DAC e o retorna em DAC.
<b>NEG</b>	2E8DH	Inverte o sinal de DAC
<b>SGN</b>	2E97H	DAC ← sinal de DAC: DAC +2, +3: 0000H = Zero 0001H = Positivo FFFFH – Negativo
<b>FCOMP</b>	2F21H	Esq: CBED
<b>ICOMP</b>	2F4DH	Esq: DE
<b>XDCOMP</b>	2F5CH	Esq: ARG
		Dir: DAC (precisão simples)
		Dir: HL (número inteiro)
		Dir: DAC (precisão dupla)
		Compara “Esq” com “Dir” e armazena o resultado em A:
	A = 1	→ Esq < Dir
	A = 0	→ Esq = Dir
	A = -1	→ Esq > Dir

### 9.3.4 – Movimento

<b>MAF</b>	2C4DH	ARG ← DAC	Precisão dupla
<b>MAM</b>	2C50H	ARG ← (HL)	
<b>MOV8DH</b>	2C53H	(DE) ← (HL)	
<b>MFA</b>	2C59H	DAC ← ARG	
<b>MFM</b>	2C5CH	DAC ← (HL)	
<b>MMF</b>	2C67H	(HL) ← DAC	
<b>MOV8HD</b>	2C6AH	(HL) ← (DE)	
<b>XTF</b>	2C6FH	(SP) ↔ DAC	
<b>PHA</b>	2CC7H	ARG ← (SP)	
<b>PHF</b>	2CCCH	DAC ← (SP)	
<b>PPA</b>	2CDCH	(SP) ← ARG	Precisão simples
<b>PPF</b>	2CE1H	(SP) ← DAC	
<b>PUSHF</b>	2EB1H	DAC ← (SP)	
<b>MOVFM</b>	2EBEH	DAC ← (HL)	
<b>MOVFR</b>	2EC1H	DAC ← (CBED)	
<b>MOVRF</b>	2ECCH	(CBED) ← DAC	
<b>MOVRMI</b>	2ED6H	(CBED) ← (HL)	
<b>MOVRM</b>	2EDFH	(BCDE) ← (HL)	
<b>MOVMF</b>	2EE8H	(HL) ← DAC	
<b>MOVE</b>	2EEBH	(HL) ← (DE)	

<b>VMOVAM</b>	2EEFH	ARG ← (HL)	VALTYP
<b>MOVVFM</b>	2EF2H	(DE) ← (HL)	
<b>VMOVE</b>	2EF3H	(HL) ← (DE)	
<b>VMOVFA</b>	2F05H	DAC ← ARG	
<b>VMOVFM</b>	2F08H	DAC ← (HL)	
<b>VMOVAF</b>	2F0DH	ARG ← DAC	
<b>VMOVMF</b>	2F10H	(HL) ← DAC	

### 9.3.5 – Conversões

<b>FRCINT</b>	2F8AH	Converte DAC em inteiro de 2 bytes (DAC+2,+3)
<b>FRCSNG</b>	2FB2H	Converte DAC em real de precisão simples.
<b>FRCDBL</b>	303AH	Converte DAC em real de precisão dupla.
<b>FIXER</b>	30BEH	DAC ← SGN(DAC) * INT(ABS(DAC)).
<b>FIN</b>	3299H	Converte uma string de um número real para o formato BCD e o armazena em DAC.  Entrada: HL – Endereço inicial da string. A – Primeiro caractere da string.  Saída: DAC – Número real. C = 0 → C/ ponto decimal; FFH → Sem ponto B – Número de dígitos após o ponto decimal. D – Número total de dígitos.
<b>FOUT</b>	3425H	Converte um número real contido em DAC para uma string sem formatar.  Entrada: A – Sempre 0 B – Nº dígitos antes do ponto, sem incluir este. C – Nº dígitos depois do ponto, incluindo este.  Saída: HL – Endereço inicial da string.
<b>PUFOUT</b>	3426H	Converte um número real contido em DAC para uma string, formatando.  Entrada: A – Formato: bit 7 – 0: s/ formato 1: formatado bit 6 – 0: s/ vírgulas 1: vírgulas c/ 3 dígitos bit 5 – 0: n/c 1: preenche espaços c/ "*" bit 4 – 0: n/c 1: adiciona "\$" antes do nº bit 3 – 0: n/c 1: adiciona "+" para nºs pos. bit 2 – 0: n/c 1: coloca sinal após número bit 1 – Não usado. bit 0 – 0: ponto fixo 1: ponto flutuante

		B – Número de dígitos antes do ponto decimal sem incluir este.	
		C – Número de dígitos depois do ponto decimal, incluindo este.	
	Saída:	HL – Endereço inicial da string.	
<b>FOUTB</b>	(371AH)	Converte um inteiro de dois bytes contido em DAC+2,+3 em uma expressão binária.	
	Entrada:	DAC+2, +3 – Número inteiro. VALTYP = 2.	
	Saída:	HL – Endereço inicial da string binária.	
<b>FOUTO</b>	(371EH)	Converte um inteiro de dois bytes contido em DAC+2,+3 em uma expressão octal.	
	Entrada:	DAC+2, +3 – Número inteiro. VALTYP = 2.	
	Saída:	HL – Endereço inicial da string octal.	
<b>FOUTH</b>	(3722H)	Converte um inteiro de dois bytes contido em DAC+2,+3 em uma expressão hexadecimal.	
	Entrada:	DAC+2, +3 – Número inteiro. VALTYP = 2.	
	Saída:	HL – Endereço inicial da string hexadecimal.	

## 9.4 – ROTINAS DO INTERPRETADOR BASIC

### 9.4.1 – Rotinas de execução

**READYR** (409BH/Main)

Função:	Retorna ao nível de comandos (partida a quente do BASIC).
Entrada:	Nenhuma
Saída:	Nenhuma

**CRUNCH** (42B2H/Main)

Função:	Converte um texto BASIC da forma ASCII para a forma tokenizada.
Entrada:	HL – Endereço do texto em ASCII a ser convertido, finalizado por um byte 00H.
Saída:	KBUF (F41FH) – Texto BASIC convertido

**NEWSTT** (4601H/Main)

Função: Executa um texto BASIC. O texto deverá estar na forma tokenizada.

Entrada: HL – Apontador para o início do texto a ser executado. O texto deverá estar na forma ilustrada abaixo:

3AH 94H 00H ...

:	NEW		...
---	-----	--	-----



(HL)

Saída: Nenhuma

**CHRGTR** (4666H/Main) – De 0010H

Função: Extrai um caractere do texto BASIC, iniciando por (HL)+1. Espaços são ignorados.

Entrada: HL – Endereço inicial do texto.

Saída: HL – Endereço do caractere extraído.

A – Código ASCII do caractere extraído.

Z = 1 se for fim de linha (00H ou 3AH “:”).

CY = 1 se for um caractere de 0 a 9.

**FRMEVL** (4C64H/Main)

Função: Avalia uma expressão e devolve o resultado.

Entrada: HL – Endereço inicial da expressão no texto BASIC.

Saída: HL – Endereço final da expressão +1.

VALTYP (F663H) = 2 – Variável inteira.

4 – Variável de precisão simples.

8 – Variável de precisão dupla.

3 – Variável string.

DAC (F7F6H) – Resultado da expressão avaliada.

**GETBYT** (521CH/Main)

Função: Avalia uma expressão e retornar um resultado de 1 byte. Quando o resultado extrapolar o valor de 1 byte será gerado erro de “Função Ilegal” e a execução retornará ao nível de comandos.

Entrada: HL – Endereço inicial da expressão a ser avaliada.

Saída: HL – Endereço final da expressão +1.

A,E – Resultado da avaliação (A e E contêm o mesmo valor).

**FRMQNT** (542FH/Main)

Função: Avalia uma expressão e retornar um resultado de 2 bytes (número inteiro). Quando o resultado extrapolar o valor de 2 bytes, será gerado um erro de “Overflow” e a execução retornará ao nível de comandos.

Entrada: HL – Endereço inicial da expressão a ser avaliada

Saída: HL – Endereço final da expressão +1.

DE – Resultado da avaliação

**SYNCHR** (558CH/Main) – De 0008H

Função: Testa se o caractere apontado por (HL) é o especificado. Se não for, gera “Syntax error”; caso contrário chama CHRGR (4666H/Main).

Entrada: HL – Aponta para o caractere a ser testado

O caractere para comparação deve ser colocado após uma intrução “RST 0008H” na forma de parâmetro em linha, conforme exemplo abaixo:

```
LD    HL, CARACT
RST  008H
DEFB 'A'
|
CARACT: DEFB 'B'
```

Saída: HL é incrementado em um e A recebe (HL). Quando o caractere testado for numérico, a flag CY é setada. O fim de declaração (00H ou 3AH “:”) seta a flag Z.

**GETYPR** (5597H/Main) – De 0028H

Função: Obtém o tipo de operando contido em DAC.

Entrada: Nenhuma

Saída: Flags CY, S, Z e P/V, conforme tabela abaixo:

Inteiro:	C=1	S=1*	Z=0	P/V=1
----------	-----	------	-----	-------

Precisão simples:	C=1	S=0	Z=0	P/V=0*
-------------------	-----	-----	-----	--------

Precisão dupla:	C=0*	S=0	Z=0	P/V=1
-----------------	------	-----	-----	-------

String:	C=1	S=0	Z=1*	P/V=1
---------	-----	-----	------	-------

Obs.: Os tipos podem ser reconhecidos unicamente pelas flags marcadas com “\*”.

**PTRGET** (5EA4H/Main)

**Função:** Obtém o endereço para o armazenamento de uma variável ou matriz. O endereço também é obtido quando a variável não foi atribuída. Quando o valor de SUBFLG (F5A5H) for diferente de 0, o endereço inicial de uma matriz será obtido; caso contrário, será obtido o endereço do elemento da matriz.

**Entrada:** HL – Endereço inicial do nome da variável no texto BASIC.  
SUBFLG (F6A5H) – 0 → Variável simples.  
Outro valor → Matriz.

**Saída:** HL – Endereço após o nome da variável.  
DE – Endereço de onde o conteúdo da variável está armazenado.

**FRESTR** (67D0H/Main)

**Função:** Registra o resultado de uma string obtida por FRMEVL (4C64H) e obtém o respectivo descritor. Quando avaliando uma string, esta rotina é, geralmente, combinada com FRMEVL da forma descrita abaixo:

```
CALL FRMEVL
PUSH HL
CALL FRESTR
EX DE, HL
POP HL
LD A, (DE)
...
```

**Entrada:** VALTYP (F663H) – Tipo de variável (deve ser 3)  
DAC (F7F6H) – Apontador para o descritor da string

**Saída:** HL – apontador para o descritor da string

#### 9.4.2 – Rotinas dos comandos e funções

Comando/ função	Token	Token de função	Endereço na tabela da rotina	Endereço da rotina
>	EEH	-	Afat	-
=	EFH	-	Afat	-
<	F0H	-	Afat	-
+	F1H	-	Afat	-
-	F2H	-	Afat	-

*	F3H	-	Afat	-
/	F4H	-	Afat	-
^	F5H	-	Afat	-
\$	FCH	-	Afat	-
ABS	06H	FF86H	39E8H	2E82H
AND	F6H	-	Afat	-
ASC	15H	FF95H	3A06H	680BH
ATN	0EH	FF8EH	39F8H	2A14H
ATTR\$	E9H	-	Afat	7C43H
AUTO	A9H	-	3973H	49B5H
BASE	C9H	-	39BEH	7B5AH
BEEP	C0H	-	39ACH	00C0H
BIN\$	1DH	FF9DH	3A16H	6FFFH
BLOAD	CFH	-	39CAH	6EC6H
BSAVE	D0H	-	39CCH	6E92H
CALL	CAH	-	39C0H	55A8H
CDBL	20H	FFA0H	3A1CH	303AH
CHR\$	16H	FF96H	3A08H	681BH
CINT	1EH	FF9EH	3A18H	2F8AH
CIRCLE	BCH	-	39A4H	5B11H
CLEAR	92H	-	3950H	64AFH
CLOAD	9BH	-	3962H	703FH
CLOSE	B4H	-	3994H	6C14H
CLS	9FH	-	396AH	00C3H
CMD	D7H	-	39DAH	7C34H
COLOR	BDH	-	39A6H	7980H
CONT	99H	-	395EH	6424H
COPY	D6H	-	39D8H	7C2FH
COS	0CH	FF8CH	39F4H	2993H
CSAVE	9AH	-	3960H	6FB7H
CSNG	1FH	FF9FH	3A1AH	2FB2H
CSRLIN	E8H	-	Afat	790AH
CVD	2AH	FFAAH	3A30H	7C70H
CVI	28H	FFA8H	3A2CH	7C66H
CVS	29H	FFA9H	3A2EH	7C6BH
DATA	84H	-	3934H	485BH
DEF	97H	-	395AH	501DH
DEFDBL	AEH	-	3988H	4721H
DEFINT	ACH	-	3984H	471BH
DEFSNG	ADH	-	3986H	471EH
DEFSTR	ABH	-	3982H	4718H
DELETE	A8H	-	397CH	53E2H

DIM	86H	-	3938H	5E9FH
DRAW	BEH	-	39A8H	5D6EH
DSKF	26H	FFA6H	3A28H	7C39H
DSKI\$	EAH	-	Afat	7C3EH
DSKO\$	D1H		39CEH	7C16H
ELSE	A1H	3AA1H	396EH	485DH
END	81H	-	396EH	63EAH
EOF	2BH	FFABH	3A32H	6D25H
EQV	F9H	-	Afat	-
ERASE	A5H	-	3976H	6477H
ERL	E1H	-	Afat	4E0BH
ERR	E2H	-	Afat	4DFDH
ERROR	A6H	-	3978H	49AAH
EXP	0BH	FF8BH	39F2H	2B4AH
FIELD	B1H	-	398EH	7C52H
FILES	B7H	-	39AAH	6C2FH
FIX	21H	FFA1H	3A1EH	30BEH
FN	DEH	-	Afat	5040H
FOR	82H	-	3920H	4524H
FPOS	27H	FFA7H	3A2AH	6D39H
FRE	0FH	FF8FH	39FAH	69F2H
GET	B2H	-	3990H	775BH
GOSUB	8DH	-	3948H	47B2H
GOTO	89H	-	393EH	47E8H
GO TO	89H	-	393EH	47E8H
HEX\$	1BH	FF9BH	3A12H	65FAH
IF	8BH	-	3942H	49E5H
IMP	FAH	-	3A20H	7940H
INKEY\$	ECH	-	Afat	7347H
INP	10H	FF90H	39FCH	4001H
INPUT	85H	-	3936H	4B6CH
INSTR	E5H	-	39F6H	29FBH
INT	05H	FF85H	39E6H	30CFH
IPL	D5H	-	39D6H	7C2AH
KEY	CCH	-	3964H	786CH
KILL	D4H	-	39D4H	7C25H
LEFT\$	01H	FF81H	39DEH	6861H
LEN	12H	FF92H	3A00H	67FFH
LET	88H	-	393CH	4880H
LFILES	BBH	-	39A2H	6C2AH
LINE	AFH	-	398AH	4B0EH
LIST	93H	-	3952H	522EH

LLIST	9EH	-	3968H	5229H
LOAD	B5H	-	3996H	6B5DH
LOC	2CH	FFACH	3A34H	6D03H
LOCATE	D8H	-	39DCH	7766H
LOF	2DH	FFADH	3A36H	6D14H
LOG	0AH	FF8AH	39F0H	2A72H
LPOS	1CH	FF9CH	3A14H	4FC7H
LPRINT	9DH	-	394CH	4A1DH
LSET	B8H	-	399CH	7C48H
MAX	CDH	-	39C6H	7E4BH
MERGE	B6H	-	3998H	6B5EH
MID\$	03H	FF83H	39E2H	689AH
MKD\$	30H	FFB0H	3A3CH	7C61H
MKI\$	2EH	FFAEH	3A38H	7C57H
MKS\$	2FH	FFAFH	3A3AH	7C5CH
MOD	FBH	-	Afat	-
MOTOR	CEH	-	39C8H	73B7H
NAME	D3H	-	39D2H	7C20H
NEW	94H	-	3954H	6286H
NEXT	83H	-	3932H	6527H
NOT	E0H	-	Afat	-
OCT\$	1AH	FF9AH	3A10H	7C70H
OFF	EBH	-	3A02H	3A02H
ON	95H	-	3956H	48E4H
OPEN	B0H	-	398CH	6AB7H
OR	F7H	-	Afat	-
OUT	9CH	-	3964H	4016H
PAD	25H	FFA5H	3A26H	7969H
PAINT	BFH	-	39AAH	59C5H
PDL	24H	FFA4H	3A24H	795AH
PEEK	17H	FF97H	3A0AH	541CH
PLAY	C1H	-	39AEH	73E5H
POINT	EDH	-	Afat	5803H
POKE	98H	-	395CH	5423H
POS	11H	FF91H	39FEH	4FCCH
PRESET	C3H	-	39B2H	57E5H
PRINT	91H	-	394EH	4A24H
PSET	C2H	-	39B0H	57EAH
PUT	B3H	-	3992H	7758H
READ	87H	-	393AH	4B9FH
REM	8FH	3A8FH	394AH	485DH
RENUM	AAH	-	3980H	5468H

RESTORE	8CH	-	3944H	63C9H
RESUME	A7H	-	397AH	495DH
RETURN	8EH	-	3948H	4821H
RIGHT\$	02H	FF82H	39E0H	6891H
RND	08H	FF88H	39ECH	2BDFH
RSET	B9H	-	399EH	7C4DH
RUN	8AH	-	3940H	479EH
SAVE	BAH	-	39A0H	6BA3H
SCREEN	C5H	-	39B6H	79CCH
SET	D2H	-	39D0H	7C1BH
SGN	04H	FF84H	39E4H	2E97H
SIN	09H	FF89H	39EEH	29ACH
SOUND	C4H	-	39B4H	73CAH
SPACE\$	19H	FF99H	3A0EH	6848H
SPC(	DFH	-	Afat	-
SPRITE	C7H	-	39BAH	7A48H
SQR	07H	FF87H	39EAH	2AFFH
STEP	DCH	-	Afat	-
STICK	22H	FFA2H	3A20H	7940H
STOP	90H	-	394CH	63E3H
STR\$	13H	FF93H	3A02H	6604H
STRIG	23H	FFA3H	3A22H	794CH
STRING\$	E3H	-	Afat	6829H
SWAP	A4H	-	3974H	643EH
TAB(	DBH	-	Afat	-
TAN	0DH	FF8DH	39F6H	29FBH
THEN	DAH	-	Afat	-
TIME	CBH	-	39C2H	7911H
TO	D9H	-	Afat	-
TROFF	A3H	-	3972H	6439H
TRON	A2H	-	3970H	6438H
USING	E4H	-	Afat	-
USR	DDH	-	Afat	4FD5H
VAL	14H	FF94H	3A04H	68BBH
VARPTR	E7H	-	39FAH	4E41H
VDP	C8H	-	39BCH	7B37H
VPEEK	18H	FF98H	3A0CH	7BF5H
VPOKE	C6H	-	39B8H	7BE2H
WAIT	96H	-	3958H	401CH
WIDTH	A0H	-	396CH	51C9H
XOR	F8H	-	Afat	-

## 9.5 – ROTINAS DA BIOS ESTENDIDA

### 9.5.1 – Entrada da BIOS estendida

#### **EXTBIO** (FFCAH/Work Area)

**Função:** Acessa funções estendidas da BIOS. Disponível apenas se o bit 0 da flag HOKVLD (FB20H) estiver setado em 1.

**Entrada:** A – Sempre 00H

D – ID do dispositivo:

00 – Comandos internos (broadcast commands)

01~03 – Livre

04 – Manipulação da Memória Mapeada do DOS2

05~07 – Livre

08 – RS232C / MSX Modem

09 – Livre

10 – MSX-Audio

11 – MSX MIDI

12~15 – Livre

16 – MSX-JE

17 – Kanji Driver

18~33 – Livre

34 – UNAPI

35~51 – Livre

52 – MWMPPLAY (MoonBlaster 4 Wave replayer)

53~76 – Livre

77 – Memman

78 – Nowind

79~204 – Livre

205 – MCDRV (Micro Cabin BGM replayer)

206~239 – Livre

240 – MGSDRV (SCC music-player)

241~254 – Livre

255 – Exclusivo do sistema

E – número da função (0 a 255).

**Saída:** Depende do dispositivo e função chamadas.

CY = 1 se o dispositivo especificado não for encontrado.

**Registradores:** Todos.

### 9.5.2 – Comandos internos (broadcast commands)

#### **EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 00H – Comando interno.

E = 00H – Examina os dispositivos presentes no sistema, pede que este grave seu próprio número na tabela, incrementa o ponteiro e passa p/ o próximo dispositivo.

B – ID do slot onde será colocada a tabela.

HL – Endereço da tabela.

Saída: B – ID do slot da tabela.

HL – Endereço do próximo byte depois da tabela.

CY = 1 se não houver dispositivos.

Registradores: Todos.

#### **EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 00H – Comando interno.

E = 01H – Obtém o número de eventos de interrupção do MSX BASIC. Os eventos são:

0 a 9 ON KEY GOSUB

10 ON STOP GOSUB

11 ON SPRITE GOSUB

12 a 16 ON STRIG GOSUB

17 ON INTERVAL GOSUB

18 a 23 Para dispositivos de expansão

24 e 25 Reservados (uso proibido)

Saída: A – Número de eventos ativos.

Registradores: Todos.

#### **EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 00H – Comando interno.

E = 02H – Declara proibição de interrupção (desabilita as interrupções pelo tempo padrão de 1 mS).

Saída: Nenhuma.

Registradores: Todos.

**EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 00H – Comando interno.

E = 03H – Declara permissão de interrupção (habilita as interrupções bloqueadas pela função 02H).

Saída: Nenhuma.

Registradores: Todos.

**9.5.3 – Memória Mapeada****EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 04H – Dispositivo de manipulação de Memória Mapeada do MSXDOS2.

E = 01H – Retorna o endereço da tabela de variáveis da Memória Mapeada.

Saída: A – ID do slot da mapper primária.

DE – Reservado.

HL – Endereço inicial da tabela de variáveis, cuja estrutura é a seguinte:

+00H ID do slot da mapper primária.

+01H Número total de segmentos de 16K.

+02H Número de segmentos de 16K livres.

+03H Número de segmentos de 16k alocados pelo sistema (mínimo de 6 para a mapper primária).

+04H Nº de segmentos de 16K alocados para o usuário.

+05H~+07H Reservados (Sempre 00H).

+08H... Entradas para outras mappers em outros slots. Se não houver nenhuma, conterá 00H.

Registradores: Todos.

**EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 04H – Dispositivo de manipulação de Memória Mapeada.

E = 02H – Retorna diversos parâmetros relativos à Memória Mapeada.

Saída: A – Número total de segmentos (páginas lógicas) da mapper primária.

B – ID do slot da mapper primária.

C – Número de segmentos (páginas lógicas) livres na mapper primária.

DE – Reservado.

HL – Endereço inicial de uma tabela de chamada de subrotinas de suporte à mapper. O formato desta tabela é o seguinte:

+00H	ALL_SEG	Aloca um segmento de 16K.
+03H	FRE_SEG	Libera um segmento de 16K.
+06H	RD SEG	Lê um byte do endereço A:HL p/ A.
+09H	WR SEG	Escreve o conteúdo de E em (A:HL).
+0CH	CAL-SEG	Chamada inter-segmento por lyh:IX.
+0FH	CALLS	Chamada inter-segmento. Parâmetros em linha após a instrução CALL.
+12H	PUT_PH	Coloca um segmento na página física (HL)
+15H	GET_PH	Retorna o segmento atual para a página física (HL)
+18H	PUT_P0	Coloca um segmento na página física 0
+1BH	GET_P0	Retorna o segmento atual da página física 0.
+1EH	PUT_P1	Coloca um segmento na página física 1
+21H	GET_P1	Retorna o segmento atual da página física 1.
+24H	PUT_P2	Coloca um segmento na página física 2
+27H	GET_P2	Retorna o segmento atual da página física 2.
+2AH	PUT_P3	Não suportada (página 3 não pode ser trocada). Se chamada, apenas retorna.
+2DH	GET_P3	Retorna o segmento atual da página física 3.

Registradores: Todos.

### 9.5.3.1 – Rotinas de manipulação da Memória Mapeada

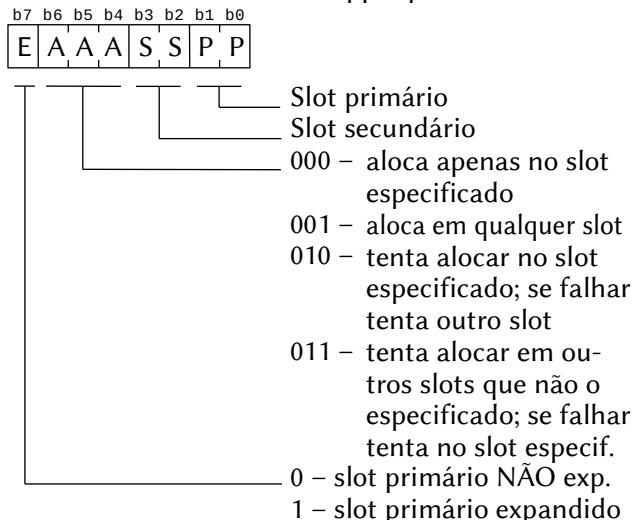
**ALL\_SEG** (HL+00H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Alocar um segmento de 16K da mapper.

Entrada: A = 00H – Aloca um segmento de usuário

01H – Aloca um segmento de sistema

B = 00H – Aloca somente na mapper primária:



Saída CY = 1 – Não há segmentos livres

CY = 0 – Segmento alocado

A – Número do segmento

B = ID do slot do segmento

Obs.: Um segmento de sistema só será liberado com o uso da rotina FRE\_SEG. Um segmento de usuário sempre será liberado quando que o programa que o utiliza for fechado.

**FRE\_SEG** (HL+03H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Liberar um segmento de 16K da mapper.

Entrada: A – número do segmento a ser liberado

B – Se for 00H, libera somente na mapper primária; se for diferente de 00H libera em qualquer outra mapper que não a primária.

Saída: CY = 0 – segmento liberado

CY = 1 – erro na liberação do segmento

**RD\_SEG** (HL+06H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Ler um byte da mapper

Entrada: A – Número do segmento de onde o byte será lido.  
HL – Endereço a ser lido (0000H to 3FFFH).

Saída: A – Byte lido.

Todos os outros registradores são preservados.

**WR\_SEG** (HL+09H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Escrever um byte na mapper.

Entrada: A – Número do segmento onde o byte será escrito.  
HL – Endereço a ser escrito (0000H to 3FFFH).

E – Valor a escrever.

Saída: A – Corrompido durante a escrita.

Todos os outros registradores são preservados.

**CAL\_SEG** (HL+0CH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina em qualquer área da mapper.

Entrada: IYh – Número do segmento a ser chamado.  
IX – Endereço a ser chamado (0000H to FFFFH).

AF, BC, DE e HL podem conter parâmetros para a rotina.  
Não usar AF', BC', DE' e HL' pois são corrompidos durante a chamada.

Saída: AF, BC, DE, HL, IX e IY podem conter valores de retorno válidos. AF', BC', DE' e HL' retornam corrompidos.

**CALLS** (HL+0FH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina em qualquer área da mapper através de parâmetros em linha.

Entrada: AF, BC, DE e HL podem conter parâmetros para a rotina.  
Não usar AF', BC', DE' e HL' pois são corrompidos durante a chamada. A sequência de chamada deve estar no seguinte formato:

CALL CALLS

DEFB SEGMENTO

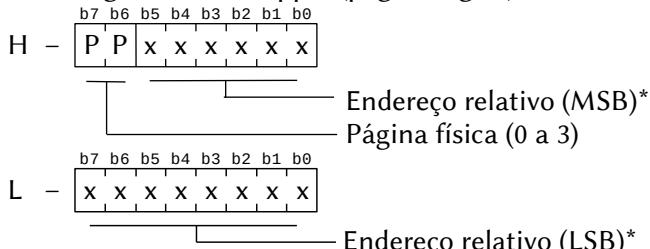
DEFW ENDEREÇO

Saída: AF, BC, DE, HL, IX e IY podem conter valores de retorno válidos. AF', BC', DE' e HL' retornam corrompidos.

**PUT\_PH** (HL+12H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita um segmento da mapper em uma página física.

Entrada: A – Segmento da mapper (página lógica)

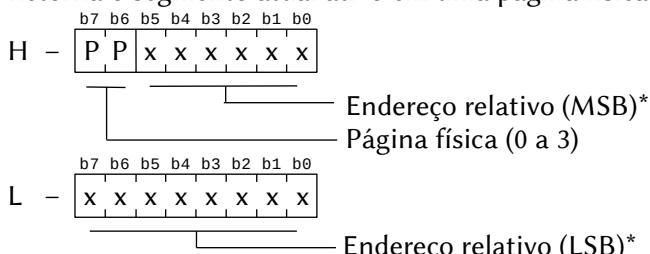


\* O endereço relativo é opcional.

Saída: Nenhuma. Todos os registradores são preservados.

**GET\_PH** (HL+15H/ExtBIOS) – Valor de HL obtido via EXTBIOS

Função: Retorna o segmento atual ativo em uma página física.



\* O endereço relativo é opcional.

Saída: A – Número do segmento.

Todos os outros registradores são preservados.

**PUT\_P0** (HL+18H/ExtBIOS) – Valor de HL obtido via EXTBIOS

Função: Habilita um segmento da mapper na página física 0.

Entrada: A – Número do segmento a ser habilitado.

Saída: Nenhuma. Todos os registradores são preservados.

**GET\_P0** (HL+1BH/ExtBIOS) – Valor de HL obtido via EXTBIOS

Função: Retorna o segmento ativo na página física 0.

Entrada: Nenhuma.

Saída: A – Número do segmento ativo.

Todos os outros registradores são preservados.

**PUT\_P1** (HL+1EH/ExtBIOS) – Valor de HL obtido via EXTBIOS

Função: Habilita um segmento da mapper na página física 1.

Entrada: A – Número do segmento a ser habilitado.

Saída: Nenhuma. Todos os registradores são preservados.

**GET\_P1** (HL+21H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o segmento ativo na página física 1.

Entrada: Nenhuma.

Saída: A – Número do segmento ativo.

Todos os outros registradores são preservados.

**PUT\_P2** (HL+24H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita um segmento da mapper na página física 2.

Entrada: A – Número do segmento a ser habilitado.

Saída: Nenhuma. Todos os registradores são preservados.

**GET\_P2** (HL+27H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o segmento ativo na página física 2.

Entrada: Nenhuma.

Saída: A – Número do segmento ativo.

Todos os outros registradores são preservados.

**PUT\_P3** (HL+2AH/ExtBIOS) – Valor de HL obtido via EXTBIO

Não suportada uma vez que a página física 3 não pode ser trocada.

Uma chamada a esta função tem efeito nulo.

**GET\_P3** (HL+2DH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o segmento ativo na página física 0.

Entrada: Nenhuma.

Saída: A – Número do segmento ativo.

Todos os outros registradores são preservados.

**CALL\_MAP** (HL+30H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina em qualquer área da RAM mapeada.

Entrada: Iyh – Número do slot.

Iyl – Número do segmento.

IX – Endereço da rotina, que deve necessariamente estar na página 1 (4000H a 7FFFH).

AF, BC, DE, HL – Parâmetros para a rotina. (Não usar AF', BC', DE' e HL' pois são corrompidos na chamada).

Saída: AF, BC, DE, HL, IX, IY – podem conter valores de retorno válidos. AF', BC', DE' e HL' retornam corrompidos.

Obs.: Rotina exclusiva para o Nextor.

**RD\_MAP** (HL+33H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Lê um byte de um segmento de RAM.

Entrada: A – Número do slot.

B – Número do segmento.

HL – Endereço a ser lido (os dois bits mais altos serão ignorados).

Saída: A – Byte de dados lido.

F, BC, DE, HL, IX, IY retornam preservados.

Obs.: Rotina exclusiva para o Nextor.

**CALL\_MAPI** (HL+36H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina em um segmento de RAM mapeada, com parâmetros em linha.

Entrada: AF, BC, DE, HL – Parâmetros para a rotina chamada. Não usar AF', BC', DE' e HL' pois são corrompidos durante a chamada. A sequência de chamada deve estar no seguinte formato:

```
CALL CALL_MAPI
DEFB SLOT
DEFB ENDEREÇO
DEFB N°_SEGMENTO
; Não é necessário o uso de RET
```

Onde:

SLOT – é o slot a ser chamado, de 0 a 3

ENDEREÇO – Endereço a ser chamado na forma de índice de uma tabela, que pode variar de 0 a 63, onde  
0=4000H, 1=4003H, 2=4006H, etc.

N°\_SEGMENTO – Pode variar de 0 a 255.

Saída: AF, BC, DE, HL, IX, IY – Parâmetros retornados pela rotina.

Obs.: Rotina exclusiva para o Nextor.

**WR\_MAP** (HL+39H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Escreve um byte em um segmento de RAM mapeada.

Entrada: A – Número do slot.

B – Número do segmento.

E – Byte a escrever.

HL – Endereço a ser escrito (os dois bits mais altos são ignorados).

Saída: A – Dado lido do endereço especificados.

F, BC, DE, HL, IX, IY retornam preservados.

Obs.: Rotina exclusiva para o Nextor.

### 9.5.4 – Porta serial RS232C e MSX Modem

#### **EXTBIO (FFCAH/Work Area)**

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 08H – Dispositivo de manipulação da RS232C.

E = 00H – Retorna o endereço da tabela de endereços de entrada das rotinas da RS232C.

B – ID do slot da tabela de endereços.

HL – Endereço da tabela.

Saída: CY = 1 → não há interfaces RS232C.

CY = 0 → HL é incrementado de 4 a cada interface encontrada e apontará para o final de uma tabela que reserva 4 bytes para cada RS232C encontrada. O valor original de HL aponta para o início da tabela, que tem a seguinte estrutura:

+00H – ID do slot

+01H – Endereço mais baixo

+02H – Endereço mais alto

+03H – Reservado para expansão

O ID de slot (+00H) e o endereço (+01H,+02H) apontarão para uma tabela com a seguinte estrutura:

+00H DB DVINF B (opcional)

+01H DB DVTYP E (opcional)

+02H DB 0

+03H JP INIT Inicializa a RS232

+06H JP OPEN Abre uma porta RS232

+09H JP STAT Retorna vários estados

+0CH JP GETCHR Lê um caractere

+0FH JP SNDCHR Envia um caractere

+12H JP CLOSE Fecha uma porta RS232

+15H JP EOF Verifica fim de arquivo

+18H JP LOC Retorna o núm. carac.

+1BH JP LOF Retorna espaço livre

+1EH JP BACKUP Salva um caractere

+21H JP SNDBRK Envia caracteres break

+24H JP DTR Liga/desliga linha DTR

+27H JP SETCHN Seleciona o canal RS232

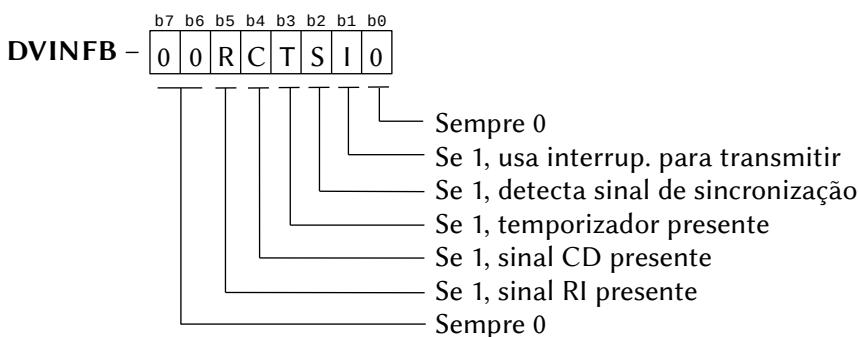
+2AH JP NCUSTA (MSX Modem)

+2DH JP SPKCNT (MSX Modem)

```
+30H JP LINSEL (MSX Modem)
+33H JP DIALST (MSX Modem)
+36H JP DIALCH (MSX Modem)
+39H JP DTMFST (MSX Modem)
+3CH JP RDDTMF (MSX Modem)
+3FH JP HOKCNT (MSX Modem)
+42H JP CONFIG (MSX Modem)
+45H JP SPCIAL (MSX Modem)
```

Registradores: Todos.

#### 9.5.4.1 – Bytes de parâmetros



**DVTYPE** – 0 → Múltiplos canais.

Outro valor → Canal único.

#### 9.5.4.2 – Rotinas de manipulação da porta serial RS232C

**INIT** (HL+03H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Inicializar a porta RS232C.

Entrada: B – ID do slot da tabela de parâmetros.

HL – Endereço da tabela de parâmetros, com a seguinte estrutura (de +00H a +07H os valores dever ser em código ASCII):

+00H – Comprimento do caractere  
("5", "6", "7" ou "8")

+01H – Paridade ("E", "O", "I" ou "N")

+02H – Stop bits ("1", "2" ou "3")

+03H – XON/XOFF ("X" ou "N")

+04H – CTR-RTS hand shake ("H" ou "N")

- +05H - Auto LF recepção ("A" ou "N")
- +06H - Auto LF transmissão ("A" ou "N")
- +07H - Controle SI/SO ("S" ou "N")
- +08H - Velocidade de recepção (low)
- +09H - Velocidade de recepção (high)  
(50 a 19200 bauds)
- +0AH - Veloc. de transmissão (low)
- +0BH - Veloc. de transmissão (high)  
(50 a 19200 bauds)
- +0CH - Contador de tempo (0 a 255)

Saída: CY = 0 → RS232C iniciada com sucesso.

$CY = 1 \rightarrow$  Erro nos parâmetros.

## Registradores: AF.

**OPEN** (HL+06H/ExtBIOS) – Valor de HL obtido via EXTBIO

Funcão: Abre uma porta serial RS232C usando o FCB.

Entrada: HL – Endereço inicial do FCB (maior que 8000H).

C – Tamanho do buffer (32 a 254).

## Modo de ab

0 – Entrada

## 2 – Saída

4 – Entrada/Saída e modo RAW

CY = 0 → Porta aberta com sucesso.

$$CY = 1 -$$

**TAT** (HL+09H/ExtBIOS) – Valor de HL obtido no bloco de configuração da placa-mãe.

**Função:** Retorna da

Entrada: Nenhuma.

## Dados retornados.

0 – Sem erro no buffer

1 - Buffer overflow.

0 – Sem erro de tempo.

I – tempo esgotado

0 – Framing correto.  
1 – Erro de framing.

bit 12 = 0 - Execução correta

0 – Execução tórica  
1 – Furo na execução

I - Erro na execução (over run)  
bit 11 0 - Não há erro de paridade

0 – Não há erro de paridade.  
1 – Erro de paridade no caractere

bit 10 – 0 – CTRL+STOP não estão pressionadas.  
                   1 – CTRL+STOP pressionadas juntas.  
 bit 09 – Reservado.  
 bit 08 – Reservado.  
 bit 07 – 0 – Estado do Clear to Send é falso.  
                   1 – Estado do Clear to Send é verdadeiro.  
 bit 06 – 0 – Timer/Contador-2 não confirmado.  
                   1 – Timer/Contador-2 confirmado.  
 bit 05 – Reservado.  
 bit 04 – Reservado.  
 bit 03 – 0 – Estado do Data Set Ready é falso.  
                   1 – Estado do Data Set Ready é verdadeiro.  
 bit 02 – 0 – Parada não detectada.  
                   1 – Parada detectada.  
 bit 01 – 0 – Estado do indicador de toque é falso.  
                   1 – Estado indicador de toque é verdadeiro.  
 bit 00 – 0 – Portadora não detectada.  
                   1 – Portadora detectada.

**GETCHR** (HL+0CH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna um caractere do buffer de recepção.

Entrada: Nenhuma.

Saída: A – Caractere recebido.

CY = 1 → EOF (fim de arquivo).

S = 1 → Erro.

Registradores: F.

**SNDCHR** (HL+0FH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Envia um caractere para a porta serial RS232C.

Entrada: A – Caractere a ser enviado.

Saída: CY = 1 → CTRL+STOP foram pressionadas juntas.

Z = 1 → Erro.

Registradores: F.

**CLOSE** (HL+12H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Fecha a porta serial RS232C.

Entrada: Nenhuma.

Saída: CY = 1 → Erro.

Registradores: AF.

**EOF** (HL+15H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Verifica se é fim de arquivo.

Entrada: Nenhuma.

Saída:  $HL = -1$  e  $CY = 1 \rightarrow$  Próximo caractere é EOF (Fim de arquivo).

$HL = 0$  e  $CY = 0 \rightarrow$  Não é fim de arquivo.

Registradores: AF.

**LOC** (HL+18H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o número de caracteres no buffer de recepção.

Entrada: Nenhuma.

Saída: HL – Número de caracteres no buffer.

Registradores: AF.

**LOF** (HL+1BH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o espaço livre no buffer de recepção.

Entrada: Nenhuma.

Saída: HL – Espaço livre em bytes.

Registradores: AF.

**BACKUP** (HL+1EH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Salva um caractere em um buffer especial. O caractere anterior é perdido.

Entrada: C – Caractere a ser salvo.

Saída: Nenhuma.

Registradores: F.

**SNDBRK** (HL+21H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Envia o número especificado de caracteres “break”.

Entrada: DE – Número de caracteres “break” a serem enviados.

Saída:  $CY = 1 \rightarrow$  CTRL+STOP foram pressionadas juntas.

Registradores: AF, DE.

**DTR** (HL+24H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Liga/desliga a linha DTR.

Entrada: A = 0 → Desliga a linha DTR.

A ≠ 0 → Liga a linha DTR.

Saída: Nenhuma.

Registradores: F.

**SETCHN** (HL+27H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Seleciona o número do canal (somente para interfaces com canais múltiplos).

Entrada: A – Número do canal.

Saída: CY = 1 → O canal não existe na interface.

Registradores: AF, BC.

#### 9.5.4.3 – Rotinas de manipulação do MSX Modem

**INIT** (HL+03H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Inicializa o MSX Modem.

Entrada: A – Tipo de modem:

0 – BELL 103	300 bps full duplex
1 – BELL 212 A	1200 bps full duplex
2 – CCITT V 21	300 bps full duplex
3 – CCITT V 22	1200 bps full duplex
4 – CCITT V22bis	2400 bps full duplex
5 – CCITT V 23	1200 bps half duplex
6 – CCITT V27ter	4800 bps half duplex
7 – CCITT V 29	9600 bps half duplex
8 – CCITT V32	9600 bps full duplex
9 a 254 – Reservado para futuras expansões.	
255 – padrão do sistema.	

C – Modo de discagem:

0 – DTMF (dicagem por tons)

1 – Reservado para futuras expansões.

2 – pulsos (20 pps)

3 – pulsos (10 pps)

4 – Automático

5 a 254 – Reservado para futuras expansões.

255 – padrão do sistema.

B – ID do slot da tabela de parâmetros.

HL – Endereço da tabela de parâmetros, com a seguinte estrutura (de +00H a +07H os valores dever ser em código ASCII):

+00H – Comprimento do caractere  
("5", "6", "7" ou "8")

+01H – Paridade ("E", "O", "I" ou "N")

+02H – Stop bits ("1", "2" ou "3")

+03H – XON/XOFF ("X" ou "N")

+04H - CTR-RTS hand shake ("H" ou "N")  
 +05H - Auto LF recepção ("A" ou "N")  
 +06H - Auto LF transmissão ("A" ou "N")  
 +07H - Controle SI/SO ("S" ou "N")  
 +08H~0BH - Não utilizados  
 +0CH - Contador de tempo (0 a 255)

Saída: CY = 0 → MSX Modem iniciado com sucesso.

CY = 1 → Erro nos parâmetros.

Registradores: AF.

**NCUSTA** (HL+2AH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o estado da NCU.

Entrada: Nenhuma.

Saída: HL – Estado.

bit 15~9 – Sempre 0.  
 bit 8 – 0 – Sem dados DTMF.  
     1 – Recebendo dados DTMF  
 bit 7 – 0 – Telefone externo no gancho  
     1 – Telefone externo fora do gancho  
 bit 6 – 0 – Sem tom de chamada  
     1 – Tom de chamada de 400 Hz detectado  
 bit 5 – Trava a inversão de polaridade da linha  
 b4,b3 – 0,0 – Loop desligado  
     0,1 – Loop DC (LB)  
     1,0 – Loop DC (LA)  
     1,1 – Indefinido  
 b2,b1 – modo de discagem  
     0,0 – DTMF  
     0,1 – pulso (10 pps)  
     1,0 – pulso (20 pps)  
     1,1 – Automático  
 bit 0 – 0 – Sem sinal da campainha (toque)  
     1 – Sinal da campainha (toque) presente

Registradores: Todos.

**SPKCNT** (HL+2DH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Liga/desliga o alto-falante.

Entrada: A = 0 → Desliga o alto-falante.

A ≠ 0 → Liga o alto-falante.

Saída: CY = 1 se esta função não for suportada.

Registradores: F.

**LINSEL** (HL+30H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Comutar a linha.

Entrada: A – bit 7~bit 5 – Reservados (sempre 0).

b4~b3 – Libera a linha (coloca o telefone interno no “gancho”). bit4=alto-falante; bit3=microfone.

b2~b1 – Conecta o telefone integrado ao modem à linha externa (bit2 = 1, conecta o alto-falante, bit1 = 1, conecta o microfone).

Bit 0 – Alterna entre o modem e o telefone externo:  
b0 = 0 → conecta o modem interno;  
b0 = 0 → conecta telefone ligado à porta “TEL”.

Saída: CY = 1 se houver erro nos parâmetros.

Registradores: Nenhum.

**DIALST** (HL+33H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Conecta o dispositivo à linha e efetua a “discagem”.

Entrada: C – Modo de “discagem”:

0 – DTMF (discagem por tons).

1 – Reservado para futuras expansões.

2 – pulsos (20 pps).

3 – pulsos (10 pps).

4 – Automático.

5 a 254 – Reservado para futuras expansões.

255 – padrão do sistema.

B – ID do slot da tabela de parâmetros.

HL – Endereço inicial dos dados de discagem a serem enviados. Os caracteres válidos para “discagem” são: “0”~“9”, “A”~“D”, “#”, “\*”, “H”, “<”, “:” e “T”. “H” significa 1 segundo no gancho, “<” significa três, “T” seleciona discagem por tom e “:” aguarda o segundo tom de discagem. A lista deve terminar com 00H.

Saída: CY = 1 se houver erro nos parâmetros.

Registradores: Nenhum.

**DIALCH** (HL+36H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Envia um único caractere por vez para a “discagem”.

Entrada: A – Caractere a ser enviado.

C – Modo de “discagem” (igual a DIALST(HL+33H)).

Saída: CY = 1 se houver erro nos parâmetros.

Registradores: Nenhum.

**DTMFST** (HL+39H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Verifica o estado do decodificador DTMF.

Entrada: Nenhuma.

Saída: Z = 1 se código DTMF estiver em modo entrada.

CY = 1 se esta função não for suportada.

Registradores: AF.

**RDDTMF** (HL+3CH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Ler dados do decodificador DTMF.

Entrada: Nenhuma.

Saída: A – Código DTMF (em ASCII)

CY = 1 se CTRL+STOP forem pressionadas ou se esta função não for suportada.

Registradores: AF.

**HOKCNT** (HL+3FH/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Conectar ou desconectar a linha.

Entrada: A = 0 → No gancho

1 → Fora do gancho

Saída: CY = 1 se esta função não for suportada.

Registradores: Nenhum.

**CONFIG** (HL+42H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna as especificações de hardware.

Entrada: A – 0 a 255.

Saída: HL – Especificações.

Quando A = 0:

bit 15 ~ bit 09: sempre 0.

bit 8 = 1 → CCITT V 32 9600 bps full duplex.

bit 7 = 1 → CCITT V 29 9600 bps half duplex.

bit 6 = 1 → CCITT V 27ter 4800 bps half duplex.

bit 5 = 1 → CCITT V 23 1200 bps half duplex.

bit 4 = 1 → CCITT V 22a 2400 bps full duplex.

bit 3 = 1 → CCITT V 22 1200 bps full duplex.

bit 2 = 1 → CCITT V 21 300 bps full duplex.

bit 1 = 1 → BELL 212 A 1200 bps full duplex.

bit 0 = 1 → BELL 103 300 bps full duplex.

Quando A = 1:

- bit 15 ~ bit 08: sempre 0.
- bit 7 = 1 → Suporta mudança 10 pps↔20 pps por software.
- Bit 6 = 1 → DTMF – Comutação suave de impulsos.
- bit 5 = 1 → Suporta "H".
- bit 4 = 1 → Suporte para "A" a "D".
- bit 3 = 1 → Automático.
- bit 2 = 1 → Pulso (20 pps).
- bit 1 = 1 → Pulso (10 pps).
- bit 0 = 1 → DTMF.

Quando A = 2:

- bit 15 ~ bit 04: Sempre 0
- bit 7 = 1 → Suporta mudança 10 pps↔20 pps por software.
- bit 3 = 1 → Telefone viva-voz integrado.
- bit 2 = 1 → Telefone padrão integrado.
- bit 1 = 1 → Modem interno.
- bit 0 = 1 → Telefone externo.

Quando A = 3:

- bit 15 ~ bit 13: Sempre 0.
- bit 12 = 1 → Função de detecção de loop longo.
- bit 11 = 1 → Função de controle da portadora.
- bit 10 = 1 → Função de comutação de potência de transmissão.
- bit 9 = 1 → RS-232C.
- bit 8 = 1 → Cartucho padrão MSX.
- bit 7 = 1 → Detecção de gancho de telefone externo (no gancho ou fora do gancho).
- bit 6 = 1 → Função “no gancho” / “fora do gancho”.
- bit 5 = 1 → Possui alto-falante.
- bit 4 = 1 → Possui decodificador DTMF.
- bit 3 = 1 → Detecção de pulso de carregamento.
- bit 2 = 1 → Detecção de polaridade de linha.
- bit 1 = 1 → Detecção de progresso de chamadas.
- bit 0 = 1 → Detecção de sinal de toque.

Quando A for 4 a 255:

HL = 0000H.

Registradores: HL.

**SPCIAL** (HL+45H/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Implementa funções especiais p/ cada modelo de modem.

Entrada: A = 0 → Enviar função de comutação de energia do modem.

C – Valor da potência de transmissão (dBm)  
se for 255, assume valor padrão.

A = 1 → Controle da onda portadora.

C = 0 – Desliga a portadora.

1 – Liga a portadora.

H – Tempo de atraso até RS ON ( $n^* 10 \text{ mS}$ )

L – Tempo de atraso de CS ON até RETURN  
( $n^* 10 \text{ mS}$ )

A = 2 → Configuração do equalizador.

C = 0 – Não usar o equalizador.

1 – Usar o equalizador.

2 – Ajuste automático do equalizador

255 – Assume o padrão.

Saída: CY = 1 se a função selecionada não for suportada.

Registradores: Depende da função chamada.

### 9.5.5 – MSX-AUDIO

**EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 0AH – Dispositivo de manipulação do MSX-Audio.

E = 00H – Retorna o apontador para a tabela de informações do MSX-Audio.

B – ID do slot da tabela de endereços.

HL – Endereço de um buffer de 64 bytes para a tabela  
(deve estar na página 3).

Saída: B – ID do slot da tabela de informações.

HL – HL é incrementado de 4 e apontará para o final de uma tabela que reserva 4 bytes para o MSX-Audio. O valor original de HL aponta para o início da tabela, que tem a seguinte estrutura:

+00H – ID do slot

+01H – Endereço mais baixo

+02H – Endereço mais alto

+03H – Reservado para expansão

O ID de slot (+00H) e o endereço (+01H,+02H) apontarão para uma tabela com a seguinte estrutura:

+00H	VERSION	Versão do software
+03H	MBIOS	Music BIOS
+06H	AUDIO	Inicialização do MSX-Audio
+09H	SYNTHE	Chama o aplicativo SYNTHE
+0CH	PLAYF	Estado instrução PLAY
+0FH	BGM	Habilita/cancela modo BGM
+12H	MKTEMP	Definir tempo de gravação/reprodução teclado musical
+15H	PLAYMK	Toca pelo teclado musical
+18H	RECMK	Grava as Notas tocadas no teclado musical
+1BH	STOPM	Reprodução/gravação teclado /ADPCM; pára instr. PLAY
+1EH	CONTMK	Continua gravação pelo teclado musical
+21H	RECMOD	Configura modo de gravação do teclado musical
+24H	STPPLY	Pára a instrução PLAY
+27H	SETPCM	Área protegida ADPCM/PCM
+2AH	RECPCM	Gravação ADPCM/PCM
+2DH	PLAYPCM	Reprodução ADPCM/PCM
+30H	PCMFREQ	Alteração da frequência de reprodução ADPCM/PCM
+33H	MKPCM	Configura/cancela dados ADPCM p/ teclado musical
+36H	PCMVOL	Configura o volume de reprodução ADPCM/PCM
+39H	SAVEPCM	Salvar dados ADPCM/PCM
+3CH	LOADPCM	Carrega dados ADPCM/PCM
+3FH	COPYPCM	Transfere dados ADPCM/PCM
+42H	CONVP	Converte dados ADPCM > PCM
+45H	CONVA	Converte dados PCM > ADPCM
+48H	VOICE	Configura dados FM
+4BH	VOICECOPY	Movimenta dados FM

Registradores: F.

### **EXTBIO (FFCAH/Work Area)**

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 0AH - Dispositivo de manipulação do MSX-Audio.

E = 01H – Retorna quantos cartuchos MSX-Audio estão conectados ao MSX (máximo de 2).

Saída: A – 0 → Não há MSX-Audio conectado.  
           1 → Há um cartucho MSX-Audio conectado.  
           2 → Há dois cartuchos MSX-Audio conectados.  
 Registradores: BC, DE, HL.

### 9.5.5.1 – Rotinas de inicialização

**VERSION** (HL+00H) – Valor de HL obtido via EXTBIO  
 Função: Versão da BIOS. Normalmente 00H-00H-00H.

**MBIOS** (HL+03H) – Valor de HL obtido via EXTBIO  
 Função: Chamar as rotinas da MBIOS (Music BIOS).  
 Entrada: HL – Endereço da rotina da MBIOS.  
 IX e IY são usados para chamada interslot e devem ser definidos em BUF (F55EH) do seguinte modo:  
 BUF +00H/+01H ← IX  
 BUF +02H/+03H ← IY

Saída: Depende da rotina da MBIOS.  
 Registradores: Depende da rotina da MBIOS.

**AUDIO** (HL+06H) – Valor de HL obtido via EXTBIO

Função: Inicializar o MSX-Audio.  
 Entrada: Setar os seguintes valores em BUF (F55EH):  
 +01H – Chave de modo  

b7 b6 b5 b4 b3 b2 b1 b0	C	P	R
0 0 0 0 0			

- 0 – Não usa ritmo
- 1 – Usa ritmo
- 0 – Não usa ADPCM via PLAY
- 1 – Usa ADPCM via PLAY
- 0 – Sem modo CMS
- 1 – MSX-Audio em modo CMS

Sempre 0

- +02H – Número de instrumentos FM usados para configurar o MSX-Audio (0 a 9)
- +03H – Número de fontes de som FM para a primeira string (0 a 9)
- +04H – Número de fontes de som FM para a segunda string (0 a 8)

- +05H - Número de fontes de som FM para a terceira string (0 a 7)
- +06H - Número de fontes de som FM para a quarta string (0 a 6)
- +07H - Número de fontes de som FM para a quinta string (0 a 5)
- +08H - Número de fontes de som FM para a sexta string (0 a 4)
- +09H - Número de fontes de som FM para a sétima string (0 a 3)
- +0AH - Número de fontes de som FM para a oitava string (0 a 2)
- +0BH - Número de fontes de som FM para a nona string (0 a 1)

Saída: CY = 1 → Inicialização falhou.

Registradores: Todos.

#### **SYNTHE** (HL+09H) – Valor de HL obtido via EXTBIO

Função: Chama o aplicativo SYNTHE integrado.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

#### **9.5.5.2 – Rotinas do PCM/ADPCM**

##### **SETPCM** (HL+27H) – Valor de HL obtido via EXTBIO

Função: Inicializa o arquivo de audio para o PCM/ADPCM.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo de áudio (0 a 15).

+01H – Número do dispositivo (0 a 5, exceto 4).

0 ou 2 → RAM externa.

1 ou 3 → ROM externa.

4 → CPU (não pode ser usado).

5 → VRAM.

+02H – Modo (0 ou 1).

+03H/+04H – Dependem do número dispositivo:

RAM → Não é necessário definir.

ROM → +3H – Número do arq. de áudio na ROM.

+4H – Sempre 0.

VRAM → +3H – Endereço VRAM (LSB).

+4H – Endereço VRAM (MSB).

+05H/+06H – Comprimento (LSB-MSB).  
+07H/+08H – Frequência de amostragem (LSB-MSB).  
+09H – Número do canal (0 ou 1).

Saída: CY = 1 → Erro de parâmetros, não configurado.

Registradores: Todos.

#### **RECP CM (HL+2AH) – Valor de HL obtido via EXTBIO**

Função: Gravar arquivo de áudio.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo de áudio (0 a 15).  
+01H – Sincronização (0 ou 1).  
+02H/+03H – Deslocamento (LSB-MSB).  
+04H/+05H – Comprimento (LSB-MSB). FFFFH para usar valores definidos por SETPCM (HL+27H).  
+06H/+07H – Frequência de amostragem (LSB-MSB). FFFFH para usar valores definidos por SETPCM (HL+27H).  
+08H – Número do canal (0 ou 1). FFH para usar o canal definido por SETPCM (HL+27H).

Saída: CY = 1 → Erro nos parâmetros, gravação cancelada.

Registradores: Todos.

#### **PLAYPCM (HL+2DH) – Valor de HL obtido via EXTBIO**

Função: Reproduzir arquivo de áudio.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo de áudio (0 a 15).  
+01H – Flag de repetição (0 ou 1).  
+02H/+03H – Deslocamento (LSB-MSB).  
+04H/+05H – Comprimento (LSB-MSB). FFFFH para usar valores definidos por SETPCM (HL+27H).  
+06H/+07H – Frequência de amostragem (LSB-MSB). FFFFH para usar valores definidos por SETPCM (HL+27H).  
+08H – Número do canal (0 ou 1). FFH para usar o canal definido por SETPCM (HL+27H).

Saída: CY = 1 → Erro nos parâmetros, gravação cancelada.

Registradores: Todos.

**PCMREQ (HL+30H) – Valor de HL obtido via EXTBIO**

Função: Alterar a frequência de reprodução.

Entrada: BC – Frequência de amostragem do primeiro canal

DE – Frequência de amostragem do primeiro canal

A frequência pode variar entre 1800 e 49.716 Hz. Se não houver segundo canal, definir o valor de DE igual a BC.

Saída: CY = 1 → Erro nos parâmetros. A frequência não é alterada.

Registradores: Todos.

**PCMVOL (HL+36H) – Valor de HL obtido via EXTBIO**

Função: Define o volume de reprodução do PCM/ADPCM.

Entrada: BC – Volume do primeiro canal (0 a 63).

DE – Volume do primeiro canal (0 a 63)

O volume máximo é 63. O valor inicial é 63 para ADPCM e 32 para PCM. Se não houver segundo canal, definir o valor de DE igual a BC.

Saída: CY = 1 → Erro nos parâmetros. O volume não é configurado.

Registradores: Todos.

**SAVEPCM (HL+39H) – Valor de HL obtido via EXTBIO**

Função: Salva arquivo de áudio PCM/ADPCM no disco.

Entrada: A – Número do arquivo de áudio.

HL – Apontador para o nome do arquivo. Deve estar entre aspas duplas (22H) e terminar com byte 00H (Ex. “FILENAME.PCM”,00H), como no MSX-BASIC.

Saída: CY = 1 → Número do arquivo de áudio incorreto. O salvamento não será feito.

Registradores: Todos.

Obs.: Se houver algum erro durante o salvamento, o controle será devolvido ao interpretador BASIC.

**LOADPCM (HL+3CH) – Valor de HL obtido via EXTBIO**

Função: Carrega arquivo de áudio PCM/ADPCM do disco.

Entrada: A – Número do arquivo de áudio.

HL – Apontador para o nome do arquivo. Deve estar entre aspas duplas (22H) e terminar com byte 00H (Ex. “FILENAME.PCM”,00H), como no MSX-BASIC.

Saída: CY = 1 → Número do arquivo de áudio incorreto. O carregamento não será feito.

Registradores: Todos.

Obs.: Se houver algum erro durante o carregamento, o controle será devolvido ao interpretador BASIC.

#### **COPYPCM (HL+3FH) – Valor de HL obtido via EXTBIO**

Função: Transfere dados PCM/ADPCM entre arquivos de áudio.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo fonte (0 a 15).

+01H – Número do arquivo destino (0 a 15).

+02H/+03H – Deslocamento do arquivo fonte (LSB-MSB).

+04H/+05H – Comprimento (LSB-MSB).

+06H/+07H – Deslocamento arquivo destino (LSB-MSB).

+08H – Especificação de fonte (0 ou 1).

Saída: CY = 1 → Erro nos parâmetros, transferência cancelada.

Registradores: Todos.

#### **CONVP (HL+42H) – Valor de HL obtido via EXTBIO**

Função: Converte dados do formato PCM para ADPCM.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo fonte (0 a 15).

+01H – Número do arquivo destino (0 a 15).

Saída: CY = 1 → Erro nos parâmetros, conversão cancelada.

Registradores: Todos.

#### **CONVA (HL+45H) – Valor de HL obtido via EXTBIO**

Função: Converte dados do formato ADPCM para PCM.

Entrada: Definir os parâmetros em BUF (F55EH):

+00H – Número do arquivo fonte (0 a 15).

+01H – Número do arquivo destino (0 a 15).

Saída: CY = 1 → Erro nos parâmetros, conversão cancelada.

Registradores: Todos.

#### **MKTEMPO (HL+18H) – Valor de HL obtido via EXTBIO**

Função: Seta o tempo para gravação e reprodução através do teclado musical, com função metrônomo.

Entrada: DE – Tempo em semínimas por minuto (25 a 360).

Saída: CY = 1 → Erro nos parâmetros, configuração cancelada.

Registradores: Todos.

**MKPCM** (HL+33H) – Valor de HL obtido via EXTBIO

Função: Especificar o arquivo de som ADPCM para tocar com o teclado musical.

Entrada: A – Número do arquivo de áudio (0 a 15). Para cancelar, use FFH.

Saída: CY = 1 → Erro nos parâmetros, reprodução cancelada.

Registradores: Todos.

### 9.5.5.3 – Rotinas do teclado musical

**PLAYMK** (HL+15H) – Valor de HL obtido via EXTBIO

Função: Toca áudio gravado através do teclado musical.

Entrada: DE – Endereço inicial de reprodução.

BC – Endereço final de reprodução.

Saída: Nenhuma.

Registradores: Todos.

**RECMK** (HL+18H) – Valor de HL obtido via EXTBIO

Função: Grava áudio através do teclado musical.

Entrada: DE – Endereço inicial para gravação.

BC – Endereço final para gravação.

Saída: Nenhuma.

Registradores: Todos.

**CONTMK** (HL+1EH) – Valor de HL obtido via EXTBIO

Função: Continuar gravando ou tocando áudio do teclado musical que foi interrompido por STOPM.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**RECMOD** (HL+21H) – Valor de HL obtido via EXTBIO

Função: Define o modo de gravação para o teclado musical.

Entrada: A – 0 → Muting (não gravar).

1 → Gravar.

2 → Reproduzir.

3 → Gravar e reproduzir simultaneamente.

Saída: CY = 1 → Erro nos parâmetros, configuração cancelada.

Registradores: Todos.

#### 9.5.5.4 – Rotinas do sintetizador FM

**PLAYF** (HL+0CH) – Valor de HL obtido via EXTBIO

Função: Verifica o estado da instrução PLAY.

Entrada: A – Número do canal da instrução PLAY (0 – Todos os canais).

Saída: HL – 0000H → o canal especificado NÃO está tocando.

FFFFH → o canal especificado está tocando.

Quando houver especificação para todos os canais, HL retornará FFFFH se qualquer um estiver ativo.

Registradores: Todos.

**BGM** (HL+0FH) – Valor de HL obtido via EXTBIO

Função: Especifica execução em segundo plano.

Entrada: A – 0 → NÃO executa processamento em segundo plano.

1 → Executa processamento em segundo plano (padrão). As funções disponíveis para segundo plano são: reprodução pelo comando PLAY, gravação e reprodução ADPCM via microfone e gravação e reprodução pelo teclado musical.

Saída: Nenhuma.

Registradores: Todos.

**STOPM** (HL+1BH) – Valor de HL obtido via EXTBIO

Função: Parar reprodução e gravação.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**STPPLY** (HL+1BH) – Valor de HL obtido via EXTBIO

Função: Parar reprodução apenas do comando PLAY.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**VOICE** (HL+48H) – Valor de HL obtido via EXTBIO

Função: Define o instrumento para cada canal FM.

Entrada: Definir os seguintes parâmetros em BUF (F55EH):

+0 → Bloco de parâmetros da voz 1.

+4 → Bloco de parâmetros da voz 2.

:

$(n-1)*4 \rightarrow$  Bloco de parâmetros da voz n.

$n*4 \rightarrow$  Marca de fim (FFH).

Especificando instrumentos fornecidos na ROM:

+0 → Número do canal (0 a 8).

+1 → 00H.

+2 → Número do instrumento na ROM (0 a 63).

+3 → 00H.

Especificando instrumento do usuário:

+0 → Número do canal (0 a 8).

+1 → FFH.

+2/+3 → Endereço dos dados do instrumento (LSB-MSB).

Saída: CY = 1 → Erro nos parâmetros, configuração cancelada.

Registradores: Todos.

### VOICECOPY (HL+4BH) – Valor de HL obtido via EXTBIO

Função: Transfere dados de instrumentos FM.

Entrada: Definir os seguintes parâmetros em BUF (F55EH):

1. Transfere instrumentos 0~63 da ROM para instrumentos 32~63 do sistema:

+0 → 00H

+1 → Número do instrumento fonte (0~63).

+2 ~ +5 → 00H

+6 → Número do instrumento destino (32~63).

+7 ~ +9 → 00H

2. Transfere instrumentos 0~63 da ROM para a área de dados do usuário:

+0 → 00H

+1 → Número do instrumento fonte (0~63).

+2 ~ +4 → 00H

+5 → FFH

+6 ~ +7 → Endereço de destino na área de dados.

+8 ~ +9 → 00H

3. Transfere instrumentos da área de dados do usuário para instrumentos 32~63 do sistema:

+0 → FFH

+1 ~ +2 → Endereço fonte na área de dados.

+3 ~ +5 → 00H

+6 → Número do instrumento destino (32~63).

+7 ~ +9 → 00H

4. Transfere todos os instrumentos 32~63 do sistema para a área de dados do usuário:
  - +0 → 00H
  - +1 → FFH
  - +2 ~ +4 → 00H
  - +5 → FFH
  - +6 ~ +7 → Endereço de destino na área de dados.
  - +8 ~ +9 → Comprimento dos dados em bytes.
5. Transfere todos os instrumentos da área de dados do usuário para instrumentos 32~63 do sistema:
  - +0 → FFH
  - +1 ~ +2 → Endereço de destino na área de dados.
  - +3 ~ +4 → Comprimento dos dados em bytes.
  - +5 → 00H
  - +6 → FFH
  - +7 ~ +9 → 00H.

#### **9.5.5.5 – Rotinas da MBIOS (Music BIOS)**

As rotinas da Music BIOS devem ser chamadas através da entrada MBIOS da tabela de salto, setando em HL o endereço de chamada da rotina desejada da Music BIOS. O formato de MBIOS é o seguinte:

**MBIOS** (JumpTable+03H) – Valor de JumpTable obtido via EXTBIO

Função: Chamar as rotinas da MBIOS (Music BIOS).

Entrada: HL – Endereço da rotina da MBIOS.

IX e IY são usados para chamada interslot e devem ser definidos em BUF (F55EH) do seguinte modo:

BUF +00H/+01H ← IX

BUF +02H/+03H ← IY

Saída: Depende da rotina da Music BIOS.

Registradores: Depende da rotina da Music BIOS.

As tabelas de dados usadas pela Music BIOS são as seguintes:

CHDB (32 bytes)	MIDB (64 bytes)
+00 YCA00_MULTI	+00~+01 Não usado
+01 YCA00_LS	+02 YM_TIM 1
+02 YCA00_AR	+03 YM_TIM 2
+03 YCA00_RR	+04~+17 Não usados
+04 YCA00_VELS	+18 YMA_BIAS
+05 YCA00_VTL	+19~+24 Não usados
+06~+07 Não usados	+25 YMA_AUDIO
+08 YCA01_MULTI	+26~+31 Não usados
+09 YCA01_LS	+32~+33 YMA_TRANS
+10 YCA01_AR	+34 YMA_LFO
+11 YCA01_RR	+35 YMA_RAM
+12 YCA01_VELS	+36 ZMA_FLAG
+13 YCA01_VTL	+37~+38 YMA_PDB
+14~+15 Não usados	+39 ZMA_PH_FILTER
+16~+17 YCA_VTRANS	+40 ZMA_PH_TL
+18~+19 YCA_TRANS	+41~+42 ZMA_PH_AR
+20 YCA_TRIG	+43~+44 ZMA_PH_DIR
+21 YCA_VOL	+45 ZMA_PH_SL
+22 YCA_FB	+46~+47 ZMA_PH_D 2 R
+23 YCA_VEL	+48~+49 ZMA_PH_RR
+24~+25 YCA_PITCH	+50~+51 ZMA_PH_EG
+26 YCA_VOICE	+52 ZMA_PH_STAT
+27 ZCA_FLAG	+53~+63 Não usados
+28 ZC_CH	
+29 ZC_OP	
+30~+31 ZC_COUNT	

## PDB (PCM Data Block)

As rotinas da Music BIOS são as seguintes:

**SV\_RESET** (0090H/MBIOS)

Função: Inicializar a MBIOS.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

Obs.: As interrupções são desabilitadas no retorno. Antes de reabilitá-las, deve ser definido o hook MBIOS.

**SV\_DI** (0093H/MBIOS)

Função: Desativa as interrupções do usuário.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**SV\_EI** (0096H/MBIOS)

Função: Permitir as interrupções do usuário.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**SV\_ADW** (0099H/MBIOS)

Função: Escrever um byte de dados em um registrador do Y8950.

Entrada: IY – Especificação master/slave por endereço MIDB.

A – Byte de dados a ser escrito.

C – Número do registrador.

Saída: CY = 1 → houve tentativa de escrita em um dispositivo “slave” inexistente.

Registradores: Todos.

**SV\_ADW\_DI** (009CH/MBIOS)

Função: Escrever um byte de dados em um registrador do Y8950, desabilitando interrupções no retorno.

Entrada: IY – Especificação master/slave por endereço MIDB.

A – Byte de dados a ser escrito.

C – Número do registrador.

Saída: CY = 1 → houve tentativa de escrita em um dispositivo “slave” inexistente.

IFF = 0 → Interrupções desabilitadas.

Registradores: Todos.

**SV\_SETUP (00ABH/MBIOS)**

Função: Configuração inicial de várias funções.

Entrada: A – Código da função.

0 – SM\_AUDIO → configuração de tom.

1 – SC\_CHDB → inicializa área de trabalho CHDB.

2 – SM\_INST → inicializa a função de instrumento.

3 – SM\_MK → inicializa leitura do teclado musical.

Outros parâmetros dependem da função.

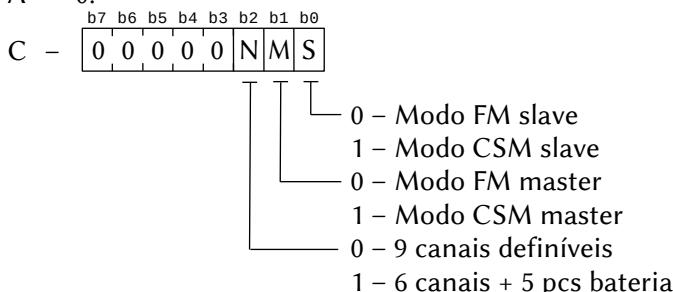
Saída: CY = 1 → Configuração falhou (geralmente pela rotina ser chamada através de interrupções).

Registradores: Todos.

**SM\_AUDIO (00ABH/MBIOS)**

Função: Definir o modo do tom musical do sintetizador FM.

Entrada: A = 0.



DE – Canal de tom do sintetizador FM.

bit0 = 1 → instrumento no canal 0

bit1 = 1 → instrumento no canal 1

:

bit8 = 1 → instrumento no canal 8

(No modo 6 canais + 5 pcs bateria, apenas os canais 0 a 5 podem ser atribuídos).

Saída: Nenhuma.

Registradores: Todos.

Obs.: Esta rotina chama internamente SC\_CHDB e SM\_INST.

**SC\_CHDB (00ABH/MBIOS)**

Função: Inicializa a área de trabalho do CHDB.

Entrada: A = 1.

IX – Endereço do CHDB a ser inicializado.

Saída: Nenhuma.  
 Registradores: Todos.

#### **SM\_INST (00ABH/MBIOS)**

Função: Inicializa o tom do instrumento com o timbre nº 0.  
 Entrada: A = 2.  
 Saída: Nenhuma.  
 Registradores: Todos.

#### **SM\_MK (00ABH/MBIOS)**

Função: Inicializa a leitura do teclado musical.  
 Entrada: A = 3.  
 B – 1 → conecta o teclado ao instrumento.  
 0 → não conecta o teclado.  
 C – Velocidade quando as teclas são pressionadas. 0 é o mais fraco e 15 o mais forte. A velocidade é referenciada em SV\_MK (music keyboard scan).

Saída: Nenhuma.  
 Registradores: Todos.

#### **SV\_REAL (00AEH/MBIOS)**

Função: Executa operações em tempo real. Esta chamada é dividida em várias funções designadas por códigos que são os seguintes:

- 00 RM\_MOVE\_DI Transf. de dados ADPCM/PCM
- 01 RM\_TRACE\_DI Rastreio de dados ADPCM
- 02 RM\_CONV\_PCM\_DI Conversão ADPCM para PCM
- 03 RM\_CONV\_ADPCM\_DI Conversão PCM em ADPCM
- 04 RMA\_DAC\_BIAS Volume para reprodução PCM
- 05 RMA\_DAC\_DI Reprodução de dados PCM
- 06 RMA\_ADC\_DI Gravação dados PCM
- 07 RMA\_ADPCM\_BIAS Configura reprodução ADPCM
- 08 RMA\_ADPLAY\_DI Reprodução de dados ADPCM
- 09 RMA\_ADREC\_DI Gravação dados ADPCM
- 10 RMA\_BREAK Interrup. reprodução/gravação
- 11 RMA\_ADPLAY Reprodução de dados ADPCM
- 12 RMA\_ADREC Gravação dados ADPCM
- 13 RMA\_PHASE\_SET\_DI Converte 256 bytes PCM
- 14 RMA\_PHASE\_EG Configura a envoltória
- 15 RMA\_PHASE\_EVENT Amostragem do pitch

```

16 RM_TIMER Ativa/desativa interrup. temp.
17 RM_TIM1 Configura o timer 1
18 RM_TIM2 Configura o timer 2
19 RM_TEMPO Define o ciclo do timer 2
20 RM_DAMP Força parada do gerador FM
21 RM_PERC Toca som de ritmo
22 RMA_MK Retorna estado teclado musical
23 RMA_LFO Configura o vibrato
24 RMA_TRANS Configura transição de sons
25 RMA_CSM_DI Reprodução de dados do CSM
26 RM_READ_DI Transf. 256 bytes ADPCM/PCM
27 RM_WRITE_DI Transf. 256 bytes ADPCM/PCM
28 RM_UTEMPR Converte o pitch temperamento
29 RM_CTEMPPR Configuração de temperamento
30 RM_PITCH Define o tom atual/subsequente
31 RM_TSPLAN Configura transposição do som
32 RC_NOTE Liga/desliga voz FM
33 RC_LEGATO Liga/desliga legato da voz FM
34 RC_DAMP Interrompe a voz do FM
35 RC_KON Liga a voz do gerador FM
36 RC_LEGATO_ON Liga legato da voz FM
37 RC_KOFF Desliga a voz do gerador FM
38 RCA_PARAM Configuração em tempo real FM
39 RCA_VOICE Configura voz para o canal FM
40 RCA_VPARAM Configura parâmetros de voz
41 RCA_VOICEP Configura voz para o canal FM
42 RMA_ADPLAYLP Reprod. dados ADPCM repete
43 RMA_ADPLOY_SAMPLE Reprodução ADPCM
44 RM_PVEL Define velocidade som de ritmo
:
48 RI_DAMP Força parada do gerador FM
49 RI_ALLOFF Ativa todos os canais FM
50 RI_EVENT Converte uma Nota
51 RI_PCHB Definição da posição do pitch
52 RI_PCHBR Configurando o grau do pitch
53 RIA_PARAM Definição em tempo real p/ FM
54 RIA_VOICE Configuração de voz do FM
55 RIA_VPARAM Definição de voz para o FM
56 RIA_VOICEP Config. tonalidade p/ FM

```

**Entrada:** A – Código da função.

Outros parâmetros dependem da função chamada.

**Saída:** CY = 1 → erro nos parâmetros de entrada.

Outros parâmetros dependem da função chamada.

**Registradores:** Depende da função chamada.

**RC\_NOTE** (00AEH/MBIOS)

Função: Liga uma voz no gerador FM e desliga automaticamente após um tempo especificado.

Entrada: A = 32.

IX – Endereço do CHDB com dados da voz FM.

DE – Intervalo (0 ~ 32.767). O valor central é 15.360 e um semitom corresponde a 256.

C – Velocidade (0~15). 0 é o mais fraco e 15 o mais forte.

B – Timer. Desliga quando SV\_TEMPO é chamado esse número de vezes.

Saída: Nenhuma.

Registradores: Todos.

**RC\_LEGATO** (00AEH/MBIOS)

Função: Liga uma voz no gerador FM e desliga automaticamente após um tempo especificado. Ao contrário de RC\_NOTE, esta função não inicia a envoltória.

Entrada: A = 33.

IX – Endereço do CHDB com dados da voz FM.

DE – Intervalo (0 ~ 32.767). O valor central é 15.360 e um semitom corresponde a 256.

C – Velocidade (0~15). 0 é o mais fraco e 15 o mais forte.

B – Timer. Desliga quando SV\_TEMPO é chamado esse número de vezes.

Saída: Nenhuma.

Registradores: Todos.

**RC\_DAMP** (00AEH/MBIOS)

Função: Força a parada da voz FM que está tocando.

Entrada: A = 34.

IX – Endereço do CHDB com dados da voz FM.

Saída: Nenhuma.

Registradores: Todos.

**RC\_KON** (00AEH/MBIOS)

Função: Liga uma voz FM.

Entrada: A = 35.

IX – Endereço do CHDB com dados da voz FM.

DE – Intervalo (0 ~ 32.767). O valor central é 15.360 e um semitom corresponde a 256.

C – Velocidade (0~15). 0 é o mais fraco e 15 o mais forte.

Saída: Nenhuma.

Registradores: Todos.

#### **RC\_LEGATO\_ON (00AEH/MBIOS)**

Função: Liga uma voz FM. Ao contrário de RC\_KON, esta função não inicia a envoltória.

Entrada: A = 36.

IX – Endereço do CHDB com dados da voz FM.

DE – Intervalo (0 ~ 32.767). O valor central é 15.360 e um semitom corresponde a 256.

C – Velocidade (0~15). 0 é o mais fraco e 15 o mais forte.

Saída: Nenhuma.

Registradores: Todos.

#### **RC\_KOFF (00AEH/MBIOS)**

Função: Desliga uma voz FM.

Entrada: A = 37.

IX – Endereço do CHDB com dados da voz FM.

Saída: Nenhuma.

Registradores: Todos.

#### **RCA\_PARAM (00AEH/MBIOS)**

Função: Ajustar os parâmetros em tempo real para uma voz FM.

Entrada: A = 38.

IX – Endereço do CHDB com dados da voz FM.

C – Offset do parâmetro a ser ajustado na lista CHDB.

DE – Dados de configuração.

Os dados que podem ser configurados com esta função são os seguintes:

YCA\_TRANS

YCA\_TRIG

YCA\_PITCH

YCA\_VOL

YCA\_VEL

Saída: Nenhuma.

Registradores: Todos.

#### **RCA\_VOICE (00AEH/MBIOS)**

Função: Associar um instrumento a uma voz do FM.

Entrada: A = 39.

IX – Endereço do CHDB com dados da voz FM.

C – Número do padrão do instrumento na ROM (0~63).

Os instrumentos disponíveis são os seguintes:

0	Piano 1	32	Piano 3
1	Piano 2	33	Electric Piano 2
2	Violin	34	Santool 2
3	Flute 1	35	Brass
4	Clarinet	36	Flute 2
5	Oboe	37	Clavicode 2
6	Trumpet	38	Clavicode 3
7	Pipe Organ 1	39	Koto 2
8	Xylophone	40	Pipe Organ 2
9	Organ	41	PohdsPLA
10	Guitar	42	RohdsPRA
11	Santool 1	43	Orch L
12	Electric Piano 1	44	Orch R
13	Clavicode 1	45	Synth Violin
14	Harpsicode 1	46	Synth Organ
15	Harpsicode 2	47	Synthe Brass
16	Vibraphone	48	Tube
17	Koto 1	49	Shamisen
18	Taiko	50	Magical
19	Engine 1	51	Huwawa
20	UFO	52	Wander Flat
21	Synthesizer bell	53	Hardrock
22	Chime	54	Machine
23	Synthesizer bass	55	Machine V
24	Synthesizer	56	Comic
25	Synth Percussion	57	SE-Comic
26	Synth Rhythm	58	SE-Laser
27	Harm Drum	59	SE-Noise
28	Cowbell	60	SE-Star 1
29	Close Hi-hat	61	SE-Star 2
30	Snare Drum	62	Engine 2
31	Bass Drum	63	Silence

Saída: Nenhuma.

Registradores: Todos.

#### RCA\_VPARAM (00AEH/MBIOS)

Função: Ajustar parâmetros de uma voz FM.

Entrada: A = 40.

IX – Endereço do CHDB com dados da voz FM.

C – Offset do parâmetro a ser ajustado na lista CHDB.

DE – Dados de configuração.

Os dados que podem ser configurados com esta função  
são os seguintes:

CA00_LS	CA01_LS	YCA00_MULTI
CA00_AR	YCA01_AR	CA01_MULTI
CA00_RR	YCA01_RR	YCA_VTRANS
CA00_VELS	YCA01_VELS	YCA_FB
CA00_VTL	YCA01_VTL	

Saída: Nenhuma.

Registradores: Todos.

### RCA\_VOICEP (00AEH/MBIOS)

Função: Definir uma voz FM com dados de tom.

Entrada: A = 41.

IX – Endereço do CHDB com dados da voz FM.

BC – Apontador para os dados, que ocupam 32 bytes com  
a seguinte estrutura:

0~7 V\_NAME Nome do som

8~9 V\_TRANS Valor de transposição

10 V-ARG Várias configurações:

bit7 – Nível trêmolo:

0- 1dB; 1- 4,8 dB

bit6 – Nível vibrato:

0- 7%; 1- 14%

bit5 – Define trêmolo/vibrato:

0- não; 1- configura

bit4 – Define tom fixo:

0- tom normal; 1- tom fixo

bit3~bit1 – Nível de feedback:

000 – 0 100 –  $\pi/2$

001 –  $\pi/16$  101 –  $\pi$

010 –  $\pi/8$  110 –  $2\pi$

011 –  $\pi/4$  111 –  $4\pi$

bit0 – Tipo de conexão entre oper.:

0- serial; 1- paralelo

11~15 – Sem uso

16 VOO\_MULTI – Dados a serem configurados para os registradores 20H  
(voz 0) a 35H (voz 1):

bit7 – Modulação de amplitude:

0- sem; 1- com

bit6 - Vibrato:  
     0- sem; 1- com  
 bit5 - EG-TYP (tipo de envoltória):  
     0- percussivo; 1- constante  
 bit4 - KSR (Key Scale Rate):  
     0- sem; 1- com  
 bit3~bit0 - Múltiplo:  
     00- $\frac{1}{2}$  04-4 08-8 12-12  
     1-1 05-5 09-9 13- 12  
     02-2 06-6 10-10 14-15  
     03-3 07-7 11-10 15-15

17 VOO\_TL - Dados a serem configurados para os registradores 40H (voz 0) a 55H (voz 1):  
 bit7~bit6 - KSL (Key Scale Level):  
     00 - 0 dB/oitava  
     01 - 1,5 dB/oitava  
     10 - 3 dB/oitava  
     11 - 6 dB/oitava  
 bit7~bit6 - Nível total:  
     bit0 - 0,75 dB  
     bit1 - 1,5 dB  
     bit2 - 3 dB  
     bit3 - 6 dB  
     bit4 - 12 dB  
     bit5 - 24 dB

18 VOO\_AR - Dados a serem configurados para os registradores 60H (voz 0) a 75H (voz 1):  
 bit7~bit4 - Attack Rate:  
     0dB a 96dB: 1111 - 0 mS  
                   1110 - 0,2 mS  
                   0000 - 2826 mS  
     10% a 90%: 1111 - 0 mS  
                   1110 - 0,11 mS  
                   0000 - 1482 mS  
 bit7~bit4 - Decay Rate:  
     0dB a 96dB: 1111 - 0 mS  
                   1110 - 2,4 mS  
                   0000 - 39280 mS  
     10% a 90%: 1111 - 0 mS  
                   1110 - 0,51 mS  
                   0000 - 8212 mS

19 VOO\_RR - Dados a serem configurados para os registradores 80H  
 (voz 0) a 95H (voz 1):  
 bit7~bit4 - Sustain Level:  
     bit7 - 24 dB  
     bit6 - 12 dB  
     bit5 - 6 dB  
     bit4 - 3 dB  
 bit3~bit0 - Release Rate:  
     bit0 - 24 dB  
     bit1 - 12 dB  
     bit2 - 6 dB  
     bit3 - 3 dB  
 20 VOO\_VELS - Sensibilidade à velocidade realizada via software através da MBIOS.  
 bit7~bit4 - Sem uso.  
 bit3~bit0 - Sensibilidade:  
     0000 - Inválido  
     0001 - Menor  
     1111 - Maior  
 21~23 - Sem uso.  
 24 VO1\_MULTI (igual a VOO\_MULTI mas atua no operador 1)  
 25 VO1\_TL (igual a VOO\_TL mas atua no operador 1)  
 26 VO1\_AR (igual a VOO\_AR mas atua no operador 1)  
 27 VO1\_RR (igual a VOO\_RR mas atua no operador 1)  
 28 VO1\_VELS (igual a VOO\_VELS mas atua no operador 1)  
 29~31 - Sem uso.

### **RM\_TIMER (00AEH/MBIOS)**

Função: Ativar/desativar funções do temporizador.

Entrada: A = 16.

C - bit7~bit2 - Sem uso.  
 bit1 - temporizador 2:  
     0 - Desativar; 1 - Ativar.  
 bit0 - temporizador 1:  
     0 - Desativar; 1 - Ativar.

Saída: Nenhuma.

Registradores: Todos.

**RM\_TIM1** (00AEH/MBIOS)

Função: Definir o valor do temporizador nº 1.

Entrada: A = 17.

C – Período com passo de 80 uS. Corresponde a 20,48 mS quando C=0 e 80 uS quando C=255.

Saída: Nenhuma.

Registradores: Todos.

**RM\_TIM2** (00AEH/MBIOS)

Função: Definir o valor do temporizador nº 2.

Entrada: A = 18.

C – Período com passo de 80 uS. Corresponde a 20,48 mS quando C=0 e 80 uS quando C=255.

Saída: Nenhuma.

Registradores: Todos.

**RM\_TEMPO** (00AEH/MBIOS)

Função: Definir o ciclo do temporizador nº 2.

Entrada: A = 19.

C – Número de semínimas por minuto.

Saída: Nenhuma.

Registradores: Todos.

**RM\_DAMP** (00AEH/MBIOS)

Função: Força parada de todos os canais ativos do gerador FM.

Entrada: A = 20.

Saída: Nenhuma.

Registradores: Todos.

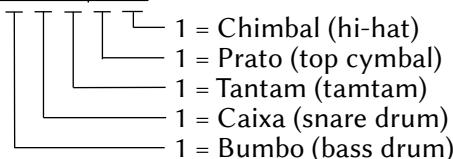
**RM\_VEL** (00AEH/MBIOS)

Função: Define a velocidade das cinco peças de bateria (ritmo).

Esta função pode definir mais de uma peça por vez.

Entrada: A = 44.

b7	b6	b5	b4	b3	b2	b1	b0
C = [ 0 0 0 B C T P C ]							



E – Velocidade. 0 é o mais forte e 31 o mais fraco.

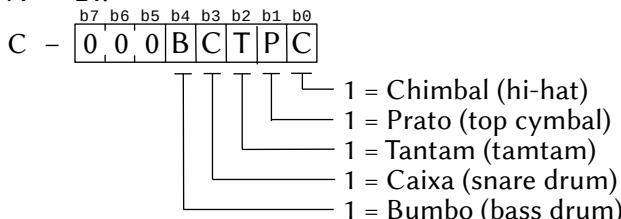
Saída: Nenhuma.

Registradores: Todos.

#### **RM\_PERC (00AEH/MBIOS)**

Função: Ativa o som peças de bateria (ritmo). Várias peças podem ser tocadas simultaneamente.

Entrada: A = 21.



E – Velocidade. 0 é o mais forte e 31 o mais fraco.

Saída: Nenhuma.

Registradores: Todos.

#### **RMA\_MK (00AEH/MBIOS)**

Função: Retorna o estado do teclado musical.

Entrada: A = 22.

DE – Apontador para um buffer de 9 bytes.

IY – Apontador do MIDB indicando master/slave.

Saída: O buffer apontado por DE contém a seguinte estrutura, onde uma tecla pressionada corresponde a um bit setado:

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0 →	0	C	B	A#	0	A	G#	G
1 →	0	F#	F	E	0	D#	D	C#
2 →	0	C	B	A#	0	A	G#	G
3 →	0	F#	F	E	0	D#	D	C#
4 →	0	C	B	A#	0	A	G#	G
5 →	0	F#	F	E	0	D#	D	C#
6 →	0	C	B	A#	0	A	G#	G
7 →	0	F#	F	E	0	D#	D	C#
8 →	0	C	0	0	0	0	0	0

Obs.: A Nota “C” (dó) do byte 8 corresponde à segunda oitava e “C” do byte 0 corresponde à sexta oitava.

Registradores: Todos.

#### **RMA\_LFO (00AEH/MBIOS)**

Função: Configura os níveis do vibrato e do trêmolo.

Entrada: A = 23.

C - 

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	0	0	0	0	V	T

Trêmolo: 0=1dB, 1=4,8 dB

Vibrato: 0=7%, 1=14%

IY – Apontador do MIDB indicando master/slave.

Saída: Nenhuma.

Registradores: Todos.

#### **RM\_A\_TRANS (00AEH/MBIOS)**

Função: Define a transição do som atual para o som subsequente.

Entrada: A = 24.

DE – Valor de transposição, em unidades correspondentes a 1% de 100/256 (~0,0039).

IY – Apontador do MIDB indicando master/slave.

Saída: Nenhuma.

Registradores: Todos.

#### **RM\_UTEMPR (00AEH/MBIOS)**

Função: Converte o tom de temperamento para o tom do temperamento definido.

Entrada: A = 28.

D – Intervalo (A Nota dó [C] central é 60).

Saída: DE – Tom convertido.

Registradores: Todos.

#### **RM\_CTEMPR (00AEH/MBIOS)**

Função: Seleciona o temperamento.

Entrada: A = 29.

C – Temperamento:

00 – Pythagoras

01 – Meanone

02 – Werk Meister

03 – Werk Meister (modificado)

04 – Werk Meister (another)

05 – Kirunberger

06 – Kirunberger (modificado)

07 – Valoty Young

08 – Lamoo

09 – Temperamento balanceado (valor inicial)

- 10 – C (dó menor)
- 11 – C# (dó maior)
- 12 – D (ré menor)
- 13 – D# (ré maior)
- 14 – E (mi)
- 15 – F (fá menor)
- 16 – F# (fá maior)
- 17 – G (sol menor)
- 18 – G# (sol maior)
- 19 – A (lá menor)
- 20 – A# (lá maior)
- 21 – B (sol)

Saída: Nenhuma.

Registradores: Todos.

#### **RM\_PITCH (00AEH/MBIOS)**

Função: Ajusta o tom da Nota atual e subsequentes.

Entrada: A = 30.

BC – Canal master.

DE – Canal slave.

O pitch deve estar no intervalo 410~459, sendo que o valor inicial é 440.

Saída: Nenhuma.

Registradores: Todos.

#### **RM\_TSTRAN (00AEH/MBIOS)**

Função: Define a transição entre o som atual e o som subsequente.

O valor de transição deve estar entre -12.799 e +12.799. O valor inicial é 0 e os valores estão em centésimos.

Entrada: A = 31.

BC – Canal master.

DE – Canal slave.

Saída: Nenhuma.

Registradores: Todos.

#### **RI\_DAMP (00AEH/MBIOS)**

Função: Força a parada de todos os canais FM.

Entrada: A = 48.

Saída: Nenhuma.

Registradores: Todos.

**RI\_ALLOFF** (00AEH/MBIOS)

Função: Desliga todos os canais FM atribuídos.

Entrada: A = 49.

Saída: Nenhuma.

Registradores: Todos.

**RI\_EVENT** (00AEH/MBIOS)

Função: Ativar/desativar tons, convertendo o temperamento.

Entrada: A = 50.

Quando keyon:

D – Pitch (60 – Nota Dó [C] central) + 80H

E – Velocidade (0=maior, 15=menor)

Quando keyoff:

D – Pitch (60 – Nota Dó [C] central)

Saída: Nenhuma.

Registradores: Todos.

**RI\_PCHB** (00AEH/MBIOS)

Função: Define a posição do pitch bender.

Entrada: A = 51.

DE – Posição do pitch bender (os 16 bits são válidos em complemento de 2, onde 7FFFH define a posição mais alta, 0 a central e 8000H a posição mais baixa).

Saída: Nenhuma.

Registradores: Todos.

Obs.: Esta função chama RCA\_PARAM (38) internamente.

**RI\_PCHBR** (00AEH/MBIOS)

Função: Define o grau que o pitch bender dará ao pitch.

Entrada: A = 52.

C – Grau (0 a 12 vezes).

Saída: Nenhuma.

Registradores: Todos.

**RIA\_PARAM** (00AEH/MBIOS)

Função: Ajusta os parâmetros em tempo real para a voz FM ativa.

Os parâmetros que podem ser ajustados por esta função são YCA\_TRANS e YCA\_VOL.

Entrada: A = 53.

IY – Apontador do MIDB indicando master/slave.

C – Offset do parâmetro em CHDB.

DE – Parâmetros de configuração.

Saída: Nenhuma.

Registradores: Todos.

#### **RIA\_VOICE (00AEH/MBIOS)**

Função: Atribui um número de instrumento a um canal FM.

Entrada: A = 54.

IY – Apontador do MIDB indicando master/slave e para a voz FM a ser atribuída.

C – Número do instrumento (0 a 63).

Saída: Nenhuma.

Registradores: Todos.

#### **RIA\_VPARAM (00AEH/MBIOS)**

Função: Define os parâmetros de um canal FM.

Entrada: A = 55.

IY – Apontador do MIDB indicando master/slave e para a voz FM a ser atribuída.

C – Offset do parâmetro em CHDB.

DE – Parâmetros de configuração.

Os parâmetros que podem ser definidos por esta função são os seguintes:

CA00_LS	CA01_LS	YCA00_MULTI
CA00_AR	YCA01_AR	CA01_MULTI
CA00_RR	YCA01_RR	YCA_VTRANS
CA00_VELS	YCA01_VELS	YCA_FB
CA00_VTL	YCA01_VTL	

Saída: Nenhuma.

Registradores: Todos.

#### **RIA\_VOICEP (00AEH/MBIOS)**

Função: Define um instrumento para um canal FM.

Entrada: A = 56.

IY – Apontador do MIDB indicando master/slave e para a voz FM a ser definida.

BC – Endereço dos dados do instrumento.

Saída: Nenhuma.

Registradores: Todos.

#### **RM\_MOVE\_DI (00AEH/MBIOS)**

Função: Transfere dados PCM/ADPCM entre dispositivos.

Entrada: A = 0.

IX – Endereço do PDB indicando a origem. Os seguintes campos dão relevantes:

PDB\_DEV (Número do dispositivo)

PDB\_ADDR (Endereço inicial)

PDB\_SIZE (Tamanho dos dados de transferência)

IY – Endereço do PDB indicando o destino. Os seguintes campos são relevantes:

PDB\_DEV (Número do dispositivo)

PDB\_ADDR (Endereço inicial)

Saída: CY = 1 → erro na transferência.

Registradores: Todos.

#### **RM\_READ\_DI (00AEH/MBIOS)**

Função: Transfere 256 bytes de dados PCM/ADPCM para a RAM.

Entrada: A = 26

DE – Endereço destino na RAM.

IX – Endereço do PDB indicando a origem. Os seguintes campos dão relevantes:

PDB\_DEV (Número do dispositivo)

PDB\_ADDR (Endereço inicial)

Saída: CY = 1 → erro na transferência.

Registradores: Todos.

#### **RM\_WRITE\_DI (00AEH/MBIOS)**

Função: Transfere 256 bytes de dados da RAM para o dispositivo PCM/ADPCM

Entrada: A = 27.

DE – Endereço fonte na RAM.

IX – Endereço do PDB indicando o destino. Os seguintes campos dão relevantes:

PDB\_DEV (Número do dispositivo)

PDB\_ADDR (Endereço inicial)

Saída: CY = 1 → erro na transferência.

Registradores: Todos.

**RM\_TRACE\_DI** (00AEH/MBIOS)

**Função:** Rastrear os dados ADPCM com base no valor de previsão inicial e na largura de quantização para localizar o próximo valor predito e a próxima largura de quantização.

**Entrada:** A = 1.

C – Modo para início de rastreio:

0 → previsão inicial em 8000H e largura de quantização em 007FH. Especificar os seguintes dados no PDB:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho da transferência)

1 → previsão inicial e largura de quantização. Além dos dados para C=0, também devem ser especificados:

PDB\_PCM (valor predito inicial)

PDB\_STEP (largura inicial de quantização)

**Saída:** Os seguintes campos retornam válidos no PDB:

PDB\_ADDR (próximo endereço de início)

PDB\_PCM (próximo valor previsto)

PDB\_STEP (próxima largura de quantização)

Se CY = 1, houve erro no rastreio.

**Registradores:** Todos.

**RM\_CONV\_PCM\_DI** (00AEH/MBIOS)

**Função:** Converter dados ADPCM em dados PCM com base no valor de previsão inicial e na largura de quantização.

**Entrada:** A = 2.

C – Modo para início de rastreio:

0 → previsão inicial em 8000H e largura de quantização em 007FH.

1 → previsão inicial e largura de quantização especificados no PDB.

IX – Endereço do PDB origem com os dados ADPCM. Os seguintes campos devem ser preenchidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (valor de conversão)

PDB\_SAMPLE (frequência de amostragem)

Se C=1, preencher também:

PDB\_PCM (valor predito inicial)

PDB\_STEP (largura inicial de quantização)

IY – Endereço do PDB destino com os dados PCM. Os seguintes campos devem ser preenchidos:  
 PDB\_DEV (número do dispositivo)  
 PDB\_ADDR (endereço inicial)

Saída: Os seguintes campos retornam válidos no PDB origem:  
 PDB\_PCM (próximo valor previsto)  
 PDB\_STEP (próxima largura de quantização)  
 Se CY = 1, houve erro na conversão.

Registradores: Todos.

### **RM\_CONV\_ADPCM\_DI (00AEH/MBIOS)**

Função: Converter dados PCM em dados ADPCM com base no valor de previsão inicial e na largura de quantização para dados ADPCM.

Entrada: A = 3.

C – Modo para início de rastreio:  
 0 → previsão inicial em 8000H e largura de quantização em 007FH.  
 1 → previsão inicial e largura de quantização especificados no PDB.

IX – Endereço do PDB origem com os dados PCM:

PDB\_DEV (número do dispositivo)  
 PDB\_ADDR (endereço inicial)  
 PDB\_SIZE (valor de conversão)  
 PDB\_SAMPLE (frequência de amostragem)  
 Se C=1, preencher também:  
 PDB\_PCM (valor predito inicial)  
 PDB\_STEP (largura inicial de quantização)

IY – Endereço do PDB destino com os dados ADPCM:

PDB\_DEV (número do dispositivo)  
 PDB\_ADDR (endereço inicial)

Saída: Os seguintes campos retornam válidos no PDB origem:  
 PDB\_SIZE (tamanho após conversão)  
 PDB\_SAMPLE (cópia da freq. amostragem da fonte PCM)  
 PDB\_PCM (próximo valor previsto)  
 PDB\_STEP (próxima largura de quantização)  
 Se CY = 1, houve erro na conversão.

Registradores: Todos.

**RM\_DAC\_BIAS (00AEH/MBIOS)**

Função: Define o volume para reprodução PCM (define o registrador 17H do Y8950).

Entrada: A = 4.

IY – Apontador para o MIDB indicando master/slave e canal PCM (dispositivo) 0 ou 1.

C – Volume (1 a 7). O volume 7 é o máximo.

Saída: Nenhuma.

Registradores: Todos.

**RMA\_DAC\_DI (00AEH/MBIOS)**

Função: Reproduzir dados PCM.

Entrada: A = 5.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de reprodução. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na reprodução.

Registradores: Todos.

**RMA\_ADC\_DI (00AEH/MBIOS)**

Função: Gravar dados PCM.

Entrada: A = 6.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de gravação. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na gravação.

Registradores: Todos.

**RMA\_ADPCM\_BIAS** (00AEH/MBIOS)

Função: Define o volume para reprodução ADPCM.

Entrada: A = 7.

IY – Apontador para o MIDB indicando master/slave e canal PCM (dispositivo) 0 ou 1.

C – Volume (0 a 63). O volume 63 é o máximo.

Saída: Nenhuma.

Registradores: Todos.

**RMA\_ADPLAY\_DI** (00AEH/MBIOS)

Função: Reproduzir dados ADPCM no modo não local.

Entrada: A = 8.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de reprodução. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na reprodução.

Registradores: Todos.

**RMA\_ADPLAY\_DI** (00AEH/MBIOS)

Função: Gravar áudio ADPCM no modo não local.

Entrada: A = 9.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de gravação. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na gravação.

Registradores: Todos.

**RMA\_ADPAY\_SAMPLE (00AEH/MBIOS)**

Função: Altera a frequência de amostragem durante a reprodução no modo local.

Entrada: A = 43.

IY – Apontador para o MIDB indicando master/slave.

DE – Frequência de amostragem.

Saída: Nenhuma.

Registradores: Todos.

**RMA\_BREAK (00AEH/MBIOS)**

Função: Interrompe gravação ou reprodução no modo local.

Entrada: A = 10.

IY – Apontador para o MIDB indicando master/slave.

Saída: Nenhuma.

Registradores: Todos.

**RMA\_ADPLAY (00AEH/MBIOS)**

Função: Reproduzir dados ADPCM no modo local.

Entrada: A = 11.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de reprodução. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na reprodução.

Registradores: Todos.

**RMA\_ADREC (00AEH/MBIOS)**

Função: Reproduzir dados ADPCM no modo local.

Entrada: A = 12.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de gravação. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na gravação.

Registradores: Todos.

#### **RMA\_ADPLAYLP (00AEH/MBIOS)**

Função: Reproduzir dados ADPCM no modo local com loop. No final, a reprodução é reiniciada indefinidamente. Para interromper, execute RMA\_BREAK (Função 10).

Entrada: A = 42.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

IX – Endereço do PDB com os dados de reprodução. Os seguintes campos devem ser definidos:

PDB\_DEV (número do dispositivo)

PDB\_ADDR (endereço inicial)

PDB\_SIZE (tamanho)

PDB\_SAMPLE (frequência de amostragem)

Saída: CY = 1 → erro na reprodução.

Registradores: Todos.

#### **RMA\_PHASE\_SET\_DI (00AEH/MBIOS)**

Função: Pegar 256 bytes de dados PCM na RAM principal, converter em dados ADPCM e armazenar na RAM externa:

End RAM ext	Pitch	núm. formas onda
0000H ~ 07FFH	24H ~ 36H	16
0800H ~ 0FFFH	37H ~ 42H	32
1000H ~ 17FFH	43H ~ 4EH	64
1800H ~ 1FFFH	4FH ~ 5AH	128

Entrada: A = 13.

IY – Apontador para o MIDB indicando master/slave.

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

DE – Endereço dos dados PCM.

Saída: Nenhuma.

Registradores: Todos.

Obs.: Antes da conversão, RMA\_BREAK (Func. 10) é executada.

### **RMA\_PHASE\_EG (00AEH/MBIOS)**

Função: Definir os dados da envoltória.

Entrada: A = 14.

IY – Apontador para o MIDB indicando master/slave.

DE – Endereço dos dados da envoltória (7 bytes):

+0 – Valor do timer 1

+1 – Nível total (Total level)

+2 – Taxa de ataque (Attack rate)

+3 – Taxa de decaimento #1 (Decay rate #1)

+4 – Nível de sustentação (Sustain Level)

+5 – Taxa de decaimento #2 (Decay rate #2)

+6 – Taxa de liberação (Release rate )

Saída: Nenhuma.

Registradores: Todos.

### **RMA\_PHASE\_EVENT (00AEH/MBIOS)**

Função: Ligar a amostragem do tom especificado ou desligar a simulação de amostragem pelo teclado.

Entrada: A = 15.

IY – Apontador para o MIDB indicando master/slave.

D – Key on: intervalo+80H (nota dó [C] central é 60)

Key off: intervalo (nota dó [C] central é 60)

A faixa válida é 24H ~ 5AH.

Saída: Nenhuma.

Registradores: Todos.

### **RMA\_CSM\_DI (00AEH/MBIOS)**

Função: Reprodução de dados CSM.

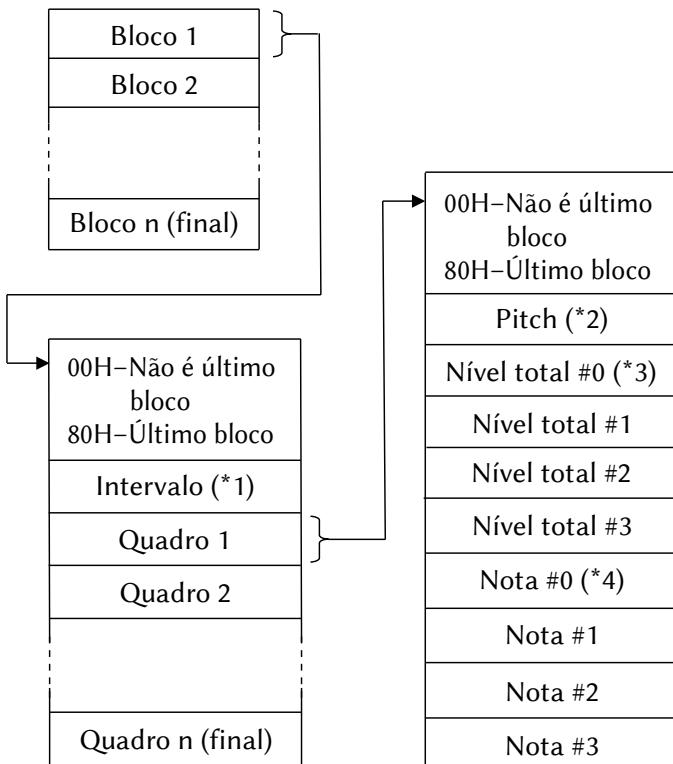
Entrada: A = 25.

IY – Apontador para o MIDB indicando master/slave.

B – Volume (0 a 127, onde 0 é o volume máximo).

C – Especificação de filtro (consultar ZMA\_PH\_FILTER).

DE – Endereço dos dados CSM com a seguinte estrutura:



(\*1) Intervalo (Timer 2 → 0 = 81,9 mS / 255 = 0,32 mS)

(\*2) Pitch (Timer 1 → 0 = 20,9 mS / 255 = 0,08 mS)

(\*3) Nível total (dados de volume de cada canal)  
0 = máximo / 127 = mínimo

(\*4) Nota (dados de pitch de cada canal)

Os 4 bits superiores especificam a oitava no intervalo de 0 a 7 e os 4 inferiores a escala:

00	- C#	08	- G
01	- D	09	- G#
02	- D#	10	- A
03	- Nenhum	11	- Nenhum
04	- E	12	- A#
05	- F	13	- B
06	- F#	14	- C
07	- Nenhum	15	- Nenhum

Saída: Nenhuma.

Registradores: Todos.

**SV\_IRQ** (00B4H/MBIOS)

Função: Manipulação de interrupção do MSX-Audio (é necessário configurar o hook HKEYI (FD9AH)).

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**9.5.6 – MSX-JE****EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = 16H – Dispositivo de manipulação do MSX-JE.

E = 00H – Retorna o apontador para a tabela de endereços de entrada das rotinas do MSX-JE.

B – ID do slot da tabela de endereços.

HL – Endereço de um buffer de 64 bytes para a tabela (deve estar na página 3).

Saída: CY = 1 → não há MSX-JE.

CY = 0 → HL é incrementado de 4 a cada MSX-JE encontrado e apontará para o final de uma tabela que reserva 4 bytes para cada MSX-JE. O valor original de HL aponta para o início da tabela, que tem a seguinte estrutura:

+00H – Vetor de capacidade

+01H – ID do slot

+02H – Endereço mais baixo

+03H – Endereço mais alto

O byte vetor de capacidade tem a seguinte estrutura:

bit 0 – 0 → compatível com MSX-JE

1 → incompatível

bit 1 – 0 → existe interface de terminal virtual

1 → não existe interface

bit 2 – 0 → existe interface de dicionário

1 → não existe dicionário

bit 3 – 0 → existe função de registro e excl. dicionário

1 → não existe função de registro e exclusão

bit 4 ~ bit 7 → sempre 0.

O ID de slot (+01H) e o endereço (+02H,+03H) especificam o ponto de entrada para as funções do MSX-JE. A chamada deve ser feita através da rotina CALSLT (0030H) da Main-ROM, colocando o número da função no registrador A.

Registradores: Todos.

### **9.5.6.1 – Chamando as funções do MSX-JE**

#### **INQUIRY** (Função 01H)

Função: Retorna o tamanho da área de trabalho.

Entrada: A = 01H.

Saída: HL – Limite máximo de tamanho da área de trabalho utilizada pelo MSX-JE.

DE – Limite inferior do tamanho da área de trabalho utilizada pelo MSX-JE.

BC – Tamanho mínimo necessário para o MSX-JE usar a função de aprendizado

#### **INVOKE** (Função 02H)

Função: Inicializa a área de trabalho.

Entrada: A = 02H.

HL – Endereço da área de trabalho protegida pelo AP.

DE – Tamanho da área de trabalho protegida pelo AP.

Saída: Nenhuma.

#### **RELEASE** (Função 03H)

Função: Libera a memória protegida pelo AP.

Entrada: A = 03H.

HL – Endereço da área de trabalho protegida pelo AP.

Saída: Nenhuma.

#### **CLEAR** (Função 04H)

Função: Limpa o buffer para conversão Kana – Kanji.

Entrada: A = 04H.

HL – Endereço da área de trabalho protegida pelo AP.

Saída: Nenhuma.

**SET\_TTB** (Função 05H – opcional)

Função: Passar o texto e ler os dados que a converter do AP para o MSX-JE, configurando-os no buffer interno do MSX-JE. Essa função faz com que o MSX-JE reconverte o texto fornecido.

Entrada: A – 05H.

HL – Endereço da área de trabalho.

DE – Endereço do texto a ser convertido novamente.

BC – Endereço do TTB (Transferable Text Block).

Saída: A = 255 → Função não suportada.

**DISPATCH** (Função 06H – opcional)

Função: Passar o controle da CPU do AP para o MSX-JE.

Entrada: A = 06H.

HL – Endereço da área de trabalho.

Saída: HL – Endereço do STB (Screen image Text Block).

A – estado de retorno:

bit 0 = 1 → o AP exibe o STB.

bit 1 = 1 → o AP pode obter resultado da conversão.

bit 2 = 1 → a conversão do MSX-JE terminou.

estados possíveis com os bits combinados:

000 – o MSX-JE ignora chave (não há entrada de chave).

001 – entrada ou conversão em andamento.

01x – feita conversão parcial.

10x – conversão interrompida e finalizada.

11x – totalmente convertido.

**GET\_RESULT** (Função 07H)

Função: Retorna o resultado da conversão.

Entrada: A – 07H.

HL – Endereço da área de trabalho.

Saída: HL – Endereço inicial do resultado da conversão, terminando com um byte 00H.

**GET\_TTB** (Função 08H – opcional)

Função: Adquire os dados do texto obtido por GET\_RESULT.

Entrada: A = 08H.

HL – Endereço da área de trabalho.

Saída: HL – Endereço do TTB (Transferable Text Block). Se esta função não for suportada, (HL) apontará para um byte 00H.

**INQUIRY\_WINDOW\_SIZE** (Função 09H – opcional)

Função: Define o formato da janela.

Entrada: A = 09H.

HL – Endereço da área de trabalho.

E – Comprimento máximo da “tail”.

B – Altura máxima da janela.

C – Largura máxima da janela.

Saída: HL – Endereço dos dados de especificação da janela.

+00H – Tipo de janela:

1 – Independente.

2 – “tail”.

+01H – Largura da janela (1~255).

+02H – Altura da janela (1~255).

**CONFLICT\_DETECT** (Função 0AH – opcional)

Função: Evitar conflitos de colisão de chaves.

Entrada: A = 0AH.

HL – Endereço da área de trabalho.

Saída: A – 00H → conflito não detectado.

FFH → conflito detectado.

**9.5.6.2 – Interface do dicionário do MSX-JE****HAN\_ZEN** (Função 40H)

Função: Converte uma string de caracteres de um byte em uma string de caracteres de dois bytes.

Entrada: A = 40H.

HL – Endereço da área de trabalho.

DE – Endereço da string de origem (de um byte).

BC – Endereço da string de caracteres de dois bytes.

Saída: A = 0 → Conversão bem-sucedida.

A ≠ 0 → Erro na conversão.

**ZEN\_HAN** (Função 41H)

Função: Converte uma string de caracteres de dois bytes em uma string de caracteres de um byte.

Entrada: A = 41H.

HL – Endereço da área de trabalho.

DE – Endereço da string de origem (de dois byte).

BC – Endereço da string de um byte.

Saída: A = 0 → Conversão bem-sucedida.  
 A ≠ 0 → Erro na conversão.

### **HAN\_KATA** (Função 42H)

Função: Converte uma string de caracteres de um byte do alfabeto romano, katakana ou uma combinação delas em uma string de caracteres katakana de dois bytes.

Entrada: A = 42H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string de origem (de um byte).  
 BC – Endereço da string de caracteres katakana.  
 Saída: A = 0 → Conversão bem-sucedida.  
 A ≠ 0 → Erro na conversão.

### **HAN\_HIRA** (Função 43H)

Função: Converte uma string de caracteres de um byte do alfabeto romano, katakana ou uma combinação delas em uma string de caracteres hiragana de dois bytes.

Entrada: A = 43H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string de origem (de um byte).  
 BC – Endereço da string de caracteres hiragana.  
 Saída: A = 0 → Conversão bem-sucedida.  
 A ≠ 0 → Erro na conversão.

### **KATA\_HIRA** (Função 44H)

Função: Converte uma string de caracteres katakana de dois bytes em uma string de caracteres hiragana de dois bytes.

Entrada: A = 44H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string katakana de dois bytes.  
 BC – Endereço da string de caracteres hiragana.  
 Saída: A = 0 → Conversão bem-sucedida.  
 A ≠ 0 → Erro na conversão.

### **HIRA\_KATA** (Função 45H)

Função: Converte uma string de caracteres hiragana de dois bytes em uma string de caracteres katakana de dois bytes.

Entrada: A = 45H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string katakana de dois bytes.  
 BC – Endereço da string de caracteres hiragana.  
 Saída: Nenhuma.

#### **OPEN\_DIC** (Função 46H)

Função: Reservado para futuras expansões.  
 Entrada: A = 46H.  
 HL – Endereço da área de trabalho.  
 DE = 0000H  
 Saída: A – Sempre retorna 5.

#### **HENKAN** (Função 47H)

Função: Converte uma string de caracteres katakana e hiragana de 2 bytes em uma string mista Kanji-Kana.  
 Entrada: A = 47H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string katakana/hiragana.  
 Saída: A – número de conversões possíveis. Se não houver nenhuma, retorna 0. As strings convertidas devem ser obtidas pela função JI\_KOHO (48H).

#### **JI\_KOHO** (Função 48H)

Função: Adquire a próxima conversão obtida por HENKAN (47H).  
 Entrada: A = 48H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da próxima string Kanji-Kana convertida.  
 BC – Endereço da string Kanji-Kana secundária.  
 Saída: A = 0 → nenhuma conversão Kanji-Kana adquirida.  
 A > 0 → número da conversão Kanji-Kana adquirida.

#### **ZEN\_KOHO** (Função 49H)

Função: Adquire a conversão anterior obtida por HENKAN (47H).  
 Entrada: A = 49H.  
 HL – Endereço da área de trabalho.  
 DE – Endereço da string Kanji-Kana anterior convertida.  
 BC – Endereço da string Kanji-Kana secundária.  
 Saída: A = 0 → nenhuma conversão Kanji-Kana adquirida.  
 A > 0 → número da conversão Kanji-Kana adquirida.

**JI\_BLOCK** (Função 4AH)

Função: Cria um grupo de conversões Kanji-Kana de menor prioridade ao lado do grupo principal.

Entrada: A = 4AH.

HL – Endereço da área de trabalho.

Saída: A = 0 → Grupo não criado.

A > 0 → Número de conversões Kanji-Kana de menor prioridade.

**ZEN\_BLOCK** (Função 4BH)

Função: Cria um grupo de conversões Kanji-Kana de maior prioridade ao lado do grupo principal.

Entrada: A = 4BH.

HL – Endereço da área de trabalho.

Saída: A = 0 → Grupo não criado.

A > 0 → Número de conversões Kanji-Kana de maior prioridade.

**KAKUTEI1** (Função 4CH)

Função: Confirma o resultado da conversão Kanji-Kana.

Entrada: A = 4CH.

HL – Endereço da área de trabalho.

E – Número da conversão Kanji-Kana dentro do grupo

BC – Endereço do buffer de conversão Kanji-Kana.

Saída: BC – 0AH + “natto curry”

**KAKUTEI2** (Função 4DH)

Função: Confirma o resultado da conversão Kanji-Kana.

Entrada: A = 4DH.

HL – Endereço da área de trabalho.

E – Número da conversão Kanji-Kana dentro do grupo

BC – Endereço do buffer de conversão Kanji-Kana.

Saída: A – Tamanho em bytes da string Kanki-Kana.

BC – 04H + “natto”

**CLOSE\_DIC** (Função 4EH)

Função: Função não implementada.

Entrada: A = 4EH.

Saída: A – Sempre 0.

**TOUROKU** (Função 4FH)

**Função:** Fornece os dados de leitura, dados de palavras e parte do texto, e inclui a palavra especificada no dicionário.

**Entrada:** A – 4FH.

HL – Endereço da área de trabalho.

DE – Endereço do buffer de leitura.

BC – Endereço do buffer de inclusão de palavras.

**Saída:** A – 00H → palavra incluída com sucesso.

01H → espaço livre insuficiente.

02H → overflow na paridade das palavras.

04H → dados de leitura incorretos.

08H → dados de palavra incorretos.

10H → parte do texto está incorreta.

FFH → não suportado.

**SAKUJO** (Função 50H)

**Função:** Fornece os dados de leitura, dados de palavras e parte do texto, excluindo a palavra especificada do dicionário.

**Entrada:** A – 50H.

HL – Endereço da área de trabalho.

DE – Endereço do buffer de leitura.

BC – Endereço do buffer de exclusão de palavras.

**Saída:** A – 00H → palavra excluída com sucesso.

01H → palavra a ser excluída não foi encontrada.

04H → dados de leitura incorretos.

08H → dados de palavra incorretos.

10H → parte do texto está incorreta.

FFH → não suportado.

**9.5.7 – MSX UNAPI****EXTBIO** (FFCAH/Work Area)

**Função:** Acessa funções estendidas da BIOS.

**Entrada:** A = 00H – obtém o número de implementações da API especificada.

A > 00H – Retorna os parâmetros da API especificada.

D = 22H – Dispositivo de manipulação do MSX UNAPI.

E = 22H – Retorna dados da API especificada.

(F487H) – Identificador de especificação da API (string alfanumérica de até 15 caracteres terminada em 00H, sem distinção de maiúsculas e minúsculas).

Saída: A = 00H – B → Número de implementações da API especificada.  
 A > 00H – A → ID do slot da rotina da implementação.  
 B → Segmento da mapper da implementação (FFH – Não está na mapper).  
 HL → Endereço do ponto de entrada das rotinas da implementação (se estiver na página física 3, os valores de A e B são desconsiderados).

Registradores: AF, BC, HL.

### 9.5.7.1 – RAM Helper

#### **EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = FFH → API: RAM helper  
 D = 22H → Dispositivo de manipulação do MSX UNAPI.  
 E = 22H → Retorna parâmetros da API.  
 HL = 0000H

Saída: HL = 0000H → RAM helper não instalada.  
 HL > 0000H → Endereço da tabela de salto na página 3.  
 BC – Endereço da tabela de mapeamento.  
 A – Número de entradas na tabela de salto.  
 A tabela de salto tem a seguinte estrutura:  
 +00H CALMAP chama rotina mapper  
 +03H RDBYTE lê byte da RAM  
 +06H CALSEG chama rotina na RAM

Registradores: AF, BC, HL.

#### **CALMAP** (HL+00H) – Valor de HL obtido via EXTBI

Função: Chama uma rotina em um segmento de RAM mapeada.

Entrada: IYh – ID de slot.  
 IYI – Número do segmento da mapper.  
 IX – Endereço da rotina (deve ser na página física 1).

AF, BC, DE, HL – Parâmetros para a rotina chamada.  
 Saída: AF, BC, DE, HL, IX, IY – Parâmetros de retorno da rotina.

Registradores: Depende da rotina chamada.

#### **RDBYTE** (HL+03H) – Valor de HL obtido via EXTBI

Função: Lê um byte de um segmento da RAM mapeada.

Entrada: A – ID do slot.

B – número do segmento.

HL – Endereço a ser lido (os dois bits mais altos são ignorados).

Saída: A - byte lido no endereço especificado.

Registradores: A.

### **CALSEG (HL+06H) – Valor de HL obtido via EXTBIO**

Função: Chama uma rotina um um segmento da RAM mapeada usando parâmetros em linha.

Entrada: AF, BC, DE e HL podem conter parâmetros para a rotina chamada (não usar IX e IY).

Parâmetros de chamada em linha, no seguinte formato:

CALL <endereço da rotina>

DB ID da rotina

DB número do segmento

ID da rotina:

b7	b6	b5	b4	b3	b2	b1	b0
M	M	E	E	E	E	E	E

Rotina a ser chamada na tabela de salto (0 a 63)

Slot da mapper especificado na tabela de mapper (0 a 3)

- A tabela de salto inicia no endereço 4000H, sendo que o índice 0 significa 4000H, o índice 1 significa 4003H e assim por diante até o valor 63, de três em três bytes.

- A tabela da mapper ocupa 8 bytes reservando dois bytes para cada mapper, podendo gerenciar até 4 mappers (0 a 3), e tem a seguinte estrutura:

+0 – ID do slot da primeira mapper

+1 – Número de segmentos disponíveis na 1<sup>a</sup> mapper

+2 – ID do slot da segunda mapper

+3 – Número de segmentos disponíveis na 2<sup>a</sup> mapper

+4 – ID do slot da terceira mapper

+5 – Número de segmentos disponíveis na 3<sup>a</sup> mapper

+6 – ID do slot da quarta mapper

+7 – Número de segmentos disponíveis na 4<sup>a</sup> mapper

Obs.: se a mapper tiver 4 Mbytes, o número de segmentos será FEH, pois o valor FFH é reservado para o sistema.

Saída: AF, BC, DE, HL, IX e IY podem conter valores válidos.  
 Registradores: Depende da rotina chamada.

### 9.5.7.2 – API para cartuchos Ethernet

#### **EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H – obtém o número de implementações da API especificada.

A > 00H – Retorna os parâmetros da API especificada.

D = 22H – Dispositivo de manipulação do MSX UNAPI.

E = 22H – Retorna dados da API especificada.

(F487H) = “ETHERNET”

Saída: A = 00H → B – Número de implementações da API.

A > 00H → A – ID do slot da rotina da implementação.

B – Segmento da mapper da implementação  
 (FFH → Não está na mapper).

HL – Endereço do ponto de entrada das rotinas da implementação (se estiver na página física 3, os valores de A e B são desconsiderados).

Registradores: AF, BC, HL.

#### **ETH\_GETINFO** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna a versão e o nome da implementação.

Entrada: A = 0.

Saída: HL – Endereço da string do nome da implementação.

B – Versão da implementação da API (primária).

C – Versão da implementação da API (secundária).

D – Especificação da versão da API (primária).

E – Especificação da versão da API (secundária).

Registradores: Todos.

#### **ETH\_RESET** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o hardware e as variáveis de estado à sua condição inicial (condição logo após o reset do micro).

Entrada: A = 1.

Saída: Nenhuma.

Registradores: Todos.

**ETH\_GET\_HWADD** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o endereço da Ethernet.

Entrada: A = 2.

Saída: L-H-E-D-C-B – Endereço.

Registradores: Todos.

**ETH\_GET\_NETSTAT** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Verifica o estado de conexão à rede.

Entrada: A = 3.

Saída: A – 0 → não há conexão com uma rede ativa.

1 → existe conexão com rede ativa.

Registradores: Todos.

**ETH\_NET\_ONOFF** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita ou desabilita a rede.

Entrada: A = 4.

B – 0 → retorna o estado atual da rede.

1 → habilita a rede.

2 → desabilita a rede.

Saída: A – 1 → rede habilitada.

2 → rede desabilitada.

Registradores: Todos.

**ETH\_DUPLEX** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Configura o modo duplex.

Entrada: A = 5

B – 0 → retorna o modo corrente.

1 → seleciona modo half-duplex.

2 → seleciona modo full-duplex.

Saída: A – 1 → modo half-duplex selecionado.

2 → modo half-duplex selecionado.

3 → modo desconhecido ou modo duplex não é aplicável.

**ETH\_FILTERS** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Configura os filtros de recepção de frames.

Entrada: A = 6.

B – bit 7 – 0 → nenhuma ação.

1 → retorna configuração atual.

bit 6 – Reservado.

bit 5 – Reservado.

bit 4 – 0 → desabilita modo promíscuo.  
                   1 → habilita modo promíscuo.  
 bit 3 – Reservado.  
 bit 2 – 0 → rejeita frames “broadcast”.  
                   1 → aceita frames “broadcast”.  
 bit 1 – 0 → rejeita frames menores que 64 bytes.  
                   1 → aceita frames menores que 64 bytes.  
 Bit 0 – Reservado.

Saída: A – Configuração de filtro após a execução (mesmo formato do registrador B na entrada)

Registradores: Todos.

#### **ETH\_IN\_STATUS** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Verifica a disponibilidade dos frames recebidos.

Entrada: A = 7.

Saída: A – 0 → Não ha frames recebidos disponíveis.  
                   1 → Ao menos um frame recebido está disponível.  
                   BC – Tamanho do frame mais antigo disponível.  
                   HL – Bytes 12 e 13 do frame mais antigo disponível.

Registradores: Todos.

#### **ETH\_GET\_FRAME** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Recupera o frame mais antigo.

Entrada: A = 8.

HL – 0 → descarta o frame.  
                   Outro valor → endereço de destino do frame.

Saída: A – 0 → frame recuperado ou descartado.  
                   1 → não há frames recebidos disponíveis.  
                   BC – Tamanho do frame recuperado.

#### **ETH\_SEND\_FRAME** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Envia um frame.

Entrada: A = 9.

HL – Endereço de destino do frame na memória.  
                   BC – Tamanho do frame.  
                   D – 0 → Execução síncrona  
                           1 → Execução assíncrona.

Saída: A – 0 → Frame enviado ou transmissão iniciada.  
                   1 → Tamanho do frame inválido.  
                   2 → Ignorado.  
                   3 → Portadora perdida.

4 → Número excessivo de colisões.

5 → Modo assíncrono não suportado.

Registradores: Todos.

#### **ETH\_OUT\_STATUS (HL/ExtBIOS) – Valor de HL obtido via EXTBIO**

Função: Recupera o frame mais antigo.

Entrada: A = 10.

Saída: A = 0 → Nenhum frame enviado desde o último reset.

1 → Transmitindo neste momento.

2 → Transmissão finalizada com sucesso.

3 → Portadora perdida.

4 → Número excessivo de colisões.

Registradores: Todos.

#### **ETH\_SET\_HWADD (HL/ExtBIOS) – Valor de HL obtido via EXTBIO**

Função: Seleciona o endereço da Ethernet.

Entrada: A = 11.

L-H-E-D-C-B – Endereço Ethernet a ser setado.

Saída: L-H-E-D-C-B – Endereço Ethernet após a execução.

Registradores: Todos.

### **9.5.8 – MemMan**

#### **EXTBIO (FFCAH/Work Area)**

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 32H → Retorna informações sobre entradas alternativas para funções do MemMan.

B = 0 → Endereço de entrada para FastUse0 (func. 0)

1 → Endereço de entrada para FastUse1 (func. 1)

2 → Endereço de entrada para FastUse2 (func. 2)

3 → Endereço de entrada para FastTsrCall (fn. 63)

4 → Endereço de entrada para BasicCall

5 → Endereço de entrada para FastCurSeg (fn. 32)

6 → Endereço de entrada para o manipulador de funções do MemMan

7 → Retorna a versão do MemMan (VerMM: #H.L)

8 → Endereço de entrada para FastXTsrCall (f. 61)

Saída: HL – Endereço ou versão.

Registradores: Todos.

### 9.5.8.1 – Fast Calls (Entradas alternativas preferenciais)

**FastUse0** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita um segmento na página física 0 (0000H~3FFFH). A habilitação somente será possível se o segmento conter os pontos de entrada para as rotinas padrão de troca de slots.

Entrada: HL – Número do segmento.

Saída: A – 00H → segmento habilitado com sucesso.  
FFH → falha na habilitação do segmento.

Obs.: Esta função é idêntica à função 0 (Use0).

**FastUse1** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita um segmento na página física 1 (4000H~7FFFH).

Entrada: HL – Número do segmento.

Saída: A – 00H → segmento habilitado com sucesso.  
FFH → falha na habilitação do segmento.

Obs.: Esta função é idêntica à função 1 (Use1).

**FastUse2** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Habilita um segmento na página física 2(8000H~BFFFH).

Entrada: HL – Número do segmento.

Saída: A – 00H → segmento habilitado com sucesso.  
FFH → falha na habilitação do segmento.

Obs.: Esta função é idêntica à função 2 (Use2).

**FastTsrCall** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina de driver da TSR.

Entrada: BC – Código ID da função TSR.

AF, DE, HL – Parâmetros para a TSR.

Saída: AF, BC, DE, HL – Parâmetros de retorno da TSR.

Obs.: Esta função é idêntica à função 63 (TsrCall), exceto que aqui o registrador DE pode ser usado sem problemas.

**BasicCall** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama uma rotina da Main-ROM.

Entrada: IX – Endereço da rotina na página física 0 ou 1.

AF, BC, DE, HL – Parâmetros a passar para a rotina.

Saída: AF, BC, DE, HL – Parâmetros de retorno da rotina.

Obs.: As interrupções são desabilitadas.

**FastCurSeg** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o número de segmento atual de uma página.

Entrada: B – Página física (0, 1, 2 ou 3).

Saída: HL – Número do segmento.

A – Tipo de segmento: 00H – PSEG; FFH – FSEG.

Obs.: Esta função é idêntica à função 32 (CurSeg).

**MemMan** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama diretamente uma função do MemMan.

Entrada: E – Número da função.

AF, BC, HL – Parâmetros a passar para a rotina.

Saída: AF, BC, DE, HL – Parâmetros de retorno da rotina.

**VerMM** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Retorna o número da versão do MemMan.

Entrada: Nenhuma.

Saída: HL – Versão no formato “H.L”.

**FastXTsrCall** (HL/ExtBIOS) – Valor de HL obtido via EXTBIO

Função: Chama a entrada de driver de uma TSR.

Entrada: IX – Código ID da entrada TSR chamada.

AF, BC, DE, HL – Parâmetros a passar para a rotina.

Saída: AF, BC, DE, HL – Parâmetros de retorno da rotina.

Obs.: Esta função é idêntica à função 61 (XtrsCall).

**9.5.8.2 – Funções do MemMan****Use0** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 00H → Função Use0. Habilita um segmento na página física 0 (0000H~3FFFH). A habilitação somente será possível se o segmento contiver os pontos de entrada para as rotinas padrão de troca de slots.

HL – Número do segmento.

Saída: A – 00H → Segmento habilitado com sucesso.

FFH → Falha na habilitação do segmento.

Obs.: Usar preferencialmente a entrada FastUse0 da função 32H (Info) do MemMan.

**Use1** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 01H → Função Use0. Habilita um segmento na página física 1 (4000H~7FFFH).

HL – Número do segmento.

Saída: A = 00H → segmento habilitado com sucesso.

FFH → falha na habilitação do segmento.

Obs.: Usar preferencialmente a entrada FastUse1 da função 32H (Info) do MemMan.

**Use2** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 02H → Função Use2. Habilita um segmento na página física 2 (8000H~BFFFH).

HL – Número do segmento.

Saída: A = 00H → Segmento habilitado com sucesso.

FFH → Falha na habilitação do segmento.

Obs.: Usar preferencialmente a entrada FastUse2 da função 32H (Info) do MemMan.

**Alloc** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 0AH → Função Alloc. Aloca um segmento.

B – Código de preferência do segmento:

b7	b6	b5	b4	b3	b2	b1	b0
P	T	0	0	0	0	S	S

Tipo de segmento:

00–PSEG0000 10–PSEG8000

01–PSEG4000 11–FSEG

1 – Prefere TPA (RAM padrão do MSXDOS).

1 – Prefere slot não expandido.

Saída: HL – Número do segmento (0000H – sem segmentos livres)

**SetRes** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 0BH → Função SetRes. Atribui a um segmento o status “reservado”.

HL – Número do segmento.

Saída: Nenhuma.

**DeAlloc** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 14H → Função DeAlloc. Libera um segmento.

HL – Número do segmento.

Saída: Nenhuma.

**IniChk** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A – Código de controle.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 1EH → Função IniChk. Inicializa o MemMan antes de um programa.

Saída: A – Código de controle + “M”.

DE – Número da versão no formato “D.E”.

**Status** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 1FH → Função Status. Retorna informações de status do MemMan.

Saída: HL – Número se segmentos disponíveis.

BC – Número de segmentos livres.

DE – Número de segmentos controlados simultaneamente pelo MemMan e pelo DOS2.

A – Estado de conexão do hardware:

bit0 = 0 → Suporte à mapper do DOS2 não disponível.

1 → Suporte à mapper do DOS2 instalado.

bit1 ~ bit 7 → sempre 0.

**CurSeg** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 20H → Função CurSeg. Retorna o número do segmento em uma página.

B – Número da página física (0, 1, 2 ou 3).

Saída: HL – Número do segmento.

A – Tipo de segmento: 00H – PSEG; FFH – FSEG.

Obs.: Usar preferencialmente a entrada FastCurSeg da função 32H (Info) do MemMan.

**StoSeg** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 28H → Função StoSeg. Armazena o estado do segmento atual.

HL – Endereço de um buffer de 9 bytes.

Saída: Nenhuma.

**RstSeg** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 29H → Função RstSeg. Reativa o estado de um segmento que foi armazenado.

HL – Buffer de estado de 9 bytes.

Saída: Nenhuma.

**XtsrCall** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 3DH → Função XTsRCall. Chama uma entrada do driver TSR.

IX – Código ID da entrada TSR chamada.

AF, BC, HL – Parâmetros a passar para a rotina.

Saída: AF, BC, DE, HL – Parâmetros de retorno da rotina.

Obs.: Usar preferencialmente a entrada FastXtrsCall da função 32H (Info) do MemMan.

**GetTsrID** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 3EH → Função GetTsrID. Determina código ID da TSR.

HL – Apontador para TsrName. As posições não usadas devem ser preenchidas com espaços.

Saída: CY = 0 → Não encontrado.

CY = 1 → ID encontrado.

BC → código ID da TSR.

**TsrCall** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 3FH → Função TsrCall. Chama uma entrada do driver TSR.

BC – Código ID da entrada TSR chamada.

AF, HL – Parâmetros a passar para a rotina.

Saída: AF, BC, DE, HL – Parâmetros de retorno da rotina.

Obs.: Usar preferencialmente a entrada FastTrsCall da função 32H (Info) do MemMan.

**HeapAlloc** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 46H → Função HeapAlloc. Aloca espaço na “heap”.

HL – Tamanho do espaço a ser alocado.

Saída: HL – 0000H → memória insuficiente para alocação.

Outro valor → endereço inicial do espaço alocado.

**HeapDeAlloc** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 47H → Função HeapDeAlloc. Libera espaço alocado na “heap”.

HL – Tamanho do espaço a ser alocado.

Saída: Nenhuma.

**HeapMax** (FFCAH/Work Area) – Execução via EXTBIO

Função: Acessa funções estendidas da BIOS.

Entrada: A = 00H.

D = 4DH → Dispositivo de manipulação do MEMMAN.

E = 48H → Função HeapMax. Retorna o tamanho máximo de espaço disponível na “heap”.

Saída: HL – Espaço disponível na “heap”.

### 9.5.9 – Comandos de sistema

**EXTBIO** (FFCAH/Work Area)

Função: Acessa funções estendidas da BIOS

Entrada: A = 00H.

D = FFH → Dispositivo de sistema.

E = 00H → Retorna o endereço inicial, o ID do slot e o código do fabricante do dispositivo.

B – ID do slot da tabela de parâmetros.

HL – Endereço da tabela de parâmetros.

Saída: CY = 1 se não houver dispositivos.

CY = 0 → há dispositivos.

B – ID do slot da tabela de parâmetros.

HL – Endereço inicial da tabela de parâmetros.

Cada dispositivo ocupa 5 bytes na tabela apontada por HL, com a seguinte estrutura:

+00H – Reservado. Sempre 0.

+01H – Código do fabricante.

+02H – Endereço MSB da tabela de salto.

+03H – Endereço LSB da tabela de salto.

+04H – ID do slot do dispositivo.

Os fabricantes são os seguintes:

00 – ASCII

01 – Microsoft

02 – Canon

03 – Casio Computer

04 – Fujitsu

05 – General Fujitsu

06 – Hitachi, Ltd.

07 – Kyocera

08 – Matsushita (Panasonic)

- 09 – Mitsubishi Electric Corporation
- 10 – NEC
- 11 – Yamaha (Nippon Gakki)
- 12 – Japan Victor Company (JVC)
- 13 – Philips
- 14 – Pioneer
- 15 – Sanyo Electric
- 16 – Sharp Japan
- 17 – Sony
- 18 – Spectravideo
- 19 – Toshiba
- 20 – Mitsumi Electric
- 21 – Telematika
- 22 – Gradiente Brazil
- 23 – Sharp do Brazil
- 24 – GoldStar (LG)
- 25 – Daewoo
- 26 – Samsung
- 128 – Image Scanner (Matsushita)
- 170 – Darky (SuperSoniqs)
- 171 – Darky (SuperSoniqs) second setting
- 212 – 1chipMSX / Zemmix Neo (KdL firmware)
- 254 – MPS2 (ASCII)

## 9.6 – ROTINAS DA INTERFACE DE DISCO

### 9.6.1 – Inicialização da interface

As rotinas abaixo são executadas uma única vez durante a inicialização do sistema ao ser ligado ou após um reset, na sequência INIHRD, DRIVES, INIENV. O endereço de chamada delas é diferente para cada interface.

**INIHRD** (????H/Interface de disco).

Função: Inicializa o hardware assim que o controle for passado ao cartucho da interface de disco.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**DRIVES** (????H/Interface de disco).

Função: Verifica os drives físicos conectados ao sistema.

Entrada: Flag Z = 0 → Duas unidades lógicas são atribuídas a uma unidade física.

1 → Apenas uma unidade lógica é atribuída a uma unidade física.

Saída: L – Número de drives conectados.

Registradores: F, HL, IX, IY.

**INIENV** (????H/Interface de disco).

Função: Inicializa a área de trabalho da interface de disco.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**9.6.2 – Rotinas padrão da interface****MALLOC** (01CBH/Interface de disco)

Função: Aloca um buffer para um segmento para o MSXDOS2.

Entrada: Número de bytes a reservar.

Saída: A > 0 → Erro na alocação.

A = 0 → Alocação realizada.

HL → Endereço inicial do buffer.

(HL-2, HL-1) → tamanho do buffer + 2.

Registradores: Todos.

**DEALOC** (2D0FH/Interface de disco)

Função: Realoca um buffer para um segmento para o MSXDOS2.

Entrada: HL – Endereço inicial do buffer.

(HL-2, HL-1) → tamanho do buffer + 2.

Saída: Desconhecido.

Registradores: Todos.

**DSKIO** (4010H/Interface de disco)

Função: Leitura/escrita direta de setores.

Entrada: HL – Apontador para a TPA (área de transferência).

DE – Número do primeiro setor a ler ou escrever.

B – Número de setores a ler ou escrever.

A – número do drive (0=A:, 1=B; 2=C; etc).

- C – ID da formatação do disco:
- F0H – 63 setores por trilha (para HD's)
  - F8H – 80 trilhas, 9 setores por trilha, face simples.
  - F9H – 80 trilhas, 9 setores por trilha, face dupla.
  - FAH – 80 trilhas, 8 setores por trilha, face simples.
  - FBH – 80 trilhas, 8 setores por trilha, face dupla.
  - FCH – 40 trilhas, 9 setores por trilha, face simples.
  - FDH – 40 trilhas, 9 setores por trilha, face dupla.

CY – 0 → Leitura.  
1 → Escrita.

- Saída: B – Número de setores efetivamente transferidos.
- CY – 1 → Transferência executada com sucesso.
- 0 → Erro na transferência. O código de erro retorna no registrador A.
- A – Código de erro:
- 00 – protegido contra escrita.
  - 02 – Não pronto.
  - 04 – Erro de CRC (setor não acessível).
  - 06 – Erro de busca.
  - 08 – Cluster não encontrado.
  - 10 – Falha na escrita.
  - 12 – Erro de disco.
  - Somente MSXDOS2 ou superior:
  - 18 – Disco não DOS.
  - 20 – Versão do MSXDOS incorreta.
  - 22 – Disco não formatado.
  - 24 – Disco trocado.
  - Restantes: erro de disco.

Registradores: Todos.

### **DSKCHG (4013H/Interface de disco)**

Função: Verificar o estado de troca do disco.

- Entrada: A – Número do drive (0=A:, 1=B:, 2=C:, etc).
- B – Sempre 00H.
- C – ID da formatação do disco (igual a DISKIO/4010H).
- HL – Apontador para o DPB respectivo.

Saída: CY = 1 → Erro na execução.  
           A – Código de erro (igual a DISKIO/4010H).  
           CY = 0 → verificado com sucesso.  
           B – 00H → estado desconhecido.  
           01H → disco não trocado.  
           FFH → disco trocado.

Registradores: Todos.

#### **GETDPB** (4016H/Interface de disco)

Função: Preenche o DPB da unidade de disco.  
           Entrada: A – número do drive.  
                   B – primeiro byte da FAT (ID do disco).  
                   C – ID da formatação do disco (igual a DISKIO/4010H).  
                   HL – Apontador para o DPB a ser preenchido (18 bytes).  
           Saída: HL – Endereço inicial do DPB preenchido:

DRIVE	+00H Número drive (0=A:, etc)
MEDIA	+01H Tipo de mídia (F8H~FFH)
SECSIZ	+02H Tamanho do setor
DIRMSK	+04H (SECSIZ/32) - 1
DIRSHFT	+05H Número de bits 1 em DIRMSK
CLUSMSK	+06H Setores por cluster - 1
CLUSHFT	+07H Núm bits 1 em CLUSMSK - 1
FIRFAT	+08H Primeiro setor da FAT
FATCNT	+0AH Número de FATS
MAXENT	+0BH N° entradas diretório raiz
FIRREC	+0CH Primeiro setor área dados
MAXCLUS	+0EH Total de clusters + 1
FATSIZ	+10H Número de setores por FAT
FIRDIR	+11H Primeiro setor diretório
FATDIR	+13H Endereço da FAT na RAM

Registradores: Todos.

#### **CHOICE** (4019H/Interface de disco)

Função: Retorna o endereço da mensagem de formatação do disco.  
           Entrada: Nenhuma.  
           Saída: HL – Endereço da mensagem, que termina com um byte  
                   00H. Se não houver escolha (somente um tipo de  
                   formatação é suportado), HL retorna 0000H.  
           Registradores: Todos.

**DSKFMT** (401CH/Interface de disco)

Função: Formatar um disco.

Entrada: A – Escolha da formatação pelo usuário (rotina CHOICE /4019H). Pode variar a 1 a 9.

D – Número do drive (00H=A; 01H=B; etc).

HL – Endereço inicial da área de trabalho usada pela rotina de formatação.

BC – Tamanho da área de trabalho usada pela rotina de formatação.

Saída: CY – 0 → Formatação concluída com sucesso.

1 → Erro durante a formatação.

A – Código de erro:

00 – protegido contra escrita.

02 – Não pronto.

04 – Erro de dados (CRC).

06 – Erro de busca.

08 – Registro não encontrado.

10 – Falha de escrita/gravação

12 – parâmetro inválido.

14 – Memória insuficiente.

16 – outros erros.

Registradores: Todos.

**MTROFF** (401FH/Interface de disco)

Função: Parar o motor dos drives.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

Obs.: Esta função está implementada em apenas algumas interfaces. Se a interface não tiver esta função implementada, o valor do endereço 401FH será 00H. Portanto, é necessário verificar se a função existe lendo o endereço 401FH antes de chamá-la.

**CALBAS** (4022H/Interface de disco)

Função: Chamar o interpretador BASIC.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**FORMAT** (4025H/Interface de disco)

Função: Formatar um disco apresentando mensagem.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: Todos.

**STPDRV** (4029H/Interface de disco)

Função: Parar o motor dos drives.  
 Entrada: Nenhuma.  
 Saída: Nenhuma.  
 Registradores: Todos.

**SLTDOS** (402DH/Interface de disco)

Função: Retorna o ID do slot do Kernel do DOS.  
 Entrada: Nenhuma.  
 Saída: A – ID do slot (igual a RDSL (000CH/Main)).  
 Registradores: Todos.

**HIGMEM** (4030H/Interface de disco)

Função: Retorna o endereço mais alto disponível na RAM.  
 Entrada: Nenhuma.  
 Saída: HL – Endereço.  
 Registradores: Todos.

**BLKDOS** (40FFH/Interface de disco)

Este endereço contém número do bloco ativo do BDOS. O sistema ocupa um total de 64Kb, que é dividido em 4 segmentos de ROM de 16Kb. Eles podem ser trocados apenas na página 1 e são numerados com 0, 1, 2 ou 3.

**9.6.3 – Rotinas para acesso a Hard-Disks padrão IDE****IDBYT** (7F80H/Interface IDE)

Função: ID da interface em 3 bytes. (“ID#” para interfaces IDE).

**RDLBLK** (7F89H/Interface IDE)

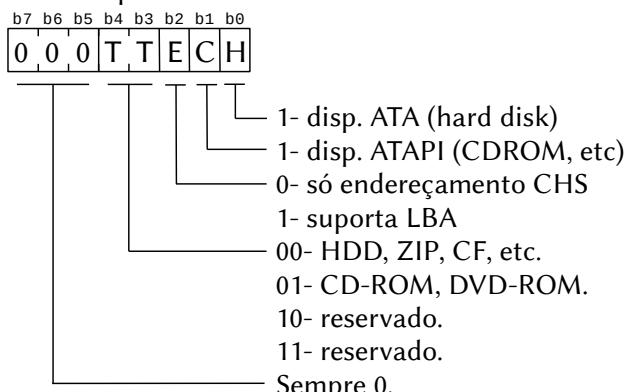
Função: Lê setores lógicos do disco ou dispositivo.

Entrada: CDE – Número do setor.

HL – Endereço na RAM para os dados lidos.

B – Quantidade de setores a ler.

A – ID do dispositivo:



Saída: HL – apontador para os dados lidos.

CY = 1 → Erro na leitura.

A – Código de erro para dispositivos IDE:

00 – Protegido contra escrita.

02 – Não pronto.

04 – Erro de CRC (setor não acessível).

06 – Erro de busca.

08 – Cluster não encontrado.

10 – Falha na escrita.

12 – Erro de disco.

Somente MSXDOS2 ou superior:

18 – Disco não DOS.

20 – Versão do MSXDOS incorreta.

22 – Disco não formatado.

24 – Disco trocado.

Restantes: erro de disco.

Registradores: Todos.

Obs.: Esta rotina também pode ler setores do CD-ROM, que têm 2048 bytes em vez de 512 bytes dos HD's.

**WRLBLK(7F8CH/Interface IDE)**

Função: Escreve setores lógicos do disco.

Entrada: CDE – Número do setor.

HL – Endereço inicial dos dados a serem escritos.

B – Quantidade de setores a escrever.

A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: CY = 1 → erro na escrita.

A – Código de erro. Igual a RDLBLK (7F89H).

Registradores: Todos.

**SELDEV (7FB9H/Interface IDE)**

Função: Seleciona mestre/escravo para dispositivos ATAPI.

Entrada: A – bit0 = 0 → mestre.

1 → escravo.

bit1~bit7: Reservados. Sempre 0.

Saída: CY = 1 se houver erro de time-out.

Registradores: A, BC, IX.

**PACKET (7FBCH/Interface IDE)**

Função: Enviar uma sequência de comandos ATAPI para o dispositivo selecionado.

Entrada: HL – Apontador para o pacote de comandos ATAPI de 12 bytes (não pode estar na página 1 – 4000H~7FFFH).

DE – Endereço para transferência de dados (se houver).

Saída: CY = 1 → Erro na execução.

Z = 1 → Erro de time-out.

A – Código de erro. Igual a RDSECT (7F89H).

Registradores: Todos.

Atenção: Esta entrada tem função diferente em interfaces SCSI.

**DRVADR (7FBFH/Interface IDE)**

Função: Retorna o endereço da área de trabalho.

Entrada: A – Número da unidade (0 a 7).

0~5 – Número do drive (0=A: ~ 5 – F:)

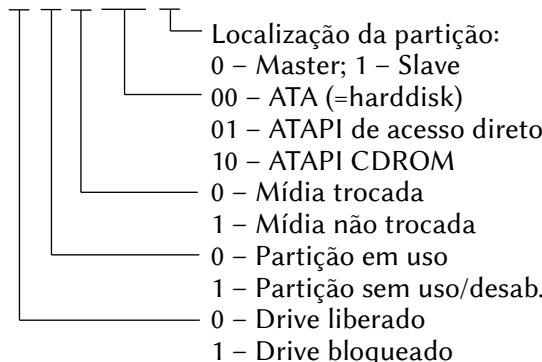
6 – Infobytes do dispositivo Y.

7 – 18 bytes de espaço livre (usado internamente para envio de sequência de comandos ATAPI).

Saída: HL – Apontador para o início dos dados:

+00H – Codebyte do dispositivo:

b7	b6	b5	b4	b3	b2	b1	b0
0	0	L	U	T	ATA	L	

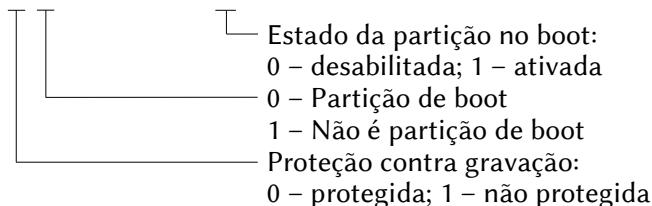


+01H~+03H – Setor de início da partição (bits 0~23).

+04H~+06H – Tamanho da partição em setores menos 1 (bits 0~23).

+07H – Informação adicional sobre a partição.

b7	b6	b5	b4	b3	b2	b1	b0
P	B	0	0	0	0	0	S



Para BIOS 3.0 ou superior:

+08H – Setor de início da partição (bits 24~31).

+08H – Tamanho da partição em setores menos um (bits 24~31).

Registradores: AF, BC, DE, HL, IX.

#### 9.6.4 – Rotinas adicionadas pelo NEXTOR

**GSLOT1** (402DH / Kernel NEXTOR)

Função: Retorna o slot do driver atual.

Entrada: Nenhuma.

Saída: A – Identificador do slot.

Registradores: AF.

Nota: Esta rotina não pode ser chamada diretamente. Deve ser chamada por meio de CALBNK (4042H), da seguinte forma:

```
XOR A
LD IX, GSLOT1
CALL CALBNK
```

### **RDBANK** (403CH / Kernel NEXTOR)

Função: Lê um byte em qualquer banco do Kernel.

Entrada: A – Número do banco.

          HL – Endereço (deve estar na página física 1).

Saída: A – Byte lido.

Registradores: AF.

Nota: Esta rotina não pode ser chamada diretamente. Deve ser chamada por meio de CALBNK (4042H), da seguinte forma:

```
LD A, <número do banco>
LD HL, <endereço do byte>
LD IX, RDBANK
CALL CALBNK
```

### **CALLBO** (403FH / Kernel NEXTOR)

Função: Alternar temporariamente o banco principal do Kernel (normalmente o banco 0, mas será 3 quando executado no modo MSX-DOS 1), e então chamar a rotina cujo endereço está em CODE\_ADD (F1D0H).

Entrada: CODE\_ADD – Endereço da rotina a ser chamada.

AF, BC, DE, HL, IX, IY – Parâmetros para a rotina.

Saída: AF, BC, DE, HL, IX, IY – Retornam dados da rotina.

Registradores: Todos.

### **CALBNK** (4042H / Kernel NEXTOR)

Função: Chamar rotina em outro banco do Kernel.

Entrada: A – Número do banco.

IX – Endereço de rotina (deve estar na página física 1).

AF' – Parâmetro de entrada para a rotina chamada.

          (será passado como AF para a rotina chamada).

BC, DE, HL, IY – Parâmetros para a rotina chamada.

Saída: AF, BC, DE, HL, IX, IY – Valores de saída da rotina.

Registradores: Todos.

**GWORK** (4045H / Kernel NEXTOR)

**Função:** Obter o endereço da entrada SLTWRK de 8 bytes para o slot passado ou para o slot atual na página 1. Os primeiros dois bytes desta área conterão um ponteiro para a área de trabalho da página 3 alocada para este driver (conforme solicitado na rotina DRV\_INIT), ou zero se nenhuma área de trabalho foi alocada.

**Entrada:** A – Número do slot (0 para o slot atual na página 1).

**Saída:** A – Slot atual da página 1 (se 0 na entrada). Inalterado se não for 0 na entrada).

IX – Endereço da entrada SLTWRK de 8 bytes

**Registradores:** F

**Nota:** Esta rotina não pode ser chamada diretamente. Deve ser chamada por meio de CALBNK (4042H), da seguinte forma:  
 LD A,<número do slot ou 0>  
 EX AF,AF'  
 XOR A  
 LD IX,GWORK  
 CALL CALBNK

**K\_SIZE** (40FEH / Kernel NEXTOR)

**Função:** Este endereço contém um byte que informa quantos bancos formam o Kernel Nextor (ou alternativamente, o primeiro número de banco do driver).

**CUR\_BANK** (40FFH / Kernel NEXTOR)

**Função:** Este endereço contém um byte com o número do banco atual. Para o primeiro banco de drivers, esse valor é o mesmo de K\_SIZE e aumenta em um para cada banco de drivers adicional (se houver).

**CHGBNK** (7FD0H / Kernel NEXTOR)

**Função:** Fazer com que o banco especificado seja visível na página 1 do Z80. Esta rotina está disponível em todos os bancos e não apenas no banco 0. Normalmente, o código do driver não precisará usar esta rotina, mas usará CALBNK em seu lugar.

**Entrada:** A – Número do banco

**Saída:** Nenhuma

**Registradores:** AF

**PROMPT** (41E8H / Kernel NEXTOR v2.1.0)

**Função:** Exibir a mensagem “Insira o disco para a unidade X: e pressione uma tecla quando pronto” e aguarda a ação solicitada.

**Entrada:** Nenhuma.

**Saída:** Nenhuma.

**Registradores:** Todos.

**Nota:** Esta rotina não pode ser chamada diretamente. Deve ser chamada por meio de CALLB0 (403FH), da seguinte forma:

```
PROMPT: EQU 041E8H
CODE_ADD: EQU 0F1D0H
CALLB0: EQU 0403FH
LD HL, PROMPT
LD (CODE_ADD), HL
CALL CALLB0
```

**OBS.:** O número do drive “zero-based” é obtido de TARGET (F33FH) e o hook HPROMPT (F24FH) é chamado com o número do drive “zero-based” em A antes que a rotina seja executada.

**9.6.4.1 – Rotinas para drivers de dispositivos de disco****DRV\_SIGN** (4100H / Kernel NEXTOR)

**Função:** Assinatura de driver válida. Usada pelo Kernel na inicialização para verificar se o banco de driver contém um driver válido. Consiste na string "NEXTOR\_DRIVER", sem as aspas, terminada em zero e em maiúsculas.

**DRV\_FLAGS** (410EH / Kernel NEXTOR)

**Função:** Byte de sinalizadores contendo informações sobre o driver:

- bit 0: 0 – Driver baseado em drive.
- 1 – Driver baseado em dispositivo.
- bit 1: Reservado, deve ser zero.
- bit 2: 1 – O driver implementa a rotina DRV\_CONFIG  
(usado pelo Nextor a partir da versão 2.0.5).
- bits 3-7: Reservados, sempre zero.

**RESERVADO** (410FH / Kernel NEXTOR)

**Função:** Byte reservado, deve ser zero.

**DRV\_NAME** (4110H / Kernel NEXTOR)

Função: String contendo o nome do driver. Deve consistir em 32 caracteres ASCII imprimíveis (códigos 32 a 126). Deve ser justificada à esquerda e preenchida à direita com espaços.

**DRV\_TIMI** (4130H / Kernel NEXTOR)

Função: Ponto de entrada para a rotina de interrupção do driver, chamada 50 ou 60 vezes por segundo dependendo da frequência do VDP selecionada. Se o driver não precisar de interrupção, essa entrada deve ser preenchida com RETs. Esta entrada só será chamada se DRV\_INIT retornar CY=1 em sua primeira execução.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**DRV\_VERSION** (4133H / Kernel NEXTOR)

Função: Retornar a versão do driver.

Entrada: Nenhuma.

Saída: A – Número da versão principal.

B – Número da versão secundária.

C – Número de revisão.

Registradores: Todos.

**DRV\_INIT** (4136H / Kernel NEXTOR)

Função: Rotina de inicialização do driver. Os drivers baseados em drive devem retornar o número de unidades de drive necessárias na saída da primeira execução desta rotina. Os drivers baseados em dispositivo podem opcionalmente solicitar um número inicial de drives a serem alocados no momento da inicialização, implementando a rotina DRV\_CONFIG, substituindo assim o procedimento de mapeamento automático. Esta rotina é chamada pelo Kernel duas vezes:

\*1. Primeira execução, para coleta de informações.

Entrada: A = 0.

B = Número de letras de unidade disponíveis.

HL = Tamanho máximo da área de trabalho alocável na página 3.

C = Sinalizadores de inicialização:

bit 5: Solicita uma contagem reduzida da unidade.

Saída: A = Número de unidades de acionamento controladas  
(apenas para motoristas baseados em unidade).

HL = Tamanho da área de trabalho necessária na página 3

C = Sinalizadores de inicialização:

bit 5: Solicita uma contagem reduzida da unidade.

CY = 1 se DRV\_TIMI deve ser conectado ao temporizador  
de interrupção do, 0 caso contrário.

\***2.** Segunda execução, para área de trabalho e inicialização  
do hardware.

Entrada: A = 1.

B = Número de letras de unidade realmente alocadas para  
este controlador.

Registradores: Todos.

Nota: Se 8 bytes ou menos forem necessários, esta rotina deve  
retornar HL = 0 em sua primeira execução, e o espaço de 8  
bytes reservado pelo sistema para este slot em SLTWRK  
deve ser usado como área de trabalho:

```
XOR A
EX AF,AF'
XOR A
LD IX,GWORK
CALL CALBNK
; IX aponta para área de trabalho
; de 8 bytes
```

Se mais de 8 bytes forem necessários, esta rotina deve  
retornar o espaço necessário em HL, obtendo ponteiro para  
o espaço alocado a partir dos primeiros dois bytes do  
espaço reservado pelo sistema para este slot em SLTWRK:

```
XOR A
EX AF,AF'
XOR A
LD IX,GWORK
CALL CALBNK
LD L,(IX)
LD H,(IX+1)
; Use o espaço apontado por HL como
; área de trabalho
```

**DRV\_BASSTAT** (4139H / Kernel NEXTOR)

- Função: Entrada para o manipulador de instruções estendidas BASIC ("CALLs"). Funciona da mesma maneira que os manipuladores padrão, exceto que se as instruções tratadas tiverem parâmetros, a rotina CALBAS do MSX BIOS não pode ser usada diretamente; em vez disso, deve ser usada a rotina CALLB0 na página 0 do Kernel. Se o driver não manipular instruções estendidas BASIC, ele deve setar o sinalizador de transporte (CY=1) e retornar.
- Entrada: Depende da rotina chamada.
- Saída: Depende da rotina chamada.
- Registradores: Depende da rotina chamada.  
CY = 1 se não houver manipulador de instrução.

**DRV\_BASDEV** (413CH / Kernel NEXTOR)

- Função: Entrada para o manipulador de dispositivos estendidos do BASIC. Funciona da mesma maneira que os manipuladores padrão. Se o driver não manipular dispositivos estendidos BASIC, ele deve fazer CY=1 retornar.
- Entrada: Depende da rotina chamada.
- Saída: Depende da rotina chamada.
- Registradores: Depende da rotina chamada.  
CY = 1 se não houver manipulador de instrução.

**DRV\_EXTBIO** (413FH / Kernel NEXTOR)

- Função: Manipulador da BIOS estendida. Funciona da mesma maneira que os manipuladores padrão, exceto que deve retornar um valor em D'(D no conjunto de registradores alternativos).
- Entrada: Depende da rotina chamada.
- Saída: D' = 0 → Retornar imediatamente.  
D' = 0 → Executar Kernel e/ou manipulador da BIOS estendida do sistema.
- Outros registradores: depende da rotina chamada.

**DRV\_DIRECT0** (4142H / Kernel NEXTOR)**DRV\_DIRECT1** (4145H / Kernel NEXTOR)**DRV\_DIRECT2** (4148H / Kernel NEXTOR)**DRV\_DIRECT3** (414BH / Kernel NEXTOR)

**DRV\_DIRECT5** (414EH / Kernel NEXTOR)

**Função:** Entradas para chamadas diretas ao driver. As chamadas para qualquer um dos cinco pontos de entrada disponíveis nos endereços 7850h a 785Ch no ROM do Kernel (banco 0 ou 3) serão mapeadas para uma chamada para o ponto de entrada DRV\_DIRECT correspondente. Todos os registradores, exceto IX e AF', são passados sem modificações.

**Entrada:** Depende da rotina chamada.

**Saída:** Depende da rotina chamada.

**Registradores:** Depende da rotina chamada.

**DRV\_CONFIG** (4151H / Kernel NEXTOR v2.0.5)

**Função:** Permitir que o driver forneça informações sobre a configuração no momento da inicialização. Atualmente, todas as configurações definidas se aplicam apenas a drivers baseados em dispositivo. Esta rotina é chamada duas vezes.

**Entrada:** A – Índice de configuração.

BC, DE, HL – Depende da configuração.

**Saída:** A = 0 – Ok.

1 – Configuração não disponível para o índice fornecido ou índice de configuração desconhecido  
BC, DE, HL = Depende da configuração.

\***1.** Para obter o número de unidades no momento da inicialização.

**Entrada:** A = 1.

B = 0 para o modo DOS 2, 1 para o modo DOS 1

C = Sinalizadores de inicialização

bit 5: O usuário está solicitando uma contagem resumida da unidade.

**Saída:** B – Número de unidades.

\***2.** Obter a configuração padrão para a unidade

**Entrada:** A = 2.

B = 0 para o modo DOS 2, 1 para o modo DOS 1.

C = Nº relativo da unidade no momento da inicialização.

**Saída:** B = Índice do dispositivo.

C = Índice LUN.

**RESERVADO** (4155H a 415FH / Kernel NEXTOR)

Área está reservada para expansão futura (preenchida com zeros).

**DRV\_DSKIO (4160H / Kernel NEXTOR)**

**Função:** Ler ou gravar setores do dispositivo de armazenamento de massa associado a uma unidade de drive. Ao contrário das rotinas padrão do MSX-DOS, esta rotina nunca receberá uma solicitação para transferir dados de / para a página 1.

**Entrada:**

- A – Unidade de drive, começando em 0
- CY – 0 para leitura, 1 para escrita
- B – Número de setores para ler / escrever
- C – Primeiro número do setor para leitura / gravação (bits 22-16) se o bit 7 não estiver definido ou Byte de ID de mídia se o bit 7 definido.
- DE – Primeiro setor para leitura/gravação (bits 15-0).
- HL – Endereço de origem/destino para a transferência.

**Saída:**

- CY = 1 se houve erro na operação
  - A – código de erro (apenas em caso de erro):
  - 0 – protegido contra gravação
  - 2 – Não está pronto
  - 4 – Erro de dados (CRC)
  - 6 – Erro de busca
  - 8 – Registro não encontrado
  - 10 – Falha de gravação
  - 12 – Outros erros
- B – Número de setores realmente lidos (somente em caso de erro)

**DRV\_DSKCHG (4163H / Kernel NEXTOR)**

**Função:** Obter informações sobre o estado de alteração da mídia associada a uma determinada unidade de drive.

**Entrada:**

- A – Unidade de drive, começando em 0.
- B = C – Descritor de mídia.
- HL – Endereço base para DPB – 1.

**Saída:**

- CY = 1 se houver erro.
  - A – Código de erro (apenas em caso de erro). Os mesmos códigos de DRV\_DSKIO são usados.
  - B – Estados da mídia (caso CY = 0):
    - 1 – A mídia não mudou desde a última vez que esta rotina foi chamada.
    - 0 – Desconhecido.
    - 1 – A mídia mudou desde a última vez que esta rotina foi chamada.

**Nota:** Se o estado da mídia for "Alterado" ou "Desconhecido", a rotina deve gerar um DPB para o disco e copiá-lo para o endereço passado em HL mais um. O formato do DPB está descrito na rotina DRV\_GETDPB.

### **DRV\_GETDPB (4166h)**

**Função:** Obter um DPB (Drive Parameter Block) para a mídia associada a uma determinada unidade de driver.

**Entrada:** A = unidade de driver, começando em 0.  
 B = C = Descritor de mídia.  
 HL – Endereço base para DPB-1.

**Saída:** HL aponta para o DPB preenchido (O DPB deve ser copiado para o endereço passado em HL mais um).  
 O formato do DPB de 18 bytes é o seguinte:  
 HL+00: Byte do descritor de mídia (F0h a FFh)  
 +01: Tamanho do setor em 2 bytes (deve ser potência de 2).  
 +03: Máscara de diretório (tamanho do setor/32 – 1).  
 +04: Mudança de diretório (nº bits 1 na másc. Diretório).  
 +05: Máscara de cluster (setores por cluster – 1).  
 +06: Mudança de cluster (nº de bits 1 másc. de cluster + 1).  
 +07: Número do primeiro setor da FAT.  
 +08: Número de FATs.  
 +0A: Número de entradas do diretório (máximo 254).  
 +0B: Número do primeiro setor de dados (2 bytes).  
 +0D: Número máximo do cluster (nº de clusters + 1).  
 +0F: Número total de setores.  
 +10: Número do primeiro setor do diretório raiz.

### **DRV\_CHOICE (4169h)**

**Função:** Retorna uma string de escolha de formato para um disco.

**Entrada:** Nenhuma.

**Saída:** HL – Endereço da string de escolha no slot do Kernel. Esta rotina é chamada pelo Kernel quando um comando FORMAT é executado, a fim de mostrar as opções de formatação para o usuário.

### **DRV\_FORMAT (416Ch)**

**Função:** Formata um disco e inicializa seu setor de boot, FAT e diretório raiz.

**Entrada:** A – Escolha de formatação, de 1 a 9 (ver DRV\_CHOICE).  
 D – Unidade de drive, começando em 0.  
 HL – Endereço da área de trabalho na memória.  
 DE – Tamanho da área de trabalho.

**Saída:** CY – 1 se houver erro..  
 A – Código de erro (apenas em caso de erro):  
     0 – Protegido contra gravação .  
     2 – Não pronto.  
     4 – Erro de CRC.  
     6 – Erro de busca.  
     8 – Registro não encontrado.  
     10 – Falha de escrita.  
     12 – Parâmetro ruim.  
     14 – Memória insuficiente.  
     16 – Outros erros.

#### **DRV\_MTOFF (416Fh)**

**Função:** Parar o motor de todos os drives. Útil apenas para unidades de disquete.

**Entrada:** Nenhuma.

**Saída:** Nenhuma.

#### **9.6.4.2 – Rotinas para drivers de outros dispositivos**

##### **DEV\_RW (4160h)**

**Função:** Ler ou gravar setores absolutos de ou para um dispositivo.

**Entrada:** CY – 0 para leitura; 1 para escrita.  
 A – Índice de dispositivo (1 a 7).  
 B – Número de setores a ler ou escrever.  
 C – Índice de unidade lógica (1 a 7).  
 HL – Endereço para transferência (não pode ser na página 1).  
 DE – Endereço para armazenamento do número do setor de 4 bytes (não pode ser na página 1).

**Saída:** A – Código de erro (mesmos códigos do MSXDOS2):  
 00H: Não houve erro.  
 B5H: Número inválido de dispositivo/unidade lógica.  
 F3H: Erro de busca.  
 F7H: Disco não formatado.  
 F8H: Protegido contra escrita ou unidade só leitura.

- F9H: Setor não encontrado.  
 FAH: Erro de CRC durante a leitura.  
 FCH: Não pronto.  
 FDH: Erro de disco.  
 FEH: Erro de escrita.  
 FFH: Disco incompatível.
- B – Número de setores efetivamente lidos ou escritos (somente em caso de erro).

### **DEV\_INFO (4163h)**

- Função: Retornar informações sobre um dispositivo.
- Entrada: A – Índice do dispositivo (1 a 7)  
 B – 0 → Informações básicas.  
     1 → String contendo o nome do fabricante.  
     2 → String contendo o nome do dispositivo.  
     3 → String contendo o número de série.
- HL – Apontador para buffer (não pode ser na página 1).
- Saída: A – Código de erro:  
 0 → Não houve erro.  
 1 → Dispositivo / informação não disponível ou índice inválido.
- HL – Buffer preenchido com a string de texto ou com informações básicas no seguinte formato:
- +0: Número de unidades lógicas (1 a 8). Deve ser 1 se não houver unidades lógicas.
  - +1: Sinalizadores com os recursos do dispositivo (00H na versão atual).

### **DEV\_STATUS (4166h)**

- Função: Verificar a disponibilidade e alterar o estado de um dispositivo ou unidade lógica.
- Entrada: A – Índice do dispositivo (1 a 7)  
 B – Número da unidade lógica (1 a 7) ou 0 para retornar o estado do próprio dispositivo
- Saída: A – Estado para a unidade lógica especificada ou para o dispositivo se 0 foi especificado:  
 0 – O dispositivo ou unidade lógica não está disponível, ou o número dispositivo / unidade lógica é inválido.  
 1 – O dispositivo ou unidade lógica está disponível e não mudou desde a última verificação de estado.

- 2 – O dispositivo ou unidade lógica está disponível e mudou desde a última verificação de estado (para dispositivos, o dispositivo foi desconectado e outro foi conectado ao qual foi atribuído o mesmo índice; para unidades lógicas, a mídia foi alterada).
- 3 – O dispositivo ou unidade lógica está disponível, mas não é possível determinar se foi alterado ou não desde a última verificação de estado.

### **LUN\_INFO (4169h)**

Função: Obter informações para uma unidade lógica.

Entrada: A – Índice do dispositivo (1 a 7).

B – Índice de unidade lógica (1 a 7).

HL – Apontador para buffer (não pode ser na página 1).

Saída: A – 0 → Ok, buffer preenchido com as informações.

1 → Dispositivo ou unidade lógica não disponível ou inválido.

HL – Buffer de 12 bytes preenchido.

+0: Tipo de mídia:

0 – Bloquear dispositivo.

1 – Leitor ou gravador de CD ou DVD.

2-254 – Não utilizado (reservado para uso futuro).

255 – Outro tipo.

+1: Tamanho do setor em 2 bytes (0 se esta informação não se aplica ou não está disponível).

+3: Total de setores disponíveis em 4 bytes (0 se esta informação não se aplica ou não está disponível).

+7: Sinalizadores da unidade lógica:

bit 0: 1 se a mídia for removível.

bit 1: 1 se a mídia for somente leitura.

bit 2: 1 se a unidade lógica for uma unidade de disquete.

bit 3: 1 se a unidade lógica não deve ser usada para mapeamento automático.

bits 4-7: Não utilizados (sempre 0).

+8: Número de cilindros (2 bytes).

+10: Número de cabeças (1 byte).

+11: Número de setores por trilha (1 byte).

### 9.6.5 – Rotinas para acesso a Hard-Disks padrão SCSI

#### **IDBYT** (7F80H/Interface SCSI)

Função: ID da interface em 3 bytes. (Ex.: “HD#”).

#### **INISYS** (7F83H/Interface SCSI)

Função: Inicia a interface SCSI.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

#### **TRMACT** (7F86H/Interface SCSI)

Função: Termina as ações do HDD.

Entrada: Nenhuma.

Saída: A – Status da interface SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo atual. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: AF, DE.

#### **RDLBLK** (7F89H/Interface SCSI)

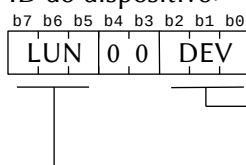
Função: Lê setores lógicos do disco ou dispositivo.

Entrada: CDE – Número do setor.

HL – Endereço na RAM para os dados lidos.

B – Quantidade de setores a ler.

A – ID do dispositivo:



Número do dispositivo SCSI  
(0 a 7, ou 000 a 111).

LUN – Logical Unit Number  
(Normalmente 0).

Saída: HL – apontador para os dados lidos.

A – Status da interface SCSI.

00H – Não houve erro.

02H – Verificar condição.

04H – Condição “MET”.

08H – Dispositivo ocupado.

0CH – Conflito de reserva.

- 10H – Condição intermediária.
- 14H – Condição intermediária “MET”.
- 18H – Conflito de reserva.
- 22H – Comando terminado.
- 28H – Fila cheia.
- 30H – ACA ativa.
- 40H – Operação abortada.

D – Status do dispositivo atual.

- 00H – Não houve erro.
- 02H – Verificar condição.
- 04H – Condição “MET”.
- 08H – Dispositivo ocupado.
- 0CH – Conflito de reserva.
- 10H – Condição intermediária.
- 14H – Condição intermediária “MET”.
- 18H – Conflito de reserva.
- 22H – Comando terminado.
- 28H – Fila cheia.
- 30H – ACA ativa.
- 40H – Operação abortada.

E – Mensagens:

- 00H – Comando completo.
- 01H, xx, 00H – Modificar apontadores dados.
- 01H, xx, 01H – Pedido de transf. síncrona de dados.
- 01H, xx, 03H – Pedido de transf. total de dados.
- 02H – Salvar apontadores de dados.
- 03H – Restaurar apontadores.
- 04H – Desconectar.
- 05H – Erro na inicialização.
- 06H – Abortar.
- 07H – Mensagem rejeitada.
- 08H – Sem operação.
- 09H – Erro de paridade na mensagem.
- 0AH – Comando anexado completo.
- 0BH – Comando anexado completo (com flag).
- 0CH – Reset no barramento do dispositivo.
- 0DH – Abort TAG.

- 0EH – Fila limpa/vazia.
- 0FH – Iniciar recuperação.
- 10H – liberar recuperaração.
- 11H – Encerrar processo de I/O.
- 20H – Tag de fila simples.
- 21H – Tag de cabeçalho de fila
- 22H – Tag de fila ordenada.
- 23H – Ignorar resíduo.
- 80H ~ 0FFH – Identificar.

Registradores: Todos.

Obs.: Esta rotina também pode ler setores do CD-ROM, que têm 2048 bytes em vez de 512 bytes dos HD's.

#### **WRLBLK (7F8CH/Interface SCSI)**

Função: Escreve setores lógicos do disco.

Entrada: CDE – Número do setor.

HL – Endereço inicial dos dados a serem escritos.

B – Quantidade de setores a escrever.

A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: HL – Apontador para os dados lidos.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status do dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: Todos.

#### **RQSENS (7F8FH/Interface SCSI)**

Função: Retorna informações “sense” sobre o dispositivo SCSI.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: A – Código de erro do DOS.

IX – Apontador para um buffer preenchido com os dados “sense”:

+00H – Código de erro:

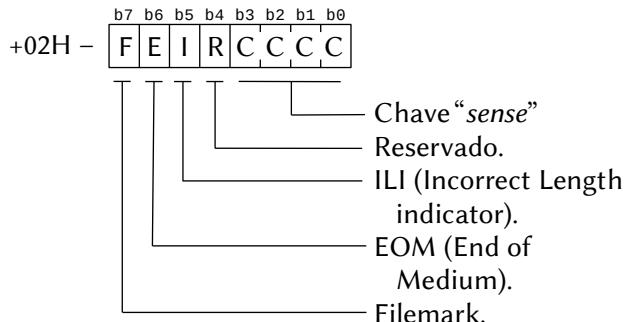
70H – Formato fixo, “sense” atual

71H – Formato fixo, “sense” anterior

72H – Formato do descriptor, “sense” atual

73H – Formato do descriptor, “sense” anterior

+01H – Número do segmento.



Códigos da chave “sense”:

00H – No sense.

01H – Erro recuperado.

02H – Não pronto.

03H – Erro de mídia.

04H – Erro de hardware.

05H – Requisição ilegal.

06H – Unidade requer atenção.

07H – Dados protegidos.

08H – Verificação em branco.

09H – Específico do fabricante.

0AH – Cópia abortada.

0BH – Comando abortado.

0DH – Volume overflow.

0EH – Erro de concordância.

0FH – Completado.

+03H ~ +06H – Informação .

+07H – Comprimento “sense” adicional (n-7).

+08H ~ +11H – Informação específica de comandos.

+12H – Código “sense” adicional.

+13H – Qualificador de código “sense” adicional.

+14H – Código substituível de unidade.

+15H – Bit7 = 0 → Não há informações válidas

1 → Há informações válidas.

+15H ( bit6 ~ bit0 ) ~ +17H – Informações específicas do fabricante.

Registradores: AF, BC, DE.

### **INQIRY (7F92H/Interface SCSI)**

Função: Retorna informações sobre o dispositivo SCSI.

Entrada: HL – Endereço do buffer para as informações lidas.

A – ID do dispositivo.

Saída: CY = 1 → Erro na leitura.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

CY = 0 → HL – Aponta para o início do buffer:

+00H – Código do dispositivo.

+01H – bit7 → RMB (mídia removível).

bit6~bit0 → tipo de dispositivos.

+02H – Versão da interface:

00H – Não especificada

01H – SCSI 1

02H – SCSI 2

+03H – bit7~bit4 → reservados

bit3~bit0 → formato dados resposta

+04H – Comprimento adicional, contém quantos bytes seguintes são válidos.

+05H ~ +07H – Reservados.

+08H ~ +15H – Nome (Ex. SEAGATE).

+16H ~ +31H – ID do dispositivo (em ASCII).

+32H – Revisão do hardware.

+33H – Revisão do firmware.

+34H – Revisão da ROM.

+35H – Reservado.

Registradores: Todos.

### **RDSIZE (7F95H/Interface SCSI)**

Função: Retorna o espaço total do dispositivo SCSI.

Entrada: HL – Endereço do buffer para as informações lidas.

A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: CY = 1 → Erro na leitura.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

CY = 0 → Dados lidos com sucesso.

(HL+0)~(HL+3) → nº total setores (MSB/LSB).

(HL+4)~(HL+7) → tamanho do setor em bytes

(MSB/LSB). Normalmente 512  
(00H-00H-02H-00H).

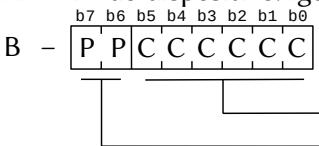
Registradores: Todos.

**MDSENS** (7F98H/Interface SCSI)

Função: Retorna os parâmetros “sense” do modo atual.

Entrada: HL – Endereço do buffer para as informações lidas.

A – ID do dispositivo. Igual a RDLBLK (7F89H).

B – 

Código de página

Campo de controle de

página

Saída: CY = 1 → erro na leitura.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

CY = 0 → HL – Aponta para o início do buffer:

+00H – parâmetros de operação (SEAGATE).

+01H – parâmetros de recuperação de erros.

+02H – parâmetros desconectados.

+03H – parâmetros de formato.

+04H – parâmetros de geometria.

+05H ~ +1FH – Reservados.

+20H – Número de série do drive.

+3FH – Retorna todas as páginas.

Registradores: Todos.

**MDSEL** (7F9BH/Interface SCSI)

Função: Seleção de modo. Usado para inicializar o HD.

Entrada: HL – Endereço do buffer.

A – ID do dispositivo. Igual a RDLBLK (7F89H).

B – Tamanho da lista de parâmetros.

Saída: CY = 1 → erro na leitura.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

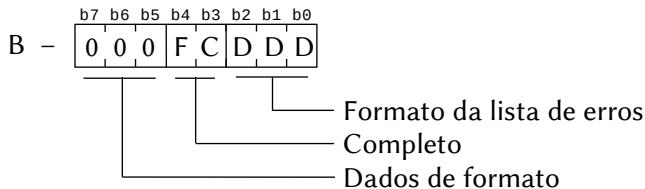
CY = 0 → HL – Aponta para a lista de parâmetros.

Registradores: AF, BC, HL, IX.

**HDFORM** (7F9EH/Interface SCSI)

Função: Formatar a unidade SCSI.

Entrada: A – ID da unidade.



DE – Interleave (MSB-LSB).

HL – Endereço de dados.

Saída: CY = 1 → erro na leitura.

A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

CY = 0 → Formatado com sucesso.

Registradores: AF, BC, DE, HL.

#### **TESTRD** (7FA1H/Interface SCSI)

Função: Testa se o dispositivo SCSI está pronto.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: A – 85H → o dispositivo está pronto.

42H → o dispositivo NÃO está pronto.

Registradores: Todos.

#### **SFBOOT** (7FA4H/Interface SCSI)

Função: Executa “softboot” do dispositivo SCSI.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

Obs.: Esta entrada não deve ser usada.

#### **INSWRK** (7FA7H/Interface SCSI)

Função: Monta a tabela de dispositivos SCSI (instala a área de trabalho).

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

Obs.: Esta entrada não deve ser usada (rotina interna).

#### **CLRLIN** (7FAAH/Interface SCSI)

Função: Limpa até o fim da linha (imprime sequência ESC).

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**VERIFY** (7FADH/Interface SCSI)

Função: Verificação.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

B – Tamanho a ser verificado (em blocos).

CDE – Número do bloco lógico.

HL – Endereço.

Saída: A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: AF, BC, HL, IX.

**STRSTP** (7FB0H/Interface SCSI)

Função: Inicia ou para o drive.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

B – 0 → Para o drive.

1 → Inicia o drive.

Saída: A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: Todos.

**SNDDGN** (7FB3H/Interface SCSI)

Função: Envia diagnósticos.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: Todos.

**RESERV** (7FB6H/Interface SCSI)

Função: Reservado.

**RESER2** (7FB9H/Interface SCSI)

Função: Reservado.

**COPY** (7FBCH/Interface SCSI)

Função: Lê lista “padrão”.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

DE – Comprimento da lista de parâmetros.

HL – Endereço dos dados.

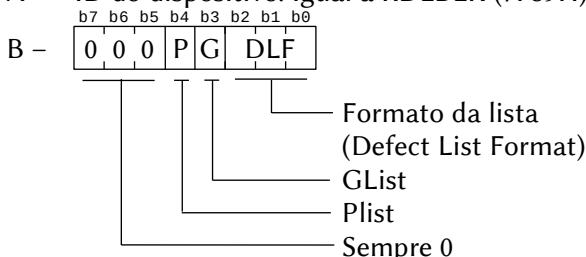
- Saída: A – Status SCSI. Igual a RDLBLK (7F89H).  
 D – Status dispositivo. Igual a RDLBLK (7F89H).  
 E – Mensagens. Igual a RDLBLK (7F89H).

Atenção: Esta entrada tem função diferente em interfaces IDE. Não é aconselhável usar esta chamada.

#### **RDEFCT (7FBFH/Interface SCSI)**

Função: Retorna dados corrompidos.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).



- Saída: A – Status SCSI. Igual a RDLBLK (7F89H).  
 D – Status dispositivo. Igual a RDLBLK (7F89H).  
 E – Mensagens. Igual a RDLBLK (7F89H).

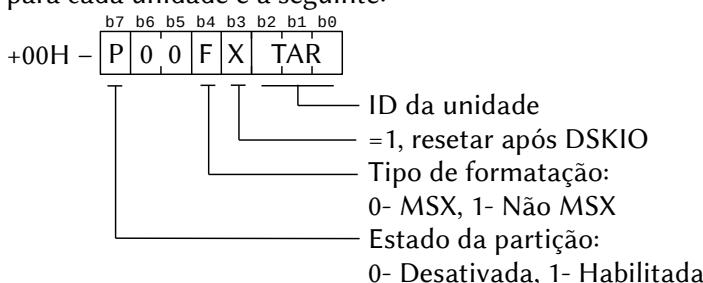
Registradores: Todos.

#### **GETWRK (7FC2H/Interface SCSI)**

Função: Retorna o endereço da área de trabalho.

Entrada: Nenhuma.

Saída: HL = IX = Apontador para o início da área de trabalho.  
 São reservados 8 bytes para cada unidade lógica (podem haver até 6 unidades lógicas, de A: até F:). A estrutura para cada unidade é a seguinte:



+01H ~ +03H	- Primeiro setor da partição.																
+04H ~ +05H	- Número de setores da partição.																
+06H -	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>W</td><td>0</td><td>0</td><td>0</td><td>0</td><td>NNN</td><td colspan="2"></td> </tr> </table> <p style="text-align: center;"> ----- ----- ----- ----- ----- ----- ----- ----- </p> <p>ID de escrita (Network). Proteção contra escrita: 0- Protegido, 1- Permitido.</p>	b7	b6	b5	b4	b3	b2	b1	b0	W	0	0	0	0	NNN		
b7	b6	b5	b4	b3	b2	b1	b0										
W	0	0	0	0	NNN												
+07H -	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>F</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>H</td><td>M</td> </tr> </table> <p style="text-align: center;"> ----- ----- ----- ----- ----- ----- ----- ----- </p> <p>Versão do MSX: 0- MSX2/2+, 1- turbo R. Suporte a múltiplos Hds: 0- Não, 1- Sim. Fast RAM transfer (sem uso atual).</p>	b7	b6	b5	b4	b3	b2	b1	b0	F	0	0	0	0	0	H	M
b7	b6	b5	b4	b3	b2	b1	b0										
F	0	0	0	0	0	H	M										

Registradores: AF, BC, HL, IX.

#### **PRTINF** (7FC5H/Interface SCSI)

Função: Retorna informações sobre a partição.

Entrada: A – Número do drive

Saída: HL = IX = Apontador para o início da área de trabalho do drive especificado. São 8 bytes com estrutura idêntica à GETWRK (7FC2H).

Registradores: AF, BC, DE, HL, IX.

#### **GTUNIT** (7FC8H/Interface SCSI)

Função: Retorna o número de unidades ativas.

Entrada: Nenhuma.

Saída: A – número de unidades ativas.

C – Vector ID.

D – Host ID.

Registradores: AF, BC, DE.

#### **HOSTID** (7FCBH/Interface SCSI)

Função: Seleciona o Host ID.

Entrada: A – Host ID (4 ~ 7)

Saída: CY = 1 se houve erro.

Registradores: AF, D.

**TARGID** (7FCEH/Interface SCSI)

Função: Seleciona o Target ID.

Entrada: A – Target ID (0 ~ 3)

Saída: CY = 1 se houver erro.

Registradores: AF, D.

**GTTARG** (7FD1H/Interface SCSI)

Função: Retorna o Target ID.

Entrada: Nenhuma.

Saída: A – Target ID.

Registradores: AF.

**GTHOST** (7FD4H/Interface SCSI)

Função: Retorna o Host ID.

Entrada: Nenhuma.

Saída: A – Host ID.

Registradores: AF.

**GTSENS** (7FD7H/Interface SCSI)

Função: Retorna dados “sense”.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

Saída: A – Chave “sense”.

C – Código “sense” adicional.

D – Target Status

IX – Endereço dados “sense”. Igual a RQSENS (7F8FH).

Registradores: AF, BC, DE.

**MEDREM** (7FDAH/Interface SCSI)

Função: Prevenir remoção de mídia.

Entrada: A – ID do dispositivo. Igual a RDLBLK (7F89H).

B – 0 → Permite remoção.

1 → Impede remoção.

Saída: A – Status SCSI. Igual a RDLBLK (7F89H).

D – Status dispositivo. Igual a RDLBLK (7F89H).

E – Mensagens. Igual a RDLBLK (7F89H).

Registradores: Todos.

## 9.7 – ROTINAS DA MSX-MUSIC (FM/OPLL)

### **WRTOPL** (4110H/FM-BIOS)

Função: Escreve um byte de dados em um registrador do OPLL.

Entrada: A – Registrador do OPLL.

E – Byte de dados a ser escrito.

Saída: Nenhuma.

Registradores: Nenhum.

### **INIOPL** (4113H/FM-BIOS)

Função: Inicializa a área de trabalho da FM-BIOS/OPLL.

Entrada: HL – Início da área de trabalho (deve ser par).

Saída: Nenhuma.

Registradores: Todos.

### **MSTART** (4116H/FM-BIOS)

Função: Começa a tocar a música.

Entrada: HL – Endereço da fila musical.

A – 0 → Loop infinito.

1~254 → Número de repetições.

255 → Reservado. Não usar.

A fila musical tem a estrutura descrita abaixo.

Cabeçalho para 6 vozes FM + 5 peças de bateria:

+00 ~ +01	0EH, 00H
+02 ~ +03	Endereço para FM1CH
+04 ~ +05	Endereço para FM2CH
+06 ~ +07	Endereço para FM3CH
+08 ~ +09	Endereço para FM4CH
+10 ~ +11	Endereço para FM5CH
+12 ~ +13	Endereço para FM6CH
+14 ...	Área de dados

Cabeçalho para 9 vozes FM:

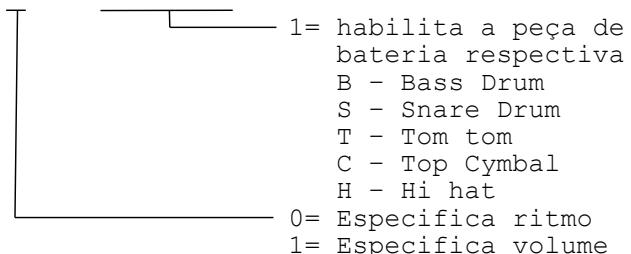
+00 ~ +01	12H, 00H
+02 ~ +03	Endereço para FM1CH
+04 ~ +05	Endereço para FM2CH
+06 ~ +07	Endereço para FM3CH
+08 ~ +09	Endereço para FM4CH
+10 ~ +11	Endereço para FM5CH
+12 ~ +13	Endereço para FM6CH
+14 ~ +15	Endereço para FM7CH
+16 ~ +17	Endereço para FM8CH
+18 ~ +19	Endereço para FM9CH
+20 ...	Área de dados

**Área de dados para a melodia:**

- +00H ~ +5FH Especifica o tom. Este número representa todas as escalas musicais, incluindo o "pitch"
- +60H ~ +6FH Volume
- +70H ~ +7FH Instrumento
- +80H Liberação do "Sustain"
- +81H Manutenção do "Sustain"
- +82H Habilita instrumento da ROM (0 a 63)
- +83H Especificar instrumento do usuário
- +84H Desligar o legato
- +85H Ligar o legato
- +86H Designação Q (1 a 8). Quando o legato estiver ligado, a designação Q não é executada.
- +87H ~ +FEH Não utilizados
- +FFH Final dos dados para cada voz

**Área de dados para o ritmo:**

b7	b6	b5	b4	b3	b2	b1	b0
V	0	1	B	S	T	C	H



FFH → final dos dados de ritmo

**Formato de armazenamento de dados de instrumento:**

+0	AM	VIB	EG TYP	KSR	MÚLTIPLO	
+1	KSL M		NÍVEL TOTAL			
+2	KSL C		XX	DC	DM	REALIMENTAÇÃO
+3	ATTACK RATE			DECAY RATE		
+4						
+5						
+6	SUSTAIN LEVEL			RELEASE RATE		
+7						

Saída: Nenhuma.

Registradores: Todos.

**MSTOP** (4119H/FM-BIOS)

Função: Parar a música.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Todos.

**RDDATA** (411CH/FM-BIOS)

Função: Retorna os dados dos instrumentos da ROM.

Entrada: HL – Endereço do buffer para os dados lidos.

A – Número do instrumento (0 a 63).

Saída: Nenhuma.

Registradores: F.

**OPLDRV** (411FH/FM-BIOS)

Função: Entrada para o driver OPLL. É a rotina que toca a música, devendo ser chamada pelo manipulador de interrupções através do hook HTIMI.

Entrada: Nenhuma.

Saída: Nenhuma.

Registradores: Nenhum.

**TSTBGM** (4122H/FM-BIOS)

Função: Verifica se ainda há dados na fila musical.

Entrada: Nenhuma.

Saída: A = 0 → não há música sendo tocada

A ≠ 0 → há música sendo tocada.

Registradores: AF.

## 10 – MSX-HID (Human Interface Device)

Fórmula para o byte de ID único:

$$\text{HIDID} = (\text{byte1} \ll 4 | 0xF) \& (\text{byte2} | 0xC0) \& (\text{byte1} \ll 2 | 0x3F) \& 0xFF$$

### 10.1 – FINGERPRINTS DE DISPOSITIVOS MSX

Não conectado ou joystick MSX:	3Fh, 3Fh, 3Fh
Mouse:	30h, 30h, 30h
Trackball:	38h, 38h, 38h
Touchpad(1):	39h, 3Dh, 39h
Touchpad(2):	3Dh, 3Dh, 3Dh
Caneta ótica:	2Fh, 2Fh, 2Fh
Paddle Arkanoid Vaus:	3Eh, 3Eh, 3Eh
Dispositivos codificados por tempo: (onde cada bit de "xx" é zero para cada canal analógico)	xxh, 3Fh, 3Fh
Paddle MSX:	3Eh, 3Fh, 3Fh
Pad musical Yamaha MMP-01:	3Ch, 3Fh, 3Fh
Adaptador de joystick IBM-PC DA15:	3Ah, 3Fh, 3Fh
Adaptador de dual-paddle Atari:	36h, 3Fh, 3Fh
Controlador analógico de eixo duplo:	30h, 3Fh, 3Fh

### 10.2 – FINGERPRINTS DE DISPOSITIVOS SEGA COMPATÍVEIS

Joypad Megadrive de 3 botões:	3Fh, 33h, 3Fh
Joypad Megadrive de 6 botões:	3Fh, 33h, 3Fh, 33h, 3Fh, 30h
Megadrive Multi-Tap:	33h, 3Fh, 33h
Joypad digital Saturn:	3Ch, 3Fh, 3Ch
Mouse Saturn:	30h, 3Bh, 30h
Dispositivo Sega 3line-handshake:	31h, 31h, 31h

### 10.3 – FINGERPRINTS DE DISP. QUE PODEM CONFLITAR

Micomsoft XE1-AP em modo analógico:	2Fh, 2Fh, 2Fh
Sega-Mouse (Megadrive):	30h, 30h, 30h

**10.4 – FINGERPRINTS DE DISPOSITIVOS CASEIROS**

Ninja-tap:	3Fh, 1Fh, 3Fh
Óculos 3D:	3Fh, 37h, 3Fh
Óculos 3D + caneta ótica:	2Fh, 27h, 2Fh
Adaptador passivo para mouse PS/2:	3Fh, 3Eh, 3Fh

**10.5 – FINGERPRINTS RESERVADOS (NÃO USAR)**

- Quaisquer fingerprints que possam ser produzidas por um joystick MSX padrão.
- Quaisquer fingerprints que definam o pino 6 e o pino 7 da porta do joystick para 0 simultaneamente nos dois primeiros bytes.

# 11 - MEMÔNICOS Z80/R800

## 11.1 – GRUPO DE CARGA DE 8 BITS

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
LD r, r'	r ← r'	• • • • • •	01 r r'	-- 04	05	01	01	
LD r, n	r ← n	• • • • • •	00 r 110 ← n →	-- 07	08	02	02	
LD u, u'	u ← u'	• • • • • •	11 011 101 01 u u'	DD --	08	10	02	02
LD v, v'	v ← v'	• • • • • •	11 111 101 01 v v'	FD --	08	10	02	02
LD u, n	u ← n	• • • • • •	11 011 101 00 u 110 ← n →	DD --	11	13	03	03
LD v, n	u ← n	• • • • • •	11 111 101 00 v 110 ← n →	FD --	11	13	03	03
LD r, (HL)	r ← (HL)	• • • • • •	01 r 110	-- 07	08	02	04	
LD r, (IX+d)	r ← (IX+d)	• • • • • •	11 011 101 01 r 110 ← d →	DD --	19	21	05	07
LD r, (IY+d)	r ← (IY+d)	• • • • • •	11 111 101 01 r 110 ← d →	FD --	19	21	05	07
LD (HL), r	(HL) ← r	• • • • • •	01 110 r	-- 07	08	02	04	
LD (IX+d), r	(IX+d) ← r	• • • • • •	11 011 101 01 110 r ← d →	DD --	19	21	05	07
LD (IY+d), r	(IY+d) ← r	• • • • • •	11 111 101 01 110 r ← d →	FD --	19	21	05	07
LD (HL), n	(HL) ← n	• • • • • •	00 110 110 ← n →	36 --	10	11	03	05
LD (IX+d), n	(IX+d) ← n	• • • • • •	11 011 101 01 110 110 ← d → ← n →	DD 36 --	19	21	05	07

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
LD (IY+d),n	(IY+d) ← n	• • • • • •	11 111 101 01 110 110 ← d → ← n →	FD 36 -- --	19	21	05	07
LD A,(BC)	A ← (BC)	• • • • • •	00 001 010	0A	07	08	02	04
LD A,(DE)	A ← (DE)	• • • • • •	00 011 010	1A	07	08	02	04
LD A,(nn)	A ← (nn)	• • • • • •	00 111 010 ← n → ← n →	1A -- --	13	14	04	06
LD (BC),A	(BC) ← A	• • • • • •	00 000 010	02	07	08	02	04
LD (DE),A	(DE) ← A	• • • • • •	00 010 010	22	07	08	02	04
LD (nn),A	(nn) ← A	• • • • • •	00 110 010 ← n → ← n →	32 -- --	13	14	04	06
LD A,I	A ← I	• ↑ I ↑ • •	11 101 101 01 010 111	ED 57	09	11	02	02
LD A,R	A ← R	• ↑ I ↑ • •	11 101 101 01 011 111	ED 5F	09	11	02	02
LD I,A	I ← A	• • • • • •	11 101 101 01 000 111	ED 47	09	11	02	02
LD R,A	R ← A	• • • • • •	11 101 101 01 001 111	ED 4F	09	11	02	02

	000	001	010	011	100	101	110	111	TZ - Ciclos T Z80
r,r'	B	C	D	E	H	L	•	A	Z1 - Z80 + M1
u,u'	B	C	D	E	IXH	IXL	•	A	TR - Ciclos T R800
v,v'	B	C	D	E	IYH	IYL	•	A	RW - R800 + Wait

Notação dos sinalizadores:

• = sinalizador não afetado

↑ = sinalizador afetado de acordo com o resultado da operação

I = O conteúdo de IFF (biestável de ativação de interrupções) é copiado no sinalizador P/V.

## 11.2 – GRUPO DE CARGA DE 16 BITS

Memônico	Operação	C Z $\frac{P}{V}$ S N H	Binário	Hex	TZ	Z1	TR	RW
LD dd,nn	dd $\leftarrow$ nn	• • • • • •	00 dd0 001 ← n → ← n →	-- -- --	10	11	03	03
LD IX,nn	IX $\leftarrow$ nn	• • • • • •	11 011 101 00 100 001 ← n → ← n →	DD 21 -- --	14	16	04	04
LD IY,nn	IY $\leftarrow$ nn	• • • • • •	11 111 101 00 100 001 ← n → ← n →	FD 21 -- --	14	16	04	04
LD HL,(nn)	H $\leftarrow$ (nn+1) L $\leftarrow$ (nn)	• • • • • •	00 101 010 ← n → ← n →	2A -- --	16	17	05	07
LD dd,(nn)	dd <sub>H</sub> $\leftarrow$ (nn+1) dd <sub>L</sub> $\leftarrow$ (nn)	• • • • • •	11 101 101 01 dd1 011 ← n → ← n →	ED -- --	20	22	06	08
LD IX,(nn)	IX <sub>H</sub> $\leftarrow$ (nn+1) IX <sub>L</sub> $\leftarrow$ (nn)	• • • • • •	11 011 101 00 101 010 ← n → ← n →	DD 2A -- --	20	22	06	08
LD IY,(nn)	IY <sub>H</sub> $\leftarrow$ (nn+1) IY <sub>L</sub> $\leftarrow$ (nn)	• • • • • •	11 111 101 00 101 010 ← n → ← n →	FD 2A -- --	20	22	06	08
LD (nn),HL	(nn+1) $\leftarrow$ H (nn) $\leftarrow$ L	• • • • • •	00 100 010 ← n → ← n →	22 -- --	16	17	05	07
LD (nn),dd	(nn+1) $\leftarrow$ dd <sub>H</sub> (nn) $\leftarrow$ dd <sub>L</sub>	• • • • • •	11 101 101 01 dd0 011 ← n → ← n →	ED -- --	20	22	06	08
LD (nn),IX	(nn+1) $\leftarrow$ IX <sub>H</sub> (nn) $\leftarrow$ IX <sub>L</sub>	• • • • • •	11 011 101 01 100 010 ← n → ← n →	DD 22 -- --	20	22	06	08

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
LD (nn), IY	(nn+1) ← IY <sub>H</sub> (nn) ← IY <sub>L</sub>	• • • • • •	11 111 101 01 100 010 ← n → ← n →	FD 22 -- --	20	22	06	08
LD SP, HL	SP ← HL	• • • • • •	11 111 001	F9	06	07	01	01
LD SP, IX	SP ← IX	• • • • • •	11 011 101 11 111 001	DD F9	10	12	02	02
LD SP, IY	SP ← IY	• • • • • •	11 111 101 11 111 001	FD F9	10	12	02	02
PUSH qq	(SP-2) ← qq <sub>L</sub> (SP-1) ← qq <sub>H</sub> SP ← SP - 2	• • • • • •	11 qq0 101	--	11	12	04	06
PUSH IX	(SP-2) ← IX <sub>L</sub> (SP-1) ← IX <sub>H</sub> SP ← SP - 2	• • • • • •	11 011 101 11 100 101	DD E5	15	17	05	07
PUSH IY	(SP-2) ← IY <sub>L</sub> (SP-1) ← IY <sub>H</sub> SP ← SP - 2	• • • • • •	11 111 101 11 100 101	FD E5	15	17	05	07
POP qq	qq <sub>H</sub> ← (SP+1) qq <sub>L</sub> ← (SP) SP ← SP + 2	• • • • • •	11 qq0 001	--	10	11	03	05
POP IX	IX <sub>H</sub> ← (SP+1) IX <sub>L</sub> ← (SP) SP ← SP + 2	• • • • • •	11 011 101 11 100 001	DD E1	14	16	04	06
POP IY	IY <sub>H</sub> ← (SP+1) IY <sub>L</sub> ← (SP) SP ← SP + 2	• • • • • •	11 111 101 11 100 001	FD E1	14	16	04	06

	00	01	10	11
dd	BC	DE	HL	SP
qq	BC	DE	HL	AF

TZ - Ciclos T Z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

### 11.3 – GRUPO ARITMÉTICO DE 8 BITS

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
ADD r	$A \leftarrow A + r$	$\downarrow \downarrow v \downarrow 0 \uparrow$	10 000 r	-- 04	05	01	01	
ADD p	$A \leftarrow A + p$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 011 101 10 000 r	DD --	08	10	02	02
ADD q	$A \leftarrow A + q$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 111 101 10 000 r	FD --	08	10	02	02
ADD (HL)	$A \leftarrow A + (HL)$	$\downarrow \downarrow v \downarrow 0 \uparrow$	10 000 110	86	07	08	02	04
ADD (IX+d)	$A \leftarrow A + (IX+d)$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 011 101 10 000 110 $\leftarrow d \rightarrow$	DD 86 --	19	21	05	07
ADD (IY+d)	$A \leftarrow A + (IY+d)$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 111 101 10 000 110 $\leftarrow d \rightarrow$	FD 86 --	19	21	05	07
ADD n	$A \leftarrow A + n$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 000 110 $\leftarrow n \rightarrow$	C6	07	08	02	02
ADC r	$A \leftarrow A+r+CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	10 001 r	-- 04	05	01	01	
ADC p	$A \leftarrow A+p+CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 011 101 10 001 r	DD --	08	10	02	02
ADC q	$A \leftarrow A+q+CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 111 101 10 001 r	FD --	08	10	02	02
ADC (HL)	$A \leftarrow A + (HL) + CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	10 001 110	8E	07	08	02	04
ADC (IX+d)	$A \leftarrow A + (IX+d) + CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 011 101 10 001 110 $\leftarrow d \rightarrow$	DD 8E --	19	21	05	07
ADC (IY+d)	$A \leftarrow A + (IY+d) + CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 111 101 10 001 110 $\leftarrow d \rightarrow$	FD 8E --	19	21	05	07
ADC n	$A \leftarrow A+n+CY$	$\downarrow \downarrow v \downarrow 0 \uparrow$	11 001 110 $\leftarrow n \rightarrow$	CE	07	08	02	02
SUB r	$A \leftarrow A - r$	$\downarrow \downarrow v \downarrow 1 \uparrow$	10 010 r	-- 04	05	01	01	
SUB p	$A \leftarrow A - p$	$\downarrow \downarrow v \downarrow 1 \uparrow$	11 011 101 10 010 r	DD --	08	10	02	02
SUB p	$A \leftarrow A - q$	$\downarrow \downarrow v \downarrow 1 \uparrow$	11 111 101 10 010 r	FD --	08	10	02	02

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
SUB (HL)	A $\leftarrow$ A - (HL)	$\uparrow \downarrow v \downarrow 1 \uparrow$	10 010 110	96	07	08	02	04
SUB (IX+d)	A $\leftarrow$ A - (IX+d)	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 011 101 10 010 110 $\leftarrow d \rightarrow$	DD 96 --	19	21	05	07
SUB (IY+d)	A $\leftarrow$ A - (IY+d)	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 111 101 10 010 110 $\leftarrow d \rightarrow$	FD 96 --	19	21	05	07
SUB n	A $\leftarrow$ A - n	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 010 110 $\leftarrow n \rightarrow$	D6	07	08	02	02
SBC r	A $\leftarrow$ A-r-CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	10 011 r	--	04	05	01	01
SBC p	A $\leftarrow$ A-p-CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 011 101 10 011 r	DD --	08	10	02	02
SBC p	A $\leftarrow$ A-q-CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 111 101 10 011 r	FD --	08	10	02	02
SBC (HL)	A $\leftarrow$ A - (HL) -CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	10 011 110	8E	07	08	02	04
SBC (IX+d)	A $\leftarrow$ A - (IX+d) -CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 011 101 10 011 110 $\leftarrow d \rightarrow$	DD 8E --	19	21	05	07
SBC (IY+d)	A $\leftarrow$ A - (IY+d) -CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 111 101 10 011 110 $\leftarrow d \rightarrow$	FD 8E --	19	21	05	07
SBC n	A $\leftarrow$ A-n-CY	$\uparrow \downarrow v \downarrow 1 \uparrow$	11 011 110 $\leftarrow n \rightarrow$	CE	07	08	02	02
INC r	r $\leftarrow$ r + 1	$\bullet \uparrow v \downarrow 0 \uparrow$	00 r 100	--	04	05	01	01
INC (HL)	(HL) $\leftarrow$ (HL) +1	$\bullet \uparrow v \downarrow 0 \uparrow$	00 110 100	--	11	12	04	07
INC (IX+d)	(IX+d) $\leftarrow$ (IX+d) +1	$\bullet \uparrow v \downarrow 0 \uparrow$	11 011 101 00 110 100 $\leftarrow d \rightarrow$	DD 34 --	23	25	07	10
INC (IY+d)	(IY+d) $\leftarrow$ (IY+d) +1	$\bullet \uparrow v \downarrow 0 \uparrow$	11 111 101 00 110 100 $\leftarrow d \rightarrow$	FD 34 --	23	25	07	10
DEC r	r $\leftarrow$ r - 1	$\bullet \uparrow v \downarrow 1 \uparrow$	00 r 101	--	04	05	01	01
DEC (HL)	(HL) $\leftarrow$ (HL) -1	$\bullet \uparrow v \downarrow 1 \uparrow$	00 110 101	--	11	12	04	07

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
DEC (IX+d)	(IX+d) ← (IX+d) -1	• ↓ v ↓ 1 ↑	11 011 101 00 110 101 ← d →	DD 34 --	23	25	07	10
DEC (IY+d)	(IY+d) ← (IY+d) -1	• ↓ v ↓ 1 ↑	11 111 101 00 110 101 ← d →	FD 34 --	23	25	07	10
MULB r	HL ← A * r	↓ ↓ 0 0 • •	11 101 101 11 r 001	ED --	--	--	14	14

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
p	•	•	•	•	IXH	IXL	•	•
q	•	•	•	•	IYH	IYL	•	•

TZ - Ciclos T Z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

**Notação dos sinalizadores:**

• = sinalizador não afetado

↓ = sinalizador afetado de acordo com o resultado da operação

0 = sinalizador desligado

1 = sinalizador ligado

? = sinalizador desconhecido

V = o sinalizador P/V contém o estouro de capacidade do resultado da operação. V=1 sinaliza estouro de capacidade; V=0 sinaliza que não houve estouro.

## 11.4 – GRUPO ARITMÉTICO DE 16 BITS

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
ADD HL,ss	HL $\leftarrow$ HL + ss	$\downarrow \bullet \bullet \bullet 0 ?$	00 ss1 001	--	11	12	01	01
ADD IX,pp	IX $\leftarrow$ IX + ss	$\downarrow \bullet \bullet \bullet 0 ?$	11 011 101 00 pp1 001	DD --	15	17	02	02
ADD IY,rr	IY $\leftarrow$ IY + ss	$\downarrow \bullet \bullet \bullet 0 ?$	11 111 101 00 rr1 001	FD --	15	17	02	02
ADC HL,SS	HL $\leftarrow$ HL+ss+CY	$\downarrow \downarrow v \downarrow 0 ?$	11 101 101 01 ss1 010	ED --	15	17	02	02
SBC HL,SS	HL $\leftarrow$ HL-ss-CY	$\downarrow \downarrow v \downarrow 1 ?$	11 101 101 01 ss0 010	ED --	15	17	02	02
INC ss	ss $\leftarrow$ ss + 1	$\bullet \bullet \bullet \bullet \bullet$	00 ss0 011	--	06	07	01	01
INC IX	IX $\leftarrow$ IX + ss	$\bullet \bullet \bullet \bullet \bullet$	11 011 101 00 100 011	DD 23	10	12	02	02
INC IY	IY $\leftarrow$ IY + ss	$\bullet \bullet \bullet \bullet \bullet$	11 111 101 00 100 011	FD 23	10	12	02	02
DEC ss	ss $\leftarrow$ ss - 1	$\bullet \bullet \bullet \bullet \bullet$	00 ss1 011	--	06	07	01	01
DEC IX	IX $\leftarrow$ IX - ss	$\bullet \bullet \bullet \bullet \bullet$	11 011 101 00 101 011	DD 2B	10	12	02	02
DEC IY	IY $\leftarrow$ IY - ss	$\bullet \bullet \bullet \bullet \bullet$	11 111 101 00 101 011	FD 2B	10	12	02	02
MULW HL,tt	DE:HL $\leftarrow$ HL * tt	$\downarrow \downarrow 0 0 \bullet \bullet$	11 101 101 11 tt0 011	ED --	--	--	36	36

	00	01	10	11
ss	BC	DE	HL	SP
pp	BC	DE	IX	SP
rr	BC	DE	IY	SP
tt	BC	•	•	SP

TZ – Ciclos T Z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

Notação dos sinalizadores:

• = sinalizador não afetado

$\downarrow$  = sinalizador afetado de acordo com o resultado da operação

0 = sinalizador desligado

1 = sinalizador ligado

? = sinalizador desconhecido

V = o sinalizador P/V contém o estouro de capacidade do resultado da operação. V=1 sinaliza estouro de capacidade; V=0 sinaliza que não houve estouro.

## 11.5 – GRUPO DE TROCA

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
EX DE,HL	DE $\leftrightarrow$ HL	• • • • • •	11 101 011	EB	04	05	01	01
EX AF,AF'	AF $\leftrightarrow$ AF'	• • • • • •	00 001 000	08	04	05	01	01
EXX	BC $\leftrightarrow$ BC' DE $\leftrightarrow$ DE' HL $\leftrightarrow$ HL'	• • • • • •	11 011 001	D9	04	05	01	01
EX (SP),HL	H $\leftrightarrow$ (SP+1) L $\leftrightarrow$ (SP)	• • • • • •	11 100 011	E3	19	20	05	07
EX (SP),IX	IX <sub>H</sub> $\leftrightarrow$ (SP+1) IX <sub>L</sub> $\leftrightarrow$ (SP)	• • • • • •	11 011 101 11 100 011	DD E3	23	25	06	08
EX (SP),IY	IY <sub>H</sub> $\leftrightarrow$ (SP+1) IY <sub>L</sub> $\leftrightarrow$ (SP)	• • • • • •	11 111 101 11 100 011	FD E3	23	25	06	08

TZ – Ciclos T Z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

Notação dos sinalizadores:  
 • = sinalizador não afetado

## 11.6 – GRUPO DE TRANSFERÊNCIA DE BLOCO

Memônico	Operação	C Z <sup>P/V</sup> S N H	Binário	Hex	TZ	Z1	TR	RW
LDI	(DE) $\leftarrow$ (HL) DE $\leftarrow$ DE+1 HL $\leftarrow$ HL+1 BC $\leftarrow$ BC-1	• • $\downarrow$ • 0 0	11 101 101 10 100 000	ED A0	16	18	04	07
LDIR	(DE) $\leftarrow$ (HL) DE $\leftarrow$ DE+1 HL $\leftarrow$ HL+1 BC $\leftarrow$ BC-1 {Até BC=0}	• • 0 • 0 0	11 101 101 10 110 000	ED B0	21	23	04	?
LDD	(DE) $\leftarrow$ (HL) DE $\leftarrow$ DE-1 HL $\leftarrow$ HL-1 BC $\leftarrow$ BC-1	• • $\downarrow$ • 0 0	11 101 101 10 101 000	ED A8	16	18	04	07
LDLR	(DE) $\leftarrow$ (HL) DE $\leftarrow$ DE-1 HL $\leftarrow$ HL-1 BC $\leftarrow$ BC-1 {Até BC=0}	• • 0 • 0 0	11 101 101 10 111 000	ED B8	21	23	04	?

TZ – Ciclos T z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

**Notação dos sinalizadores:**

• = sinalizador não afetado

$\downarrow$  = sinalizador afetado de acordo com o resultado da operação

0 = sinalizador desligado

**OBS.** Quando houver duas descrições de ciclos, elas referem às duas condições que a instrução pode assumir. Assim, para LDIR, o tempo em ciclos T para o Z80 é 21; quando BC atinge 0, são gastos 16 ciclos T.

## 11.7 – GRUPO DE PESQUISAS

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
CPI	A - (HL) HL $\leftarrow$ HL+1 BC $\leftarrow$ BC-1	• $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$ 1 $\uparrow$	11 101 101 10 100 001	ED A1	16	18	04	06
CPIR	A - (HL) HL $\leftarrow$ HL+1 BC $\leftarrow$ BC-1 {Até BC=0 ou A=(HL) }	• $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$ 1 $\uparrow$	11 101 101 10 110 001	ED B1	21	23	05	?
CPD	A - (HL) HL $\leftarrow$ HL-1 BC $\leftarrow$ BC-1	• $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$ 1 $\uparrow$	11 101 101 10 101 001	ED A9	16	18	04	06
CPDR	A - (HL) HL $\leftarrow$ HL-1 BC $\leftarrow$ BC-1 {Até BC=0 ou A=(HL) }	• $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$ 1 $\uparrow$	11 101 101 10 111 001	ED B9	21	23	05	?

TZ – Ciclos T Z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

**Notação dos sinalizadores:**

• = sinalizador não afetado

$\uparrow$  = sinalizador afetado de acordo com o resultado da operação

1 = sinalizador ligado

OBS. Quando houver duas descrições de ciclos, elas referem às duas condições que a instrução pode assumir. Assim, para CPIR, o tempo em ciclos T para o Z80 é 21; quando BC atinge 0, são gastos 16 ciclos T.

## 11.8 – GRUPO DE COMPARAÇÃO

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
CP A,r	A - R	↓ ↓ v ↓ 1 ↓	10 111 r	-- 04	05	01	01	
CP A,p	A - p	↓ ↓ v ↓ 1 ↓	11 011 101 10 111 p	DD --	08	10	02	02
CP A,q	A - q	↓ ↓ v ↓ 1 ↓	11 111 101 10 111 p	FD --	08	10	02	02
CP A,(HL)	A - (HL)	↓ ↓ v ↓ 1 ↓	10 111 110	BE	07	08	02	04
CP A,(IX+d)	A - (IX+d)	↓ ↓ v ↓ 1 ↓	11 011 101 10 111 110 ← d →	DD BE --	19	21	05	07
CP A,(IY+d)	A - (IY+d)	↓ ↓ v ↓ 1 ↓	11 111 101 10 111 110 ← d →	FD BE --	19	21	05	07
CP A,n	A - n	↓ ↓ v ↓ 1 ↓	11 111 110 ← n →	FE --	07	08	02	02

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
p	•	•	•	•	IXH	IXL	•	•
q	•	•	•	•	IYH	IYL	•	•

TZ - Ciclos T z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

Notação dos sinalizadores:

↓ = sinalizador afetado de acordo com o resultado da operação

1 = sinalizador ligado

V = o sinalizador P/V contém o estouro de capacidade do resultado da operação. V=1 sinaliza estouro de capacidade; V=0 sinaliza que não houve estouro.

## 11.9 – GRUPO LÓGICO

Memônico	Operação	C Z $\frac{P}{V}$ S N H	Binário	Hex	TZ	Z1	TR	RW
AND r	$A \leftarrow A \wedge r$	0 $\uparrow$ P $\uparrow$ 0 1	10 100 r	--	04	05	01	01
AND p	$A \leftarrow A \wedge p$	0 $\uparrow$ P $\uparrow$ 0 1	11 011 101 10 100 p	DD --	08	10	02	02
AND q	$A \leftarrow A \wedge q$	0 $\uparrow$ P $\uparrow$ 0 1	11 111 101 10 100 p	FD --	08	10	02	02
AND (HL)	$A \leftarrow A \wedge (HL)$	0 $\uparrow$ P $\uparrow$ 0 1	10 100 110	A6	07	08	02	04
AND (IX+d)	$A \leftarrow A \wedge (IX+d)$	0 $\uparrow$ P $\uparrow$ 0 1	11 011 101 10 100 110 $\leftarrow d \rightarrow$	DD A6 --	19	21	05	07
AND (IY+d)	$A \leftarrow A \wedge (IY+d)$	0 $\uparrow$ P $\uparrow$ 0 1	11 111 101 10 100 110 $\leftarrow d \rightarrow$	FD A6 --	19	21	05	07
AND n	$A \leftarrow A \wedge n$	0 $\uparrow$ P $\uparrow$ 0 1	11 100 110 $\leftarrow n \rightarrow$	E6 --	07	08	02	02
OR r	$A \leftarrow A \vee r$	0 $\uparrow$ P $\uparrow$ 0 1	10 110 r	--	04	05	01	01
OR p	$A \leftarrow A \vee p$	0 $\uparrow$ P $\uparrow$ 0 1	11 011 101 10 110 p	DD --	08	10	02	02
OR q	$A \leftarrow A \vee q$	0 $\uparrow$ P $\uparrow$ 0 1	11 111 101 10 110 p	FD --	08	10	02	02
OR (HL)	$A \leftarrow A \vee (HL)$	0 $\uparrow$ P $\uparrow$ 0 1	10 110 110	B6	07	08	02	04
OR (IX+d)	$A \leftarrow A \vee (IX+d)$	0 $\uparrow$ P $\uparrow$ 0 1	11 011 101 10 110 110 $\leftarrow d \rightarrow$	DD B6 --	19	21	05	07
OR (IY+d)	$A \leftarrow A \vee (IY+d)$	0 $\uparrow$ P $\uparrow$ 0 1	11 111 101 10 110 110 $\leftarrow d \rightarrow$	FD B6 --	19	21	05	07
OR n	$A \leftarrow A \vee n$	0 $\uparrow$ P $\uparrow$ 0 1	11 110 110 $\leftarrow n \rightarrow$	F6 --	07	08	02	02
XOR r	$A \leftarrow A \oplus r$	0 $\uparrow$ P $\uparrow$ 0 1	10 110 r	--	04	05	01	01
XOR p	$A \leftarrow A \oplus p$	0 $\uparrow$ P $\uparrow$ 0 1	11 011 101 10 110 p	DD --	08	10	02	02
XOR q	$A \leftarrow A \oplus q$	0 $\uparrow$ P $\uparrow$ 0 1	11 111 101 10 110 p	FD --	08	10	02	02
XOR (HL)	$A \oplus (HL)$	0 $\uparrow$ P $\uparrow$ 0 1	10 110 110	B6	07	08	02	04

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
XOR (IX+d)	A $\leftarrow$ A $\oplus$ (IX+d)	0 $\uparrow$ P $\uparrow$ 0 1 $\leftarrow$ d $\rightarrow$	11 011 101 10 110 110 $\leftarrow$ d $\rightarrow$	DD B6 --	19	21	05	07
XOR (IY+d)	A $\leftarrow$ A $\oplus$ (IY+d)	0 $\uparrow$ P $\uparrow$ 0 1 $\leftarrow$ d $\rightarrow$	11 111 101 10 110 110 $\leftarrow$ d $\rightarrow$	FD B6 --	19	21	05	07
XOR n	A $\leftarrow$ A $\oplus$ n	0 $\uparrow$ P $\uparrow$ 0 1 $\leftarrow$ n $\rightarrow$	11 110 110 $\leftarrow$ n $\rightarrow$	F6 --	07	08	02	02

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
p	•	•	•	•	IXH	IXL	•	•
q	•	•	•	•	IYH	IYL	•	•

TZ - Ciclos T Z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

**Notação dos sinalizadores:**

↑ = sinalizador afetado de acordo com o resultado da operação

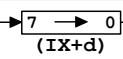
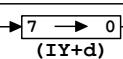
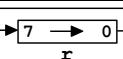
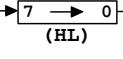
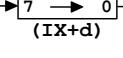
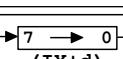
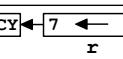
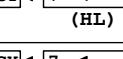
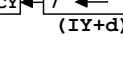
0 = sinalizador desligado

1 = sinalizador ligado

P = o sinalizador P/V contém a paridade. P=1 significa que a paridade do resultado é par; P=0 significa que é ímpar.

## 11.10 – GRUPO DE DESLOCAMENTO E ROTAÇÃO

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
RLCA		↓ • • • 0 0	00 000 111	07	04	05	01	01
RLA		↓ • • • 0 0	00 010 111	0F	04	05	01	01
RRCA		↑ • • • 0 0	00 001 111	17	04	05	01	01
RRA		↑ • • • 0 0	00 011 111	1F	04	05	01	01
RLC r		↑↑ P ↓ 0 0	11 001 011 00 000 r	CB --	08	10	02	02
RLC (HL)		↑↑ P ↓ 0 0	11 001 011 00 000 110	CB 06	15	17	05	08
RLC (IX+d)		↑↑ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 000 110	DD CB -- 06	23	25	07	10
RLC (IY+d)		↑↑ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 000 110	FD CB -- 06	23	25	07	10
RL r		↑↑ P ↓ 0 0	11 001 011 00 010 r	CB --	08	10	02	02
RL (HL)		↑↑ P ↓ 0 0	11 001 011 00 010 110	CB 16	15	17	05	08
RL (IX+d)		↑↑ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 010 110	DD CB -- 16	23	25	07	10
RL (IY+d)		↑↑ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 010 110	FD CB -- 16	23	25	07	10
RRC r		↑↑ P ↓ 0 0	11 001 011 00 001 r	CB --	08	10	02	02

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
RRC (HL)		↓ ↓ P ↓ 0 0	11 001 011 00 001 110	CB OE	15	17	05	08
RRC (IX+d)		↓ ↓ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 001 110	DD CB -- OE	23	25	07	10
RRC (IY+d)		↓ ↓ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 001 110	FD CB -- OE	23	25	07	10
RR r		↓ ↓ P ↓ 0 0	11 001 011 00 011 r	CB --	08	10	02	02
RR (HL)		↓ ↓ P ↓ 0 0	11 001 011 00 011 110	CB 1E	15	17	05	08
RR (IX+d)		↓ ↓ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 011 110	DD CB -- 1E	23	25	07	10
RR (IY+d)		↓ ↓ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 011 110	FD CB -- 1E	23	25	07	10
SLA r		↓ ↓ P ↓ 0 0	11 001 011 00 100 r	CB --	08	10	02	02
SLA (HL)		↓ ↓ P ↓ 0 0	11 001 011 00 100 110	CB 26	15	17	05	08
SLA (IX+d)		↓ ↓ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 100 110	DD CB -- 26	23	25	07	10
SLA (IY+d)		↓ ↓ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 100 110	FD CB -- 26	23	25	07	10
SRA r		↓ ↓ P ↓ 0 0	11 001 011 00 101 r	CB --	08	10	02	02

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
SRA (HL)		↓ ↓ P ↓ 0 0	11 001 011 00 101 110	CB 2E	15	17	05	08
SRA (IX+d)		↓ ↓ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 101 110	DD CB -- 2E	23	25	07	10
SRA (IY+d)		↓ ↓ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 101 110	FD CB -- 2E	23	25	07	10
SRL r	0 →	↓ ↓ P ↓ 0 0	11 001 011 00 111 r	CB --	08	10	02	02
SRL (HL)	0 →	↓ ↓ P ↓ 0 0	11 001 011 00 111 110	CB 3E	15	17	05	08
SRL (IX+d)	0 →	↓ ↓ P ↓ 0 0	11 011 101 11 001 011 ← d → 00 111 110	DD CB -- 3E	23	25	07	10
SRL (IY+d)	0 →	↓ ↓ P ↓ 0 0	11 111 101 11 001 011 ← d → 00 111 110	FD CB -- 3E	23	25	07	10
RLD		• ↓ P ↓ 0 0	11 101 101 01 101 111	ED 6F	18	20	05	08
RRD		• ↓ P ↓ 0 0	11 101 101 01 100 111	ED 67	18	20	05	08

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A

TZ - Ciclos T z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

**Notação dos sinalizadores:**

• = sinalizador não afetado

↓ = sinalizador afetado de acordo com o resultado da operação

0 = sinalizador desligado

P = o sinalizador P/V contém a paridade. P=1 significa que a paridade do resultado é par; P=0 significa que é ímpar.

## 11.11 – GRUPO DE LIGAR, DESLIGAR E TESTAR BITS

Memônico	Operação	C Z $\frac{P}{V}$ S N H	Binário	Hex	TZ	Z1	TR	RW
BIT b,r	$Z \leftarrow \overline{x_b}$	0 $\downarrow$ ? ? 0 1	11 001 011 01 b r	CB --	08	10	02	02
BIT b,(HL)	$Z \leftarrow \overline{(HL)_b}$	0 $\downarrow$ ? ? 0 1	11 001 011 01 b 110	CB --	12	04	03	05
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	0 $\downarrow$ ? ? 0 1	11 011 101 11 001 011 $\leftarrow$ d $\rightarrow$ 01 b 110	DD CB -- --	20	22	05	07
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	0 $\downarrow$ ? ? 0 1	11 111 101 11 001 011 $\leftarrow$ d $\rightarrow$ 01 b 110	FD CB -- --	20	22	05	07
SET b,r	$\overline{x_b} \leftarrow 1$	• • • • •	11 001 011 11 b r	CB --	08	10	02	02
SET b,(HL)	$\overline{(HL)_b} \leftarrow 1$	• • • • •	11 001 011 11 b 110	CB --	15	17	05	08
SET b,(IX+d)	$\overline{(IX+d)_b} \leftarrow 1$	• • • • •	11 011 101 11 001 011 $\leftarrow$ d $\rightarrow$ 11 b 110	DD CB -- --	23	25	07	10
SET b,(IY+d)	$\overline{(IY+d)_b} \leftarrow 1$	• • • • •	11 111 101 11 001 011 $\leftarrow$ d $\rightarrow$ 11 b 110	FD CB -- --	23	25	07	10
RES b,r	$\overline{x_b} \leftarrow 0$	• • • • •	11 001 011 10 b r	CB --	08	10	02	02
RES b,(HL)	$\overline{(HL)_b} \leftarrow 0$	• • • • •	11 001 011 10 b 110	CB --	15	17	05	08
RES b,(IX+d)	$\overline{(IX+d)_b} \leftarrow 0$	• • • • •	11 011 101 11 001 011 $\leftarrow$ d $\rightarrow$ 10 b 110	DD CB -- --	23	25	07	10

Memônico	Operação	C Z $\frac{P}{V}$ S N H	Binário	Hex	TZ	Z1	TR	RW
RES b, (IY+d)	$(IY+d)_b \leftarrow 0$	• • • • •	11 111 101 11 001 011 ← d → 10 b 110	FD CB -- --	23	25	07	10

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
b	b0	b1	b2	b3	b4	b5	b6	b7

TZ - Ciclos T Z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

**Notação dos sinalizadores:**

• = sinalizador não afetado

↓ = sinalizador afetado de acordo com o resultado da operação

0 = sinalizador desligado

1 = sinalizador ligado

? = sinalizador desconhecido

## 11.12 – GRUPO DE SALTO

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
JP nn	PC ← nn	• • • • • •	10 000 011 ← n → ← n →	C3 -- --	10	11	03	05
JP cc,nn	Se cc=verd, PC ← nn	• • • • • •	10 cc 011 ← n → ← n →	-- -- --	10	11	03	03
JR e	PC ← PC+e	• • • • • •	00 011 000 ← e-2 →	18 --	12	13	03	03
JR C,e	Se CY=1, PC ← PC+e	• • • • • •	00 111 000 ← e-2 →	38 --	07	08	02	02
JR NC,e	Se CY=0, PC ← PC+e	• • • • • •	00 110 000 ← e-2 →	30 --	07	08	02	02
JR Z,e	Se Z=1, PC ← PC+e	• • • • • •	00 101 000 ← e-2 →	28 --	07	08	02	02
JR NZ,e	Se Z=0, PC ← PC+e	• • • • • •	00 100 000 ← e-2 →	20 --	12	13	03	03
JP (HL)	PC ← HL	• • • • • •	11 101 001	E9	04	05	01	03
JP (IX)	PC ← IX	• • • • • •	11 011 101 11 101 001	DD E9	08	10	02	04
JP (IY)	PC ← IY	• • • • • •	11 111 101 11 101 001	FD E9	08	10	02	04
DJNZ e	B ← B-1 Se B0, PC ← PC+e	• • • • • •	00 010 000 ← e-2 →	10 --	08	09	02	02
	000 001 010 011 100 101 110 111				TZ - Ciclos T	Z80		
cc	NZ Z NC C PO PE P M				Z1 -	Z80 + M1		
					TR - Ciclos T	R800		
					RW -	R800 + Wait		

Notação dos sinalizadores:

• = sinalizador não afetado

OBS. Quando houver duas descrições de ciclos, elas referem às duas condições que a instrução pode assumir. Assim, para JR C,e, o tempo em ciclos T para o Z80 é 7 quando a condição é falsa e 12 quando é verdadeira.

## 11.13 – GRUPO DE CHAMADA E RETORNO

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC $\leftarrow nn$	• • • • • •	11 001 101 ← n → ← n →	CD -- --	17	18	05	08 07
CALL cc,nn	Se cc=verd, $(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC $\leftarrow nn$	• • • • • •	11 cc 100 ← n → ← n →	CD -- --	10	11	03	07 03
						17	18	05 08
RET	$PC_H \leftarrow (SP+1)$ $PC_L \leftarrow (SP)$	• • • • • •	11 101 001	C9	10	11	03	05
RET cc	Se cc=verd, $PC_H \leftarrow (SP+1)$ $PC_L \leftarrow (SP)$	• • • • • •	11 cc 000	--	05	06	01	01
					11	12	03	05
RETI	Retorna da interrupção	• • • • • •	11 101 101 01 001 101	ED 4D	14	16	05	07
RETN	Retorna da interrupção não masca- ráv.	• • • • • •	11 101 101 01 000 101	ED 45	14	16	05	07
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow t^*8$	• • • • • •	11 t 111	--	11	12	04	06 07

	000	001	010	011	100	101	110	111
cc	NZ	Z	NC	C	PO	PE	P	M
p	00H	08H	10H	18H	20H	28H	30H	38H

TZ – Ciclos T z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

Notação dos sinalizadores:

• = sinalizador não afetado

OBS. Quando houver duas descrições de ciclos, elas referem às duas condições que a instrução pode assumir. Assim, para CALL cc,nn, o tempo em ciclos T para o Z80 é 10 quando a condição é falsa e 17 quando é verdadeira.

OBS1. Os testes mostraram que um CALL seguido por uma série de NOPs leva 8 ciclos, enquanto se for seguido por um RET ou POP AF combinados levam 12 ciclos (7 para CALL + 5 para RET/POP AF). Isso também se aplica ao RST (observação aplicável apenas ao valor RW destacado para o R800).

## 11.14 – GRUPO DE ENTRADA E SAÍDA

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
IN A, (n)	A $\leftarrow$ (n)	• • • • • •	11 011 011 $\leftarrow$ n $\rightarrow$	28 --	11	12	03	10 09
IN r, (C)	r $\leftarrow$ (C)	• $\uparrow$ P $\uparrow$ 0 $\uparrow$	11 101 101 01 r 000	ED --	12	14	03	10 09
INI	(HL) $\leftarrow$ (C) B $\leftarrow$ B-1 HL $\leftarrow$ HL+1	• $\uparrow$ ? ? 1 ?	11 101 101 10 100 010	ED A2	16	18	04	12 11
INIR	(HL) $\leftarrow$ (C) B $\leftarrow$ B-1 HL $\leftarrow$ HL+1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 110 010	ED B2	21	23	04	? 11
IND	(HL) $\leftarrow$ (C) B $\leftarrow$ B-1 HL $\leftarrow$ HL-1	• $\uparrow$ ? ? 1 ?	11 101 101 10 101 010	ED AA	16	18	04	12 11
INDR	(HL) $\leftarrow$ (C) B $\leftarrow$ B-1 HL $\leftarrow$ HL-1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 111 010	ED BA	21	23	04	? 11
OUT (n), A	(n) $\leftarrow$ A	• • • • • •	11 010 111 $\leftarrow$ n $\rightarrow$	D3 --	11	03	03	10 9
OUT (C), r	(C) $\leftarrow$ r	• • • • • •	11 101 101 01 r 001	ED --	11	12	03	10 9
OUTI	(C) $\leftarrow$ (HL) B $\leftarrow$ B-1 HL $\leftarrow$ HL+1	• $\uparrow$ ? ? 1 ?	11 101 101 10 100 011	ED A3	16	18	04	12 11
OTIR	(C) $\leftarrow$ (HL) B $\leftarrow$ B-1 HL $\leftarrow$ HL+1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 110 011	ED B3	21	23	04	? 11
OUTD	(C) $\leftarrow$ (HL) B $\leftarrow$ B-1 HL $\leftarrow$ HL-1	• $\uparrow$ ? ? 1 ?	11 101 101 10 101 011	ED AB	16	18	04	12 11

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
OTDR	(C) $\leftarrow$ (HL) B $\leftarrow$ B-1 HL $\leftarrow$ HL-1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 111 011	ED BB	21	23	04	?

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	F	A

TZ - Ciclos T Z80

Z1 - Z80 + M1

TR - Ciclos T R800

RW - R800 + Wait

**Notação dos sinalizadores:**

- = sinalizador não afetado
- ↓ = sinalizador afetado de acordo com o resultado da operação
- 0 = sinalizador desligado
- 1 = sinalizador ligado
- ? = sinalizador desconhecido
- P = o sinalizador P/V contém a paridade. P=1 significa que a paridade do resultado é par; P=0 significa que é ímpar.

OBS. Nas instruçõesINI, IND, OUTI e OUTD o sinalizador Z é ligado quando B-1=0; caso contrário é desligado.

OBS1. Para as instruções IN A,(n) e OUT (n),A, n vai para A0~A7 e A vai para A8~A15. Nas outras instruções, C vai para A0~A7 e B vai para A8~A15.

OBS2. As instruções de E/S são alinhadas ao clock do barramento, portanto, uma espera extra é inserida dependendo do alinhamento. Isso significa que, entre dois OUTs, pode haver uma redução de um ciclo (observação aplicável apenas ao valor RW destacado para o R800).

## 11.15 – GRUPO DE CONTROLE E DE PROPÓSITO GERAL

Memônico	Operação	C Z P/V S N H	Binário	Hex	TZ	Z1	TR	RW
CCF	$CY \leftarrow \overline{CY}$	1 • • • 0 ?	00 111 111	3F	04	05	01	01
CPL	$A \leftarrow \bar{A}$	• • • • 1 1	00 101 111	2F	04	05	01	01
DAA	Converte A para BCD	↑ ↓ P ↑ • ↑	00 100 111	27	04	05	01	01
DI	$IFF \leftarrow 0$	• • • • •	11 110 011	F3	04	05	02	02
EI	$IFF \leftarrow 1$	• • • • •	11 111 011	FB	04	05	01	01
HALT	CPU parada	• • • • •	01 110 110	76	04	05	02	02
IM 0	Modo 0 de interrupção	• • • • •	11 101 101 01 000 110	ED 46	08	10	03	03
IM 1	Modo 1 de interrupção	• • • • •	11 101 101 01 010 110	ED 56	08	10	03	03
IM 2	Modo 2 de interrupção	• • • • •	11 101 101 01 011 110	ED 5E	08	10	03	03
NEG	$A \leftarrow 0 - A$	↑ ↓ V ↑ 1 ↑	00 101 101 01 000 100	ED 44	08	10	02	02
NOP	Nenhuma ação	• • • • •	00 000 000	00	04	05	01	01
SCF	$CY \leftarrow 1$	1 • • • 0 0	00 110 111	37	04	05	01	01

TZ – Ciclos T Z80

Z1 – Z80 + M1

TR – Ciclos T R800

RW – R800 + Wait

**Notação dos sinalizadores:**

- = sinalizador não afetado

- ↓ = sinalizador afetado de acordo com o resultado da operação

- 0 = sinalizador desligado

- 1 = sinalizador ligado

- ? = sinalizador desconhecido

- V = o sinalizador P/V contém o estouro de capacidade do resultado da operação. V=1 sinaliza estouro de capacidade; V=0 sinaliza que não houve estouro.

- P = o sinalizador P/V contém a paridade. P=1 significa que a paridade do resultado é par; P=0 significa que é ímpar.

OBS. IFF indica o circuito biestável de ativação das interrupções.  
CY indica o circuito biestável de transporte.

## 12 - MAPAS DE REGISTRADORES DE CHIPS PADRÃO

### 12.1 – MAPA DOS REGISTRADORES DOS V9918/38/58

Regist.		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida				
R#0	W	•	•	•	•	•	•	m3	EV	Regist. de modo #1 (9918)				
		b7~b2	Não usados (sempre 00 000)											
		b1	m3: Modos de tela (em conjunto com R#1)											
		b0	EV: 0=desabilita entrada externa; 1=habilita											
		•	DG	IE2	IE1	m5~m3		•	Reg. de modo #1 (9938/58)					
		b7	Não usado (sempre 0)											
		b6	DG: 0=normal; 1=color bus como entrada											
		b5	IE2: Interrupção caneta ótica (eliminada no 9958)											
		b4	IE1: 0=ativa interrupção de linha #1; 1=desliga											
		b3~b1	M5~M3: Modos de tela (em conjunto com R#1)											
		b0	Não usado (sempre 0)											
R#1		W	16K	BL	IE0	m2~m1	BC	SI	MA	Registrador de modo #2				
					b7	(9918) → 0=4027(4K x 1-bit); 1=4108(8K x 1-bit)/4116(16K x 1-bit)								
						(9938/58) → Sem uso (sempre 0)								
					b6	BL: 0=tela desligada; 1=tela habilitada								
					b5	IE0: 9918: 0=ativa interrupção; 1=desliga interrup. 9938/58: 0=ativa interrup. linha #0; 1=desliga								
					b4~b3	M2~M1: Modos de tela (em conjunto com R#0)								
						M5 M4 M3 M2 M1 b4 b3 → (de R#25/9958)								
						0	0	0	1	0	0	0	Screen 0 Wth 40	
						0	1	0	1	0	0	0	Screen 0 Wth 80	
						0	0	0	0	0	0	0	Screen 1	
						0	0	1	0	0	0	0	Screen 2	
						0	0	0	0	1	0	0	Screen 3	
						0	1	0	0	0	0	0	Screen 4	
						0	1	1	0	0	0	0	Screen 5	
						1	0	0	0	0	0	0	Screen 6	
						1	0	1	0	0	0	0	Screen 7	
						1	1	1	0	0	0	0	Screen 8	
						1	1	1	0	0	1	1	Screen 10/11	
						1	1	1	0	0	0	1	Screen 12	

		b2	FAFCH = <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>•</td><td>•</td><td>•</td><td>x</td><td>x</td><td>•</td><td>•</td><td>•</td></tr></table>	•	•	•	x	x	•	•	•	xx=10 → Screen 10 (MODE)	xx=11 → Screen 11
•	•	•	x	x	•	•	•						
		b1		Bit Cadari									
		b0		SI: 0=sprites 8x8; 1=sprites 16x16									
				MA: 0=sprites normais; 1=sprites expandidos									
R#2	W	0	a16	a15	a14	a13	a12	a11	a10	End. Tab. Nomes dos Padrões			
R#3	W	a13	a12	a11	a10	a9	a8	a7	a6	End. Tab. Cores dos Padrões			
R#4	W	0	0	a16	a15	a14	a13	a12	a11	End. Tab. Geradora Padrões			
R#5	W	a14	a13	a12	a11	a10	a9	a8	a7	End. Tab. Atributos Sprites			
R#6	W	0	0	a16	a15	a14	a13	a12	a11	End. Tab. Geral. Padr. Sprites			
R#7	W	f3~f0			b3~b0			f3~f0 – Cor de frente (0~15) b3~b0 – Cor de fundo (0~15)					

### Registradores adicionados para o V9938

R#8	W	MS	LP	TP	CB	VR	•	SPD	BW	Registrador de modo #3
		b7								MS: 0=color bus saída; 1=color bus entrada
		b6								LP: 0=desab. caneta ótica; 1=habilita (elim. 9958)
		b5								TP: 0=cor 0 transparente; 1=cor 0 pode ser definida
		b4								CB: 0=color bus saída; 1=color bus entrada
		b3								VR: VRAM 0=16K/1bit, 16K/4bits; 1=64K/1bits, 64K/4bits
		b2								Não usado (sempre 0)
		b1								SPD: 0=liga sprites; 1=desliga sprites
		b0								BW: 0=saida colorida; 1=tons de cinza
R#9	W	LN	•	S1~S0	IL	EO	NT	DC		Registrador de modo #4
		b7								LN: 0=192 linhas; 1=212 linhas
		b6								Não usado (sempre 0)
		b5~b4								S1-S0: 00 – Sincronização interna 01 – Sincronização mixada 10 – Sincronização externa (digitalização) 11 – Sem sincronização
		b3								IL: 0=modo normal; 1=modo entrelaçado
		b2								EO: 0=uma tela; 1=duas telas simultâneas
		b1								NT: 0=NTSC (262 linhas); 1=PAL (313 linhas)
		b0								DC: 0=DLCLK saída; 1=DLCLK entrada

R#10	W	0	0	0	0	0	a16	a15	a14	End. Tab. Cores dos Padrões
R#11	W	0	0	0	0	0	0	a16	a15	End. Tab. Atributos Sprites
R#12	W	f3~f0				b3~b0			f3~f0 – Cor frente p/ blink b3~b0 – Cor fundo p/ blink	
R#13	W	e3~e0 unidade: 1/6 segundo				o3~o0 unidade: 1/6 segundo			Tempo de “blink” R#7/R#12 e3~e0 – página par (even) o3~o0 – página ímpar (odd)	
R#14	W	0	0	0	0	0	a16	a15	a14	Endereço base da VRAM
R#15	W	•	•	•	•	0 ~ 9			Apontador p/ reg. estado	
R#16	W	•	•	•	•	0 ~ 15			Apontador p/ reg. paleta	
R#17	W	AI	•	R#0 a R#46					Controle de ponteiro	
R#18	W	Vert -8 a +7				Hor -8 a +7			Ajuste de imagem na tela	
R#19	W	Número de linha (0 a 255)							Reg. Interrupção de linha	
R#20	W	0	0	0	0	0	0	0	0	Burst de cor #1
R#21	W	0	0	1	1	1	0	1	1	Burst de cor #2
R#22	W	0	0	0	0	0	1	0	1	Burst de cor #3
R#23	W	Número de linha (0 a 255)							Ajuste/scroll vertical	
R#24	W	----							Este registrador não existe	

## Registradores adicionados para o V9958

R#25	W	•	CMD	VDS	YAE	YJK	WTE	MSK	SP	Registrador de modo #5
		b7								Não usado (sempre 0)
		b6								0=comandos só nos modos gráficos 5 a 7 1=comandos ativos em todos os modos
		b5								VDS: 0=saida CPUCLK; 1=saida VDS
		b4								YAE: 0=só YJK; 1=YJK+RGB
		b3								YJK: 0=modo RGB; 1=modo YJK
		b2								WTE: 0=função espera desligada; 1=espera ativa
		b1								MSK: 0=máscara scroll desligada; 1=máscara ligada
		b0								SP: 0=scroll horizontal c/ 1 página; 1=scroll c/ 2 pgs



			0 1 0 0 POINT Lê código de cor de um ponto 0 1 0 1 PSET Desenha ponto e avança coords 0 1 1 0 SRCH Procura código de cor de um ponto 0 1 1 1 LINE Desenha uma linha (Pontos) 1 0 0 0 LMMV Desenha um retângulo (Pontos) 1 0 0 1 LMMM Transf. VRAM → VRAM (Pontos) 1 0 1 0 LMCM Transf. VRAM → CPU (Pontos) 1 0 1 1 LMMC Transf. CPU → VRAM (Pontos) 1 1 0 0 HMMV Desenha um retângulo (bytes) 1 1 0 1 HMMM Transf. VRAM → VRAM (bytes) 1 1 1 0 YMMM Transf. VRAM → VRAM direção Y 1 1 1 1 HMMC Transfere CPU → VRAM (bytes)
		b3	Cor transparente: 0=faz operação lógica 1=não faz operação lógica
	b2~b0	Logopr	000 IMP DC = SC 001 AND DC = SC and DC 010 OR DC = SC or DC 011 XOR DC = SC xor DC 100 NOT DC – Not (SC) 101~111 Não implementados

### Registrador de estado (TMS9918)

S#0	R	FLG	5S	C	5º sprite (0~31)	Registrador de estado
		b7	FLG: Flag de interrupção vertical			
		b6	5S: 0=normal; 1=mais de 4 (ou 8) sprites mesma linha			
		b5	C: 0=normal; 1=dois sprites colidindo			
		b4~b0	número do 5º (ou 9º) sprite			

### Registradores de estado (V9938 e V9958)

S#1	R	LP	KEY	ID number	HI	Caneta ótica / ID / Int.Hor.				
		b7	LPF: 0=LP normal; 1=LP detect. luz (elim. V9958)							
		b6	LPK: 0=chave LP não pressionada; 1=pressionada							
		b5~b1	Número ID do MSX-video							
		b0	FH: 0=interrupção horizontal desabilitada; 1=ligada							
S#2	R	TR	VR	HR	BD	•	•	EO	CE	Estado dos comandos
		b7	TR: 0=VDP não pronto p/ dados; 1=VDP pronto							
		b6	VR: 0=quadro não está sendo varrido; 1=está sendo							
		b5	HR: 0=linha não está sendo varrida; 1=está sendo							

		b4 b3~b2 b1 b0	BD: 0=SRCH não encontrou; 1=SRCH bem sucedido Sem uso (sempre 11) EO: 0=1 <sup>a</sup> tela apresentada; 1=2 <sup>a</sup> tela apresentada CE: 0=VDP livre; 1=VDP executando comando									
S#3	R	x7	x6	x5	x4	x3	x2	x1	x0	Coordenada X+12 (colisão de sprites)		
S#4	R	•	•	•	•	•	•	•	x8			
S#5	R	y7	y6	y5	y4	y3	y2	y1	y0	Coordenada Y+8 (colisão de sprites)		
S#6	R	•	•	•	•	•	•	•	y9			
S#7	R	Cor (0~3; 0~15; 0~255)								Cor do ponto especificado		
S#8	R	x7	x6	x5	x4	x3	x2	x1	x0	Coordenada horizontal do ponto (comando SRCH)		
S#9	R	•	•	•	•	•	•	•	•			

### 12.1.1 – Portas de acesso aos VDPs 9918/9938/9958

Porta			b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida				
P#0	98H	R/W	Byte de dados					Escreve/lê dados VRAM							
P#1	99H	R	FLG	5S	C	5º sprite (0~31)					Lê registrador de estado				
			b7	FLG: Flag de interrupção vertical					5S: 0=normal; 1=mais de 4 (ou 8) sprites mesma linha						
P#1	99H	W	b6	C: 0=normal; 1=dois sprites colidindo					número do 5º (ou 9º) sprite						
			b5												
			b4~b0												
				Endereços a7~a0					Seleciona end. VRAM f: 0=leitura; 1=escrita						
P#2	9AH	W	0	f	Endereços a13~a8										
					Byte de dados					Escreve reg. controle (9918) Seleciona reg. (9938/58)					
			1	0	Nº reg. (0~46)										
P#3	9BH	W	0	r	r	r	0	b	b	b	Escreve nos registradores de paleta				
			0	0	0	0	0	g	g	g					
P#3	9BH	W	Byte de dados								Escreve reg. indireto				

### 12.1.2 – Tabela de cores padrão

Nº paleta	Cor	Nível de vermelho	Nível de azul	Nível de verde
0	Transparente	0	0	0
1	Preto	0	0	0
2	Verde	1	1	6
3	Verde claro	3	3	7
4	Azul escuro	1	7	1
5	Azul	2	7	3
6	Vermelho escuro	5	1	1
7	Azul claro	2	7	6
8	Vermelho	7	1	1
9	Vermelho claro	7	3	3
10	Amarelo	6	1	6
11	Amarelo claro	6	3	6
12	Verde escuro	1	1	4
13	Roxo	6	5	2
14	Cinza	5	5	5
15	Branco	7	7	7

## 12.2 – MAPA DOS REGISTRADORES DO V9990

Reg		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida			
R#0	W	a7	a6	a5	a4	a3	a2	a1	a0	Endereço de escrita na VRAM AI = 0 → autoincremento			
R#1	W	a15	a14	a13	a12	a11	a10	a9	a8				
R#2	W	AI	•	•	•	•	a18	a17	a16				
R#3	W	a7	a6	a5	a4	a3	a2	a1	a0	Endereço de leitura da VRAM			
R#4	W	a15	a14	a13	a12	a11	a10	a9	a8				
R#5	W	AI	•	•	•	•	a18	a17	a16	AI = 0 → autoincremento			
R#6	R/W	DSPM	DCKM	XIMM	CLRM	Modo de tela							
		b7~b6	XIMM	00-Y=256pixels 01-Y=512pixels	10-Y=1024pixels 11-Y=2048pixels								
		b5~b4	DSPM	00-modo P1 01-modo P2	10-Bit Map 11-Stand-by								
		b3~b2	DCKM	00-XTAL=1/4 01-XTAL=1/2	10-XTAL=1/1 11 – N/A								
		b1~b0	CLRM	00-2bits/pixel 01-4bits/pixel	10-8bits/pixel 11-16bits/pixel								
R#7	R/W	•   CM	S1	S2	PL	EO	IL	HS	Modo de tela				
		b7	Não usado (sempre 0)										
		b6	C25M	0-outros modos	1-modo B6								
		b5	SM1	0-262 linhas	1-263 linhas								
		b4	SM	0-1H=fsc/228	1-1H=fsc/227.5								
		b3	PAL	0-NTSC	1-PAL								
		b2	EO	0-Y=normal	1-Y=dobrado								
		b1	IL	0-não entrelaç.	1-entrelaçado								
		b0	HSCN	0-outros modos	1-modos B5-B6								
R#8	R/W	DP   SP	YS	VE	VM	DM	V1~V0	Controle de sistema					
		b7	DISP	0=só cor de fundo, 1=apresent. de tela normal									
		b6	SPD	0=Apresenta sprite/cursor, 1=Não apresenta									
		b5	YSE	0=Saída sinal YS desabilitada, 1=Habilitada									
		b4	VWTE	0=VRAM serial data bus em saída; 1=entrada									

		b3 b2 b1~b0	VWM Escrita na VRAM digitalização: 0=Não, 1=Durante retraço horizontal DMAE 0=Saída DREQ em nível alto, 1=Habilitada VSL1/0 00=64K x 4-bit, 128K total 01=128K x 8-bit, 256K total 10=256K x 4-bit, 512K total			
R#9	R/W	• •	• • • IE IH IV	Controle interrupções		
		b7~b3 b2 b1 b0	Não usados (sempre "00 000") IECE 0=interrupções desabilitadas, 1-habilitadas IEH 0=interrup. de linha desabilitada, 1-habilitada IEV 0=interrup. de quadro desabilitada, 1-ativa			
R#10	R/W	I7 I6	I5 I4 I3 I2 I1 I0	Linha para interrup. (I9~I0) IEHM – 0=linha específica 1=todas as linhas		
R#11	R/W	IE •	• • • • I9 I8			
R#12	R/W	• •	• • ix3 ix2 ix1 ix0	Posição hor. p/ interrupção (IX) × (64) × (master clock)		
R#13	W	PLTM	IE AI	PLTO5~2	Controle de paleta	
		b7~b6 b5 b4 b3~b0	PLTM - 00=paleta; 01=256 cores; 10=YJK; 11=YUV YAE - 0=Somente YJK/YUV; 1=YJK/YUV + RGB PLTAIH – Autoinc de leitura paleta (0=Sim, 1=Não) PLTO5-2 – Offset da paleta (0 a 15)			
R#14	W	PLTA5~0			PL2~1	Controle de paleta
		b7~b2 b1~b0	PLTA – Número da cor na paleta (0 a 63) PLTP – 00-Vermelho; 01-Verde; 10-Azul; 11-N/C			
R#15	R/W	• •	b5 b4 b3 b2 b1 b0	Cor de fundo		
R#16	R/W	ADJV (-8 a +7)	ADJH (-8 a +7)	Ajuste de tela		
R#17	R/W	SCAY (b7~b0)				Controle de scroll
R#18	R/W	ROLL	•	SCAY (b12~b8)	Coord. "Y" plano A	
		b7~b0 b4~b0 b5 b7~b6	SCAY – Coordenada Y de início de exibição para o plano "A" do modo P1 e das telas dos outros modos Não usado (sempre "0") ROLL – Rolagem dir. Y: 00-tela inteira 10-512 lin. 01-256 linhas 11-N/C			

R#19	R/W	•	•	•	•	•	SX (b2~b0)	Controle de scroll Coord. "X" plano A e B0~B7
R#20	R/W	SCAX (b10~b3)						
R#21	R/W	SCBY (b7~b0)						Controle de scroll Coord. "Y" plano B
R#22	R/W	A	B	•	•	•	•	b8
		b7~b0	SCBY – Coordenada Y de início de exibição para plano "B" modo P1 (b8~b0)					
		b0	SDB – Se 1, desabilita plano "A" e sprites					
		b7	SDB – Se 1, desabilita plano "B" e sprites					
		b6						
		b5~b1	Não usados (sempre "00 000")					
R#23	R/W	•	•	•	•	•	BX(b2~b0)	SCBX -Coord X de início de exibição plano "B"
R#24	R/W	•	•	SCBX (b8~b3)				
R#25	R/W	•	•	•	•	a17	a16	a15
		•	•	•	•	a18	a17	a16
		•	•	•	•	a15		
R#26	R/W	•	•	•	VR	PNS	PL	PD
				PNE				
		b7~b5	Controle de painel LCD					
		b4	VRI	0=igual CRT	1=uma linha retraço vert.			
		b3	PNSL	0=400 pontos vert.	1=480 pontos vert.			
		b2	PLVO	0=binário tons de cinza (terminais D3~0)				
				1=colorido (terminais CB7~0)				
		b1	PDUAL	0=uma tela	1=duas telas			
		b0	PNEN	0=ciclo CRT	1=ciclo LCD			
R#27	R/W	•	•	•	•	PRY	PRX	Prioridade modo P1
		b7~b4	Não usados (sempre "0000")					
		b3~b2	PRX – 00 – X=256	10 – X=128				
			01 – X=64	11 – X=192				
		b1~b0	PRY – 00 – Y=256	10 – Y=128				
			01 – Y=64	11 – Y=192				
R#28	W	•	•	•	•	CSPO (b5~b2)	Offset paleta cursor	
–	–	Os registradores R#29 a R#31 não existem						

R#32	W	SX, SA, KA (b7~b0)						Comandos VDP – FONTE Coordenadas XY (SX, SY) Endereço linear (SA) Endereço Kanji-ROM (KA)			
R#33	W	•	•	•	•	•	SX(b10~b8)				
R#34	W	SY (b7~b0); SA, KA (b15~b8)									
R#35	W	•	•	•	•	SY (b7~b0)					
		•	•	•	•	•	SA(b18~b16)				
		•	•	•	•	•	•	K(b17~16)			
R#36	W	DX, DA (b7~b0)						Comandos VDP – DEST Coordenadas XY (DX, DY) Endereço linear (DA)			
R#37	W	•	•	•	•	•	DX(b10~b8)				
R#38	W	DY (b7~b0); DA (b15~b8)									
R#39	W	•	•	•	•	DY (b7~b0)					
		•	•	•	•	•	DA(b18~b16)				
R#40	W	NX, NA, MJ (b7~b0)						Comandos VDP (DIVERSOS)			
R#41	W	•	•	•	•	MJ (b11~b8)					
		•	•	•	•	•	NX(b10~b8)				
R#42	W	NY, MI (b7~b0); NA (b15~b8)						Número de pixels a transferir XY (NX, NY) Nº bytes a transferir (NA) Lado maior linha (MJ) Lado menor linha (MI)			
R#43	W	•	•	•	•	NY,MI(b11~b8)					
		•	•	•	•	•	NA(b18~b16)				
R#44	W	•	•	•	•	DIY DIX NEQ MAJ	Reg. argumento (só escrita)				
		b7~b4	Não usados (sempre “0000”)								
		b3	DIY: 0 – para baixo; 1 – para cima								
		b2	DIX: 0 – para a direita; 1 – para a esquerda Obs. BMXL e BMLX são fixados em “+” e para BMLL, X e Y são fixados na mesma direção.								
		b1	NEQ (término SRCH): 0=cor espec. 1=outra cor								
		b0	MAJ (Lado maior LINE): 0=horizontal 1=vertical								
R#45	W	•	•	•	T	Logopr	Operação lógica				
		b7~b5	Sempre “000”								
		b4	Cor transparente: 0=faz operação lógica 1=não faz operação lógica								

		b3~b0	Logopr	0 0 0 1 WC – Not (SC .or. DC) 0 0 1 1 WC – Not (SC) 0 1 1 0 WC = SC .xor. DC 0 1 1 1 WC – Not (SC .and. DC) 1 0 0 0 WC = SC .and. DC 1 0 0 1 WC – Not (SC .xor. DC) 1 1 0 0 WC = SC 1 1 1 0 WC = SC .or. DC						
R#46	W	m7	m6	m5	m4	m3	m2	m1	m0	Máscara de escrita
R#47	W	m15	m14	m13	m12	m11	m10	m9	m8	Bit=1 → leitura permitida Bit=0 → não permitida
No modo P1, R#46 é usado p/ plano A e R#47 p/ plano B										
R#48	W	f7	f6	f5	f4	f3	f2	f1	f0	Cor de frente
R#49	W	f15	f14	f13	f12	f11	f10	f9	f8	Cor de fundo
R#50	W	b7	b6	b5	b4	b3	b2	b1	b0	
R#51	W	b15	b14	b13	b12	b11	b10	b9	b8	
R#52	W	OP-CODE		AY		AX		Controle de Operação		
		b7~b4	Código de comando (OP-CODE) 0 0 0 0 STOP Comando de parada 0 0 0 1 LMMC Transf. CPU → VRAM (coord) 0 0 1 0 LMMV Pinta retângulo na VRAM 0 0 1 1 LMCM Transf. VRAM → CPU (coord) 0 1 0 0 LMMM Transf. VRAM → VRAM (coord) 0 1 0 1 CMMC Transf. de caractere CPU → VRAM 0 1 1 0 CMMK Transf. Kanji ROM → VRAM 0 1 1 1 CMMI Transf. de caract. VRAM → VRAM 1 0 0 0 BMXL VRAM → VRAM (linear → coord) 1 0 0 1 BMLX VRAM → VRAM (coord → linear) 1 0 1 0 BMLL VRAM → VRAM (linear → linear) 1 0 1 1 LINE Desenha uma linha 1 1 0 0 SRCH Procura código de cor de um ponto 1 1 0 1 POINT Lê código de cor de um ponto 1 1 1 0 PSET Desenha um ponto e avança coord. 1 1 1 1 ADVN Avança coordenadas sem desenhar							

		b3~b2	AY:	00-sem deslocamento	10-desloc. para baixo					
		b1~b0	AX:	01-sem deslocamento	11-desloc. para cima					
R#53	R	x7	x6	x5	x4	x3	x2	x1	x0	Coordenada horizontal do ponto (SRCH bem sucedido)
R#54	R	•	•	•	•	•	x10	x9	x8	

### **12.2.1 – Portas de acesso ao V9990**



### 12.3 – MAPA DOS REGISTRADORES DO PSG (AY-3-8910)

Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
R#0	7	6	5	4	3	2	1	0	Frequência da voz A
R#1	.	.	.	.	11	10	9	8	111860,87 / f_num (b11~b0)
R#2	7	6	5	4	3	2	1	0	Frequência da voz B
R#3	.	.	.	.	11	10	9	8	111860,87 / f_num (b11~b0)
R#4	7	6	5	4	3	2	1	0	Frequência da voz C
R#5	.	.	.	.	11	10	9	8	111860,87 / f_num (b11~b0)
R#6	.	.	.	4	3	2	1	0	Frequência do ruído branco 111860,87 / f_num (b11~b0)
R#7	ioB	ioA	rC	rB	rA	tC	tB	tA	habilita/desabilita sons
	b0~b2	Habilita/desabilita tons (0=habilita)							
	b5~b4	Habilita/desabilita ruído branco (0=habilita)							
	b6	Configura porta A de I/O (0=in, 1=out)							
	b7	Configura porta B de I/O (0=in, 1=out)							
R#8	.	.	.	m	v	v	v	v	Volume da voz A
R#9	.	.	.	m	v	v	v	v	Volume da voz B
R#10	.	.	.	m	v	v	v	v	Volume da voz C
	b7~b5	Não utilizados (sempre “000”)							
	b4	0=não usa a envoltória; 1=usa a envoltória							
	b3~b0	0000=volume mínimo; 1111=volume máximo							
R#11	7	6	5	4	3	2	1	0	Frequência da envoltória
R#12	15	14	13	12	11	10	9	8	6983,3 / f_num (b15~b0)
R#13	.	.	.	.	e	e	e	e	Forma da envoltória
	b7~b4	Não usados (sempre “0000”)							
	b3~b0	Define a forma da envoltória							
	00xx								
	01xx								
	1000								
	1001								
	1010								
R#14	a7	a6	a5	a4	a3	a2	a1	a0	Envia/recebe porta A de I/O
R#15	b7	b6	b5	b4	b3	b2	b1	b0	Envia/recebe porta B de I/O

**12.3.1 – Portas de acesso ao PSG**

Porta		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida		
A0H	W	•	•	•	•	nº reg. (0 a 15)		Seleciona registrador				
A1H	W	Byte de dados						Escreve dados no PSG				
A2H	R	Byte de dados						Lê dados do PSG				

## 12.4 – MAPA DOS REGISTRADORES DO FM-OPLL (YM2413)

\$0FH	b7	•	b5	•	b3	b2	b1	b0	Registrador de teste do OPLL
	b7		1=Reseta os LFOs (amplitude máx. / desvio fase zero)						
	b5		1=Seleciona forma de onda						
	b3		1=Atualiza o LFO a cada amostra (a frequência do trémolo é multiplicada por 64 e do vibrato por 1024)						
	b2		1=Mantém mas coloca a forma de onda em “0”.						
	b1		1=Mantém mas coloca a fase do LFO em “0”.						
	b0		1=Coloca os geradores de envoltória (modulador e portador) em volume total.						
\$10H : \$18H	Frequência LSB (8 bits)						Registradores usados para a seleção de frequências do gerador de tons		
\$20H : \$28H	•	•	Sustain	Key	Oitava	Freq.	Frequência MSB 1 bit Oitava Key / Sustain on/off		
	b7~b6		Não usados (sempre “00”)						
	b5		0=Sem “sustain”; 1=Release Rate decairá gradativam.						
	b4		0=Voz respectiva desligada (key off); 1=voz ativa						
	b3~b1		Define a oitava. A quarta é 011.						
	b0		Frequência MSB 1 bit. A Nota Lá central de 440 Hz é obtida com b0=1 e \$10H~18H=00 100 000						
\$30H : \$38H	Instrumentos			Volume			Registradores usados para seleção de instrumentos e de volume		
	b7~b4		Definição de instrumento: 0000 – A ser definido      1000 – Órgão 0001 – Violino              1001 – Piston 0010 – Violão              1010 – Sintetizador 0011 – Piano              1011 – Cravo 0100 – Flauta              1100 – Vibrafone 0101 – Clarinete              1101 – Baixo elétrico 0110 – Oboé              1110 – Baixo acústico 0111 – Trompete              1111 – Guitarra elétrica b3-b0              Volume (0000=mínimo; 1111=máximo)						

Mapa dos registradores para o modo bateria (\$0EH, b5=1)										
\$36H	•	•	•	•	BD volume		Registradores de volume para as peças de bateria			
\$37H	HH volume		SD volume							
\$38H	TOM volume		TCY volume							

#### 12.4.1 – Portas de acesso ao OPLL

Porta		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
7CH	W	Nº do registrador (00H a 38H)						Seleciona registrador		
7DH	W	Byte de dados						Escreve dados no OPLL		

## 12.5 – MAPA DOS REGISTRADORES DO MSX-AUDIO (Y8950)

Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
\$01H	b7	b6	•	•	b3	b2	b1	b0	Registrador de teste
	b7		1=Reseta os LFOs (amplitude máx. / desvio fase zero)						
	b6		1=Afeta o bit 0 (PCMBSY) do registrador de estado.						
	b3		1=Atualiza o LFO a cada amostra (a frequência do trêmolo é multiplicada por 64 e do vibrato por 1024)						
	b2		1=Mantém mas coloca a forma de onda em “0”.						
	b1		1=Mantém mas coloca a fase do LFO em “0”.						
	b0		1=Coloca os geradores de envoltória (modulador e portador) em volume total.						
\$02H			1º Temporizador (80 µS)						Registradores de tempo
\$03H			2º Temporizador (320 µS)						
\$04H	IRQ	T1M	T2M	EOS	BR	•	ST2	ST1	Registrador de sinalizadores
	b7		IRQ – Se colocado em 1, reseta todas as flags.						
	b6		T1M – Se colocado em 1, b0 será colocado em 0.						
	b5		T2M – Se colocado em 1, b1 será colocado em 0.						
	b4		EOS – Máscara p/ b3, indicando fim da operação atual						
	b3		BR – Máscara para ADPCM / Mem. Áudio (1=ativada)						
	b2		Não usado (sempre 0)						
	b1		ST2 – Controla início/parada de \$03 (1=inicia cont.)						
	b0		ST1 – Controla início/parada de \$02 (1=inicia cont.)						
\$05H			Teclado externo (entrada)						Registradores para acesso ao teclado externo
\$06H			Teclado externo (saída)						
\$07H	STA	REC	MEM	REP	OFF	•	•	RST	Registrador de controle (1)
	b7		STA – Deve ser 1 para iniciar leitura/gravação de dados						
	b6		REC – Deve ser 1 para gravação de dados na memória						
	b5		MEM – Deve ser 1 ao acessar a memória de áudio						
	b4		REP – Quando 1, habilita repetição de dados ADPCM						
	b3		OFF – Quando 1, desliga a saída de áudio						
	b1~b2		Não usados (sempre “00”)						
	b0		RST – Quando 1, coloca o ADPCM no estado inicial						

\$08H	CSM	SEL	•	•	SAM	DAD	64K	ROM	Registrador de controle (2)	
	b7		CSM – 1=modo de modulação senoidal composta							
	b6		SEL – Ponto de separação oitavas p/ teclado externo							
	b5~b4		Não usados (sempre "00")							
	b3		SAM – 0=inicia conversão DA; 1=inicia conversão AD							
	b2		DAD – 0=conv. AD / saída música; 1=\$15~\$16 → saída							
	b1		64K – Tamanho da memória: 0=256K; 1=64K							
	b0		ROM – Tipo de memória: 0=RAM; 1=ROM							
\$09H	Endereço inicial (b7~b0)									
\$0AH	Endereço inicial (b15~b8)									
\$0BH	Endereço final (b7~b0)									
\$0CH	Endereço final (b15~b8)									
\$0DH	f7	f6	f5	f4	f3	f2	f1	f0	Frequência para o ADPCM 3580 / F_num (1,8 ~ 16 KHz)	
\$0EH	•	•	•	•	•	f10	f9	f8		
\$0FH	Dados para o ADPCM								Registrador de dados	
\$10H	i7	i6	i5	i4	i3	i2	i1	i0	Fator de interpolação ADPCM (i15~i0) = 1310,72 * taxa amostr.	
\$11H	i15	i14	i13	i12	i11	i10	i9	i8		
\$12H	Volume do ADPCM								Volume do ADPCM (0~255)	
\$15H	f9	f8	f7	f6	f5	f4	f3	f2	Dados para conversão DA Out: $V_{cc}/2 + V_{cc}/4^*(-1+f9+f8*2^{-1}+...+f1*2^{-8}+f0*2^{-9}+2-10)*2^{-E}$ $E=S2*4+S1*2+S0*1$ ( $S0+S1+S2>0$ )	
&16H	f1	f0	•	•	•	•	•	•		
\$17H	•	•	•	•	•	S2	S1	S0	Controle das portas I/O (\$18H → 0=entrada; 1=saida)	
\$18H	•	•	•	•	Controle I/O					
\$19H	•	•	•	•	Dados I/O					
\$1AH	Dados para o ADPCM								Registrador de dados	
\$20H	AM	VIB	EGT	KSR	Múltiplo				Definição de instrumentos	
⋮	b7				AM (1=liga trêmolo – Variação de amplitude 3,7Hz)					
\$35H	b6				VIB (1=liga vibrato – Variação de frequência 6,4Hz)					
	b5				EG-TYP (0=tom percussivo; 1=tom constante)					
	b4				Se 0, KSR→0~3; Se 1, KSR→0~15					
	b3~b0				Fator de multiplicação (0=1/2, 1=1, 2=2, 3=3, ..., 15=15)					

\$40H : \$55H	KSL	Nível total			KSL (00= 0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB) Nível total (b0=0,75dB, b1=1,5dB ..... b5=24dB)				
\$60H : \$75H	Attack Rate (AR)		Decay Rate (DR)		Attack (0dB a 96dB → mín. 0,2 mS; máx 2826 mS) Decay (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)				
\$80H : \$95H	Sustain Level (SL)		Release Rate (RL)		Sustain (b7=24dB, b6=12dB, b5=6dB, b4=3dB) Release (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)				
\$A0H : \$A8H	Frequência (LSB 8 bits)			Frequência FM (b7~b0)					
\$B0H : \$B8H	•	•	KEY	Oitava	Freq. MSB 2 bits	Frequência FM (b9~b8) Oitava (FM) Key on/off (FM)			
	b7~b6	Não usados (sempre “00”)							
	b5	0=Voz respectiva desligada (key off); 1=voz ativa							
	b4~b2	Define a oitava. A quarta é 011.							
	b1~b0	Frequência MSB 2 bits. A Nota Lá central de 440 Hz é obtida com b1~b0=10 e \$A0H~A8H=01000 001							
	Operadores (para \$20H~\$35H e \$A0H~\$A8H)								
	Oper: 01 02 03 04 05 06 07 08 09 Voz: 1 2 3 1 2 3 4 5 6 Reg: \$20 \$21 \$22 \$23 \$24 \$25 \$28 \$29 \$2A Freq: \$A0 \$A1 \$A2 \$A0 \$A1 \$A2 \$A3 \$A4 \$A5								
	Oper: 10 11 12 13 14 15 16 17 18 Voz: 4 5 6 7 8 9 7 8 9 Reg: \$2B \$2C \$2D \$30 \$31 \$32 \$33 \$34 \$35 Freq: \$A3 \$A4 \$A5 \$A6 \$A7 \$A8 \$A6 \$A7 \$A8								
\$BDH	AM	VIB	BAT	BD	SD TOM TCY HH Controle da bateria do FM				

Os operadores são associados da seguinte forma:  
 \$20/\$40/\$60/\$80/\$A0/\$B0/\$C0  
 ou  
 \$23/\$43/\$63/\$83/\$A0/\$B0/\$C0

	b7	Grau do trêmolo (0=1dB. 1=4,8dB)						
	b6	Grau do vibrato (0=7%; 1=14%)						
	b5	0=Modo Melodia; 1=Modo Bateria						
	b4	1=Bass Drum						
	b3	1=Snare Drum						
	b2	1=Tom-tom						
	b1	1=Top Cymbal						
	b0	1=High-Hat						
\$C0H ⋮ \$C8H	•    •	•    •	Feedback	CON	Fator de realimentação para o FM e tipo de conexão			
	b7~b4	Não usados (sempre “0000”)						
	b3~b1	Realimentação (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π)						
	b0	Tipo de conexão dos operadores (0=série; 1=paralelo)						
STAT	INT	T1	T2	EOS	BUF	•    •	PCM	Registrador de estado
	b7	Será 1 quando um ou mais bits b3 a b6 forem 1						
	b6	Será 1 após o tempo contado pelo timer 1 (\$02)						
	b5	Será 1 após o tempo contado pelo timer 2 (\$03)						
	b4	Será 1 quando análise/síntese ADPCM completar						
	b3	Será 1 no fim da leitura/gravação/análise/síntese						
	b2-b1	Não usados (sempre “00”)						
	b0	Será 1 durante a análise/síntese ADPCM (se b7 de \$07 for 1)						

### 12.5.1 – Portas de acesso ao MSX-Audio

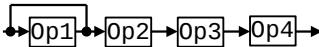
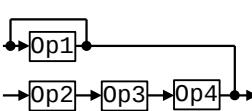
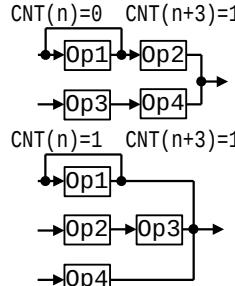
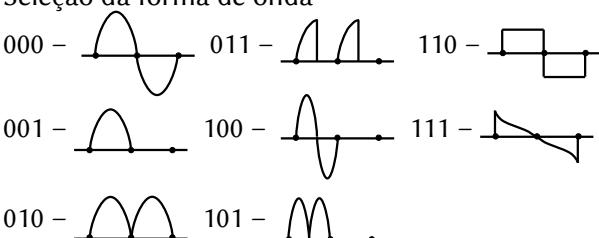
Porta		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
C0H	W	Nº registrador (01H a C8H)					Seleciona registrador			
	R	INT	T1	T2	EOS	BUF	•	•	PCM	Lê registrador de estado
C1H	W/R	Byte de dados						Escreve/lê dados no/do MSX-Audio		

## 12.6 – MAPA DOS REGISTRADORES DO OPL4 (YMF278)

### 12.6.1 – Register Array #0

Gerador FM – Register Array 0 (A1 = “1”)														
Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida					
\$00H	Teste						Registradores de teste							
\$01H														
\$02H	1º Temporizador (80,8 µS)						Registradores de tempo							
\$03H	2º Temporizador (323,1 µS)													
\$04H	RST	MT1	MT2	•	•	•	ST2	ST1	Sinalizadores					
	b7	RST – Se colocado em 1, reseta b5, b6 e b7.												
	b6	MT1 – Se colocado em 1, b0 será colocado em 0.												
	b5	MT2 – Se colocado em 1, b1 será colocado em 0.												
	b4~b2	Não usados (sempre “000”)												
	b1	ST2 – Controla início e parada de \$03 (1=inicia contagem)												
	b0	ST1 – Controla início e parada de \$02 (1=inicia contagem)												
\$05H	Somente no Register Array 1													
\$08H	•	NTS	•	•	•	•	•	•	Configuração de teclado					
	b7	Não usado (sempre “0”)												
	b6	NTS – Se 0, as oitavas serão determinadas pelos 2 bits mais altos de F_number. Se 1, serão determinadas apenas pelo bit mais alto de F_number.												
	b5~b0	Não usados (sempre “000 000”)												
\$20H	AM	VIB	EGT	KSR	Múltiplo		Definição de instrumentos							
⋮														
\$35H	b7	AM (1=liga trémolo – Variação de amplitude 3,7Hz)												
	b6	VIB (1=liga vibrato – Variação de frequência 6,4Hz)												
	b5	EG-TYP (0=tom percussivo; 1=tom constante)												
	b4	Se 0, KSR→0~3; Se 1, KSR→0~15												
	b3~b0	Fator de multiplicação (0=1/2, 1=1, 2=2, 3=3, ..., 15=15)												
\$40H	KSL		Nível total				KSL (00= 0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB) Nível total (b0=0,75dB,							
⋮														
\$55H														

					b1=1,5dB ..... b5=24dB)						
\$60H ⋮ \$75H	Attack Rate (AR)		Decay Rate (DR)	Attack (0dB a 96dB → mín. 0,2 mS; máx 2826 mS) Decay (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)							
\$80H ⋮ \$95H	Sustain Level (SL)		Release Rate (RL)	Sustain (b7=24dB, b6=12dB, b5=6dB, b4=3dB) Release (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)							
\$A0H ⋮ \$A8H	Frequência (LSB 8 bits)			Frequência (b7~b0)							
\$B0H ⋮ \$B8H	•	•	KEY	Oitava	Freq. MSB 2 bits (b9~b8) Oitava (FM) Key on/off (FM)						
	b7~b6 b5 b4~b2 b1~b0	Não usados (sempre “00”) 0=Voz respectiva desligada (key off); 1=voz ativa Define a oitava. A quarta é 011. Frequência MSB 2 bits. A Nota Lá central de 440 Hz é obtida com b1~b0=10 e \$A0H~A8H=01 000 110									
\$BDH	AM	VIB	BAT	BD	SD	TOM	TCY	HH	Controle da bateria do FM		
	b7 b6 b5 b4 b3 b2 b1 b0	Grau do trêmolo (0=1dB. 1=4,8dB) Grau do vibrato (0=7%; 1=14%) 0=Modo Melodia; 1=Modo Bateria 1=Bass Drum 1=Snare Drum 1=Tom-tom 1=Top Cymbal 1=High-Hat									
\$C0H ⋮ \$C8H	•	•	•	•	Feedback	CON	Fator de realim. tipo conexão Não usados (sempre “0000”) Realimentação (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π) Tipo de conexão (0=série; 1=paralelo) Para 4 operadores: A1      Canal      CNT(n)      CNT(n+3) 0            1            C0H            C3H				

		<table border="1"> <tr><td>0</td><td>2</td><td>C1H</td><td>C4H</td></tr> <tr><td>0</td><td>3</td><td>C2H</td><td>C5H</td></tr> <tr><td>1</td><td>4</td><td>C0H</td><td>C3H</td></tr> <tr><td>1</td><td>5</td><td>C1H</td><td>C4H</td></tr> <tr><td>1</td><td>6</td><td>C2H</td><td>C5H</td></tr> </table> <p>Para 4 operadores:  <math>CNT(n)=0 \quad CNT(n+3)=0</math>    <math>CNT(n)=1 \quad CNT(n+3)=0</math>    <math>CNT(n)=0 \quad CNT(n+3)=1</math>  </p>	0	2	C1H	C4H	0	3	C2H	C5H	1	4	C0H	C3H	1	5	C1H	C4H	1	6	C2H	C5H																																																												
0	2	C1H	C4H																																																																															
0	3	C2H	C5H																																																																															
1	4	C0H	C3H																																																																															
1	5	C1H	C4H																																																																															
1	6	C2H	C5H																																																																															
		<p>Operadores (para \$20H~\$35H e \$C0H~\$C8H)</p> <table> <tr><td>Oper:</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td></tr> <tr><td>Voz:</td><td>1</td><td>2</td><td>3</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Reg:</td><td>\$20</td><td>\$21</td><td>\$22</td><td>\$23</td><td>\$24</td><td>\$25</td><td>\$28</td><td>\$29</td><td>\$2A</td></tr> <tr><td>Freq:</td><td>\$A0</td><td>\$A1</td><td>\$A2</td><td>\$A0</td><td>\$A1</td><td>\$A2</td><td>\$A3</td><td>\$A4</td><td>\$A5</td></tr> </table> <table> <tr><td>Oper:</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td></tr> <tr><td>Voz:</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>Reg:</td><td>\$2B</td><td>\$2C</td><td>\$2D</td><td>\$30</td><td>\$31</td><td>\$32</td><td>\$33</td><td>\$34</td><td>\$35</td></tr> <tr><td>Freq:</td><td>\$A3</td><td>\$A4</td><td>\$A5</td><td>\$A6</td><td>\$A7</td><td>\$A8</td><td>\$A6</td><td>\$A7</td><td>\$A8</td></tr> </table> <p>Os operadores são associados da seguinte forma:  \$20/\$40/\$60/\$80/\$A0/\$B0/\$C0  ou  \$23/\$43/\$63/\$83/\$A0/\$B0/\$C0</p>	Oper:	01	02	03	04	05	06	07	08	09	Voz:	1	2	3	1	2	3	4	5	6	Reg:	\$20	\$21	\$22	\$23	\$24	\$25	\$28	\$29	\$2A	Freq:	\$A0	\$A1	\$A2	\$A0	\$A1	\$A2	\$A3	\$A4	\$A5	Oper:	10	11	12	13	14	15	16	17	18	Voz:	4	5	6	7	8	9	7	8	9	Reg:	\$2B	\$2C	\$2D	\$30	\$31	\$32	\$33	\$34	\$35	Freq:	\$A3	\$A4	\$A5	\$A6	\$A7	\$A8	\$A6	\$A7	\$A8
Oper:	01	02	03	04	05	06	07	08	09																																																																									
Voz:	1	2	3	1	2	3	4	5	6																																																																									
Reg:	\$20	\$21	\$22	\$23	\$24	\$25	\$28	\$29	\$2A																																																																									
Freq:	\$A0	\$A1	\$A2	\$A0	\$A1	\$A2	\$A3	\$A4	\$A5																																																																									
Oper:	10	11	12	13	14	15	16	17	18																																																																									
Voz:	4	5	6	7	8	9	7	8	9																																																																									
Reg:	\$2B	\$2C	\$2D	\$30	\$31	\$32	\$33	\$34	\$35																																																																									
Freq:	\$A3	\$A4	\$A5	\$A6	\$A7	\$A8	\$A6	\$A7	\$A8																																																																									
\$E0H : \$F5H	.	• . • . •	Wave Select Seleção da forma de onda																																																																															
	b7~b3	Não usados (sempre "00 000")																																																																																
	b2-b1	Seleção da forma de onda																																																																																

## 12.6.2 – Register Array #1

Gerador FM – Register Array 1 (A1 = "H")									
Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida

\$00H \$01H	Teste				Registradores de teste				
\$02H \$03H	Somente no Register Array 0								
\$04H	.	.	Connection SEL		Seleção modo 4 operadores				
	b7~b6	Não usados (sempre “00”)							
\$05H	b5~b0	Liga o modo de 4 operadores p/ o slot respectivo: Bit: b5 b4 b3 b2 b1 b0 Slot: 6 5 4 3 2 1							
	.	.	.	.	NEW2	NEW Registrador de expansão			
\$08H	Somente no Register Array 0								
\$20H ⋮ \$35H	AM VIB ⋮ b7 b6 b5 b4 b3~b0	EGT KSR ⋮ AM (1=liga trêmolo – Variação de amplitude 3,7Hz) VIB (1=liga vibrato – Variação de frequência 6,4Hz) EG-TYP (0=tom percussivo; 1=tom constante) Se 0, KSR→0~3; Se 1, KSR→0~15 Fator de multiplicação (0=1/2, 1=1, 2=2, 3=3, ..., 15=15)	Múltiplo						
\$40H ⋮ \$55H	KSL	Nível total		KSL (00= 0dB/oitava, 01=1,5dB, 10=3dB, 11=6dB) Nível total (b0=0,75dB, b1=1,5dB ..... b5=24dB)					
\$60H ⋮ \$75H	Attack Rate (AR)		Decay Rate (DR)	Attack (0dB a 96dB → mín. 0,2 mS; máx 2826 mS) Decay (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)					
\$80H ⋮ \$95H	Sustain Level (SL)		Release Rate (RL)	Sustain (b7=24dB, b6=12dB, b5=6dB, b4=3dB) Release (0dB a 96dB → mín. 2,4 mS; máx 39 280 mS)					
\$A0H ⋮ \$A8H	Frequência (LSB 8 bits)				Frequência (b7-b8)				

\$B0H ⋮ \$B8H	•	•	KEY	Oitava	Freq. MSB 2 bits	Freq. MSB 2 bits (FM) Oitava (FM) Key on/off (FM)																												
	b7~b6 b5 b4~b2 b1~b0	<p>Não usados (sempre “00”)</p> <p>0=Voz respectiva desligada (key off); 1=voz ativa</p> <p>Define a oitava. A quarta é 011.</p> <p>Frequência MSB 2 bits. A Nota Lá central de 440 Hz é obtida com b1~b0=10 e \$A0H~A8H=01000 110</p>																																
\$BDH	Somente no Register Array 0																																	
\$C0H ⋮ \$C8H	•	•	•	•	Feedback	CON	Fator de realim. tipo conexão																											
	b7~b4 b3~b1 b0	<p>Não usados (sempre “0000”)</p> <p>Realimentação (0=0; 1=π/16; 2=π/8; ...; 6=2π; 7=4π)</p> <p>Tipo de conexão (0=série; 1=paralelo)</p> <p>Para 4 operadores:</p> <table> <thead> <tr> <th>A1</th> <th>Canal</th> <th>CNT(n)</th> <th>CNT(n+3)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>C0H</td> <td>C3H</td> </tr> <tr> <td>0</td> <td>2</td> <td>C1H</td> <td>C4H</td> </tr> <tr> <td>0</td> <td>3</td> <td>C2H</td> <td>C5H</td> </tr> <tr> <td>1</td> <td>4</td> <td>C0H</td> <td>C3H</td> </tr> <tr> <td>1</td> <td>5</td> <td>C1H</td> <td>C4H</td> </tr> <tr> <td>1</td> <td>6</td> <td>C2H</td> <td>C5H</td> </tr> </tbody> </table> <p>Para 4 operadores:</p> <p>CNT(n)=0 CNT(n+3)=0      CNT(n)=0 CNT(n+3)=1</p> <p>CNT(n)=1 CNT(n+3)=0      CNT(n)=1 CNT(n+3)=1</p>					A1	Canal	CNT(n)	CNT(n+3)	0	1	C0H	C3H	0	2	C1H	C4H	0	3	C2H	C5H	1	4	C0H	C3H	1	5	C1H	C4H	1	6	C2H	C5H
A1	Canal	CNT(n)	CNT(n+3)																															
0	1	C0H	C3H																															
0	2	C1H	C4H																															
0	3	C2H	C5H																															
1	4	C0H	C3H																															
1	5	C1H	C4H																															
1	6	C2H	C5H																															
	Operadores (para \$20H~\$35H e \$C0H~\$C8H)																																	
	Oper: Voz: Reg: Freq: Oper: Voz: Reg: Freq:	<p>19 20 21 22 23 24 25 26 27</p> <p>1 2 3 1 2 3 4 5 6</p> <p>\$20 \$21 \$22 \$23 \$24 \$25 \$28 \$29 \$2A</p> <p>\$A0 \$A1 \$A2 \$A0 \$A1 \$A2 \$A3 \$A4 \$A5</p> <p>28 29 30 31 32 33 34 35 36</p> <p>4 5 6 7 8 9 7 8 9</p> <p>\$2B \$2C \$2D \$30 \$31 \$32 \$33 \$34 \$35</p> <p>\$A3 \$A4 \$A5 \$A6 \$A7 \$A8 \$A6 \$A7 \$A8</p>																																
	<p>Os operadores são associados da seguinte forma:</p> <p>\$20/\$40/\$60/\$80/\$A0/\$B0/\$C0</p> <p>ou</p> <p>\$23/\$43/\$63/\$83/\$A0/\$B0/\$C0</p>																																	

\$E0H : \$F5H	•   •   •   •   •	Wave Select	Seleção da forma de onda
	b7~b3	Não usados (sempre “00 000”)	
	b2~b1	Seleção da forma de onda	<p>000 -  011 -  110 - </p> <p>001 -  100 -  111 - </p> <p>010 -  101 - </p>

### 12.6.3 – Síntese Wave

Síntese Wave										
Reg	b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida	
\$00H \$01H	Teste						Registradores de teste			
\$02H	ID disp		Cabeç. Wave		MT	MM	Funções especiais			
	b7~b5 b4~b2		ID do OPL4 (b7=0; b6=0; b7=1) Cabeçalho da tabela wave: 000=0 a 511 (000 000H)      100=384 a 511 (200 000H) 001=384 a 511 (080 000H)      101=384 a 511 (280 000H) 010=384 a 511 (100 000H)      110=384 a 511 (300 000H) 011=384 a 511 (180 000H)      111=384 a 511 (380 000H) b1      Tipo de memória de áudio (0=ROM; 1=RAM) b0      Acesso à memória de áudio (0=OPL4; 1=CPU)							
\$03H	•	•	a21	a20	a19	a18	a17	a16	Endereço da memória de áudio	
\$04H	a15	a14	a13	a12	a11	a10	a9	a8		
\$05H	a7	a6	a5	a4	a3	a2	a1	a0		
\$06H	Dados da memória						Registrador de dados			
\$08H : \$1FH	Número da tabela Wave LSB (n7~n0)						24 registradores com o número LSB (n7~n0) da tabela Wave reproduzida			

\$20H : \$37H	F_number (f6~f0)			Tab Wave (n8)	24 registradores com a Frequência 7 bits LSB e nº da tabela Wave MSB (n8)
\$38H : \$4FH	Oitava (o3~o0)		Pseudo-rev	F_number (f9~f7)	Oitava (-7 a +7) Pseudo-reverberação Frequência (3 bits MSB)
	b7~b0 b7~b6 b3 b7~b4	Com “b0” (n8) seleciona até 512 samples (0~511) Com “b2-b1-b0” (f9~f7) define a frequência Se “1” liga a pseudo-reverberação; se “0”, desligada Oitava. Varia de -7 a +7 (-8 não é permitido). Em conjunto com F_number define a frequência. Para oitava = 1 e F_number = 0, a frequência é de 44,1 KHz. $f(\ell) = 1200 * (\text{oitava} - 1) + 1200 * \log_2 \frac{1024 + \text{F\_number}}{1024}$			
\$50H : \$67H	Nível total (l6~l0)			ND	Nível total 7 bits (l6~l0) Nível direto
\$68H : \$7FH	b7-b1 b0	Nível total (b7=-24dB, b6=-12dB, ... b1=-0,375dB) Nível direto (0=altera envoltória durante a interpolação; 1=altera imediatamente)			
	Key on Damp	LFO RST CH	Panpot	Funções diversas e balanceamento estéreo (Panpot)	
	b7 b6 b5 b4 b3~b0	0=key on; 1=key off 0=Damp desligado; 1=Damp ativo LFO RST (0=liga o LFO; 1=desliga o LFO) 0=Wave mixado com FM; 1=Sem mixagem Panpot: 0 1 2 ... 6 7 8 9 ... 13 14 15 Esq(dB) 0 -3 -6 ... -18 -∞ -∞ 0 ... 0 0 0 Dir(dB) 0 0 0 ... 0 0 -∞ -18 ... -9 -6 -3			
\$80H : \$97H	• •	LFO (s2~s0)	VIB (v2~v0)	Frequência do trêmolo e do vibrato (LFO) Grau do vibrato (VIB)	
	b7~b6 b5~b3 b2~b0	Não usados LFO (0=0,168Hz, 1=2,019Hz, ... 7=7,066Hz) Grau Vibrato (0=off, 1=3,378; 2=5,065, ... 7=79,31)			

\$98H ⋮ \$AFH	Attack Rate		Decay Rate (1)	Attack Rate 10-90% → (1=3715 mS; 14=0,23 mS) Decay 1 Rate 10-90% → (1=19 040 mS; 14=1,18 mS)
\$B0H ⋮ \$C7H	Decay Level		Decay Rate (2)	Decay Level (b7=-24; b6=-12; b5=-6; b4=-3 dB) Decay 2 Rate 10-90% → (1=19 040 mS; 14=1,18 mS)
\$C8H ⋮ \$DFH	Rate Correction		Release Rate	Rate Correction Release Rate
	b7~b4	Rate Correction: ( RATE = (OCT + RC)*2 + f9 + RD ) OCT = Oitava (-7 a +7 em \$38H~\$4FH) RC = Rate correction (0 ~ 14 em \$C8H~\$DFH) f9 = bit “f9” de F_number (\$38H~\$4FH) RD = valores de AR, D1R, D2R e RR (0001=04; 0010=08; ...; 1111=63)		
	b3~b0	Release Rate 10-90% → (1=19 040 mS; 14=1,18 mS)		
\$E0H ⋮ \$F7H	• •	• • •	AM(a2~a0)	Grau do trêmolo
	b7~b3	Não usados		
	b2~b0	Amplitude do trêmolo (0=off; 1=1,781; ...; 7=11,91)		
\$F8H	• •	Mix FM_R	Mix FM_L	Nível de saída FM
	b7~b6	Não usados (sempre “00”)		
	b5~b3	Nível FM direito (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
	b2~b0	Nível FM esquerdo (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
\$F9H	• •	Mix PCM_R	Mix PCM_L	Nível de saída PCM
	b7~b6	Não usados (sempre “00”)		
	b5~b3	Nível PCM direito (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		
	b2~b0	Nível PCM esquerdo (0=0; 1=-3; 2=-6; ... 6=-18dB; 7=∞)		

#### 12.6.4 – Portas de acesso ao OPL4

Porta		b7	b6	b5	b4	b3	b2	b1	b0	Descrição Resumida
C4H	W	Nº registrador (00H a F5H)					Seleciona reg. FM array 0			
	R	IRQ	FT1	FT2	•	•	•	LD	BSY	Lê registrador de estado

	b7 b6 b5 b4~b2 b1 b0	IRQ – Interrupt Request (será “1” quando FT1 ou FT2 for “1”) FT1 – Será “1” quando o timer 1 terminar a contagem FT2 – Será “1” quando o timer 2 terminar a contagem Não usados (sempre “000”) LD – Será “1” durante a leitura do header PCM pelo OPL4 (Válido quando o bit 05H_NEW2 do array 1 for “1”.) BUSY – Será “1” durante a escrita de dados nos registradores (Válido quando o bit 05H_NEW2 do array 1 for “1”.)	
C5H	W	Byte de dados	Escreve dado nos regs
C4H	W	Nº registrador (00H a F5H)	Seleciona reg. FM array 0
C7H	R	Espelho de C5H	Acesso por C5H é preferido
7EH	W	Nº registrador (00H a F9H)	Seleciona regs PCM
7FH	W/R	Byte de dados	Escreve ou lê dado regs

### 12.6.5 – Cabeçalho da “Wave Table Synthesis”

End	b7	b6	b5	b4	b3	b2	b1	b0	
00H	d1	d0	s21	s20	s19	s18	s17	s16	d1, d0 → 00=8bits;
01H	s15	s14	s13	s12	s11	s10	s9	s8	01=12 bits; 10=16 bits
02H	s7	s6	s5	s4	s3	s2	s1	s0	s21~s0 – Endereço inicial
03H	l15	l14	l13	l12	l11	l10	l9	l8	Endereço de loop
04H	l7	l6	l5	l4	l3	l2	l1	l0	
05H	e15	e14	e13	e12	e11	e10	e9	e8	Endereço final
06H	e7	e6	e5	e4	e3	e2	e1	e0	
07H	•	•	f2	f1	f0	v2	v1	v0	Freq. LFO e grau vibrato
08H	ar3	ar2	ar1	ar0	dr3	dr2	dr1	dr0	Attack Rate; Decay 1 Rate
09H	dl3	dl2	dl1	dl0	dr3	dr2	dr1	dr0	Decay Level; Decay 2 Rate
0AH	rc3	rc2	rc1	rc0	rr3	rr2	rr1	rr0	Rate correct; Release Rate
0BH	•	•	•	•	•	am2	am1	am0	Grau AM (trêmolo)

### 12.6.6 – Tamanho dos dados “wave”

16 bits	d15	d14	d13	d12	d11	d10	d9	d8	+00H
	d7	d6	d5	d4	d3	d2	d1	d0	+01H
12 bits	d11	d10	d9	d8	d7	d6	d5	d4	+00H
	d3	d2	d1	d0	d3 d2 d1 d0				+01H
	d11	d10	d9	d8	d7	d6	d5	d4	+02H
8 bits	d7	d6	d5	d4	d3	d2	d1	d0	+00H

## 12.7 – MAPA DOS REGISTRADORES DO SCC (2212/2312)

Endereços	Descrição resumida (SCC)
9800H~981FH	Forma de onda da voz #1
9820H~983FH	Forma de onda da voz #2
9840H~985FH	Forma de onda da voz #3
9860H~987FH	SCC : Esc/leit: Forma de onda das vozes #4 e #5 SCC+: Leitura: Forma de onda da voz #4
9880H~9881H	Frequência da voz #1
9882H~9883H	Frequência da voz #2
9884H~9885H	Frequência da voz #3
9886H~9887H	Frequência da voz #4
9888H~9889H	Frequência da voz #5
	Exemplo: 9880= [f7 f6 f5 f4 f3 f2 f1 f0] 9881= [• • • • f11 f10 f9 f8] $F_{\text{tone}} = \frac{F_{\text{clock}}}{32 * ((f11-f0)+1)}$ ( $F_{\text{clock}}=3,579545 \text{ MHz}$ )
988AH	Volume da voz #1 (0 a 15)
988BH	Volume da voz #2 (0 a 15)
988CH	Volume da voz #3 (0 a 15)
988DH	Volume da voz #4 (0 a 15)
988EH	Volume da voz #5 (0 a 15)
998FH	[• • • v5 v4 v3 v2 v1] v5=1 → liga voz 5 v4=1 → liga voz 4, etc
9890H~989FH	Espelho de 9880H~988FH
98A0H	SCC: sem função SCC+: leitura forma de onda voz #5 (escrita proibida)
98A1H~98BFH	Espelhos de 98A0H
98C0H	SCC: Espelho de 98A0H SCC+: Registrador de deformação

98C1H~98DFH	SCC: Espelho de 98A0H SCC+: Espelho de 98C0H							
98E0H	SCC: Registrador de deformação <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>R</td> <td>R</td> <td>.</td> <td>.</td> <td>.</td> <td>P</td> <td>P</td> </tr> </table> PP: 11/10 → Ftone *16; 01 → Ftone *256; 00 → Ftone *1 RR: 11 → ruído branco vozes 4 e 5 cfe forma onda 01 → ruído branco contínuo 00 → sem ruído branco SCC+: Sem função	R	R	.	.	.	P	P
R	R	.	.	.	P	P		
98E1H~98FFH	Espelhos de 98E0H							

### 12.7.1 – Endereços de acesso ao SCC

Endereços	Descrição resumida (SCC+)																
B800H~B81FH	Forma de onda da voz #1																
B820H~B83FH	Forma de onda da voz #2																
B840H~B85FH	Forma de onda da voz #3																
B860H~B87FH	Forma de onda da voz #2																
B880H~B89FH	Forma de onda da voz #5																
B8A0H~B8A1H	Frequência da voz #1																
B8A2H~B8A3H	Frequência da voz #2																
B8A4H~B8A5H	Frequência da voz #3																
B8A6H~B8A7H	Frequência da voz #4																
B8A8H~B8A9H	Frequência da voz #5																
	Exemplo: B8A0 = <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>f7</td><td>f6</td><td>f5</td><td>f4</td><td>f3</td><td>f2</td><td>f1</td><td>f0</td></tr> </table> B8A1 = <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>.</td><td>.</td><td>.</td><td>.</td><td>f11</td><td>f10</td><td>f9</td><td>f8</td></tr> </table> $F_{clock} = \frac{F_{clock}}{32 * ((f11-f0)+1)} \quad (F_{clock}=3,579545 \text{ MHz})$	f7	f6	f5	f4	f3	f2	f1	f0	.	.	.	.	f11	f10	f9	f8
f7	f6	f5	f4	f3	f2	f1	f0										
.	.	.	.	f11	f10	f9	f8										
B8AAH	Volume da voz #1 (0 a 15)																
B8ABH	Volume da voz #2 (0 a 15)																
B8ACH	Volume da voz #3 (0 a 15)																
B8ADH	Volume da voz #4 (0 a 15)																

B8AEH	Volume da voz #5 (0 a 15)								
B8AFH	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>•</td><td>•</td><td>•</td><td>v5</td><td>v4</td><td>v3</td><td>v2</td><td>v1</td></tr></table> v5=1 → liga voz 5 v4=1 → liga voz 4, etc	•	•	•	v5	v4	v3	v2	v1
•	•	•	v5	v4	v3	v2	v1		
B8B0H~B8BFH	Espelho de B8A0H~B8AFH								
B8C0H	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>R</td><td>R</td><td>•</td><td>•</td><td>•</td><td>P</td><td>P</td></tr></table> – Registrador de deformação PP: 11/10→Ftone * 16; 01→Ftone * 256; 00→Ftone * 1 RR: 11 → ruído branco vozes 4 e 5 cfe forma onda 01 → ruído branco contínuo 00 → sem ruído branco	R	R	•	•	•	P	P	
R	R	•	•	•	P	P			
B8C1H~B8DFH	Espelhos de B8C0H								
B8E0H~B8FFH	Sem função								
B900H~BFFDH	???								
BFFEH~BFFFH	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>•</td><td>•</td><td>S</td><td>M</td><td>•</td><td>B3</td><td>B2</td><td>B1</td></tr></table> – Registrador de modo S – Modo SCC (0=SCC; 1=SCC+) M – Modo memória (0=bank select; 1=RAM) B3 – Banco de mem. #3 (0=bank select; 1=RAM) B2 – Banco de mem. #2 (0=bank select; 1=RAM) B1 – Banco de mem. #1 (0=bank select; 1=RAM)	•	•	S	M	•	B3	B2	B1
•	•	S	M	•	B3	B2	B1		

## REFERÊNCIAS BIBLIOGRÁFICAS

A PROFUNDANDO-SE NO MSX

Piazz – Maldonado – Oliveira (Editora Aleph, 1986)

CARTÃO DE 80 COLUNAS & RS232C – MANUAL DE OPERAÇÕES

Gradiente (1989)

FM MUSIC MACRO YRM-104 – Owner's Manual

Yamaha (1984)

FUDEBA ASSEMBLER – Manual de Referência e Arquitetura MSX

Felipe Bergo (2002)

GR8NET Technical Databook and Programmer's Guide

Age Labs (2019)

HBI-232MKII – Basic Manual

Age Labs & Ebsoft (2014)

LIVRO VERMELHO DO MSX, O (The Red Book)

McGraw Hill / Avalon Software (1988 / 1985)

MANUAL DE INSTRUÇÕES DA INTERFACE DE DRIVE DDX

MANUAL DO MICROPROCESSADOR Z-80

William Barden Jr. (Editora Campus, 1985)

MIDI MACRO MONITOR YRM-303 – Owner's Manual

Yamaha (1986)

MSX DATAPACK volumes 1, 2 e 3

ASCII Corporation (1991)

MSX-DOS version 2 – The advanced disk operating system for

MSX 2 computers – ASCII Corp (1988)

MSX MAGAZINE, Edição Dezembro de 1990  
ASCII Corporation (1990)

MSX MAGAZINE, Edição ???  
ASCII Corporation (1990)

MSX MOZAÏK, Edição nº 33  
Editora desconhecida, Ano desconhecido

MSX TECHNICAL GUIDE BOOK  
Ayumu Kimura (ASCAT Ashigaka, NIPPON, 1992)

MSX TECHNICAL DATA BOOK  
Sony Corp (1984)

MSX turbo R TECHNICAL HANDBOOK  
ASCII Corporation (1991)

MSX2 TECHNICAL HANDBOOK  
ASCII Corporation (1985)

NEXTOR 2.0 User Manual  
Konamiman (2014)

Nextor 2.1 Driver Development Guide  
Konamiman (2020)

OPL4 YMF278B – APPLICATION MANUAL  
Yamaha Corporation (1994)

PROGRAMAÇÃO AVANÇADA EM MSX  
Figueroedo – Maldonado – Rosseto (Editora Aleph, 1986)

PX-7 P-BASIC Reference Manual  
Pioneer (1985)

TMS9918A/TMS9928A/TMS9929A Video Display Processors Data  
Manual – Texas Instruments (1982)

V9938 MSX-VIDEO – APPLICATION MANUAL  
Nippon Gakki Co. Ltd. (Yamaha, 1985)

V9938 MSX-VIDEO – TECHNICAL DATA BOOK  
Nippon Gakki Co. Ltd. (Yamaha, 1985)

V9958 MSX-VIDEO – TECHNICAL DATA BOOK  
Yamaha Corporation (1989)

V9990 E-VDP-III – APPLICATION MANUAL  
Yamaha Corporation (1992)

Y9850 MSX-AUDIO – APPLICATION MANUAL  
Nippon Gakki Co. Ltd. (Yamaha, 1985)

YM2413 FM OPERATOR TYPE LL (OPLL) – APPLICATION MANUAL  
Yamaha Corporation (1987)

<https://www.gigamix.jp/ds2/>

[https://www.msx.org/wiki/I/O\\_Ports\\_List#The\\_register\\_of\\_internal\\_I.2FO\\_ports\\_control](https://www.msx.org/wiki/I/O_Ports_List#The_register_of_internal_I.2FO_ports_control)

<http://msxbanzai.tni.nl/v9990/manual.html>

<https://github.com/Konamiman/MSX-UNAPI-specification/tree/master/docs>

<http://www.symbos.de/>

<https://www.msx.org/wiki/MSX-HID>

## OUTROS LIVROS DO MESMO AUTOR



YBYMARÃ – A Cidade do Outro Lado



MSX Top Secret 2



MSX Top Secret