Autor:
EDISON ANTONIO PIRES DE MORAES
e-mail: eapmoraes@ig.com.br

2ª Edição - Abril/2004

Edição revisada e ampliada

NOTA DO AUTOR

Finalmente, depois de cinco anos, resolvi encarar o desafio de revisar e aprimorar o *MSX Top Secret.* O resultado é este *MSX Top Secret II*, decorrente de muito trabalho e marcado por inúmeros contratempos, mas que acabou saindo com quase o dobro de informações em relação ao primeiro.

Humildemente, pretendo que esta obra seja a maior coletânea de informações existente sobre o MSX, reconhecendo, entretanto, que ainda faltam nela muitos elementos para que o assunto possa ser esgotado. É direcionada basicamente ao desenvolvimento de software para o nosso querido MSX, mesmo para aqueles que não têm interesse em se aprofundar na linguagem de máquina e preferem programar em BASIC. Foi escrita com a mesma dedicação e carinho que ao primeiro *MSX Top Secret* e, como ele, resultou de quase três anos de pesquisas, várias delas executadas no próprio micro MSX.

Alguns termos usados no *MSX Top Secret II* podem causar estranheza, como colocar BIOS no masculino e não no feminino. Isso decorre do fato de que a tradução para o português leva-o ao masculino, por isso resolvi mantê-lo assim. Outros termos, como "MSXDOS1" em vez de "MSXDOS" ou "SCC simples" em vez de "SCC", estão lá para tornar mais clara a compreensão. Uma certa redundância em alguns casos visa a facilitar a consulta.

Espero, enfim, que esta obra seja do agrado de todos os que se dispuserem a lê-la. Abraços a todos.

Agradecimentos

Agradeço aos meus amigos
Adriano Camargo Rodrigues da Cunha,
por fornecer prontamente informações sobre o UZIX,
Alex Mitsio Sato,
por traduzir vários textos em japonês,
Hans Otten,
por traduzir vários textos do holandês para o inglês,
à minha esposa, a quem chamo carinhosamente de amoi,
pela paciência, compreensão e incentivo
à minha filha Lívia,
que me deu novo alento para a vida,
ao meu pai ✝ e à minha mãe,
pela vida,
à minha irmã,
pelo companheirismo,
e a todos que, de forma direta ou indireta,
ajudaram na conclusão desta obra.

Í N D I C E

CAPÍTULO 1 - INTRODUÇÃO AO SISTEMA

CAPÍTULO 2 - SLOTS E CARTUCHOS

CAPÍTULO 3 - A MEMÓRIA ROM

CAPÍTULO 4 - A MEMÓRIA RAM

CAPÍTULO 5 - O VÍDEO E O VDP

CAPÍTULO 6 - GERADORES DE ÁUDIO

CAPÍTULO 7 - OS SISTEMAS DE DISCO

CAPÍTULO 8 - DISPOSITIVOS ADICIONAIS

CAPÍTULO 9 - O MSX TURBO R

APÊNDICE

GUIAS DE CONSULTA RÁPIDA

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.

Atualmente, já existe 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 RAM 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 abaixo estão ilustradas as principais características e diferenças entre os quatro modelos MSX que existem atualmente.

	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	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, só 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 (256 páginas lógicas), embora alguns deles reconheçam a Mapper em slots diferentes.

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 - 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 processamen-

to desse chip, entretanto, se mostrou insuficiente. Por isso, apareceram MSX turbinados a 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 - 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.

1.2 - 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, e houve uma tentativa de uso do V9990, bem mais rápido, mas não foi muito bem aceito pelos usuários, possivelmente por não ser totalmente compatível com o V9958.

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 extremamente limitado, logo surgiram alternativas. A primeira foi o MSX-Audio, que não se popularizou. Depois veio o OPLL, 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, um chip muito avançado com qualidade de som igual à dos CD´s, mas obteve aceitação bastante reduzida. 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).

1.4 - A PPI

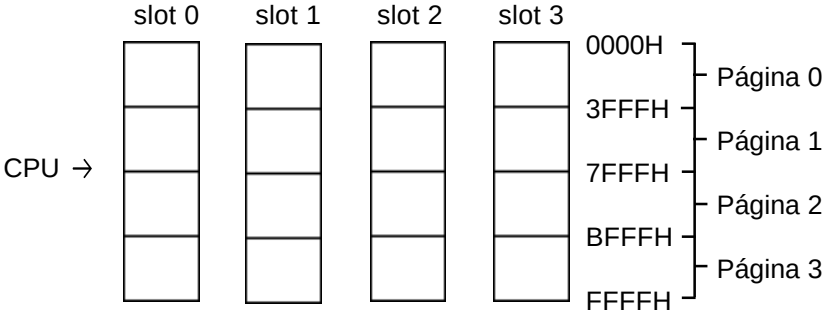
A PPI (Peripheral Programmable Interface - Interface de Periféricos Programável) é a “pedra de toque” do MSX. É ela que faz o MSX ser tão flexível. Suas duas principais tarefas são o controle de teclado e a seleção de slots e páginas de memória. O chip responsável, o 8255A da Intel, foi mantido até o modelo MSX turbo R. Apesar de ter sido embutida no MSX Engine nos modelos mais recentes, seu funcionamento é idêntico ao MSX1. No capítulo 2 pode ser visto como funciona o sistema de slots e páginas.

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[1].

1 - SLOTS

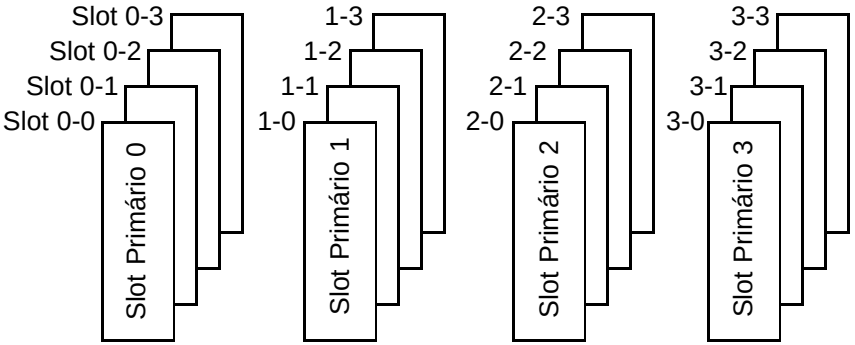
Há dois tipos de slots: os slots primários e os slots secundários. Os slots primários são em número de quatro e estão conectados diretamente à CPU. Cada slot é dividido em quatro partes de 16 Kbytes, perfazendo 64 Kbytes, denominadas "páginas". Uma página de mesmo número ocupa sempre o mesmo espaço de 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.



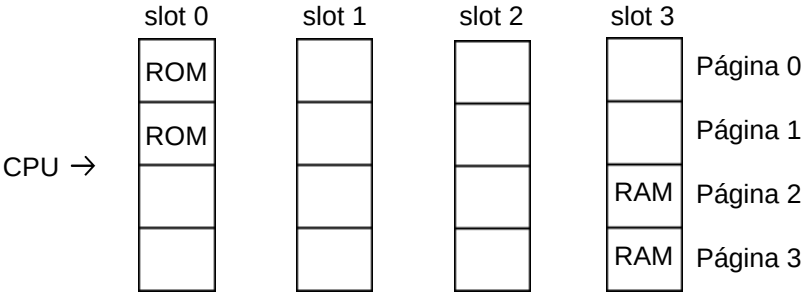
Cada slot primário pode suportar até quatro slots secundários. A escolha das páginas continua sendo possível da mesma forma que nos slots primários: apenas quatro páginas podem ficar ativas ao mesmo tempo, ainda que em slots primários e secundários diferentes.

Na página seguinte está ilustrado como ficam estruturados os slots primários e secundários. Podem haver 16 slots no máximo, dos quais usualmente 8 são reservados para a expansão do sistema e os outros 8 ficam disponíveis para o usuário.

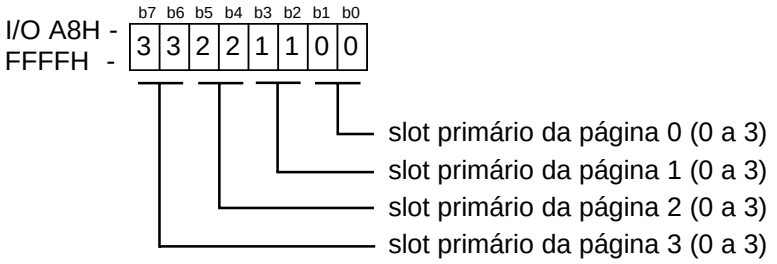
Nota 1: É necessário um certo cuidado para não confundir a técnica de slots com a expansão de memória "Memory Mapper" que utiliza outro artifício para que cada slot possa acessar efetivamente até 4 megabytes não lineares. A introdução da nova CPU R800, de 16 bits, nos modelos MSX turbo R, não altera a técnica de slots e páginas.



Seleção inicial de memória pela CPU:



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 nada mais é que o endereço FFFFH da memória[2]. O formato dos registradores de slots está descrito abaixo.



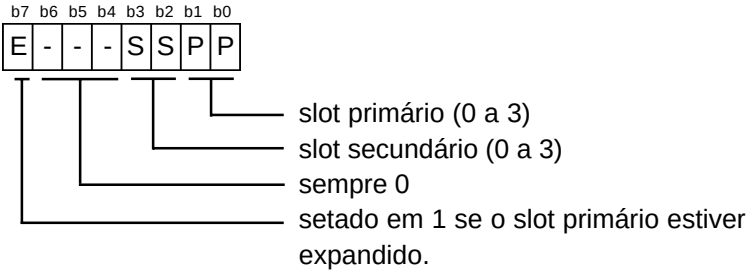
Nota 2: Não é recomendável que se troque slots e páginas diretamente e é necessário um cuidadoso planejamento para se chavear páginas e slots. Para utilizar rotinas em outras páginas é recomendável sempre usar o BIOS, que além de ser mais seguro e garantir a compatibilidade, simplifica muito a operação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

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.

Os slots onde ficam instaladas a Main-ROM, a Sub-ROM e a RAM dependem de cada máquina. Para os modelos MSX turbo R, entretanto, houve uma padronização: Main-ROM no slot 0.0 e Main-RAM no slot 3.0. 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 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.



1.1 - CHAMADAS INTER-SLOT

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 a um determinado software o acesso a rotinas em outros slots na mesma página física onde ele está sendo executado. Assim, um software rodando sob o MSXDOS, que ocupa toda a RAM linear disponível, pode acessar o BIOS ou o DOS Kernel residente na interface de disco. A partir do MSX2, há uma ROM de extensão do BIOS denominada Sub-ROM; as chamadas que o próprio BIOS faz à Sub-ROM também são chamadas inter-slot.

Para facilitar e assegurar a compatibilidade, existe um grupo de rotinas do BIOS denominado “grupo de chamadas inter-slot”, sendo que

algumas dessas rotinas também estão disponíveis para o MSXDOS, para que este possa acessar todas as rotinas do BIOS. As rotinas disponíveis para o MSXDOS estão descritas no capítulo sobre o sistema de disco.

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

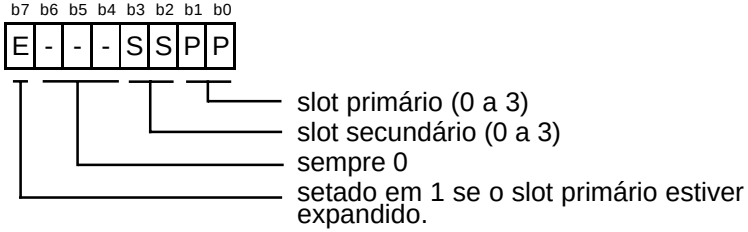
RDSLT (000CH) - Lê um byte em qualquer slot;
WRSLT (0014H) - Escreve um byte em qualquer slot;
CALSLT (001CH) - Chama uma rotina em qualquer slot;
ENASLT (0024H) - Troca páginas e slots;
CALLF (0030H) - Chama uma rotina em qualquer slot;
RSLREG (0138H) - Lê o registrador de slot primário;
WSLREG (013BH) - 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 capítulo 3, na seção “BIOS EM ROM”.

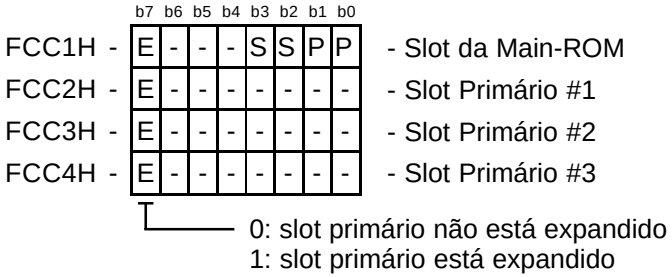
1.2 - ÁREA DE TRABALHO

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 capítulo 3):

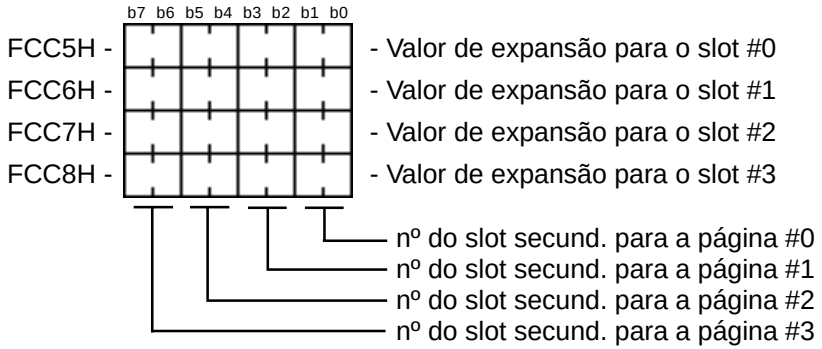
EXBRSA (FAF8H,1) - Slot da SUB-ROM



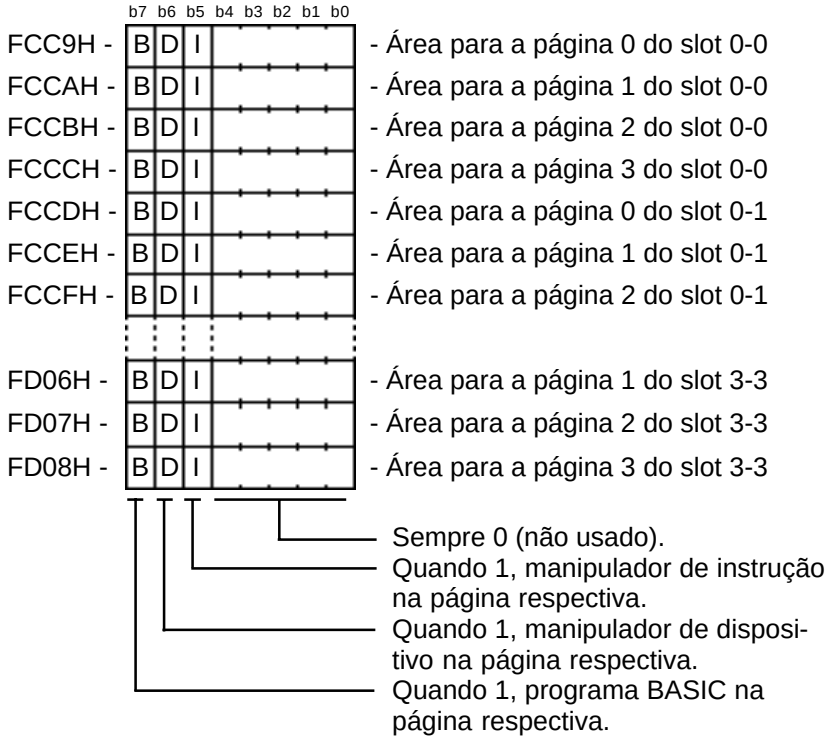
EXPTBL (FCC1H,4) - Indica se slot primário está expandido ou não.



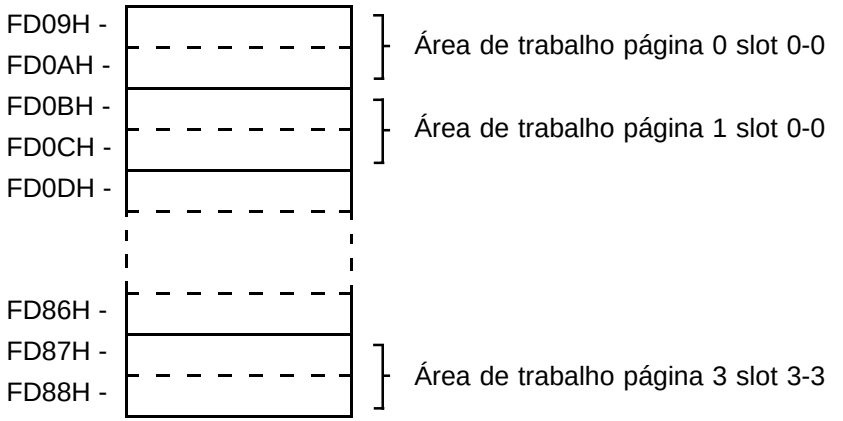
SLTTBL (FCC5H,4) - Área onde ficam registrados os valores de expansão de cada slot primário.



SLTATR (FCC9H,64) - Guarda a existência de rotinas em qualquer página e slot.



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



2 - DESENVOLVENDO SOFTWARE EM CARTUCHO

Normalmente, os micros MSX possuem dois slots 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	

Obs.: a área reservada deve ser obrigatoriamente preenchida com bytes 00H.

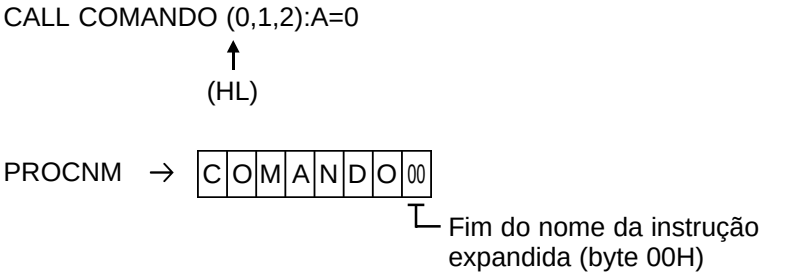
ID - São dois bytes de indentificação. No caso de cartuchos ROM, esses bytes devem ter o código "AB" (bytes 41H e 42H) e no caso de cartuchos de Sub-ROM, os bytes devem ser "CD" (43H e 44H).

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[3] 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[3] tem o seguinte formato:
CALL <nome da instrução de expansão> (argumento)

O nome da instrução 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:



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

---

Nota 3: A abreviação da instrução CALL é o caractere sublinhado "_" . Pode ser usado no lugar de CALL sem nenhum problema.

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
                       ↑
                     (HL)
Flag CY=0
```

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 descritor de arquivo (00H)

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

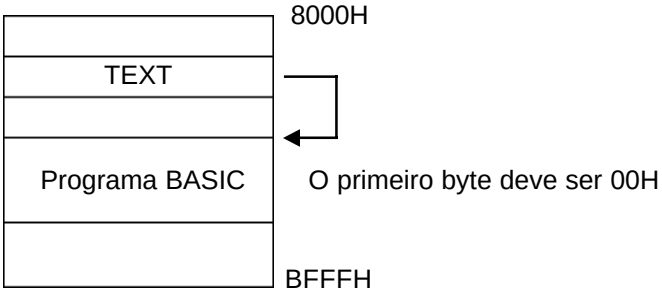
O interpretador procura cartucho após cartucho, e se ao final o nome de dispositivo não for reconhecido (ou seja, a flag CY sempre for 1), a mensagem de erro “Bad file name” (nome de arquivo errado) será mostrada.

Quando a operação de I/O atual é processada, 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	Reg. A	Dispositivo
0	OPEN	10	Função LOC
2	CLOSE	12	Função LOF
4	Acesso Aleatório	14	Função EOF
6	Saída Seqüencial	16	Função FPOS
8	Entrada Seqüencial	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.



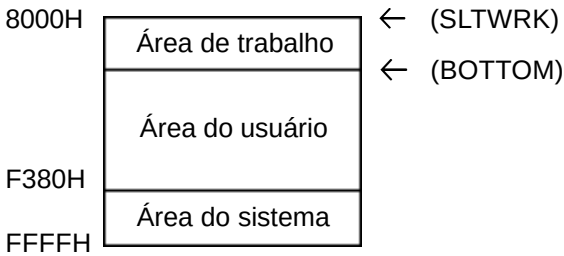
2.1 - ALOCANDO ÁREA DE TRABALHO PARA CARTUCHOS

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

Capítulo 3
A MEMÓRIA ROM

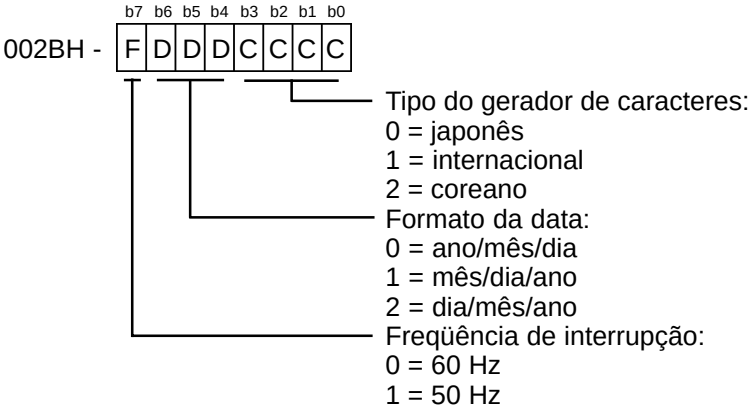
A memória ROM é vital para o funcionamento do micro. No caso do MSX, ela incorpora a rotina de inicialização, o BIOS, a tabela inicial de caracteres, o MSX-DOS (DOS Kernel), etc.

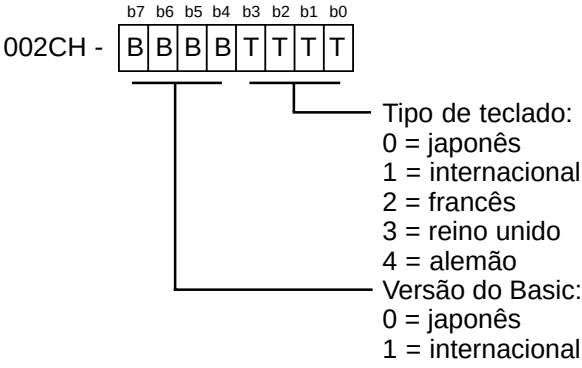
Além disso, existem alguns bytes no início da ROM que contêm algumas informações importantes que podem ser úteis ao programador. Esses bytes são:

0004H/0005H - Endereço da tabela de caracteres na ROM.

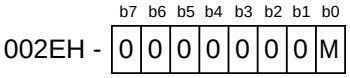
0006H - Porta de leitura de dados do VDP.

0007H - Porta de escrita de dados no VDP.





002DH - Versão do Hardware
00H = MSX1
01H = MSX2
02H = MSX2+
03H = MSX turbo R



MSX-MIDI:
0 = sem MSX-MIDI
1 = MSX-MIDI inclusa (MSX turbo R)

1 - BIOS

Praticamente todo programa, 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 do BIOS, que significa “Basic Input/Output System”, ou “Sistema Básico de Entrada e Saída”.

Esse capítulo fornece a descrição de 137 rotinas do BIOS disponíveis ao usuário, se o micro for um MSX turbo R. Para versões anteriores, o número de rotinas disponíveis diminui, mas isso estará descrito detalhadamente.

Existem dois tipos de rotinas do 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 seqüências de chamada. Para a Main-ROM, pode ser usada uma instrução CALL ou RST. As chamadas para a Sub-ROM serão descritas posteriormente.

As rotinas estão listadas conforme a seguinte notação:

LABEL (Endereço da rotina) *n
Função: descreve a função da rotina
Entrada: descreve os parâmetros para chamada da rotina
Saída: descreve os parâmetros de retorno da rotina
Registradores: descreve os registradores da CPU modificados pela rotina

A referência “*n” significa o seguinte:

*1 - rotina igual ao MSX1
*2 - chama a Sub-ROM internamente para as Screens 5 a 8
*3 - sempre chama a Sub-ROM
*4 - não chama a Sub-ROM enquanto as Screens 4 a 8 são trocadas.

1.1 - ROTINAS RST

Das rotinas RST listadas, de *RST 00H* até *RST 28H* são rotinas usadas pelo interpretador BASIC, mas também podem se usadas livremente por programas em Assembly. A *RST 30H* bem como a *RST 1CH* são usadas para chamadas inter-slot e a *RST 38H* é utilizada para interrupções de hardware.

CHKRAM (0000H) *1

Função: Testa a RAM na partida e inicializa as variáveis de sistema. Uma chamada a esta rotina provocará um reset por software.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

SYNCHR (0008H) *1

Função: Testa se o caractere apontado por (HL) é o especificado. Se não for, gera “Syntax error” (Erro de sintaxe), 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:

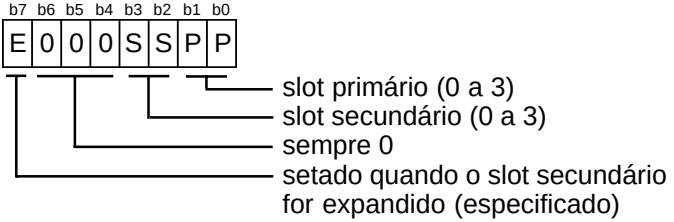
```
        LD   HL,CARACT
        RST  008H
        DEFB 'A'
        |
CARACT: DEFB 'B'
```

Saída: HL é incrementado em 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.
Registradores: AF, HL.

RDSLT (000CH) *1

Função: Lê um byte em qualquer slot. As interrupções são desabilitadas durante a leitura.

Entrada: A contém o indicador de slot.



HL - endereço de memória a ser lido.

Saída: A - contém o valor do byte lido.

Registradores: AF, BC, DE

CHRGTR (0010H) *1

Função: Lê um caractere ou um token do texto BASIC.

Entrada: (HL) aponta para o caracter a ser lido.

Saída: HL é incrementado em um 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.

WRSLT (0014H) *1

Função: Escreve um byte na RAM em qualquer slot. As interrupções ficam desabilitadas durante a escrita.

Entrada: A - indicador de slot (igual a RDSLT - 000CH).
HL - endereço para a escrita do byte.
E - byte a ser escrito.

Saída: Nenhuma.

Registradores: AF, BC, D.

OUTDO (0018H) *2

Função: Saída para o dispositivo atual.

Entrada: A - byte a ser enviado.
Se PTRFLG (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) *1
Função: Chama rotina em qualquer slot (chamada inter-slot).
Entrada: Especificar o byte ID de slot (igual a RDSLT - 000CH) nos 8 bits mais altos de IY. IX deve conter o endereço a ser chamado.
Saída: Depende da rotina chamada.
Registradores: Depende da rotina chamada.

DCOMPR (0020H) *1
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) *1
Função: Habilita uma página em qualquer slot. As interrupções são desativadas durante a habilitação. Somente as páginas 1 e 2 podem ser habilitadas por esta rotina, a 0 e 3 não.
Entrada: A - indicador de slot (igual a RDSLT - 000CH).
HL - qualquer endereço da página a ser habilitada.
Saída: Nenhuma.
Registradores: Todos.

GETYPR (0028H) *1
Função: Obtém o tipo de operando em DAC ou indicado por VALTYP.
Entrada: Nenhuma
Saída: Inteiro: A=FFH; flags C=1, S=1*, P/V=1, Z=0;
Simples precisão: A=01H; flags C=1, S=0, Z=0, P/V=0*
Dupla precisão: A=05H; flags C=0*, S=0, Z=0, P/V=1
String: A=00H; flags C=1, S=0, Z=1*, P/V=1
Obs: Os tipos podem ser reconhecidos unicamente pelas flags marcadas com "*".
Registradores: AF.

CALLF (0030H) *1
Função: Chama rotina em qualquer slot. Entretanto, ela usa parâmetros em linha ao invés de carregar diretamente os registradores como CALSLT, a fim de caber dentro dos hooks. A seqüência 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 da rotina chamada
```

Entrada: Pelo método já descrito.
Saída: Depende da rotina chamada.
Registradores: Depende da rotina chamada.

KEYINT (0038H) *1
Função: Executa a rotina de interrupção e varredura de teclado.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Nenhum.

1.2 - ROTINAS PARA INICIALIZAÇÃO DE I/O

INITIO (003BH) *1
Função: Inicializar o PSG e a porta de status da impressora.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

INIFNK (003EH) *1
Função: Inicializa o conteúdo das teclas de função.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

1.3 - ROTINAS PARA ACESSO AO VDP

DISSCR (0041H) *1
Função: Desabilita a saída de vídeo.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF, BC.

ENASCR (0044H) *1
Função: Habilita a saída de vídeo.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF, BC.

WRVDP (0047H) *2
Função: Escreve dados nos registradores do VDP.
Entrada: B - byte a ser escrito.
C - registrador que receberá o dado. Pode variar de 0 a 7 para o MSX1, de 0 a 23 / 32 a 46 para o MSX2 e de 0 a 23 / 25 a 27 / 32 a 46 para o msx2+ ou superior.
Saída: Nenhuma.
Registradores: AF, BC

RDVRM (004AH) *1
Função: Lê um byte da VRAM. Essa rotina acessa somente os 14 bits mais baixos do bus de endereços da VRAM (16 Kbytes para o VDP 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 - contém o valor do byte lido.
Registradores: AF.

WRTVRM (004DH) *1
Função: Escreve um byte na VRAM. Essa rotina acessa somente os 14 bits mais baixos do bus de endereços da VRAM (16 Kbytes para o VDP 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) *1
Função: Prepara a VRAM para leitura seqüencial usando a função de auto-incremento de endereço do VDP. É um meio de leitura mais rápido do que usando um loop com a rotina RDVRM. Esta rotina acessa somente os 14 bits mais baixos do bus de endereços da VRAM (16 Kbytes para o VDP TMS 9918 do MSX1). Para acessar toda a VRAM, é necessário usar a rotina NSETRD (016EH).
Entrada: HL - endereço da VRAM para início da leitura.
Saída: Nenhuma.
Registradores: AF.

SETWRT (0053H) *1
Função: Prepara a VRAM para escrita seqüencial usando a função de auto-incremento de endereço do VDP. As características são as mesmas de SETRD. Para acessar toda a VRAM, é necessário usar a rotina NSETRD (016EH).
Entrada: HL - endereço da VRAM para início da escrita.
Saída: Nenhuma.
Registradores: AF.

FILVRM (0056H) *4
Função: Preenche uma área da VRAM com um único byte de dados. Essa rotina acessa somente os 14 bits mais baixos do bus de endereços da VRAM (16 Kbytes para o TMS9918). Para acessar toda a VRAM, é necessário usar a rotina BIGFIL (016BH).

Entrada: HL - endereço da VRAM para início da escrita.
BC - quantidade de bytes a serem escritos (comprimento).
A - byte a ser escrito.
Saída: Nenhuma.
Registradores: AF, BC.

LDIRMV (005CH) *4
Função: Transfere um bloco de memória da VRAM para a RAM.
Entrada: HL - endereço fonte na VRAM.
DE - endereco destino na RAM.
BC - tamanho do bloco (comprimento).
Obs.: todos os bits de endereço são válidos.
Saída: Nenhuma.
Registradores: Todos.

LDIRVM (005CH) *4
Função: Transfere um bloco de memória da RAM para a VRAM.
Entrada: HL - endereço fonte na RAM.
DE - endereço destino na VRAM.
BC - tamanho do bloco (comprimento)
Obs.: todos os bits de endereço são válidos.
Saída: Nenhuma.
Registradores: Todos.

CHGMOD (005FH) *3
Função: Troca os modos de tela. No caso de micros MSX2 ou superior, a paleta de cores não é inicializada. Para inicializá-la, é necessário usar a rotina CHGMDP (01B5H/SUBROM).
Entrada: A - modo screen (0 a 3 para MSX1, 0 a 8 para MSX2 e 0 a 12 para MSX2+ ou superior. Obs.: modo 9 não existe).
Saída: Nenhuma
Registradores: Todos.

CHGCLR (0062H) *1
Função: Troca as cores da tela. No modo texto 40 ou 80 colunas, a cor da borda é sempre igual à cor de fundo.
Entrada: A - modo
FORCLR (F3E9H) - cor do primeiro plano.
BAKCLR (F3EAH) - cor de fundo.
BDRCLR (F3EBH) - cor da borda.
Saída: Nenhuma.
Registradores: Todos.

NMI (0066H) *1
Função: Executa a rotina NMI (Non-Maskable Interrupt Interrupt - Interrupção não mascarável). A rigor, apenas faz uma chamada ao hook HNMI (FDD6H) sem nenhum processamento.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Nenhum.

CLRSPR (0069H) *3
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 (screens 0 a 3) ou 217 (screens 4 a 8 ou 10 a 12).
Entrada: SCRMOD (FCAFH) deve conter o modo screen.
Saída: Nenhuma.
Registradores: Todos.

INITXT (006CH) *3
Função: Inicializa a tela no modo texto 1 (40 x 24). Nessa rotina, a paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/SUBROM).
Entrada: TXTNAM (F3B3H) - endereço da tabela de nomes dos caracteres.
TXTCGP (F3B7H) - endereço da tabela geradora de padrões dos caracteres.
LINL40 (F3AEH) - largura das linhas em caracteres.
Saída: Nenhuma.
Registradores: Todos.

INIT32 (006FH) *3
Função: Inicializa a tela no modo gráfico 1 (32 x 24). Nessa rotina, a paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/SUBROM).
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.
T32ATR (F3C3H) - endereço da tabela de atributos dos sprites.
T32PAT (F3C5H) - endereço da tabela de padrões dos sprites.
Saída: Nenhuma.
Registradores: Todos.

INIGRP (0072H) *3

Função: Inicializa a tela no modo gráfico de alta resolução (screen 2). Nessa rotina, a paleta de cores não é inicializada. Para inicializá-la, é necessário chamar INIPLT (0141H/SUBROM).

Entrada: GRPNAM (F3C7H) - endereço da tabela de nomes dos caracteres.
GRPCOL (F3C9H) - endereço da tabela de cores dos caracteres.
GRPCGP (F3CBH) - endereço da tabela de padrões dos caracteres.
GRPATR (F3CDH) - endereço da tabela de atributos dos sprites.
GRPPAT (F3CFH) - endereço da tabela de padrões dos sprites.

Saída: Nenhuma.

Registradores: Todos.

INIMLT (0075H) *3

Função: Inicializa a tela no modo multicor (screen 3). Nessa rotina, a paleta de cores não é inicializada. Para inicializá-la, é necessário chamar a rotina INIPLT (0141H/SUBROM).

Entrada: MLTNAM (F3D1H) - endereço da tabela de nomes dos caracteres.
MLTCOL (F3D3H) - endereço da tabela de cores dos caracteres.
MLTCGP (F3D5H) - endereço da tabela de padrões dos caracteres.
MLTATR (F3D7H) - endereço da tabela de atributos dos sprites.
MLTPAT (F3D9H) - endereço da tabela de padrões dos sprites.

Saída: Nenhuma.

Registradores: Todos.

SETTXT (0078H) *3

Função: Coloca apenas o VDP no modo texto 1 (40 x 24).

Entrada: Igual a INITXT (006CH).

Saída: Nenhuma.

Registradores: Todos.

SET32 (007BH) *3

Função: Coloca apenas o VDP no modo gráfico 1 (32 x 24).

Entrada: Igual a INIT32 (006FH).

Saída: Nenhuma.

Registradores: Todos.

SETGRP (007EH) *3
Função: Coloca apenas o VDP no modo gráfico 2 (screen 2).
Entrada: Igual a INIGRP (0072H).
Saída: Nenhuma.
Registradores: Todos.

SETMLT (0081H) *3
Função: Coloca apenas o VDP no modo multicor (screen 3).
Entrada: Igual a INIMLT (0075H).
Saída: Nenhuma.
Registradores: Todos.

CALPAT (0084H) *1
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) *1
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) *1
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) *2
Função: Apresenta um caractere numa tela gráfica.
Entrada: A - código ASCII do caractere. Quando a screen for 5 a 8 ou 10 a 12, é necessário colocar o código de operação lógica em LOGOPR (FB02H).
Saída: Nenhuma.
Registradores: Nenhum.

1.4 - ROTINAS PARA ACESSO AO PSG

GICINI (0090H) *1
Função: Inicializa o PSG para o comando PLAY do BASIC. O volume das três vozes é colocado em 0 e o registrador 7 inicializado com B8H, ativando os geradores de tom e desativando o gerador de ruído branco.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

WRTPSG (0093H) *1
Função: Escreve dados nos registradores do PSG.
Entrada: A - número do registrador para escrita.
E - byte a ser escrito.
Saída: Nenhuma
Registradores: Nenhum.

RDPSG (0096H) *1
Função: Lê o conteúdo dos registradores do PSG.
Entrada: A - número do registrador do PSG a ser lido.
Saída: A - byte lido.
Registradores: Nenhum.

STRTMS (0099H) *1
Função: Testa se o comando PLAY está sendo executado. Se não estiver, inicia a execução, desempilhando as filas musicais.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

1.5 - ROTINAS PARA ACESSO AO TECLADO, TELA E IMPRESSORA

CHSNS (009CH) *1
Função: Verifica o buffer de teclado.
Entrada: Nenhuma.
Saída: A flag Z é setada se o buffer estiver vazio, caso contrário é resetada.
Registradores: AF.

CHGET (009FH) *1
Função: Entrada de um caractere pelo teclado, com espera.
Entrada: Nenhuma.
Saída: A - código ASCII do caractere.
Registradores: AF.

CHPUT (00A2H) *1
Função: Apresenta um caractere na tela.
Entrada: A - código ASCII do caractere a ser apresentado.
Saída: Nenhuma.
Registradores: Nenhuma.

LPTOUT (00A5H) *1
Função: Envia um caractere para a impressora.
Entrada: A - código ASCII do caractere a ser enviado.
Saída: A flag CY é setada se a operação falhar e resetada se a operação for realizada com sucesso.
Registradores: F.

LPTSTT (00A8H) *1
Função: Retorna o status da impressora.
Entrada: Nenhuma.
Saída: Quando A=FFH e a flag Z estiver resetada, a impressora está pronta. Quando A=0 e a flag Z estiver setada, a impressora não está pronta para receber dados.
Registradores: AF.

CNVCHR (00ABH) *1
Função: Testa o cabeçalho gráfico e o converte se necessário.
Entrada: A - código ASCII do caractere.
Saída: A flag CY é resetada se não houver cabeçalho gráfico; as flags CY e Z são setadas e o código convertico colocado em A; se a flag CY é setada e a flag Z resetada, o código não convertido é colocado em A.
Registradores: AF.

PINLIN (00AEH) *1
Função: Coleta uma linha de texto do console e a armazena em um buffer especificado até que a tecla RETURN ou CTRL+STOP seja pressionada.
Entrada: Nenhuma.
Saída: HL - endereço de início do buffer menos 1.
Se a flag CY estiver setada, foi pressionado CTRL+STOP.
Registradores: Todos.

INLIN (00B1H) *1
Função: Mesma que PINLIN, exceto que AUTFLG (F6AAH) é setada.
Entrada: Nenhuma.
Saída: Mesma de PINLIN (00AEH).
Registradores: Todos.

QINLIN (00B4H) *1
Função: Executa INLIN (00B1H) apresentando "?" e um espaço.
Entrada: Nenhuma.
Saída: Mesma de PINLIN (00AEH).
Registradores: Todos.

BREAKX (00B7H) *1
Função: Verifica diretamente as teclas CTRL+STOP. Nessa rotina, as interrupções são desabilitadas.
Entrada: Nenhuma.
Saída: A flag CY é setada se CTRL+STOP estiverem pressionadas.
Registradores: AF.

ISCNTC (00BAH) *?
Função: Verifica as teclas CTRL+STOP ou STOP. É usada principalmente pelo interpretador BASIC. Se CTRL+STOP forem pressionadas, o controle é devolvido ao interpretador; se STOP for pressionada, paralisa a execução de um programa BASIC até que CTRL+STOP ou STOP sejam pressionadas novamente.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

CKCNTC (00BDH) *?
Função: Mesma que ISCCNT (00BAH), exceto que o programa BASIC não poderá ser continuado pela instrução CONT.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

BEEP (00C0H) *3
Função: Gera um beep.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

CLS (00C3H) *3
Função: Limpa a tela.
Entrada: A flag Z deve estar setada.
Saída: Nenhuma.
Registradores: AF, BC, DE

POSIT (00C6H) *1
Função: Move o cursor pelas telas de texto.
Entrada: H - coordenada X (horizontal) do cursor.
L - coordenada Y (vertical) do cursor.
Saída: Nenhuma.
Registradores: AF.

FNKSB (00C9H) *1
Função: Testa se o display das teclas de função está ligado através de FNKFLG (FBCEH). Se estiver, desliga e se não estiver, liga.
Entrada: FNKFLG (FBCEH).
Saída: Nenhuma.
Registradores: Todos.

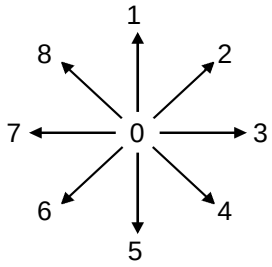
ERAFNK (00CCH) *1
Função: Desliga o display das teclas de função.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

DSPFNK (00CFH) *2
Função: Liga o display das teclas de função.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

TOTEXT (00D2H) *1
Função: Força a tela para o modo texto (screen 0 ou 1).
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

1.6 - ROTINAS DE ACESSO I/O PARA GAMES

GTSTCK (00D5H) *1
Função: Retorna o status do joystick ou 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 teclas de função, conforme ilustração da página seguinte.



Registradores: Todos.

GTTRIG (00D8H) *1

Função: Retorna o estado dos botões do joystick, do mouse ou da barra de espaço.

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 se o botão testado não estiver pressionado, e FFH se o botão testado estiver pressionado.

Registradores: AF, BC.

GTPAD (00DBH) *1

Função: Retorna o status do touch-pad (digitalizador) ligado a um dos conectores do joystick. Também utilizada para retornar os valores de um mouse ligado ao mesmo conector.

Entrada: A - código de função (0 a 3 para a porta 1 e 4 a 7 para a porta 2):
0 ou 4 - retorna o status de atividade.
1 ou 5 - retorna a coordenada “X” (horizontal).
2 ou 6 - retorna a coordenada “Y” (vertical).
3 ou 7 - retorna o status da tecla.

Saída: A - status ou valor. Para coordenada X ou Y, varia de 0 a 255; para status de atividade, devolve 255 se o touch-pad estiver sendo tocado e 0 caso contrário; para status de tecla, devolve 255 se esta estiver sendo pressionada e 0 caso contrário.

Registradores: Todos.

Obs.: Essa rotina foi modificada nos modelos MSX turbo R.

GTPDL (00DEH) *2

Função: Retorna o status dos paddles ligados aos conectores de joystick.

Entrada: A - identificação do paddle (1 a 12)
1, 3, 5, 7, 9, 11 - paddles ligados na porta A;
2, 4, 6, 8, 10 ,12 - paddles ligados na porta B.
Saída: A - valor lido (0 a 255)
Registradores: Todos.
Obs.: Essa rotina foi modificada nos modelos MSX turbo R.

1.7 - ROTINAS PARA O CASSETE

TAPION (00E1H)
TAPIN (00E4H)
TAPIOF (00E7H)
TAPOON (00EAH)
TAPOUT (00EDH)
TAPOOF (00F0H)
STMOTR (00F3H)

As rotinas acima se referiam à interface de cassete e se tornaram totalmente obsoletas, sendo eliminadas nos modelos MSX turbo R, motivos pelos quais não serão descritas.

1.8 - ROTINAS PARA A FILA DO PSG

LFTQ (00F6H) *?
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) *?
Função: Coloca um byte em uma das três filas musicais do PSG.
A - número da fila (0, 1 ou 2)
Entrada: E - byte da dados
Saída: Flag Z setada se a fila estiver cheia.
Registradores: AF, BC, HL.

1.9 - ROTINAS PARA AS TELAS GRÁFICAS DO MSX1

RIGHTC (00FCH) *1
Função: Move o endereço físico do pixel atual uma posição à direita.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

LEFTC (00FCH) *1
Função: Move o endereço físico do pixel atual uma posição à esquerda.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

UPC (0102H) *1
Função: Move endereço físico do pixel atual uma posição para cima.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

TUPC (0105H) *1
Função: Testa e move o endereço físico do pixel atual uma posição para cima.
Entrada: Nenhuma.
Saída: Se o pixel excedeu a parte superior da tela, a flag C é setada.
Registradores: AF.

DOWNC (0108H) *1
Função: Move o endereço físico do pixel atual uma posição para baixo.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF.

TDOWNC (010BH) *1
Função: Testa e move o endereço físico do pixel atual uma posição para baixo.
Entrada: Nenhuma.
Saída: Se o pixel excedeu a parte inferior da tela, a flag C é setada.
Registradores: AF.

SCALXY (010EH) *1
Função: Converte as coordenadas do pixel para a área visível da tela.
Entrada: BC - Coordenada X (horizontal).
DE - Coordenada Y (vertical).
Saída: Se houver corte (clipping), a flag C será resetada.
Registradores: AF.

MAPXYC (0111H) *1
Função: Converte um par de coordenadas gráficas no endereço físico atual do pixel.
Entrada: BC - Coordenada X (horizontal).
DE - Coordenada Y (vertical).
Saída: Nenhuma.
Registradores: AF, D, HL.

FETCHC (0114H) *1
Função: Retorna o edereço físico do pixel atual.
Entrada: Nenhuma.
Saída: A - recebe CMASK (F92CH).
HL - recebe CLOC (F92AH).
Registradores: A, HL.

STOREC (0117H) *1
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.

SETATR (011AH) *1
Função: Estabelece a cor de frente para as rotinas SETC (0120H) e NSETCX (0123H).
Entrada: A - código de cor (0 a 15).
Saída: Se o código for inválido, flag C retorna setada.
Registradores: F.

READC (011DH) *1
Função: Devolve o código de cor do pixel atual.
Entrada: Nenhuma.
Saída: A - código de cor do pixel atual.
Registradores: AF, EI.

SETC (0120H) *1
Função: Estabelece uma cor ao pixel atual.
Entrada: ATRBYT (F3F2H) - código de cor
Saída: Nenhuma.
Registradores: AF, EI.

NSETCX (0123H) *1
Função: Estabelece a cor de múltiplos pixels horizontais a partir do pixel atual, para a direita.
Entrada: HL - número de pixels a colorir.
Saída: Nenhuma.
Registradores: AF, BC, DE, HL, EI.

GTASPC (0126H) *1
Função: Retorna as razões de aspecto da instrução CIRCLE.
Entrada: Nenhuma.
Saída: DE - recebe ASPCT1 (F40BH).
HL - recebe ASPCT2 (F40DH).
Registradores: DE, HL.

PNTINI (0129H) *1
Função: Estabelece a cor de contorno para a instrução PAINT.
Entrada: A - código de cor do contorno (0 a 15).
Saída: Se a cor for inválida, a flag C será setada.
Registradores: AF.

SCANR (012CH) *1
Função: Usada pelo manipulador da instrução PAINT para percorrer, para a direita, a partir do endereço físico 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 com a cor especificada em BDRATR (FCB2H).
DE - número de pulos (pixels da mesma cor ignorados).
Saída: HL - número de pixels percorridos.
DE - número de pulos restantes.
Registradores: AF, BC, DE, HL, EI.

SCANL (012FH) *1
Função: Mesma que SCANR, exceto que a área percorrida será para a esquerda e será sempre preenchida.
Entrada: Nenhuma.
Saída: HL - número de pixels percorridos.
Registradores: AF, BC, DE, HL, EI.

1.10 - MISCELÂNEA

CHGCAP (0132H) *1
Função: Altera o estado do LED do Caps Lock.
Entrada: A - 0 = apaga o LED; outro valor = acende o LED.
Saída: Nenhuma.
Registradores: AF.

CHGSND (0135H) *1
Função: Liga ou desliga o “click” das teclas.
Entrada: A - 0 = desliga o “click”; outro valor = liga o “click”.
Saída: Nenhuma.
Registradores: AF.

RSLREG (0138H) *1
Função: Lê o conteúdo do registrador de slot primário.
Entrada: Nenhuma.
Saída: A - valor lido.
Registradores: A.

WSLREG (013BH) *1
Função: Escreve um valor no registrador de slot primário.
Entrada: A - valor a ser escrito.
Saída: Nenhuma.
Registradores: Nenhum.

RDVDP (013EH) *1
Função: Lê o registrador de status do VDP.
Entrada: Nenhuma.
Saída: A - valor lido.
Registradores: A.

SNSMAT (0141H) *1
Função: Lê uma linha da matriz do teclado.
Entrada: A - número da linha do teclado a ser lida.
Saída: A - colunas lidas da linha especificada. O bit correspondente a uma tecla pressionada é 0.
Registradores: AF, C.

ISFLIO (014AH) *1
Função: Testa se está ocorrendo I/O de arquivo.
Entrada: Nenhuma.
Saída: A - 0 se estiver ocorrendo uma operação de I/O de arquivo; outro valor caso contrário.
Registradores: AF.

OUTDLP (014DH) *1
Função: Saída formatada para a impressora. Difere de LPTOUT (00A5H) nos seguintes pontos: se o caractere for um código TAB (09H), serão enviados espaços até atingir um múltiplo de 8; para impressoras não MSX, caracteres gráficos são transformados em caracteres de 1 byte; se houver falha, ocorre um erro de I/O.
Entrada: A - byte a ser enviado para a impressora.
Saída: Nenhuma.
Registradores: F.

GETVCP (0150H) *?
Função: Retorna o endereço do byte 2 no buffer de voz especificado do PSG.
Entrada: A - número da voz.
Saída: HL - endereço no buffer de voz.
Registradores: AF, HL.

GETVC2 (0153H) *?
Função: Retorna o endereço de qualquer byte no buffer de voz especificado pelo número da voz em VOICEN (FB38H)
Entrada: L - número do byte do bloco (0 a 36).
Saída: HL - endereço no buffer de voz.
Registradores: AF, HL.

KILBUF (0156H) *1
Função: Limpa o buffer do teclado.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: HL.

CALBAS (0159H) *1
Função: Executa uma chamada inter-slot para qualquer rotina do interpretador BASIC.
Entrada: IX - endereço da rotina a ser chamada.
Saída: Depende da rotina chamada.
Registradores: Depende da rotina chamada.

1.11 - ROTINAS PARA ACESSO AO SISTEMA DE DISCO

PHYDIO (0144H) *1
Função: Ler ou gravar um ou mais setores no drive especificado.
Entrada: HL - endereço da RAM a partir do qual serão colocados os setores a ler ou retirados os setores a gravar.
DE - número do primeiro setor a ser lido ou gravado.
B - número de setores a ler ou gravar.
C - parâmetro de formatação do disco:
F8H - 80 trilhas, face simples;
F9H - 80 trilhas, face dupla;
FCH - 40 trilhas, face simples;
FDH - 40 trilhas, face dupla.
A - número do drive (0=A, 1=B, etc.)
Flag CY - resetada para fazer leitura, setada para fazer gravação.
Saída: Flag CY - se estiver setada, houve algum tipo de erro (leitura ou gravação; resetada se a operação foi realizada com sucesso.
Registradores: Todos.

FORMAT (0147H) *1
Função: Formatar um disquete. Ao ser chamada, serão apresentadas uma série de perguntas que deverão ser respondidas para iniciar a formatação. Infelizmente, não há um padrão, e as perguntas podem ser diferentes para cada interface de drive.

Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

1.12 - ROTINAS ADICIONADAS PARA O MSX2

SUBROM (015CH)
Função: Executa uma chamada inter-slot para a Sub-ROM.
Entrada: IX - endereço da rotina a ser chamada (ao mesmo tempo, salva IX na pilha).
Saída: Depende da rotina chamada.
Registradores: O registrador de fundo e IY são reservados.

EXTROM (015FH)
Função: Executa uma chamada inter-slot para a Sub-ROM.
Entrada: IX - endereço da rotina a ser chamada.
Saída: Depende da rotina chamada.
Registradores: O registrador de fundo e IY são reservados.

EOL (0168H)
Função: Apaga até o fim da linha.
Entrada: H - coordenada X (horizontal) do cursor.
L - coordenada Y (vertical) do cursor.
Saída: Nenhuma.
Registradores: Todos.

BIGFIL (016BH)
Função: Mesma de FILVRM (0056H), com as seguintes diferenças: na FILVRM, são testadas as screens 0 a 3, e nesse caso o VDP é acionado para acessar 16 Kbytes apenas, para compatibilidade com o MSX1. Na BIGFIL, o modo não é testado e as ações são levadas para fora pelos parâmetros dados.
Entrada: HL - endereço na VRAM para o início da escrita.
BC - comprimento (número de bytes a escrever).
A - dado a ser escrito.
Saída: Nenhuma.
Registradores: AF, BC.

NSETRD (016EH)
Função: Prepara a VRAM para leitura seqüencial, usando a função de auto-incremento 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)
Função: Prepara a VRAM para escrita seqüencial, usando a função de auto-incremento 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: Nenhum.

NRDVRM (0174H)
Função: Lê o conteúdo de um byte da VRAM.
Entrada: HL - endereço na VRAM do byte a ser lido.
Saída: A - byte lido.
Registradores: F.

NWRVRM (0177H)
Função: Escreve um byte de dados na VRAM.
Entrada: HL - endereço na VRAM do byte a ser escrito.
A - byte a ser escrito.
Saída: Nenhuma
Registradores: AF.

1.13 - ROTINAS ADICIONADAS PARA O MSX2+

RDRES (017AH)
Função: Verifica o status do reset.
Entrada: Nenhuma.
Saída: A - b7=0 indica reset total[4]
b7=1 indica reset parcial[4]
Registradores: Nenhum.

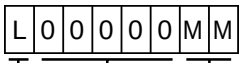
WRRES (017DH)
Função: Modifica o status do reset.
Entrada: A - b7=0 para reset total[4]
b7=1 para reset parcial[4]
Saída: Nenhuma.
Registradores: Nenhum.
Obs. Essa rotina deve sempre ser chamada antes de executar um salto para o endereço 0000H, da seguinte forma: lê-se o estado por meio da rotina RDRES (017AH), efetua-se um OR 80H no valor lido e escreve-se o dado novamente usando essa rotina. Só então executa-se o salto para 0000H.

Nota 4: No reset total, o conteúdo da RAM é apagado e aparece o logo “MSX” na inicialização. No caso de reset parcial, não aparece o logo na tela e o conteúdo da RAM não é apagado (apenas a área de trabalho é inicializada).

1.14 - ROTINAS ADICIONADAS PARA O MSX turbo R

CHGCPU (0180H)
Função: Trocar de microprocessador (modo de operação).

Entrada: A -



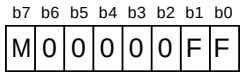
Modo de operação:
00 - Z80
01 - R800 ROM
10 - R800 DRAM
Sempre 0
LED de modo no painel
0 - apagado
1 - aceso
Obs.: o estado do LED está atrelado ao modo de operação. Este tem que ser R800 (ROM ou DRAM) para que o LED acenda.

Saída: Nenhuma.
Registradores: AF.

GETCPU (0183H)
Função: Verificar em qual modo o computador está operando.
Entrada: Nenhuma.
Saída: A - 0=Z80; 1=R800 ROM; 2=R800 DRAM.
Registradores: AF.

PCMPLY (0186H)
Função: Reproduzir o som pelo PCM.
Entrada: HL - endereço de início para leitura.
BC - tamanho do bloco a reproduzir (comprimento).

A -

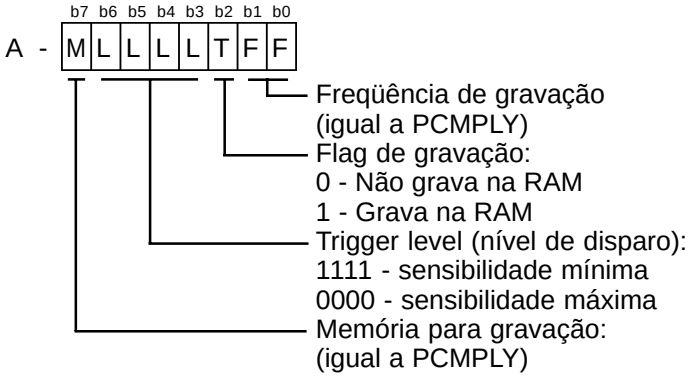


Freqüência de reprodução:
00 - 15,75 KHz
01 - 7,875 KHz
10 - 5,25 KHz
11 - 3,9375 KHz
Sempre 0
Memória para leitura:
0 - Main RAM
1 - VRAM

Obs.: A freqüência de 15,75 KHz só pode ser usada no modo R800 DRAM.

Saída: Flag CY: resetada: parou
setada: parou porque houve erro
A: 1: tem erro na freqüência.
2: foi pressionada STOP.
Registradores: AF, BC, HL.

PCMREC (0189H)
Função: Digitalizar sons através do PCM.
Entrada: Igual a PCMPLY, exceto para o registrador A:



Saída: Mesma de PCMPLY(0186H).
Registradores: AF, BC, HL.

1.15 - ROTINAS DA SUB-ROM

A seqüência de chamada para as rotinas da Sub-ROM é feita com o auxílio da rotina EXTROM (015FH) ou SUBROM (015CH), 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
                        ;da rotina
        CALL EXTROM     ;executa a rotina
        ...             ;retorno da rotina aqui
```

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

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

DESCRIÇÃO DAS ROTINAS

GRPPRT (0089H)
Função: Imprime um caractere na tela gráfica (válida somente para as screens 5 a 8 e 10 a 12).
Entrada: A - código ASCII do caractere.
Saída: Nenhuma.
Registradores: Nenhum.

NVBXLN (00C9H)
Função: Desenha uma caixa nas telas gráficas.
Entrada: Ponto inicial: BC - coordenada X (horizontal).
DE - coordenada Y (vertical).
Ponto final: GXPOS (FCB3H) - coordenada X (horizontal).
GYPOS (FCB5H) - coordenada Y (vertical)
Cor: ATRBYT (F3F2H) - atributo.
Código de operação lógica: LOGOPR (FB02H).
Saída: Nenhuma.
Registradores: Todos.

NVBXFL (00CDH)
Função Desenha uma caixa pintada.
Entrada: Mesma de NVBXLN (00C9H).
Saída: Nenhuma.
Registradores: Todos.

CHGMOD (00D1H)
Função: Troca os modos de tela.
Entrada: A - modo screen (0 a 8 ou 10 a 12).
Saída: Nenhuma.
Registradores: Todos.

INITXT (00D5H)
Função: Inicializa a tela no modo texto (40 x 24).
Entrada: TXTNAM (F3B3H) - endereço da tabela de nomes dos caracteres.
TXTCGP (B3B7H) - endereço da tabela geradora de padrões dos caracteres.
Saída: Nenhuma.
Registradores: Todos.

INIT32 (00D9H)
Função: Inicializa a tela no modo texto (32 x 24).
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.
T32ATR (F3C3H) - endereço da tabela de atributos dos sprites.
T32PAT (F3C5H) - endereço da tabela de padrões dos sprites.

Saída: Nenhuma.
Registradores: Todos.

INIGRP (00DDH)
Função: Inicializa a tela no modo gráfico screen 2.
Entrada: GRPNAM (F3C7H) - endereço da tabela de nomes dos carac-teres.
GRPCOL (F3C9H) - endereço da tabela de cores dos caracteres.
GRPCGP (F3CBH) - endereço da tabela de padrões dos caracteres.
GRPATR (F3CDH) - endereço da tabela de atributos dos sprites.
GRPPAT (F3CFH) - endereço da tabela de padrões dos sprites.
Saída: Nenhuma.
Registradores: Todos.

INIMLT (00E1H)
Função: Inicializa a tela no modo multicor (screen 3).
Entrada: MLTNAM (F3D1H) - endereço da tabela de nomes dos caracteres.
MLTCOL (F3D3H) - endereço da tabela de cores dos caracteres.
MLTCGP (F3D5H) - endereço da tabela de padrões dos caracteres.
MLTATR (F3D7H) - endereço da tabela de atributos dos sprites.
MLTPAT (F3D9H) - endereço da tabela de padrões dos sprites.
Saída: Nenhuma
Registradores: Todos.

SETTXT (00E5H)
Função: Coloca apenas o VDP no modo texto (40 x 24).
Entrada: Mesma de INITXT (00D5H/SUBROM).
Saída: Nenhuma.
Registradores: Todos.

SET32 (00E9H)
Função: Coloca apenas o VDP no modo texto 2 (32 x 24).
Entrada: Mesma de INIT32 (00D9H/SUBROM).
Saída: Nenhuma.
Registradores: Todos.

SETGRP (00EDH)
Função: Coloca apenas o VDP no modo gráfico de alta resolução (Screen 2).
Entrada: Mesma que INIGRP (00DDH/SUBROM).
Saída: Nenhuma.
Registradores: Todos.

SETMLT (00F1H)
Função: Coloca apenas o VDP no modo multicor (Screen 3).
Entrada: Mesma que INIMLT (00E1H/SUBROM).
Saída: Nenhuma.
Registradores: Todos.

CLRSPR (00F5H)
Função: Inicializa todos os sprites. A tabela de padrões dos sprites é limpa (preenchida com zeros), os números dos sprites são inicalizados com a série 0~31, a cor dos sprites é igualada à cor de fundo e a localização vertical dos sprites é colocada em 217.
Entrada: SCRMOD (FCAFH) deve conter o modo screen.
Saída: Nenhuma.
Registradores: Todos.

CALPAT (00F9H)
Função: Retorna o endereço da tabela geradora do padrão de um sprite (essa rotina é a mesma que CALPAT (0084H) na Main-ROM).
Entrada: A - número do sprite
Saída: HL - endereço na VRAM
Registradores: AF, DE, HL

CALATR (00FDH)
Função: Retorna o endereço da tabela de atributos de um sprite (essa rotina é a mesma que CALATR (0087H) na Main-ROM).
Entrada: A - número do sprite
Saída: HL - endereço na VRAM.
Registradores: AF, DE, HL.

GSPSIZ (0101H)
Função: Retorna o tamanho atual dos sprites (essa rotina é a mesma que GSPSIZ (008AH) na Main-ROM).
Entrada: Nenhuma.

Saída: A - tamanho dos sprites em bytes. A flag CY é setada se o tamanho for 16x16 e resetada caso contrário.
Registradores: AF.

GETPAT (0105H)
Função: Retorna o padrão de um caractere.
Entrada: A - código ASCII do caractere.
Saída: PATWRK (FC40H) - padrão do caractere.
Registradores: Todos.

WRTVRM (0109H)
Função: Escreve um byte de dados na VRAM.
Entrada: HL - endereço da VRAM (0000H a FFFFH).
A - byte a ser escrito.
Saída: Nenhuma.
Registradores: AF.

RDVRM (010DH)
Função: Lê o conteúdo de um byte da VRAM.
Entrada: HL - endereço da VRAM (0000H a FFFFH).
Saída: A - byte lido.
Registradores: AF.

CHGCLR (0111H)
Função: Troca as cores da tela.
Entrada: A - modo screen da tela.
FORCLR (F3E9H) - cor da frente.
BAKCLR (F3EAH) - cor do fundo.
BDRCLR (F3EBH) - cor da borda.
Saída: Nenhuma.
Registradores: Todos.

CLSSUB (0115H)
Função: Limpar a tela.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

DSPFNK (011DH)
Função: Apresenta o conteúdo das teclas de função.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

WRTVDP (012DH)
Função: Escreve um byte de dados em um registrador do VDP.

Entrada: C - número do registrador.
B - byte a ser escrito.
Saída: Nenhuma.
Registradores: AF, BC.

VDPSTA (0131H)
Função: Lê o conteúdo de um registrador do VDP.
Entrada: A - número do registrador a ser lido (0 a 9).
Saída: A - byte de dado lido.
Registradores: F.

SETPAG (013DH)
Função: Alterna as páginas de vídeo.
Entrada: DPPAGE (FAF5H) - número da página apresentada no vídeo.
ACPAGE (FAF6H) - número da página ativa para comandos.
Saída: Nenhuma.
Registradores: AF.

INIPLT (0141H)
Função: Inicializa a paleta de cores (a paleta atual é gravada na VRAM).
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF, BC, DE.

RSTPLT (0145H)
Função: Recupera a paleta de cores gravada na VRAM.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: AF, BC, DE.

GETPLT (0149H)
Função: Retorna os códigos de cores da paleta.
Entrada: A - número da paleta (0 a 15).
Saída: B - 4 bits altos para o código do vermelho;
B - 4 bits baixos para o código do azul;
C - 4 bits baixos para o código do verde.
Registradores: AF, DE.

SETPLT (014DH)
Função: Modifica os códigos de cores da paleta.
Entrada: D - número da paleta (0 a 15).
A - 4 bits altos para o código do vermelho;
A - 4 bits baixos para o código do azul;
E - 4 bits baixos para o código do verde.
Saída: Nenhuma.
Registradores: AF, DE.

BEEP (017DH)
Função: Gera um beep.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

PROMPT (0181H)
Função: Apresenta o sinal de prompt.
Entrada: Nenhuma.
Saída: Nenhuma.
Registradores: Todos.

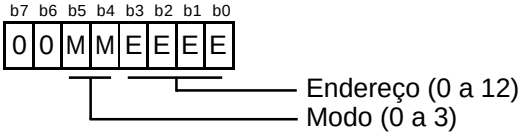
NEWPAD (01ADH)
Função: Lê o estado do mouse ou da caneta ótica.
Entrada: A - deve conter os valores para chamada descritos abaixo (as descrições entre parênteses são valores de retorno, sempre em A).
0 a 7 - sem efeito;
8 - checa se a caneta ótica está conectada (se estiver, A=255);
9 - retorna a coordenada X (horizontal) em A;
10 - retorna a coordenada Y (vertical) em A;
11 - retorna o estado da chave da caneta ótica (se estiver pressionada, A=255);
12 - checa se o mouse está conectado na porta 1 do joystick (se estiver, A=255);
13 - retorna a coordenada na direção X (horizontal) em A, para mouse conectado na porta 1;
14 - retorna a coordenada na direção Y (vertical) em A, para mouse conectado na porta 1;
15 - sempre 0
16 - checa se o mouse está conectado na porta 2 do joystick (se estiver, A=255);
17 - retorna a coordenada na direção X (horizontal) em A, para mouse conectado na porta 2;
18 - retorna a coordenada na direção Y (vertical) em A, para mouse conectado na porta 2;
19 - sempre 0.
Saída: A - contém os valores de retorno, conforme descrito acima.
Registradores: Todos.

Obs: essa rotina foi modificada nos modelos MSX turbo R.

CHGMDP (01B5H)
Função: Troca o modo do VDP. A paleta de cores é inicializada.
Entrada: A - modo screen (0 a 8 para MSX2 e 0 a 8 / 10 a 12 para MSX2+ ou superior).

Saída: Nenhuma.
Registradores: Todos.

REDCLK (01F5H)
Função: Lê um nibble de dados da memória do relógio.
Entrada: C - endereço da SRAM do relógio, conforme abaixo:



Saída: A - nibble lido (apenas os 4 bits mais baixos são válidos)
Registradores: AF.

WRTCLK (01F9H)
Função: Escreve um nibble de dados na memória do relógio.
Entrada: A - nibble a ser escrito (apenas os 4 bits baixos são válidos).
C - endereço da SRAM do relógio (igual a REDCLK).
Saída: Nenhuma.
Registradores: F.

1.16 - ROTINAS DE TRANSFERÊNCIA DE DADOS (BIT BLOCK TRANSFER)

Esse conjunto de rotinas da Sub-ROM foi desenvolvido para a transferência de dados entre a RAM, VRAM e disco, de forma semelhante ao comando COPY do BASIC. Essas rotinas são de fácil execução, tornando disponíveis para programas Assembly funções de transferência de dados de forma fácil, rápida e segura.

BLTVV (0191H)
Função: Transfere dados de uma área da VRAM para outra.
Entrada:

HL		Deve conter o valor F562H.
SX	(F562H,2)	- coordenada X (horizontal) da fonte;
SY	(F564H,2)	- coordenada Y (vertical) da fonte;
DX	(F566H,2)	- coordenada X (horizontal) do destino;
DY	(F568H,2)	- coordenada Y (vertical) do destino;
NX	(F56AH,2)	- número de pontos na direção X
NY		(horizontal);
CDUMMY	(F56CH,2)	- número de pontos na direção Y
ARGT		(vertical);
LOGOP	(F56EH,2)	- dummy (não requer dados);
	(F56FH,1)	- seleciona a direção e expansão da VRAM (igual a R#45 do VDP);
	(F570H,1)	- código de operação lógica (igual aos códigos do VDP).

Saída: A flag CY é resetada.
Registradores: Todos.
Obs.: o número após os endereços dados representa a quantidade de bytes que a variável de sistema requer. Essa representação será usada daqui em diante.

As rotinas seguintes requerem que o espaço de memória a ser movido seja alocado da seguinte forma para cada screen:

SCREEN 6:
[(pontos na direção X) * (pontos na direção Y)] / 4 + 4

SCREENS 5 e 7:
[(pontos na direção X) * (pontos na direção Y)] / 2 + 4

SCREENS 8, 10, 11 E 12:
[(pontos na direção X) * (pontos na direção Y)] + 4

BLTVM (0195H)
Função: Transfere dados da RAM para a VRAM
Entrada: HL Deve conter o valor F562H.
DPTR (F562H,2) - endereço-fonte na RAM;
DUMMY (F564H,2) - dummy (não requer dados);
DX (F566H,2) - coordenada X (horizontal) de destino;
DY (F568H,2) - coordenada Y (vertical) de destino;
NX (F56AH,2) - número de pontos na direção X (não requer dados; já está setada);
NY (F56CH,2) - número de pontos na direção Y (não requer dados; já está setada);
CDUMMY (F56EH,2) - dummy (não requer dados);
ARGT (F56FH,1) - seleciona a direção e a expansão da VRAM (igual a R#45 do VDP);
LOGOP (F570H,1) - código de operação lógica (igual aos códigos do VDP)
Saída: A flag CY é setada se o número de bytes a transferir estiver incorreto.
Registradores: Todos.

BLTMV (0199H)
Função: Tranfere dados da VRAM para a Main-RAM.
Entrada: HL Deve conter o valor F562H.
SX (F562H,2) - coordenada X (horizontal) da fonte;
SY (F564H,2) - coordenada Y (vertical) da fonte;
DPTR (F566H,2) - endereço de destino na Main-RAM;
DUMMY (F568H,2) - dummy (não requer dados);
NX (F56AH,2) - número de pontos na direção X;

NY (F56CH,2) - número de pontos na direção Y;
CDUMMY (F56EH,2) - dummy (não requer dados);
ARGT (F56FH,1) - seleciona a expansão e a direção da RAM (igual a R#45 do VDP.

Saída: A flag CY é resetada.
Registradores: Todos.

As rotinas seguintes transferem dados entre Main-RAM, VRAM e o disco. Para isso, deve-se especificar o nome do arquivo no disco como no exemplo abaixo.

```
        LD   HL,FNAME    ;pega o end. do nome do arq.
        LD   (FNPTR),HL  ;seta o end. na variável de
        |                ;sistema
        |
FNAME: DEFB 22H,'A:TESTE.PIC',22H,00H ;nome do arqui-
                                      ;vo + marca fim
```

Como essas rotinas também são usadas pelo interpretador BASIC, se ocorrer algum erro durante a transferência, o controle é passado automaticamente ao manipulador de erro que depois devolve o controle ao interpretador.

Para evitar que isso ocorra, basta usar o hook HERRO (FEFDH) para interceptar o erro antes que este seja transferido ao interpretador. O código de erro fica no registrador E, podendo ser usado pelo programa Assembly.

BLTVD (019DH)
Função: Transfere dados do disco para a VRAM.
Entrada: HL Deve conter o valor F562H.
FNPTR (F562H,2) - endereço do nome do arquivo;
DUMMY (F564H,2) - dummy (não requer dados);
DX (F566H,2) - coordenada X (horizontal) do destino;
DY (F568H,2) - coordenada Y (vertical) do destino);
NX (F56AH,2) - número de pontos na direção X (não requer dados; já está setada);
NY (F56CH,2) - número de pontos na direção Y (não requer dados; já está setada);
CDUMMY (F56EH,2) - dummy (não requer dados);
ARGT (F56FH,1) - seleciona e expansão e a direção da VRAM (igual a R#45 do VDP);
LOGOP (F570H,1) - código de operação lógica (igual aos códigos do VDP).
Saída: A flag CY é setada se houver algum erro nos parâmetros.
Registradores: Todos.

BLTDV (01A1H)
Função: Transfere dados da VRAM para o disco.
Entrada: HL Deve conter o valor F562H.
SX (F562H,2) - coordenada X (horizontal) da fonte;
SY (F564H,2) - coordenada Y (vertical) da fonte;
FNPTR (F566H,2) - endereço do nome do arquivo;
DUMMY (F568H,2) - dummy (não requer dados);
NX (F56AH,2) - número de pontos na direção X;
NY (F56CH,2) - número de pontos na direção Y;
CDUMMY (F56EH,2) - dummy (não requer dados).

Saída: A flag CY é resetada.
Registradores: Todos.

BLTMD (01A5H)
Função: Carrega dados do disco para a Main-RAM.
Entrada: HL Deve conter o valor F562H.
FNPTR (F562H,2) - endereço do nome do arquivo;
SY (F564H,2) - dummy (não requer dados);
SPTR (F566H,2) - endereço inicial dos dados a serem carregados;
EPTR (F568H,2) - endereço final dos dados a serem carregados.
Saída: A flag CY é resetada.
Registradores: Todos.

BLTDM (01A9H)
Função: Grava dados da Main-RAM no disco.
Entrada: HL Deve conter o valor F562H.
SPTR (F562H,2) - endereço inicial dos dados a serem gravados;
EPTR (F564H,2) - endereço final dos dados a serem gravados;
FNPTR (F566H,2) - endereço do nome do arquivo.
Saída: A flag CY é resetada.
Registradores: Todos.

2 - O MATH-PACK (PACOTE MATEMÁTICO)

O Math-Pack (Pacote Matemático) é um conjunto de rotinas matemáticas que não pertencem ao 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 (-32768 a + 32767), 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.

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
simples precisão	1º dígito				2º dígito				1
	3º dígito				4º dígito				2
	5º dígito				6º dígito				3
dupla precisão	7º dígito				8º dígito				4
	9º dígito				10º dígito				5
	11º dígito				12º dígito				6
	13º dígito				14º dígito				7

mantissa

Formato BCD para expressar números reais

+/-	expoente							
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	indefinido (-0?)
x	0	0	0	0	0	0	0	-63ª potência de 10
x	1	0	0	0	0	0	0	0ª potência de 10
x	1	1	1	1	1	1	1	+63ª potência de 10

Expressão binária formadora do expoente e do sinal da mantissa

Exemplo de número de precisão simples:
123.456 → 0,123456E+6

	0	1	2	3
DAC →	46	12	34	56

Exemplo de número de precisão dupla
123.456,78901234 → 0,12345678901234E+6

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

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

2.1 - ÁREA DE TRABALHO

Para realizar operações com o Math-Pack, 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 dupla precisão, simples precisão 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 distingüir 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 do 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 mostradas as localizações das variáveis de sistema para o 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.

2.2 FUNÇÕES MATEMÁTICAS EM PONTO FLUTUANTE

DECSUB	(268CH)	DAC ← DAC - ARG
DECADD	(269AH)	DAC ← DAC + ARG
DECMUL	(27E6H)	DAC ← DAC * ARG
DECDIV	(289FH)	DAC ← DAC / ARG
SGNEXP	(37C8H)	DAC ← DAC ^ ARG - Simples precisão
DBLEXP	(37D7H)	DAC ← DAC ^ ARG - Dupla precisão
COS	(2993H)	DAC ← COS (DAC)
SIN	(29ACH)	DAC ← SIN (DAC)
TAN	(29FBH)	DAC ← TAN (DAC)
ATN	(2A14H)	DAC ← ATN (DAC)
SQR	(2AFFH)	DAC ← SQR (DAC)
LOG	(2A72H)	DAC ← LOG (DAC)
EXP	(2B4AH)	DAC ← EXP (DAC)

2.3 - OPERAÇÕES COM NÚMEROS INTEIROS

UMULT	(314AH)	DE ← DE * BC
ISUB	(3167H)	HL ← DE - HL
IADD	(3172H)	HL ← DE + HL
IMULT	(3193H)	HL ← DE * HL
IDIV	(31E6H)	HL ← DE / HL
INTEXP	(383FH)	DAC ← DE ^ HL
IMOD	(323AH)	HL ← DE mod HL
		DE ← DE / HL

2.4 - OUTRAS FUNÇÕES

DECNRM	(26FAH)	Normaliza DAC[5]
RND	(2BDFH)	DAC ← RND (DAC)
SIGN	(2E71H)	A ← Sinal da mantissa em DAC
ABSFN	(2E82H)	DAC ← ABS (DAC)
NEG	(2E8DH)	DAC ← NEG (DAC)
SGN	(2E97H)	DAC ← SGN (DAC)[6]

Nota 5: Zeros excessivos na mantissa são removidos. Por exemplo, 0,00123 → 0,123E-2.
Nota 6: Na função SGN, o resultado é representado por un número inteiro de 2 bytes.

2.5 - CONVERSÃO DE TIPO

FRCINT (2F8AH) Converte DAC para número inteiro de 2 bytes (DAC+2, +3).
FRCSNG (2FB2H) Converte DAC para número de precisão simples.
FRCDBL (303AH) Converte DAC para número de precisão dupla.
FIXER (30BEH) DAC ← SGN (DAC) * INT (ABS (DAC))

Obs.: Depois da conversão, VALTYP (F663H) conterá o valor que representa o tipo de número convertido armazenado em DAC (2, 4 ou 8).

2.6 - MOVIMENTO

MAF	(2C4DH)	ARG	←	DAC	Dupla precisão
MAM	(2C50H)	ARG	←	(HL)	Dupla precisão
MOV8DH	(2C53H)	(DE)	←	(HL)	Dupla precisão
MFA	(2C59H)	DAC	←	ARG	Dupla precisão
MFM	(2C5CH)	DAC	←	(HL)	Dupla precisão
MMF	(2C67H)	(HL)	←	DAC	Dupla precisão
MOV8HD	(2C6AH)	(HL)	←	(DE)	Dupla precisão
XTF	(2C6FH)	(SP)	↔	DAC	Dupla precisão
PHA	(2CC7H)	ARG	←	(SP)	Dupla precisão
PHF	(2CCCH)	DAC	←	(SP)	Dupla precisão
PPA	(2CDCH)	(SP)	←	ARG	Dupla precisão
PPF	(2CE1H)	(SP)	←	DAC	Dupla precisão
PUSHF	(2EB1H)	DAC	←	(SP)	Simples precisão
MOVFM	(2EBEH)	DAC	←	(HL)	Simples precisão
MOVFR	(2EC1H)	DAC	←	CBED	Simples precisão
MOVRF	(2ECCH)	CBED	←	DAC	Simples precisão
MOVRMI	(2ED6H)	CBED	←	(HL)	Simples precisão
MOVRM	(2EDFH)	BCDE	←	(HL)	Simples precisão
MOVMF	(2EE8H)	(HL)	←	DAC	Simples precisão
MOVE	(2EEBH)	(HL)	←	(DE)	Simples precisão
VMOVAM	(2EEFH)	ARG	←	(HL)	VALTYP
MOVVFM	(2EF2H)	(DE)	←	(HL)	VALTYP
VMOVE	(2EF3H)	(HL)	←	(DE)	VALTYP
VMOVFA	(2F05H)	DAC	←	ARG	VALTYP
VMOVFM	(2F08H)	DAC	←	(HL)	VALTYP
VMOVAF	(2F0DH)	ARG	←	DAC	VALTYP
VMOVMF	(2F10H)	(HL)	←	DAC	VALTYP

Obs.: (HL) e (DE) significam os endereços de memória apontados por HL e DE. Quatro nomes de registradores juntos 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 como o tipo indicado por VALTYP (F663H), ou seja, 2, 4 ou 8 bytes.

2.7 - COMPARAÇÕES

			Esquerdo	Direito
ICOMP	(2F4DH)	Inteiro de 2 bytes	DE	HL
DCOMP	(2F21H)	Precisão simples	CBED	DAC
XDCOMP	(2F5CH)	Precisão dupla	ARG	DAC

O resultado da comparação será colocado no registrador A, conforme mostrado abaixo:

A=01H → esquerdo < direito
A=00H → esquerdo = direito
A=FFH → esquerdo > direito

2.8 - OUTRAS OPERAÇÕES DE 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;
- 00H - 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 uma 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 caracxtere 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: com vírgulas cada 3 dígitos
bit 5 - 0: sem significado 1: preenche espaços com "*"
bit 4 - 0: sem significado 1: adiciona "$" antes do número
bit 3 - 0: sem significado 1: coloca "+" para números positivos
bit 2 - 0: sem significado 1: coloca sinal depois do número
bit 1 - 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 - 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.

3.1 - 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 bem mais compacta. A finalidade dos tokens não é apenas tornar o texto BASIC mais compacto, mas também mais rápido, visto que, durante o processamento, ao invés de decodificar toda a seqüência 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:

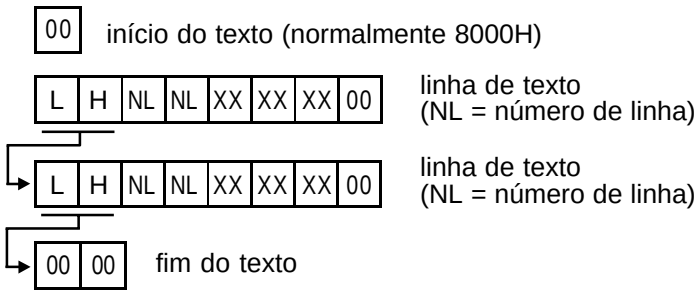
byte 58H - código ASCII da variável ‘X’
byte EFH - token do sinal ‘=’
byte FFH - indentificador de função
byte 89H - token setado da função SIN
byte 28H - código ASCII de ‘(‘
byte 41H - código ASCII da variável ‘A’
byte 29H - código ASCII de ‘)’

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

3.2 - 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 65529) 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 - 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. Por isso, são divididos em três grupos: 0 a 9, 10 a 255 e 256 a 32767. Para os números inteiros de 0 a 9, há uma espécie de token que o identifica como tal, conforme a tabela abaixo:

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

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 32767, também existe um byte de identidicação (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).

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.

Já os números que referem linhas de programas (nas instruções GOTO e GOSUB, por exemplo) também 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 da página seguinte.

Número Octal	0B XX XX
Número Hexadecimal	0C XX XX
Número Binário	26 42 Número na forma ASCII (0~1)
Inteiro de 0 a 9	XX XX pode valer de 11 a 1A
Inteiro de 10 a 255	0F XX
Inteiro de 255 a 32767	1C XX XX
Precisão simples	1D XX XX XX
Precisão dupla	1F XX XX XX XX XX XX XX
Linha (antes de RUN)	0E XX XX
Linha (depois de RUN)	0D XX XX

3.4 - A 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)[7]. 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.

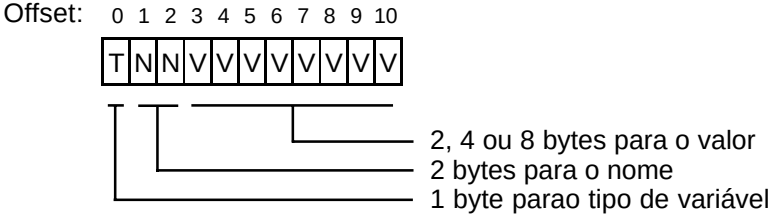
Sempre que uma nova linha BASIC é introduzida, deletada ou o comando CLEAR é executado, o valor de STREND é igualado ao valor de VARTAB e conseqüentemente todas as variáveis do programa são limpas e ficam nulas.

Existem 4 tipos de variáveis do BASIC:

numéricas inteiras:	ocupam 2 bytes
numéricas de precisão simples:	ocupam 4 bytes
numéricas de precisão supla:	ocupam 8 bytes
alfanuméricas (strings):	ocupam 3 bytes

Nota 7: as variáveis string, bem como as matrizes, são tratadas diferentemente. O tratamento de cada uma está descrito no final do item 3.4.

As variáveis numéricas possuem a sintaxe de armazenamento ilustrada abaixo:



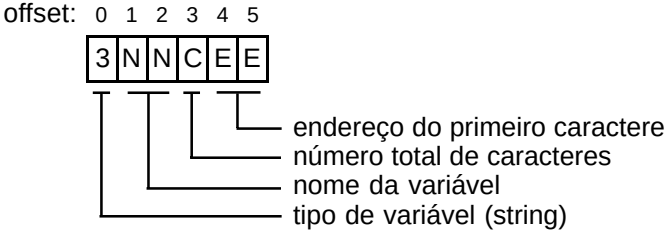
O primeiro byte indica o tipo de variável numérica que está armazenada: 2, 4 ou 8. Esse valor também já indica o número de bytes ocupados pela variável.

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 - simplea precisão
03 - string 08 - dupla precisão

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

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

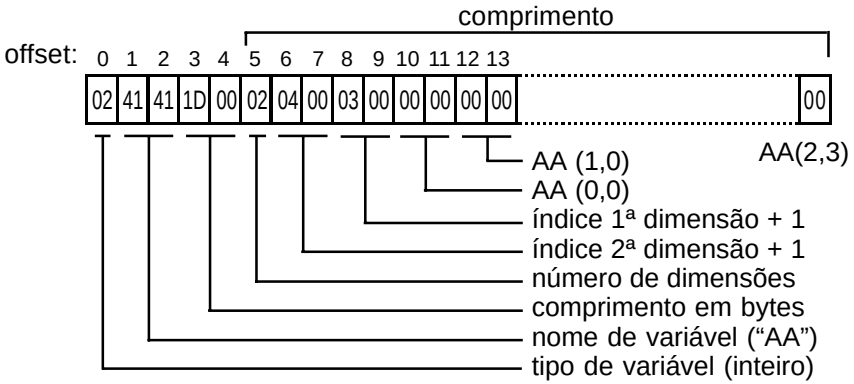


A variável de sistema FRETOP (F69BH) armazena o endereço que receberá o último caractere da string que está sendo armazenada.

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. Entretando, 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 um pouco 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 seqüências 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).



3.5 - CHAMANDO PROGRAMAS ASSEMBLY NO BASIC

Para usar programas Assembly juntamente com o BASIC, existem 3 comandos resevados 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;

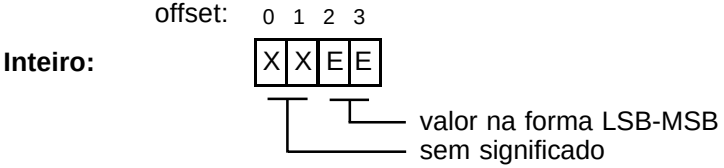
3 - Uma instrução RET na rotina Assembly retorna o controle ao interpretador.

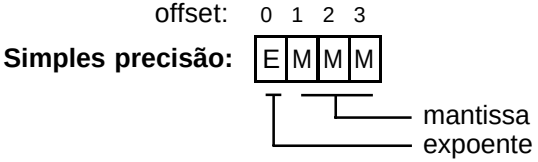
Um argumento qualquer pode ser passado ao programa Assembly pela função USR. Nesse caso, o registrador A conterá o tipo de variável passada e o registrador DE conterá o endereço de um apontador no caso de variáveis string ou HL conterá o endereço da própria variável, caso seja numérica, conforme a ilustração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

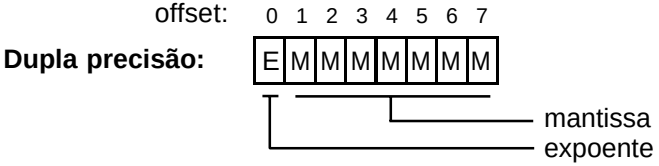
Valores de A:

2 - numérico inteiro
3 - alfanumérica
4 - simples precisão
8 - dupla precisão

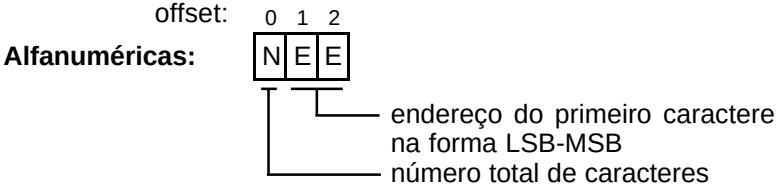
Endereços apontados por HL:







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.

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

Se executado um 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 esses comandos, o par HL apontará o primeiro caractere após o comando, conforme ilustrado abaixo:

```
CMD “ARGUMENTO”
     ↑
   (HL)
```

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).

3.6 - 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, é necessário usar o hook HERRO (FFB1H) para interceptar o erro. O código de erro fica no registrador E, podendo ser usado pela rotina assembly sem problemas. 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 do BIOS quando estas puderem realizar o mesmo trabalho, pois são muito mais rápidas 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 um algoritmo 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 do BIOS para executar o comando. Também podem ser usadas as rotinas CALSLT (001CH) ou CALLF (0030H), setando em IY o slot da Main-ROM.

O passo seguinte é verificar em qual endereço está a rotina que executa o comando desejado. Para isso existe uma tabela de endereços que inicia em 392EH e 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, endereço na tabela e pontos de entrada.

Comando	Token	Tabela	Entrada	Comando	Token	Tabela	Entrada
ABS	06	39E8	2E82	CLS	9F	396A	79A9
AND	F6	Afat		CMD	D7	39DA	7C34
ASC	15	3A06	680B	COLOR	BD	39A6	7980
ATN	0E	39F8	2A14	COS	0C	39F4	2993
ATTR$	E9	Afat	7C43	CONT	99	395E	6424
AUTO	A9	397E	49B5	COPY	D6	39D8	7C2F
BASE	C9	39BE	7B5A	CSAVE	9A	3960	6FB7
BEEP	C0	39AC	00C0	CSNG	1F	3A1A	2FB2
BIN$	1D	3A16	65FF	CSRLIN	E8	Afat	790A
BLOAD	CF	39CA	6EC6	CVD	2A	3A30	7C70
BSAVE	D0	39CC	6E92	CVI	28	3A2C	7C66
CALL	CA	39C0	55A8	CVS	29	3A2E	7C6B
CDBL	20	3A1C	303A	DATA	84	3934	485B
CHR$	16	3A08	681B	DEF	97	395A	501D
CINT	1E	3A18	2F8A	DEFDBL	AE	3988	4721
CIRCLE	BC	39A4	5B11	DEFINT	AC	3984	471B
CLEAR	92	3950	64AF	DEFSNG	AD	3986	471E
CLOAD	9B	3962	703F	DEFSTR	AB	3082	4718
CLOSE	B4	3994	6C14	DELETE	A8	397C	53E2

Comando	Token	Tabela	Entrada
DIM	86	3938	5E9F
DRAW	BE	39A8	5D6E
DSKF	26	3A28	7C39
DSKI$	EA	Afat	7C3E
DSKO$	D1	39CE	7C16
ELSE	A1	396E	485D
END	81	392E	63EA
EOF	2B	3A32	6D25
EQV	F9	Afat	
ERASE	A5	3976	6477
ERL	E1	Afat	4E0B
ERR	E2	Afat	4DFD
ERROR	A6	3978	49AA
EXP	0B	39F2	2B4A
FIELD	B1	398E	7C52
FILES	B7	399A	6C2F
FIX	21	3A1E	30BE
FN	DE	Afat	5040
FOR	82	3930	4524
FPOS	27	3A2A	6D39
FRE	0F	39FA	69F2
GET	B2	3990	775B
GOSUB	8D	3946	47B2
GOTO	89	393E	47E8
GO TO	89	393E	47E8
HEX$	1B	3A12	65FA
IF	8B	3942	49E5
IMP	FA	3A20	7940
INKEY$	EC	Afat	7347
INP	10	39FC	4001
INPUT	85	3936	4B6C
INSTR	E5	39F6	29FB
INT	05	39E6	30CF
IPL	D5	39D6	7C2A
KEY	CC	39C4	786C
KILL	D4	39D4	7C25
LEFT$	01	39DE	6861
LEN	12	3A00	67FF
LET	88	393C	4880
LFILES	BB	39A2	6C2A
LINE	AF	398A	4B0E
LIST	93	3952	522E
LLIST	9E	3968	5229
LOAD	B5	3996	6B5D
LOC	2C	3A34	6D03
LOCATE	D8	39DC	7766
LOF	2D	3A36	6D14
LOG	0A	39F0	2A72
LPOS	1C	3A14	4FC7
LPRINT	9D	3966	4A1D
LSET	B8	399C	7C48
MAX	CD	39C6	7E4B
MERGE	B6	3998	6B5E
MID$	03	39E2	689A
MKD$	30	3A3C	7C61
MKI$	2E	3A38	7C57
MKS$	2F	3A3A	7C5C
MOD	FB	3A22	794C
MOTOR	CE	39C8	73B7
NAME	D3	39D2	7C20
NEW	94	3954	6286
NEXT	83	3932	6527
NOT	E0	Afat	
OCT$	1A	3A10	65F6
OFF	EB	3A02	3A02
ON	95	3956	48E4
OPEN	B0	398C	6AB7
OR	F7	3A1A	2FB2
OUT	9C	3964	4016
PAD	25	3A26	7969
PAINT	BF	39AA	59C5
PDL	24	3A24	795A
PEEK	17	3A0A	541C
PLAY	C1	39AE	73E5
POINT	ED	Afat	5803
POKE	98	395C	5423
POS	11	39FE	4FCC
PRESET	C3	39B2	57E5
PRINT	91	394E	4A24
PSET	C2	39B0	57EA
PUT	B3	3992	7758
READ	87	393A	4B9F
REM	8F	394A	485D
RENUM	AA	3980	5468
RESTORE	8C	3944	63C9
RESUME	A7	397A	495D
RETURN	8E	3948	4821
RIGHT$	02	39E0	6891

Comando	Token	Tabela	Entrada	Comando	Token	Tabela	Entrada
RND	08	39EC	2BDF	SWAP	A4	3974	643E
RSET	B9	399E	7C4D	TAB(	DB	Afat	
RUN	8A	3940	479E	TAN	0D	39F6	29FB
SAVE	BA	39A0	6BA3	THEN	DA	Afat	
SCREEN	C5	39B6	79CC	TIME	CB	39C2	7911
SET	D2	39D0	7C1B	TO	D9	Afat	
SGN	04	39E4	2E97	TROFF	A3	3972	6439
SIN	09	39EE	29AC	TRON	A2	3970	6438
SOUND	C4	39B4	73CA	USING	E4	Afat	
SPACE$	19	3A0E	6848	USR	DD	Afat	4FD5
SPC(	DF	Afat		VAL	14	3A04	68BB
SPRITE	C7	39BA	7A48	VARPTR	E7	Afat	4E41
SQR	07	39EA	2AFF	VDP	C8	39BC	7B37
STEP	DC	Afat		VPEEK	18	3A0C	7BF5
STICK	22	3A20	7940	VPOKE	C6	39B8	7BE2
STOP	90	394C	63E3	WAIT	96	3958	401C
STR$	13	3A02	6604	WIDTH	A0	396C	51C9
STRIG	23	3A22	794C	XOR	F8	Afat	
STRING$	E3	Afat	6829				

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.

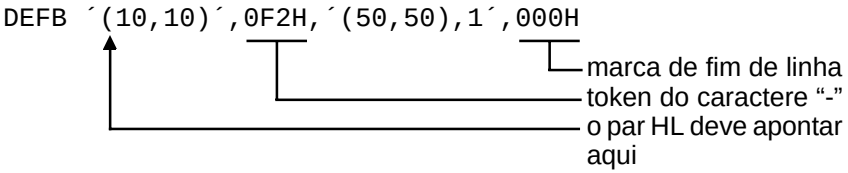
No início dessa seção foi dito que a linha BASIC deveria estar preferencialmente na forma tokenizada. Entretanto, é possível usá-la na forma ASCII. O único cuidado nesse caso é substituir dez caracteres-chave pelos tokens respectivos, podendo o restante do texto estar na forma ASCII. Esses caracteres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

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

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

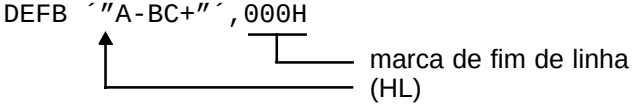
```
LINE (10,10)-(50,50),1
```

deve ser colocada na linha em código de máquina da forma mostrada na página seguinte.



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:



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 BASIC CIRCLE. Essa rotina funciona tanto sob o DOS quando sob o BASIC, em qualquer endereço.

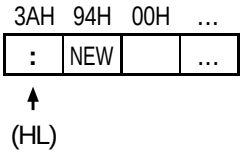
```
CIRCLE: EQU  05B11H
INIGRP: EQU  00072H
CHGET:  EQU  0009FH
CALSLT: EQU  0001CH
SLTROM: EQU  0FCC1H
KBUF:   EQU  0F41FH
        LD   HL,LINBAS
        LD   DE,KBUF
        LD   BC,12
        LDIR
        LD   IX,INIGRP
        LD   IY,(SLTROM+1)
        CALL CALSLT
        LD   HL,KBUF
        LD   IX,CIRCLE
        LD   IY,(SLTROM+1)
        CALL CALSLT
        LD   IX,CHGET
        LD   IY,(SLTROM+1)
        CALL CALSLT
        RET
LINBAS: DEFB ´(128,96),70´,000H
```

3.7 - 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 do BIOS. Como são rotinas do interpretador, caso ocorra algum erro, o controle será transferido ao manipulador de erro e devolvido ao interpretador. Para evitar que isso ocorra, deve 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.

NEWSTT (4601H/Main)
Função: Executar um texto BASIC. O texto deverá estar na forma tokenizada.
Entrada: HL - início do texto a ser executado, como ilustrado abaixo:



Saída: Nenhuma.

CHRGTR (4666H/Main)
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 - caractere extraído
Flag Z - ligada se for fim de linha (00H ou 3AH ":")
Flag CY - ligada de for um caractere de 0 a 9

FRMEVL (4C64H/Main)
Função: Avaliar uma expressão e devolver o resultado.
Entrada: HL - endereço inicial da expressão no texto
Saída: HL - endereço após a expressão
VALTYP (F663H) - 2, 3, 4 ou 8, de acordo com a expressão[8].
DAC (F7F6H) - resultado da expressão avaliada.

Nota 8: O significado dos valores está descrito na seção "ÁREA DE VARIÁVEIS DO INTERPRETADOR"

GETBYT (521C/Main)

Função: Avaliar 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 após a expressão.
A, E - resultado da avaliação (A e E contêm o mesmo valor)

FRMQNT (542FH/Main)

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

Entrada: HL - endereço inicial da expressão a ser avaliada.

Saída: HL - endereço após a expressão.
DE - resultado da avaliação

PTRGET (5EA4/Main)

Função: Obter 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 (F6A5H) for diferente de 0, o endereço inicial da matriz será obtido; caso contrário, será obtido o endereço do elemento da matriz.

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

Saída: HL - endereço após o nome da variável
DE - endereço onde o conteúdo da variável está armazenado

FRESTR (67D0H/Main)

Função: Registrar o resultado de uma string obtida por FRMEVL (4C64H) e obter 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 (deve ser 3)
DAC (F7F6H) - apontador para o descritor da string

Saída: HL - apontador para o descritor da string

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 diretamente.

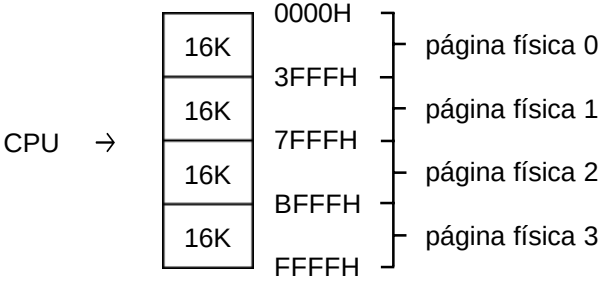
1 - EXPANSÕES DE MEMÓRIA

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, tão ruim que não chegou a ser usado comercialmente.

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.

1.1 - MEMÓRIA MAPEADA

A Memória Mapeada 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:



Assim, para cada página física, há uma porta de I/O correspondente, como ilustrado na página seguinte.

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; assim pode-se ter até 256 *Páginas Lógicas*. Como cada página tem 16 Kbytes, fazemos 16 Kbytes vezes 256, o que dá 4 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 inicial de páginas é a seguinte:

Página física 0 = página lógica 3 (porta FCH = 3)
Página física 1 = página lógica 2 (porta FDH = 2)
Página física 2 = página lógica 1 (porta FEH = 1)
Página físcia 3 = página lógica 0 (porta FFH = 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 seqüência de instruções é usada:

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

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 tem precedência sobre a seleção de páginas lógicas. Por isso, ao selecionar uma página lógica, é necessário 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 o 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.

1.2 - 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 megarom 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 seu 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)”. Na página seguinte está listada a seqüência 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
XOR  A          ;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.

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 do BIOS ou qualquer software residente na ROM para manipulação dessas expansões, à exceção do BDOS do MSXDOS2, que manipula precariamente a Memória Mapeada.

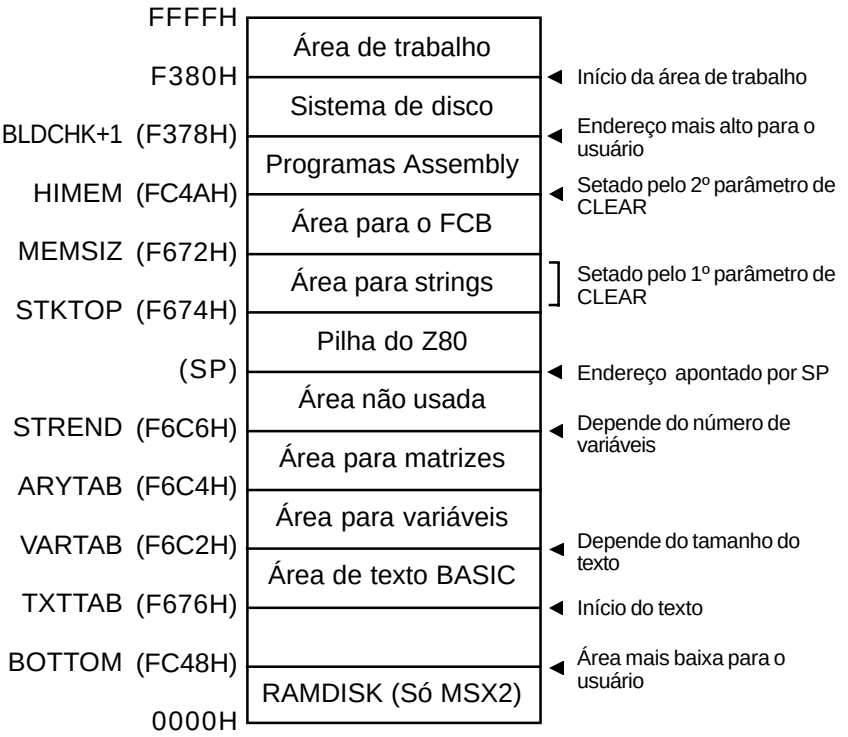
Uma dúvida que pode surgir aos programadores é sobre qual expansão de memória usar: Megaram ou Memória Mapeada. Como já descrito, a Memória Mapeada é a expansão padrão do MSX; entretanto a Megaram é muito popular no Brasil.

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.

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.



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).

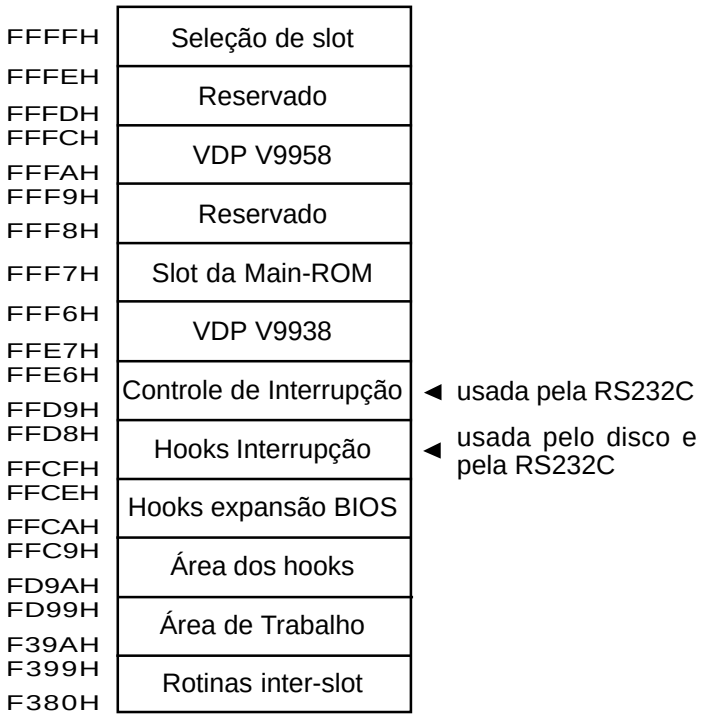
2.1 - O FCB (FILE CONTROL BLOCK)

O FCB (File Control Block, ou Bloco de Controle de Arquivos) é um buffer de 267 bytes usado para comunicação com periféricos. Esse FCB não tem nada a ver com o FCB do MSXDOS. Podem ser abertos até 15 FCB´s simultaneamente, especificados pelo comando MAXFILES do BASIC, mas no reset a área é alocada para apenas um. Eles são atribuídos a partir do FCB de endereço mais baixo. O formato do FCB está ilustrado na página seguinte.

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	Localização da cabeça virtual
+9...	FL.BUF	Início do buffer de 256 bytes

3 - 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.



A seguir, estão listadas todas as variáveis de sistema da área de trabalho. A notação é a seguinte:

LABEL (endereço,comprimento)
Valor inicial
Conteúdo

Onde “LABEL” é o nome da variável de sistema, “endereço” é o endereço inicial da variável, “comprimento” é o tamanho da variável em bytes, valor inicial é o valor atribuído à variável no reset e conteúdo é o que a variável armazena. As três primeiras são rotinas para chamada inter-slot, e não variáveis de sistema.

3.1 - SUBROTINAS INTER-SLOT[9]

RDPRIM[9] (F380H,5)
Função: lê um byte de um slot qualquer.

WRPRIM[9] (F385H,5)
Função: escreve um byte em um slot qualquer

CLPRIM[9] (F38CH,14)
Função: chama um endereço em um slot qualquer

3.2 - FUNÇÃO USR E MODOS TEXTO

USRTAB (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.

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.

Nota 9: essas subrotinas são usadas pelas rotinas inter-slot do BIOS. É desaconselhado o uso das mesmas pelo programador.

LINLEN (F3B0H,1)
Valor inicial: 39
Conteúdo: Largura da tela de texto atual.

CRTCNT (F3B1H,1)
Valor inicial: 24
Conteúdo: Número de linhas dos modos de texto.

CLMSLT (F3B2H,1)
Valor inicial: 14
Conteúdo: Tabulação horizontal em itens divididos por vírgula no comando PRINT.

3.3 - VALORES DOS MODOS DE TELA (SCREENS 0 A 3)

SCREEN 0:

TXTNAM (F3B3H,2)
Valor inicial: 0000H
Conteúdo: Endereço na VRAM da tabela de nomes dos padrões.

TXTCOL (F3B5H,2) - Sem significado.

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.

TXTATR (F3B9H,2) - Sem significado.

TXTPAT (F3BBH,2) - Sem significado.

SCREEN 1:

T32NAM (F3BDH,2)
Valor inicial: 1800H
Conteúdo: Endereço na VRAM 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.

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.

SCREEN 3:

MLTNAM (F3D1H,2)
Valor inicial: 0800H
Conteúdo: Endereço da tabela de nomes dos padrões.

MLTCOL (F3D3H,2) - Sem significado.

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.

3.4 - 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 pelos comando KEY ON/OFF.

3.5 - ÁREA DOS REGISTRADORES DO VDP

RG0SAV (F3DFH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#0 do VDP.

RG1SAV (F3E0H,1)
Valor inicial: E0H
Conteúdo: Cópia do registrador R#1 do VDP.

RG2SAV (F3E1H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#2 do VDP.

RG3SAV (F3E2H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#3 do VDP.

RG4SAV (F3E3H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#4 do VDP.

RG5SAV (F3E4H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#5 do VDP.

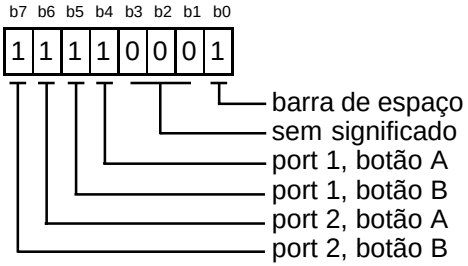
RG6SAV (F3E5H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#6 do VDP.

RG7SAV F3E6H,1)
Valor inicial: 00H
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.

3.6 - MISCELÂNEA

TRGFLG (F3E8H,1)
Valor inicial: 11110001B
Conteúdo: Estado dos botões do joystick. (0=pressionado, 1=não pressionado). Essa variável é 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.

3.7 - 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.

3.8 - 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 autorepetição 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.

3.9 - ÁREA USADA PELO CASSETE

CS1200 (F3FCH,5)
CS2400 (F401H,5)
LOW (F406H,2)
HIGH (F408H,2)
HEADER (F40AH,1)

Essas cinco variáveis eram usadas para o cassete, mas foram eliminadas nos modelos MSX turbo R, por terem se tornado totalmente obsoletas.

3.10 - ÁREA USADA PELO COMANDO CIRCLE

ASPCT1 (F40BH,2)
Valor inicial: 0000H
Conteúdo: 256 / relação de aspecto. Pode ser alterada pelo comando SCREEN para uso do comando CIRCLE.

ASPCT2 (F40DH,2)
Valor inicial: 0
Conteúdo: 256 * relação de aspecto. Pode ser alterada pelo comando SCREEN para uso do comando CIRCLE.

3.11 - ÁREA USADA PELO INTERPRETADOR

ENDPRG (F40FH,5)
Valor inicial: "**:**", 00H, 00H, 00H, 00H
Conteúdo: Falso fim de linha para os comandos RESUME e NEXT.

ERRFLG (F414H,1)
Valor inicial: 0
Conteúdo: Área para salvar o número de erro.

LPTPOS (F415H,1)
Valor inicial: 0
Conteúdo: Posição horizontal atual da cabeça da impressora.

PRTFLG (F416H,1)
Valor inicial: 0
Conteúdo: Flag para selecionar saída para tela ou impressora (0=tela; outro valor, impressora).

NTMSXP (F417H,1)
Valor inicial: 0
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: 0
Conteúdo: Flag para determinar se os caracteres gráficos e códigos 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: 0
Conteúdo: Caractere que deve ser substituído por 0 pela função VAL.

CURLIN (F41CH,2)
Valor inicial: 0000H
Conteúdo: Numero da linha BASIC atualmente sendo processada. O valor FFFFH indica modo direto.

KBFMIN (F41EH,1)
Valor inicial: "**:**"
Conteúdo: Prefixo fictício para o texto tokenizado contido em BUF.

KBUF (F41FH,318)
Valor inicial: DEFS 318 (00H ..... 00H)
Conteúdo: Buffer que guarda a linha BASIC tokenizada coletada pelo interpretador. Quando uma instrução direta é executada, esse buffer guarda os respectivos comandos.

BUFMIN (F55DH,1)
Valor inicial: "**,**"
Conteúdo: Prefixo fictício para o texto contido em KBUF.

BUF (F55EH,258)
Valor inicial: DEFS 258 (00H ..... 00H)
Conteúdo: Buffer que guarda, no formato ASCII, os caracteres coletados diretamente pelo teclado.

ENDBUF (F660H,1)
Valor inicial: 00H
Conteúdo: Byte para prevenir overflow em BUF.

TTYPOS (F661H,1)
Valor inicial: 0
Conteúdo: Usada pelo comando PRINT para guardar a posição virtual do cursor.

DIMFLG (F662H,1)
Valor inicial: 0
Conteúdo: Usada internamente pelo comando DIM.

VALTYP (F663H,1)
Valor inicial: 2
Conteúdo: Tipo de variável contida em DAC (F3F6H). 2=inteira; 3=string; 4=precisão simples; 8=precisão dupla.

DORES (F664H,1)
Valor inicial: 0
Conteúdo: Usada pelo comando DATA para manter o texto no formato ASCII.

DONUM (F665H,1)
Valor inicial: 0
Conteúdo: Flag usada internamente pelo interpretador para atomizar uma constante numérica em número de linha.

CONTXT (F666H,2)
Valor inicial: 0000H
Conteúdo: Endereço do texto usado pela rotina CHRGTR (0010H).

CONSAV (F668H,1)
Valor inicial: 0
Conteúdo: Token de uma constante numérica usada pela rotina CHRGTR (0010H).

CONTYP (F669H,1)
Valor inicial: 0
Conteúdo: Tipo de constante numérica do texto BASIC. Usada pela rotina CHRGTR (0010H).

CONLO (F66AH,8)
Valor inicial: DEFS 8 (00H ..... 00H)
Conteúdo: Valor de uma constante numérica do texto BASIC. Usada pela rotina CHRGTR.

MEMSIZ (F672H,2)
Valor inicial: Variável.
Conteúdo: Endereço mais alto de memória disponível para o BASIC.

STKTOP (F674H,2)
Valor inicial: Variável.
Conteúdo: Endereço do topo da pilha do Z80. Pode ser alterada exclusivamente pelas instruções CLEAR ou MAXFILES. Usada internamente pelo BASIC.

TXTTAB (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.

TEMP8 (F69FH,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: 0
Conteúdo: Número de linha do comando DATA para o comando READ.

SUBFLG (F6A5H,1)
Valor inicial: 0
Conteúdo: Flag usada para controlar o processamento de índices na busca de variáveis tipo matriz.

FLGINP (F6A6H,1)
Valor inicial: 0
Conteúdo: Flag usada pelos comandos INPUT e READ (0=INPUT; outro valor, READ).

TEMP (F6A7H,2)
Valor inicial: 0000H
Conteúdo: Usada internamente pelo interpretador para armazenamento temporário de várias rotinas.

PTRFLG (F6A9H,1)
Valor inicial: 0
Conteúdo: Usada internamente pelo interpretador para conversão de número de linha em apontadores (0=operando não convertido; outro valor, operando convertido).

AUTFLG (F6AAH,1)
Valor inicial: 0
Conteúdo: Flag usada pelo comando AUTO (0=comando AUTO inativo; outro valor, comando AUTO ativo).

AUTLIN (F6ABH,2)
Valor inicial: 0
Conteúdo: Número da última linha BASIC entrada.

AUTINC (F6ADH,2)
Valor inicial: 10
Conteúdo: Valor de incremento para o comando AUTO.

SAVTXT (F6AFH,2)
Valor inicial: KBFMIN (F41EH)
Conteúdo: Valor atual de execução do texto BASIC.

SAVSTK (F6B1H,2)
Valor inicial: F09EH
Conteúdo: Endereço atual da pilha do Z80. Usada pelo manipulador de erro e pela instrução RESUME.

ERRLIN (F6B3H,2)
Valor inicial: 0000H
Conteúdo: Número de linha BASIC onde ocorreu algum erro. O valor FFFFH indica modo direto.

DOT (F6B5H,2)
Valor inicial: 0
Conteúdo: Último número de linha BASIC que foi listado ou entrado para uso com o parâmetro "**.**".

ERRTXT (F6B7H,2)
Valor inicial: KBFMIN (F41EH)
Conteúdo: Endereço do texto BASIC onde ocorreu algum erro. Usada pelo comando RESUME.

ONELIN (F6B9H,2)
Valor inicial: 0000H
Conteúdo: Endereço da linha BASIC que deve ser executada ao ocorrer algum erro. Setada pelo comando ON ERROR GOTO.

ONEFLG (F6BBH,2)
Valor inicial: 0
Conteúdo: Flag para indicar a execução de rotina de erro BASIC (0= não executando; outro valor, rotina em execução).

TEMP2 (F6BCH,2)
Valor inicial: 0
Conteúdo: Usada internamente pelo interpretador para armazenamento temporário de várias rotinas.

OLDLIN (F6BEH,2)
Valor inicial: 0
Conteúdo: Última linha BASIC executada pelo interpretador. É atualizada pelos comandos END e STOP para ser usada pelo comando CONT.

OLDTXT (F6C0H,2)
Valor inicial: 0
Conteúdo: Endereço da última instrução do texto BASIC.

VARTAB (F6C2H,2)
Valor inicial: 8003H
Conteúdo: Endereço inicial da área de armazenamento das variáveis do BASIC.

ARYTAB (F6C4H,2)
Valor inicial: 8003H
Conteúdo: Endereço inicial da área de armazenamento das matrizes do BASIC.

STREND (F6C6H,2)
Valor inicial: 8003H
Conteúdo: Primeiro endereço após a área de armazenamento das matrizes, variáveis ou texto BASIC.

DATPTR (F6C8H,2)
Valor inicial: 8000H
Conteúdo: Endereço do comando DATA atual para uso do comando READ.

DEFTBL (F6CAH,26)
Valor inicial: DEFS 26 (08H ..... 08H)
Conteúdo: Área de armazenamento do tipo de variável por nome em ordem alfabética (F6CAH = “A”; F6CBH = “B”; .....; F6E3H = “Z”). Podem ser alteradas pelo grupo de comandos “DEF xxx”.

3.12 - ÁREA PARA AS FUNÇÕES DO USUÁRIO

PRMSTK (F6E4H,2)
Valor inicial: 0000H
Conteúdo: Definição prévia do bloco FN na pilha do Z80 para coleta de lixo.

PRMLEN (F6E6H,2)
Valor inicial: 0
Conteúdo: Comprimento do bloco de parâmetro FN atual em PARM1.

PARM1 (F6E8H,100)
Valor inicial: DEFS 100 (00H ..... 00H)
Conteúdo: Buffer para armazenamento das variáveis da função FN que está sendo atualmente avaliada.

PRMPRV (F74CH,2)
Valor inicial: PRMSTK (F6E4H)
Conteúdo: Endereço do bloco de parâmetros FN anterior.

PRMLN2 (F74EH,2)
Valor inicial: 0
Conteúdo: Comprimento do bloco de parâmetros FN que está sendo montado em PARM2.

PARM2 (F750H,100)
Valor inicial: DEFS 100 (00H ..... 00H)
Conteúdo: Buffer usado para construir as variáveis locais pertencentes à função FN atual.

PRMFLG (F7B4H,1)
Valor inicial: 0
Conteúdo: Flag para indicar quando PARM1 está sendo avaliada.

ARYTA2 (F7B5H,2)
Valor inicial: 0000H
Conteúdo: Endereço final da área de armazenamento das variáveis que estão sendo examinadas.

NOFUNS (F7B7H,1)
Valor inicial: 0
Conteúdo: Flag para indicar à função FN a existência de variáveis locais (0=não há variáveis; outro valor, há variáveis).

TEMP9 (F7B8H,2)
Valor inicial: 0
Conteúdo: Usada internamente pelo interpretador para armazenamento temporário de várias rotinas.

FUNACT (F7BAH,2)
Valor inicial: 0
Conteúdo: Número de funções FN atualmente ativas.

SWPTMP (F7BCH,8)
Valor inicial: DEFS 8 (00H ..... 00H)
Conteúdo: Buffer usado para conter o primeiro operando de um comando SWAP.

TRCFLG (F7C4H,1)
Valor inicial: 0
Conteúdo: Flag usada pelo comando TRACE (0=TRACE OFF; outro valor, TRACE ON.

3.13 - ÁREA PARA O MATH-PACK

FBUFFR (F7C5H,43)
Valor inicial: DEFS 43 (00H ..... 00H)
Conteúdo: Usada internamente pelo Math-Pack.

DECTMP (F7F0H,2)
Valor inicial: 0
Conteúdo: Usada para transformar um número decimal inteiro em um número de ponto flutuante.

DECTM2 (F7F2H,2)
Valor inicial: 0
Conteúdo: Usada pela rotina de divisão de dupla precisão.

DECCNT (F7F4H,1)
Valor inicial: 0
Conteúdo: Usada pelo rotina de divisão de dupla precisão.

DAC (F7F6H,16)
Valor inicial: DEFS 16 (00H ..... 00H)
Conteúdo: Acumulador primário que contém o valor a ser calculado.

HOLD8 (F806H,64)
Valor inicial: DEFS 64 (00H ..... 00H)
Conteúdo: Buffer usado pela rotina de multiplicação de dupla precisão para armazenar múltiplos de 2 do primeiro operando.

HOLD2 (F836H,8)
Valor inicial: DEFS 8 (00H ..... 00H)
Conteúdo: Usado internamente pelo Math-Pack. Sobrepõe-se ao final de HOLD8.

HOLD (F83EH,8)
Valor inicial: DEFS 8 (00H ..... 00H)
Conteúdo: Usado internamente pelo Math-Pack. Sobrepõe-se ao final de HOLD8.

ARG (F847H,16)
Valor inicial: DEFS 16 (00H ..... 00H)
Conteúdo: Acumulador secundário que contém o segundo operando a ser calculado com DAC (F7F6H).

RNDX (F857H,8)
Valor inicial: DEFS 8 (00H ..... 00H)
Conteúdo: Último número aleatório de dupla precisão gerado. Usada pela função RND.

3.14 - ÁREA DE DADOS DO INTERPRETADOR

MAXFIL (F85FH,1)
Valor inicial: 1
Conteúdo: Total de buffers de I/O existentes. Pode ser alterada pela instrução MAXFILES.

FILTAB (F860H,2)
Valor inicial: F16AH
Conteúdo: Endereço inicial da tabela de apontadores dos FCB´s dos buffers de I/O.

NULBUF (F862H,2)
Valor inicial: F177H
Conteúdo: Endereço inicial do buffer de dados do primeiro FCB. Esse buffer é usado pelos comandos SAVE e LOAD.

PTRFIL (F864H,2)
Valor inicial: 0000H
Conteúdo: Endereço inicial do buffer de dados do FCB atualmente ativo.

RUNFLG (F866H,1)
Valor inicial: 0
Conteúdo: Não zero, se algum programa foi carregado e executado. Usada pelo operando ",R" do comando LOAD. Sobrepõe-se ao início de FILNAM.

FILNAM (F866H,11)
Valor inicial: DEFS 11 (00H ..... 00H)
Conteúdo: Área para armazenamento de um nome de arquivo.

FILNM2 (F871H,11)
Valor inicial: DEFS 11 (00H ..... 00H)
Conteúdo: Área para armazenamento de um nome de arquivo para ser comparado com FILNAM.

NLONLY (F87CH,1)
Valor inicial: 00H
Conteúdo: Flag para indicar se um programa está sendo carregado. O bit 0 é usado para impedir que o buffer 0 de I/O seja fechado durante o carregamento e o bit 7 é usado para impedir que os buffers de I/O do usuário sejam fechados caso um autoprocessamento seja solicitado.

SAVEND (F87DH,2)
Valor inicial: 0000H
Conteúdo: Usada pelo comando BSAVE para conter o endereço final do bloco de memória a ser salvo.

FNKSTR (F87FH,160)
Valor inicial: Conteúdo inicial das teclas de função.
Conteúdo: Buffer usado para armazenar o conteúdo das teclas de função. Divide-se em 10 segmentos de 16 bytes.

CGPNT (F91FH,3)
Valor inicial: 00H, 1BBFH
Conteúdo: Localização do conjunto de caracteres. O primeiro byte é o ID do slot e os outros dois o endereço inicial.

NAMBAS (F922H,2)
Valor inicial: Variável
Conteúdo: Endereço da tabela de nomes no modo texto atual.

CGPBAS (F924H,2)
Valor inicial: Variável
Conteúdo: Endereço da tabela geradora de padrões no modo texto atual.

PATBAS (F926H,2)
Valor inicial: 3800H
Conteúdo: Endereço da tabela geradora de sprites atual.

ATRBAS (F928H,2)
Valor inicial: 1B00H
Conteúdo: Endereço da tabela de atributos dos sprites atual.

CLOC (F92AH,2)
Valor inicial: 0000H
Conteúdo: Endereço do byte da VRAM que contém o pixel atual.

CMASK (F92CH,1)
Valor inicial: 10000000B
Conteúdo: Define o bit dentro do byte que representa o pixel atual.

MINDEL (F92DH,2)
Valor inicial: 0
Conteúdo: Diferença mínima entre os dois pontos extremos de uma linha. Usada pelo comando LINE.

MAXDEL (F92FH,2)
Valor inicial: 0
Conteúdo: Diferença máxima ente os dois pontos extremos de uma linha. Usada pelo comando LINE.

3.15 - ÁREA DE DADOS PARA O 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 setado se uma linha for requerida a partir do ângulo inicial e o bit 7 será setado 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.

CPLOTF (F938H,1)
Valor inicial: 0
Conteúdo: Flag usada para verificar se o ângulo final é menor que o inicial (0=não é; outro valor, é).

CPCNT (F939H,2)
Valor inicial: 0
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: Flag para indicar em qual direção a compressão elíptica deve ser feita (00H=vertical; 01H=horizontal).

CSAVEA (F942H,2)
Valor inicial: 0
Conteúdo: Usada para armazenamento temporário pelo rotina padrão SCANR (012CH).

CSAVEM (F944H,1)
Valor inicial: 0
Conteúdo: Usada para armazenamento temporário pelo rotina padrão SCANR (012CH).

CXOFF (F945H,2)
Valor inicial: 0
Conteúdo: Coordenada X a partir do centro da circunferência.

CYOFF (F947H,2)
Valor inicial: 0
Conteúdo: Coordenada Y a partir do centro da circunferência.

3.16 - 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)
Valor inicial: 0
Conteúdo: Direção da pintura (40H=para baixo; C0H=para cima; 00H=terminar).

LFPROG (F954H,1)
Valor inicial: 0
Conteúdo: Flag usada para indicar se houve progresso à esquerda (0=não houve progresso; outro valor, houve progresso).

RTPROG (F955H,1)
Valor inicial: 0
Conteúdo: Flag usada para indicar se houve progresso à direita (0=não houve progresso; outro valor, houve progresso).

3.17 - ÁREA USADA PELO COMANDO PLAY

MCLTAB (F956H,2)
Valor inicial: 0000H
Conteúdo: Endereço do topo da tabela de comandos usada pelos macro-comandos DRAW ou PLAY.

MCLFLG (F958H,1)
Valor inicial: 0
Conteúdo: Flag usada para indicar qual comando está sendo processado (0=DRAW; não zero=PLAY).

QUETAB (F959H,24)
Valor inicial: Vide conteúdo.
Conteúdo: Essa tabela contém os apontadores para as três filas musicais e para a fila RS232C, reservando seis bytes para cada uma.
+0: posição relativa para colocar
+1: posição relativa para pegar
+2: indicação para devolver
+3: tamanho do da fila
+4/+5: endereço da fila - F975H = voz A
F9F5H = voz B
FA75H = voz C
0000H = RS232C

QUEBAK (F971H,4)
Valor inicial: 00H, 00H, 00H, 00H
Conteúdo: Caracteres de devolução, respectivamente, voz A, voz B, voz C e RS232C.

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.

3.18 - ÁREA ADICIONADA PARA O MSX2 E 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.

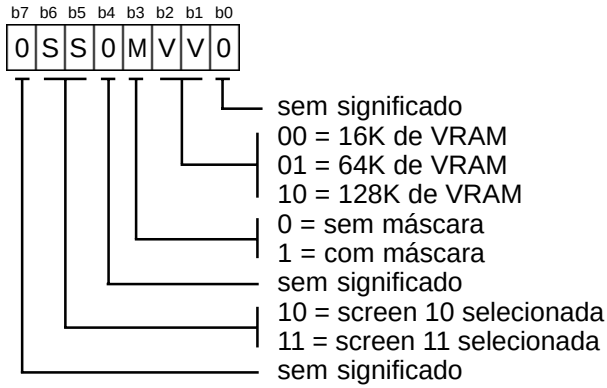
AVCSAV (FAF7H,1)
Valor inicial: 0
Conteúdo: Usada pela porta de controle AV.

EXBRSA (FAF8H,1)
Valor inicial: 10000111B
Conteúdo: Slot da Sub-ROM.

CHRCNT (FAF9H,1)
Valor inicial: 0
Conteúdo: Contador de caracteres no buffer. Usada para a 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 versão japonesa).

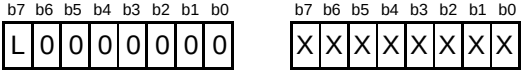
MODE (FAFCH,1)
Valor inicial: 10001001B
Conteúdo: Flag de modo e tamanho da VRAM:



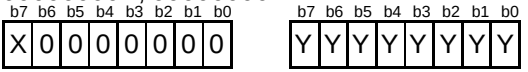
Obs.: a máscara é usada para limitar o endereçamento da VRAM em 16 Kbytes para as screens 0 a 3; de screen 4 para cima não é usada.

NORUSE (FAFDH,1)
Valor inicial: 00H
Conteúdo: Usado pelo Kanji-Driver.
bit 7 - se for 1, indica modo gráfico.
bit 6 - se for 1, rola a tela usando SHIFT + setas.
bit 5,4 - uso interno
bit 3~0 - código de operação lógica do VDP.

XSAVE (FAFEH,2)
Valor inicial: 00000000B, 00000000B
Conteúdo:



YSAVE (FB00H,2)
Valor inicial: 00000000B, 00000000B
Conteúdo:



L=1, requisição de interrupção da caneta ótica
0000000 = sem significado
XXXXXXXXX = coordenada X
YYYYYYYY = coordenada Y

LOGOPR (FB02H,1)
Valor inicial: 00H
Conteúdo: Código de operação lógica para o VDP.

3.19 - ÁREA USADA PELA RS232C

RSTMP (FB03H,50)
Valor inicial: DEFS 50 (00H ..... 00H)
Conteúdo: Área de trabalho para a RS232C ou disco.

TOCNT (FB03H,1)
Valor inicial: 00H
Conteúdo: Usada internamente pela RS232C.

RSFCB (FB04H,2)
Valor inicial: 0000H
Conteúdo: Endereço da RS232C.

RSIQLN (FB06H,1)
Valor inicial: 00H
Conteúdo: Usada internamente pela RS232C.

MEXBIH (FB07H,5)
Valor inicial: C9H, C9H, C9H, C9H, C9H
Conteúdo: +0: RST 030H
+1: Byte ID do slot
+2: Endereço (low)
+3: Endereço (high)
+4: RET

OLDSTT (FB0CH,5)
Valor inicial: C9H, C9H, C9H, C9H, C9H
Conteúdo: +0: RST 030H
+1: Byte ID do slot
+2: Endereço (low)
+3: Endereço (high)
+4: RET

OLDINT (FB12H,5)
Valor inicial: C9H, C9H, C9H, C9H, C9H
Conteúdo: +0: RST 030H
+1: Byte ID do slot
+2: Endereço (low)
+3: Endereço (high)
+4: RET

DEVNUM (FB17H,1)
Valor inicial: 00H
Conteúdo: Usada internamente pela RS232C.

DATCNT (FB18H,3)
Valor inicial: 00H, 00H, 00H
Conteúdo: +0: Byte de dados
+1: Byte apontador
+2: Byte apontador

ERRORS (FB1BH,1)
Valor inicial: 00H
Conteúdo: Usada internamente pela RS232C.

FLAGS (FB1CH,1)
Valor inicial: 00000011B
Conteúdo: Usada internamente pela RS232C.

ESTBLS (FB1DH,1)
Valor inicial: FFH
Conteúdo: Usada internamente pela RS232C.

COMMSK (FB1EH,1)
Valor inicial: C1H
Conteúdo: Usada internamente pela RS232C.

LSTCOM (FB1FH,1)
Valor inicial: E8H
Conteúdo: Usada internamente pela RS232C.

LSTMOD (FB20H,1)
Valor inicial: 01H
Conteúdo: Usada internamente pela RS232C.

3.20 - ÁREA USADA PELO SISTEMA DE DISCO

NONAME (FB21H,20)
Valor inicial: DEFS 20 (00H ..... 00H)
Conteúdo: Usada internamente pelo sistema de disco.

3.21 - ÁREA USADA PELO COMANDO PLAY

PRSCNT (FB35H,1)
Valor inicial: 00H
Conteúdo: Usada pelo comando PLAY para contar o número de operandos completados. O bit 7 será setado após cada um dos três operandos serem analisados.

SAVSP (FB36H,2)
Valor inicial: 0000H
Conteúdo: Salva o valor do registrador SP antes da execução do comando PLAY.

VOICEN (FB38H,1)
Valor inicial: 00H
Conteúdo: Número da voz que está atualmente sendo processada (0=voz A; 1=voz B; 2=voz C).

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)
Valor inicial: 00H
Conteúdo: Usada pelo manipulador de interrupção para armazenar o número da fila musical que está sendo processada.

MUSICF (FB3FH,1)
Valor inicial: 00000000B
Conteúdo: Flag usada para indicar quais filas musicais devem ser iniciadas pelo manipulador de interrupção (bit 0, VOICAQ (F975H); bit 1, VOICBQ (F9F5H); bit 2, VOICCQ (FA75H)).

PLYCNT (FB40H,1)
Valor inicial: 00H
Conteúdo: Número de seqüências do comando PLAY armazenados nas filas musicais.

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
NOTELX	(+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

ÁREA DE DADOS PARA O BUFFER DE PARÂMETROS

VCBA (FB41H,37)
Valor inicial: DEFS 37 (00H ..... 00H)
Conteúdo: Parâmetros para a voz A

VCBB (FB66H,37)
Valor inicial: DEFS 37 (00H ..... 00H)
Conteúdo: Parâmetros para a voz B

VCBC (FB8BH,37)
Valor inicial: DEFS 37 (00H ..... 00H)
Conteúdo: Parâmetros para a voz C

3.22 - ÁREA DE DADOS GERAIS

ENSTOP (FBB0H,1)
Valor inicial: 00H
Conteúdo: Flag usada para habilitar uma saída forçada para o interpretador ao detectar as teclas CTRL+SHIFT+GRAPH+CODE pressionadas juntas (0=desabilitada; outro valor, habilitada).

BASROM (FBB1H,1)
Valor inicial: 00H
Conteúdo: Localização do texto BASIC (0=RAM; outro valor, ROM).

LINTTB (FBB2H,24)
Valor inicial: DEFS 24 (FFH ..... FFH)
Conteúdo: São 24 flags para indicar se cada uma das linhas de texto avançou para a linha seguinte (0=avançou; outro valor, não avançou).

FSTPOS (FBCAH,2)
Valor inicial: 0000H
Conteúdo: Coordenadas do cursor ao iniciar a coleta de caracteres pela rotina INLIN (00B1H) do BIOS.

CODSAV (FBCCH,1)
Valor inicial: 00H
Conteúdo: Caractere substituído pelo cursor nas telas de texto.

FNKSW1 (FBCDH,1)
Valor inicial: 01H
Conteúdo: Flag usada para indicar quais teclas de função são mostradas e habilitadas (1=F1 a F5; 0=F6 a F10).

FNKFLG (FBCEH,10)
Valor inicial: DEFS 10 (00H ..... 00H)
Conteúdo: Flags usadas para habilitar, inibir ou paralisar a execução de uma linha definida pelo comando ON KEY GOSUB. Podem ser modificadas pelo comando KEY(n). [0=KEY(n) OFF/STOP; 1=KEY (n) ON].

ONGSBF (FBD8H,1)
Valor inicial: 00H
Conteúdo: Flag usada para indicar se algum dispositivo requereu uma interrupção de execução (0=não requereu; outro valor, interrupção ativa).

CLIKFL (FBD9H,1)
Valor inicial: 00H
Conteúdo: Flag de “click” das teclas. Usada pelo manipulador de interrupção.

OLDKEY (FBDAH,11)
Valor inicial: DEFS 11 (FFH ..... FFH)
Conteúdo: Estado anterior da matriz do teclado.

NEWKEY (FBE5H,11)
Valor inicial: DEFS 11 (FFH ..... FFH)
Conteúdo: Estado atual da matriz do teclado. As transições de teclas são detectadas por comparação com OLDKEY.

KEYBUF (FBF0H,40)
Valor inicial: DEFS 40 (00H ..... 00H)
Conteúdo: Buffer circular que contém os caracteres decodificados digitados no teclado.

LINWRK (FC18H,40)
Valor inicial: DEFS 40 (00H ..... 00H)
Conteúdo: Buffer que contém uma linha completa de caracteres de uma tela de texto.

PATWRK (FC40H,8)
Valor inicial: DEFS 8 (FFH ..... FFH)
Conteúdo: Buffer que contém um padrão de caractere 8x8.

BOTTOM (FC48H,2)
Valor inicial: 8000H
Conteúdo: Endereço mais baixo de RAM que pode ser usado pelo interpretador.

HIMEM (FC4AH,2)
Valor inicial: F380H
Conteúdo: Endereço mais alto de RAM disponível. Pode ser modificado pelo comando CLEAR.

TRPTBL (FC4CH,78)
Valor inicial: DEFS 78 (00H ..... 00H)
Conteúdo: Essa tabela contém o estado atual dos comandos de interrupção. Cada comando 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.

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	

RTYCNT (FC9AH,1)
Valor inicial: 00H
Conteúdo: Usada internamente pelo interpretador.

INTFLG (FC9BH,1)
Valor inicial: 00H
Conteúdo: Contém 03H ou 04H, se CTRL+STOP ou STOP forem pressionadas, respectivamente.

PADY (FC9CH,1)
Valor inicial: 00H
Conteúdo: Coordenada vertical do paddle, mouse ou touchpad.

PADX (FC9DH,1)
Valor inicial: 00H
Conteúdo: Coordenada horizontal do paddle, mouse ou touchpad.

JIFFY (FC9EH,2)
Valor inicial: 0000H
Conteúdo: Essa 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.

INTVAL (FCA0H,2)
Valor inicial: 0000H
Conteúdo: Duração do intervalo estabelecido pela instrução ON INTERVAL.

INTCNT (FCA3H,2)
Valor inicial: 0000H
Conteúdo: Contador para a instrução ON INTERVAL.

LOWLIM (FCA4H,1)
Valor inicial: 31H
Conteúdo: Usada pelo cassete. Atualmente obsoleta.

WINWID (FCA5H,1)
Valor inicial: 22H
Conteúdo: Usada pelo cassete. Atualmente obsoleta.

GRPHED (FCA6H,1)
Valor inicial: 00H
Conteúdo: Flag usada para o envio de um caractere gráfico (00H= normal; 1=cabeçalho gráfico).

ESCCNT (FCA7H,1)
Valor inicial: 00H
Conteúdo: Contador de parâmetros de escape.

INSFLG (FCA8H,1)
Valor inicial: 00H
Conteúdo: Flag para indicar modo de inserção (00H=normal; FFH= modo de inserção ativo).

CSRSW (FCA9H,1)
Valor inicial: 00H
Conteúdo: Flag usada para indicar se o cursor será mostrado (0=não, outro valor, sim). Pode ser alterada pelo comando LOCATE.

CSTYLE (FCAAH,1)
Valor inicial: 00H
Conteúdo: Forma do cursor (0=bloco; outro valor, sub-alinhado).

CAPST (FCABH,1)
Valor inicial: 00H
Conteúdo: Estado de CAPS LOCK (0=desligado; outro valor, ligado).

KANAST (FCACH,1)
Valor inicial: 00H
Conteúdo: Estado de KANA LOCK (0=desligado; outro valor, ligado).

KANAMD (FCADH,1)
Valor inicial: 00H
Conteúdo: Modo do teclado em máquinas japonesas.

FLBMEM (FCAEH,1)
Valor inicial: 00H
Conteúdo: Flag para indicar carregamento de programas em BASIC (0=está carregando; outro valor, não).

SCRMOD (FCAFH,1)
Valor inicial: 00H
Conteúdo: Modo de tela atual até Screen 8. Acima de Screen 8, deve ser usada em associação com MODE (FAFCH) e R25SAV (FFFAH), casos em que sempre conterá 8.

OLDSCR (FCB0H,1)
Valor inicial: 00H
Conteúdo: Modo de tela do último modo texto.

CASPRV (FCB1H,1)
Valor inicial: 00H
Conteúdo: Usada pelo cassete nos modelos MSX, MSX2 e MSX2+. Nos modelos MSX turbo R guarda o valor da porta A7H.

BDRATR (FCB2H,1)
Valor inicial: 00H
Conteúdo: Código da cor de contorno de um polígono. Usado pela instrução PAINT.

GXPOS (FCB3H,2)
Valor inicial: 0000H
Conteúdo: Armazenamento temporário da coordenada horizontal gráfica.

GYPOS (FCB5H,2)
Valor inicial: 0000H
Conteúdo: Armazenamento temporário da coordenada vertical gráfica.

GRPACX (FCB7H,2)
Valor inicial: 0000H
Conteúdo: Coordenada horizontal gráfica atual.

GRPACY (FCB9H,2)
Valor inicial: 0000H
Conteúdo: Coordenada vertical gráfica atual.

DRWFLG (FCBBH,1)
Valor inicial: 00H
Conteúdo: Flag usada pelo comando DRAW.

DRWSCL (FCBCH,1)
Valor inicial: 00H
Conteúdo: Fator de escala para o comando DRAW. O valor 0 indica que não será usada a escala.

DRWANG (FCBDH,1)
Valor inicial: 00H
Conteúdo: Ângulo para o comando DRAW.

RUNBNF (FCBEH,1)
Valor inicial: 00H
Conteúdo: Flag para indicar execução automática após carregamento pelo comando BLOAD (0=não, outro valor, sim).

SAVENT (FCBFH,2)
Valor inicial: 0000H
Conteúdo: Endereço inicial para os comando BSAVE e BLOAD.

3.23 - 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.

SLTTBL (FCC5H,4)
Valor inicial: Variável.
Conteúdo: Esses quatro bytes contêm o estado possível dos quatro registradores de slot primário, no caso do slot estar expandido.

SLTATR (FCC9H,64)
Valor inicial: Variável.
Conteúdo: Tabela de atributos de cada slot.

SLTWRK (FD09H,128)
Valor inicial: Variável.
Conteúdo: Esta tabela aloca dois bytes como área de trabalho para cada página de cada slot.

PROCNM (FD89H,16)
Valor inicial: DEFS 16 (00H ..... 00H)
Conteúdo: Armazena o nome de uma instrução expandida (comando CALL) ou expansão de dispositivo (comando OPEN). Um byte 00H indica o fim do nome.

DEVICE (FD99H,1)
Valor inicial: 00H
Conteúdo: Usada para passar um código de dispositivo de 0 a 3 para uma ROM de expansão.

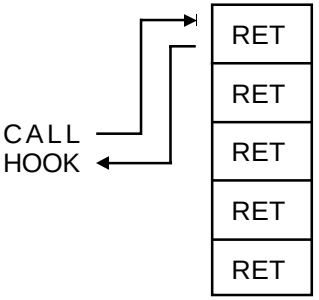
3.24 - OS HOOKS

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).

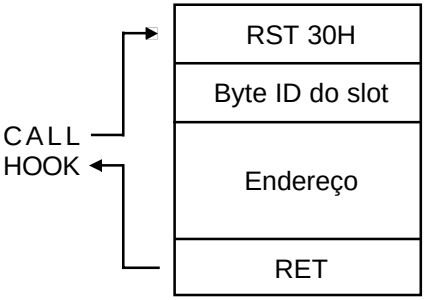
Os hooks são chamados de posições estratégicas do BIOS de modo que as operações do BIOS/Interpretador possam ser modificadas ou ampliadas. Cada hook tem espaço suficiente para uma chamada distante para qualquer slot. Como são preenchidos inicialmente com instruções RET, isso causa apenas um retorno para o 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 na página seguinte.



condição normal



Hook expandido para chamada inter-slot pela rotina CALLF

DESCRIÇÃO DOS HOOKS

A notação para a descrição dos hooks é a seguinte:

LABEL (Endereço)
Chamada:
Objetivo:

Onde label é o nome do hook, endereço é o endereço inicial do hook; chamada é o ponto da rotina que chama o hook e objetivo é a função de ampliação para a qual o hook foi criado. Segue a descrição de todos os hooks.

HKEYI (FD9AH)
Chamada: Início do manipulador de interrupção.
Objetivo: Adicionar rotinas que requeiram interrupção.

HTIMI (FD9FH)
Chamada: Início da rotina do manipulador de interrupção..
Objetivo: Adicionar rotinas de manipulação de interrupção.

HCHPU (FDA4H)
Chamada: Início da rotina CHPUT (saída de caractere).
Objetivo: Conectar outros dispositivos de console.

HDSPC (FDA9H)
Chamada: Início da rotina DSPSCR (apresenta cursor).
Objetivo: Conectar outros dispositivos de console.

HERAC (FDAEH)
Chamada: Início da rotina ERASCR (apaga cursor)
Objetivo: Conectar outros dispositivos de console.

HDSPF (FDB3H)
Chamada: Início da rotina DSPFNK (apresenta teclas de função).
Objetivo: Conectar outros dispositivos de console.

HERAF (FDB8H)
Chamada: Início da rotina ERAFNK (apaga teclas de função)
Objetivo: Conectar outros dispositivos de console.

HTOTE (FDBDH)
Chamada: Início da rotina TOTEXT (força tela para modo texto)
Objetivo: Conectar outros dispositivos de console.

HCHGE (FDC2H)
Chamada: Início da rotina CHGET (pega um caractere).
Objetivo: Conectar outros dispositivos de console.

HINIP (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.

HKEYA (FDD1H)
Chamada: Início de MSXIO NMI (Key easy)
Objetivo: Mudar a configuração do teclado.

HNMI (FDD6H)
Chamada: Início do manipulador de interrupção não mascarável.
Objetivo: Ganchos NMI.

HPINL (FDDBH)
Chamada: Início da rotina PINLIN (pega uma linha)
Objetivo: Usar outros dispositivos e/ou métodos de entrada.

HQINL (FDE0H)
Chamada: Início da rotina QINLIN (pega uma linha apresentando "?").
Objetivo: Usar outros dispositivos e/ou métodos de entrada.

HINLI (FDE5H)
Chamada: Início da rotina INLIN.
Objetivo: Usar outros dispositivos e/ou métodos de entrada.

HONGO (FDEAH)
Chamada: Início do manipulador do comando ON GOTO.
Objetivo: Usar outros dispositivos de manipulação de interrupção.

HDSKO (FDEFH)
Chamada: Início do manipulador do comando DSKO$.
Objetivo: Conectar dispositivos de disco.

HSETS (FDF4H)
Chamada: Início do manipulador do comando SET.
Objetivo: Conectar dispositivos de disco e/ou expandir o comando SET.

HNAME (FDF9H)
Chamada: Início do manipulador do comando NAME.
Objetivo: Conectar dispositivos de disco.

HKILL (FDFEH)
Chamada: Início do manipulador do comando KILL.
Objetivo: Conectar dispositivos de disco.

HIPL (FE03H)
Chamada: Início do manipulador do comando IPL
Objetivo: Conectar dispositivos de disco ou expandir o comando IPL.

HCOPY (FE08H)
Chamada: Início do manipulador do comando COPY.
Objetivo: Conectar dispositivos de disco.

HCMD (FE0DH)
Chamada: Início do manipulador do comando CMD.
Objetivo: Conectar dispositivos de disco ou expandir o comando CMD.

HDSKF (FE12H)
Chamada: Início do manipulador do comando DSKF.
Objetivo: Conectar dispositivos de disco.

HDSKI (FE17H)
Chamada: Início do manipulador do comando DSKI$.
Objetivo: Conectar dispositivos de disco.

HATTR (FE1CH)
Chamada: Início do manipulador do comando ATTR$.
Objetivo: Conectar dispositivos de disco.

HLSET (FE21H)
Chamada: Início do manipulador do comando LSET.
Objetivo: Conectar dispositivos de disco.

HRSET (FE26H)
Chamada: Início do manipulador do comando 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 arquivo 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 seqüencial.
Objetivo: Conectar dispositivos de disco.

HINDS (FE8AH)
Chamada: Manipulador de entrada seqüencial.
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: Pegar um nome de dispositivo.
Objetivo: Expandir nome lógico de dispositivo.

HNODE (FEB7H)
Chamada: Dispositivo sem nome.
Objetivo: Seta nome default de um dispositivo em 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: Despachar função I/O (assinalar dispositivo).
Objetivo: Expandir nome lógico de dispositivo.

HRUNC (FECBH)
Chamada: Inicializar variáveis do interpretador para comando RUN.
Objetivo: Expandir funções do interpretador.

HCLEA (FED0H)
Chamada: Inicializar variáveis do interpretador para comando CLEAR.
Objetivo: Expandir funções do interpretador.

HLOPD (FED5H)
Chamada: Inicializar variáveis do interpretador (geral).

HSTKE (FEDAH)
Chamada: Repor pilha (erro na pilha).

HISFL (FEDFH)
Chamada: Início da rotina ISFLIO (I/O de arquivo).

HOUTD (FEE4H)
Chamada: Início da rotina OUTDO.

HCRDO (FEE9H)
Chamada: Executar CR+LF para a rotina OUTDO.

HDSKC (FEEEH)
Chamada: Entrada de atributo de disco.

HDOGR (FEF3H)
Chamada: Início da rotina que traça uma linha cujas coordenadas estão em GXPOS e GYPOS.

HPRGE (FEF8H)
Chamada: Início da rotina de término de um programa BASIC.

HERRP (FEFDH)
Chamada: Início da rotina de apresentação de mensagens de erro.

HERRF (FF02H)
Chamada: Manipulador de arquivo.

HREAD (FF07H)
Chamada: "Ok" do loop principal (interpretador pronto).

HMAIN (FF0CH)
Chamada: Início do loop principal de execução de texto BASIC do interpretador.

HDIRD (FF11H)
Chamada: Executar comando direto (declaração direta).

HFINI (FF16H)
Chamada: Término do loop principal para comando AUTO ativo.

HFINE (FF1BH)
Chamada: Término do loop principal.

HCRUN (FF20H)
Chamada: Atomização de linha de texto (42B9H).

HCRUS (FF25H)
Chamada: Atomização de linha de texto (4353H).

HISRE (FF2AH)
Chamada: Atomização de linha de texto (437CH).

HNTFN (FF2FH)
Chamada: Atomização de linha de texto (43A4H).

HNOTR (FF34H)
Chamada: Atomização de linha de texto (44EBH).

HSNGF (FF39H)
Chamada: Início do manipulador do comando FOR.

HNEWS (FF3EH)
Chamada: Início da rotina NEWSTT (4601H) do interpretador.

HGONE (FF43H)
Chamada: Ponto de execução de instruções de NEWSTT.

HCHRG (FF48H)
Chamada: Início da rotina CHRGTR.

HRETU (FF4DH)
Chamada: Início do manipulador do comando RETURN.

HPTRF (FF52H)
Chamada: Início do manipulador do comando PRINT.

HCOMP (FF57H)
Chamada: Manipulador do comando PRINT (4A94H).

HFINP (FF5CH)
Chamada: Início da rotina que zera PRTFLG e PRTFIL para finalização do comando PRINT.

HTRMN (FF61H)
Chamada: Início do manipulador de erro dos comando READ e INPUT.

HFRME (FF66H)
Chamada: Rotina FRMEVL (4C64H) - Avaliador de Expressões.

HNTPL (FF6BH)
Chamada: Rotina FRMEVL (4CA6H) - Avaliador de Expressões.

HEVAL (FF70H)
Chamada: Avaliador de Fatores (4DD9H)

HOKNO (FF75H)
Chamada: Processamento de tokens de função prefixados por FFH (4F2CH). Usada pelo Avaliador de Fatores.

HFING (FF7AH)
Chamada: Processamento de tokens de função prefixados por FFH (4F3EH). Usada pelo Avaliador de Fatores.

HISMI (FF7FH)
Chamada: Confirma se está executando o cmando MID$ ou não.

HWIDT (FF84H)
Chamada: Início do manipulador do comando WIDTH.

HLIST (FF89H)
Chamada: Início do manipulador do comando LIST.

HBUFL (FF8EH)
Chamada: Linha de buffer (de-simbolizar para comando LIST (532DH)).

HFRQI (FF93H)
Chamada: Converte para inteiro (543FH).

HSCNE (FF98H)
Chamada: Converte número de linha para apontador (5514H).

HFRET (FF9DH)
Chamada: Examina se é último descritor de string em TEMPST (67EEH).

HPTRG (FFA2H)
Chamada: Procura apontador de variável (5EA9H).
Objetivo: Usar outro valor default para as variáveis.

HPHYD (FFA7H)
Chamada: Início da rotina PHYDIO.
Objetivo: Conectar dispositivos de disco.

HFORM (FFACH)
Chamada: Início da rotina FORMAT.
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.
Objetivo: Usar outros modelos de impressoras.

HLPTS (FFBBH)
Chamada: Início da rotina LPTSTT.
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.

CALL (FFCAH)
Objetivo: Usado internamente pelo BIOS expandido.

DISINT (FFCFH)
Objetivo: Usado internamente pelo BDOS.

ENAINT (FFD4H)
Objetivo: Usado internamente pelo BDOS.

3.25 - ÁREA USADA PARA O VDP V9938

RG8SAV (FFE7H,1)
Valor inicial: 08H
Conteúdo: Cópia do registrador R#8 do VDP.

RG9SAV (FFE8H,1)
Valor inicial: 08H
Conteúdo: Cópia do registrador R#9 do VDP.

R10SAV (FFE9H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#10 do VDP.

R11SAV (FFEAH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#11 do VDP.

R12SAV (FFEBH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#12 do VDP.

R13SAV (FFECH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#13 do VDP.

R14SAV (FFEDH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#14 do VDP.

R15SAV (FFEEH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#15 do VDP.

R16SAV (FFEFH,1)
Valor inicial: 0FH
Conteúdo: Cópia do registrador R#16 do VDP.

R17SAV (FFF0H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#17 do VDP.

R18SAV (FFF1H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#18 do VDP.

R19SAV (FFF2H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#19 do VDP.

R20SAV (FFF3H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#20 do VDP.

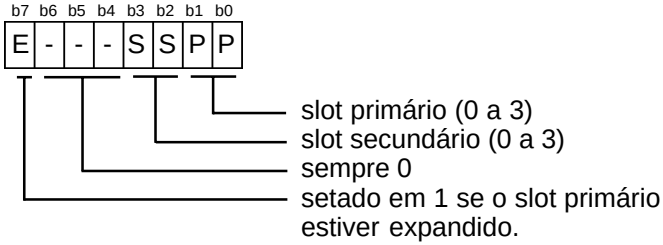
R21SAV (FFF4H,1)
Valor inicial: F4H
Conteúdo: Cópia do registrador R#21 do VDP.

R22SAV (FFF5H,1)
Valor inicial: 5BH
Conteúdo: Cópia do registrador R#22 do VDP.

R23SAV (FFF6H,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#23 do VDP.

3.26 - SLOT DA MAIN-ROM.

(FFF7H,1) - Slot da Main-ROM, no formato abaixo:



3.27 - ÁREA USADA PARA O VDP V9958

(FFF8H,1) - ?
(FFF9H,1) - ?

R25SAV (FFFAH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#25 do VDP.

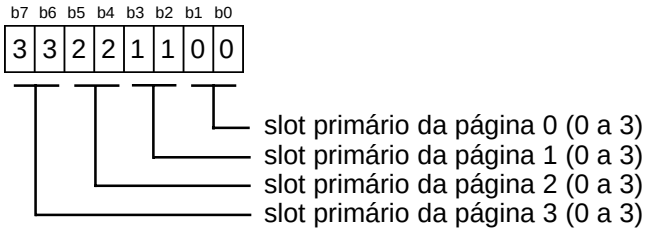
R26SAV (FFFBH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#26 do VDP.

R27SAV (FFFCH,1)
Valor inicial: 00H
Conteúdo: Cópia do registrador R#27 do VDP.

(FFFDH,1) - ?
(FFFEH,1) - ?

3.28 - REGISTRADOR DE SLOT SECUNDÁRIO

(FFFFH,1) - Registrador de slot secundário, no formato abaixo:



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 processamentoe 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; foi lançado apenas um cartucho de vídeo que o utilizava.

1 - CONFIGURAÇÕES DO MSX-VIDEO

O V9938 tem as seguintes características principais:

- Paleta de 512 cores;
- Resolução máxima de 512 x 424 pontos com 16 cores;
- Máximo de 256 cores apresentadas simutaneamente;
- 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 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”.

O V9958 acrescentou novas características ao vídeo, dentre as quais:

- Máximo de 19268 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 vertical e horizontal por hardware;
- DAC de 5 bits por cor primária.

O V9958 também foi usado nos modelos MSX turbo R, lançados em 1990 e 1991.

1.1 - DESCRIÇÃO DOS REGISTRADORES

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 status podem ser acessados diretamente pelo BASIC e pelo BIOS. O terceiro grupo, inexistente no TMS9918A, é o de paletas, e não pode ser acessado diretamente.

O grupo de registros 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#27[10] 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. A tabela abaixo descreve resumidamente as funções de cada registrador desse grupo.

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.
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)	Endereço 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 00000000B).
R#16	VDP(17)	Especificação indireta para P#n (preset 00000000B).
R#17	VDP(18)	Especificação indireta para R#n (preset 00000000B).
R#18	VDP(19)	Ajuste de tela
R#19	VDP(20)	Examina linha ao ocorrer interrupção

Nota 10: O registrador R#24 não existe.

R#20	VDP(21)	Burst de cor para a fase 0 (preset 00000000B).
R#21	VDP(22)	Burst de cor para a fase 1/3 (preset 00111011B).
R#22	VDP(23)	Burst de cor para a fase 2/3 (preset 00000101B).
R#23	VDP(24)	Scroll vertical
R#24		Esse registrador não existe.
R#25	VDP(26)	Registrador de modo #4 (V9958).
R#26	VDP(27)	Scroll horizontal (V9958).
R#27	VDP(28)	Scroll horizontal fino (V9958).
R#32	VDP(33)	SX: coordenada horizontal a ser transferida (low).
R#33	VDP(34)	SX: coordenada horizontal a ser transferida (high).
R#34	VDP(35)	SY: coordenada vertical a ser transferida (low).
R#35	VDP(36)	SY: coordenada vertical a ser transferida (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 tranferir na direção horizontal (low).
R#41	VDP(42)	NX: número de pontos a tranferir na direção horizontal (high).
R#42	VDP(43)	NX: número de pontos a tranferir na direção vertical (low).
R#43	VDP(44)	NX: número de pontos a tranferir 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 um comando ao VDP.

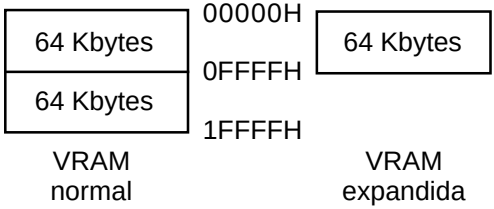
O grupo seguinte é o grupo de registradores de estado. São registradores de 8 bits somente de leitura designados por S#0 a S#9. O único presente no TMS9918 é o S#0. A listagem abaixo descreve suas funções.

S#0	VDP(8)	Informação de interrupção.
S#1	VDP(-1)	Informação de interrupção.
S#2	VDP(-2)	Registro 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 um comando do VDP.
S#8	VDP(-8)	Coordenada horizontal obtida por um comando de procura (low).
S#9	VDP(-9)	Coordenada vertical obtida por um comando de procura (low).

1.2 - A VRAM

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

Opcionalmente pode ser conectado mais um banco de 64 Kbytes de expansão (no caso do V9938 em diante). Entretanto, não há especificações de como acessar essa expansão, de forma que podem haver problemas de incompatibilidade entre máquinas diferentes que usem a expansão. Ela é mapeada como se segue:



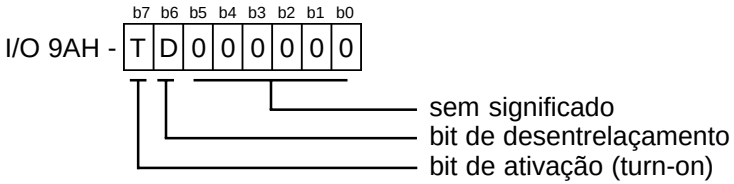
1.3 - A ADVRAM

A ADVRAM é 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 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  ;04000H a 07FFFH
 |         |      |        |
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 se acessado pela porta de I/O 9AH. A estrutura desse resgistrador está ilustrada na página seguinte.



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 seqüência de instruções:

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

Parece estranho usar uma instrução IN para escrever no registrador de controle, mas é assim mesmo. Isso acontece por características técnicas inerentes à ADVRAM.

1.4 - PORTAS DE ACESSO AO VDP

Os VDP´s V9938 e V9958 têm quatro portas de I/O 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 as operações de I/O com o VDP requererem alta velocidade, essas portas podem ser usadas para acessar o VDP diretamente. Entretando, o VDP costuma ser lento em algumas operações, requerendo até 8 μs de intervalo entre acessos consecutivos. Por isso, é bom evitar instruções tipo OTIR, o que pode acarretar falhas na leitura de dados pelo VDP. 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)
Porta #3	(escrita)	w+3	Escreve no registrador especificado indiretamente (MSX2)

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

2 - ACESSO À VRAM E AO VDP

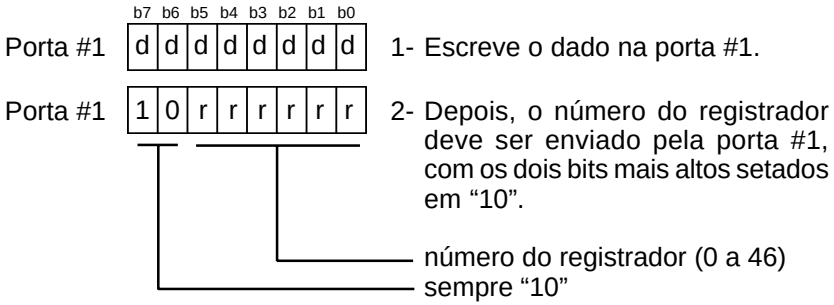
O VDP e a VRAM podem ser acessados diretamente pelas portas de I/O já descritas. Essa seção descreve como fazê-lo.

2.1 - ACESSO AOS REGISTRADORES DE CONTROLE

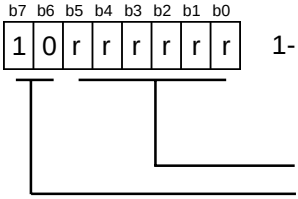
Os registradores de controle são apenas de escrita. Entretanto, o conteúdo do primeiro subgrupo (R#0 a R#27) pode ser obtido pelo comando VDP(n) do BASIC porque há uma cópia deles na área de trabalho do sistema (endereços F3DFH a F3E6H, FFE7H a FFF6H e FFFAH a FFFCH). Essa cópia é feita pelo BIOS.

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

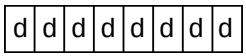
Acesso direto. O primeiro meio é especificar o dado e escrevê-lo diretamente. O dado é escrito primeiro, na porta #1, seguido do número do registrador respectivo, conforme ilustrado abaixo:



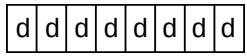
Acesso indireto. O segundo meio é escrever o dado no registrador especificado por R#17. Para isso, é necessário usar o método direto para colocar o número do registrador desejado em R#17, com os dois bits mais altos setados em "10". Depois pode-se enviar dados continuamente pela porta #3 para o mesmo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



1- O número do registrador deve ser colocado em R#17 com os dois bits mais altos setados em “10”.

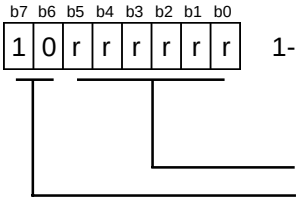


2- O dado enviado pela porta #3 será colocado no registrador especificado em R#17.

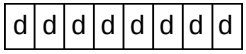


3- Outros dados podem ser enviados pela porta #3 para o mesmo registrador, idefinidamente.

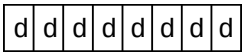
Acesso indireto com autoincremento. Esse meio é similar ao anterior, com a diferença que, cada vez que o dado for escrito pela porta #3, R#17 é incrementado em 1 e o próximo dado enviado será escrito no registrador seguinte. Para usá-lo, é necessário colocar, pelo método direto, o número do primeiro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



1- O número do registrador deve ser colocado em R#17 com os dois bits mais altos setados em “00”



2- O dado enviado pela porta #3 será colocado no registrador R#n (n é o número do registrador especificado em R#17).

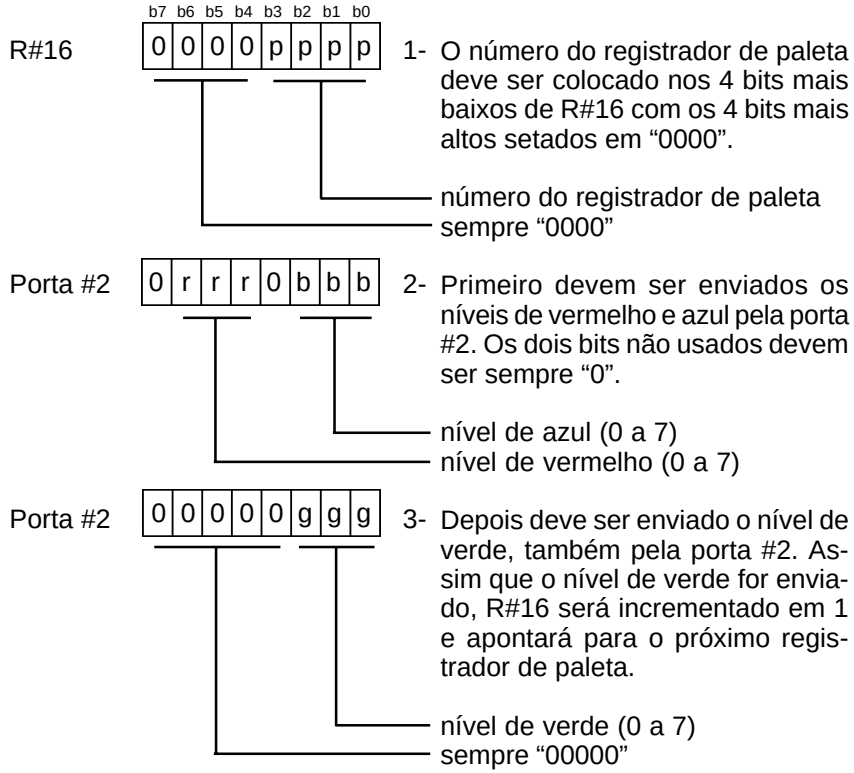


3- O próximo dado enviado pela porta #3 será colocado em R#(n+1) e assim por diante.

2.2 - ACESSO AOS REGISTRADORES DE PALETA

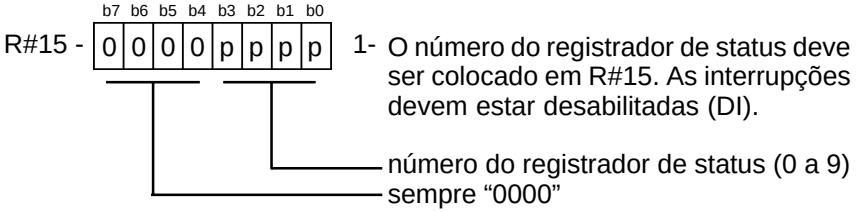
Para escrever nos registradores de paleta (P#0 a P#15), é necessário especificar o número da paleta nos quatro bits mais baixos de R#16 e enviar os dados pela porta #2. Como cada registrador tem 9 bits, os dados devem ser enviados por dois bytes consecutivos. Depois que os

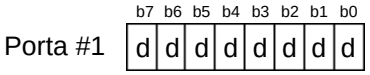
dois bytes forem enviados, R#16 é automaticamente incrementado em 1, apontando para o próximo registrador de paleta. Essa característica torna a paleta fácil de ser inicializada.



2.3 - LENDO OS REGISTRADORES DE STATUS

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



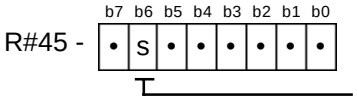


2- O dado fica disponível para ser lido pela porta #1.

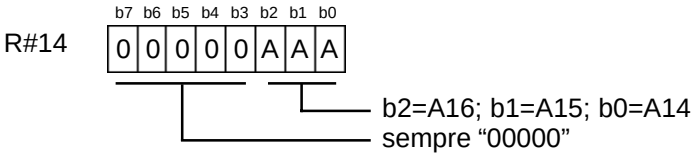
3- R#15 deve ser zerado antes de reabilitar as interrupções.

2.4 - ACESSO À VRAM PELA CPU

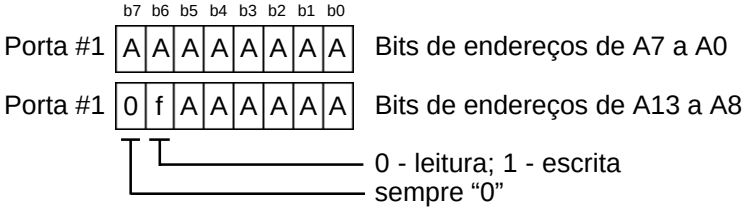
Usando a RAM expandida. Os primeiros 64 Kbytes da VRAM e a VRAM expandida ocupam o mesmo espaço de endereçamento do VDP. Se o micro não possuir VRAM expandida, sempre será selecionada a VRAM principal. O bit 6 de R#45 pode ser usado para alternar entre os dois bancos de 64 Kbytes.



Setando a página da VRAM. Para acessar os 128 Kbytes de VRAM, o VDP usa 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 de memória.

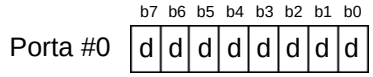


Selecionando o endereço da VRAM. Os 14 bits mais baixos de endereçamento da VRAM devem ser enviados pela porta #1 em dois bytes consecutivos. O bit 6 do segundo byte é uma flag para indicar leitura ou escrita: 0 para leitura e 1 para escrita.



Lendo e escrevendo na VRAM. 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.



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

3 - MODOS DE TELA DOS VDP´s TMS9918, V9938 e V9958

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	40 caracteres por linha de texto; uma cor para todos os caracteres.
TEXTO 2 *	SCREEN 0 WIDTH = 80	80 caracteres por linha de texto; função de "blink" inclusa.
MULTICOR	SCREEN 3	Pseudo-gráfico; um caractere é dividido em 4 blocos.
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 de 2 em 2 para cada 8 pontos horizontais.
GRÁFICO 3 *	SCREEN 4	Igual a Gráfico 2, mas usa sprites modo 2
GRÁFICO 4 *	SCREEN 5	256 x 192 pontos; 16 cores de 512 para cada ponto
GRÁFICO 5 *	SCREEN 6	512 x 212 pontos; 4 cores de 512 para cada ponto
GRÁFICO 6 *	SCREEN 7	512 x 212 pontos; 16 cores de 512 para cada ponto
GRÁFICO 7 *	SCREEN 8	256 x 212 pontos; 256 cores para cada ponto
GRÁFICO 8 **	SCREEN 10 SCREEN 11	256 x 212 pontos; 65536 cores para cada 4 pontos horizontais ou 16 cores de 512 para cda ponto; máximo de 12499 cores simultâneas
GRÁFICO 9 **	SCREEN 12	256 x 212 pontos; 131072 cores para cada 4 pontos horizontais; máximo de 19268 cores simultâneas

3.1 MODO TEXTO 1

- 24 linhas de até 40 caracteres cada;
- uma cor de fundo e uma para os caracteres, selecionadas de 16 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.

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

	b7	b6	b5	b4	b3	b2	b1	b0
R#0	•	•	•	•	0	0	0	•
R#1	•	•	•	1	0	•	•	•

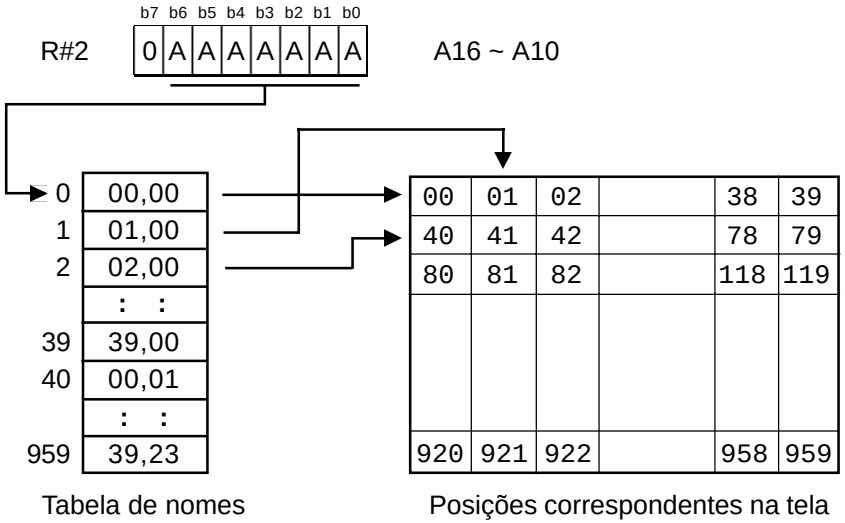
A área onde a fonte de caracteres é armazenada chama-se *tabela geradora de padrões*. Nela, cada caractere é definido por 8 bytes, mas os dois bits mais baixos de cada byte não são exibidos. Por isso, a célula onde cada caractere é mostrado tem 6 x 8 pontos. A fonte contém 256 caracteres distintos, numerados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

A localização da tabela geradora de padrões está especificada em R#4. Os 11 bits mais baixos 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 00000H. Esse endereço pode ser obtido pela variável de sistema BASE(2) do BASIC.

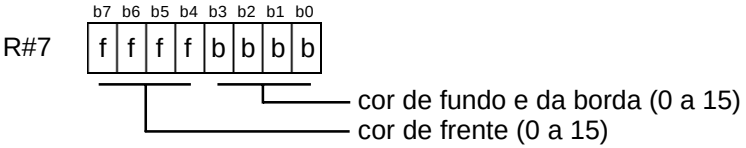
	b7	b6	b5	b4	b3	b2	b1	b0	
R#4	0	0	A	A	A	A	A	A	A16 ~ A11

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 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 00000H. Esse endereço pode ser obtido pela variável de sistema BASE(0) do BASIC.



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.



3.2 - MODO TEXTO 2

- 24 ou 26,5 linhas de 41 a 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 caractere;
- requer 2048 bytes para a fonte (256 caracteres x 8 bytes);
- 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 80).

O modo texto 2 é selecionado por R#0 e R#1 conforme ilustração da página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0
R#0	•	•	•	•	0	1	0	•
R#1	•	•	•	1	0	•	•	•

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.

	b7	b6	b5	b4	b3	b2	b1	b0
R#9	x	•	•	•	•	•	•	•

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 00000H.

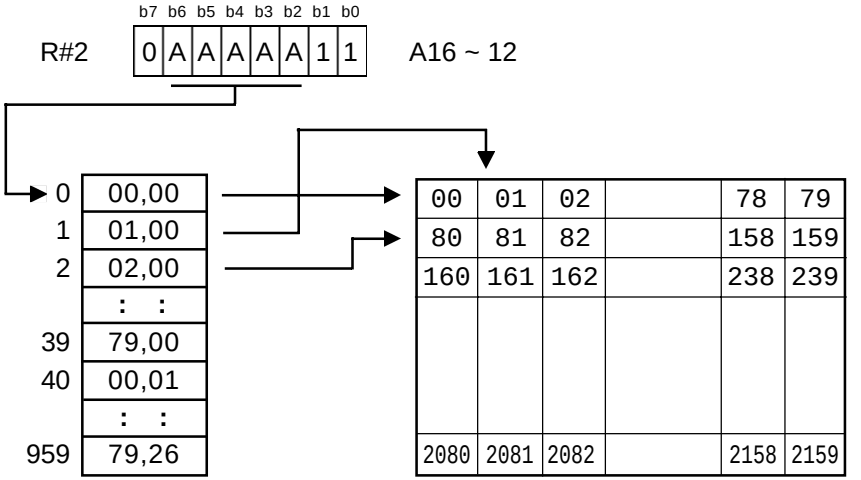


Tabela de nomes

Posições correspondentes na tela

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 os endereço e os 9 bits mais baixos são sempre 0. Por isso, o endereço da tabela de “blink” é sempre um múltiplo de 512 bytes a partir de 00000H.

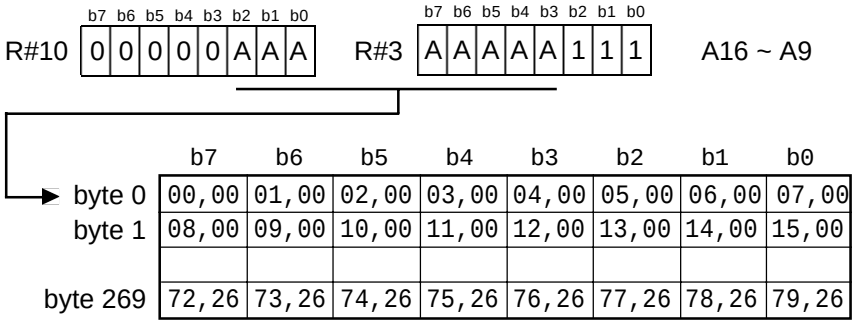
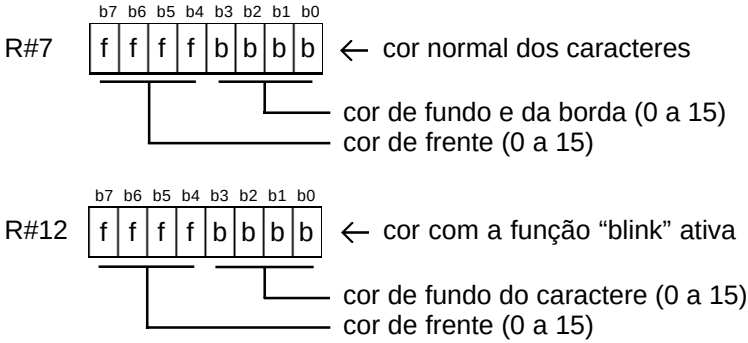
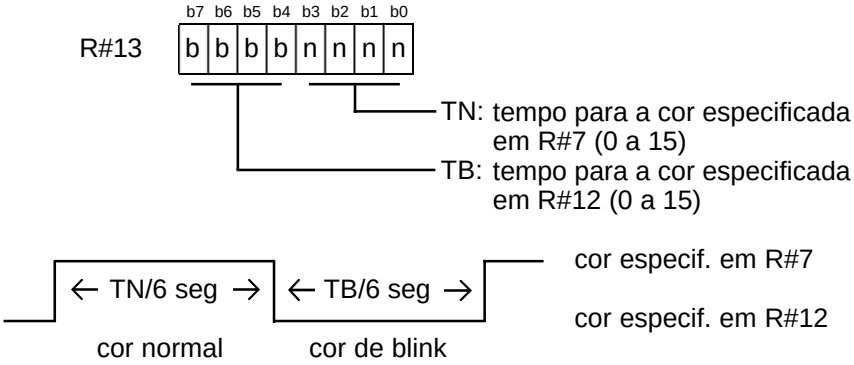


Tabela de “blink”

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.



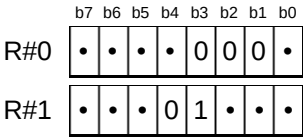
O tempo de “blinking” (tempo que o caractere assume as cores de “blink” e depois volta às cores normais) é especificado em R#13. Os quatro bits mais altos de R#13 definem o tempo em que o caractere fica com a cor original e os quatro bits mais baixos definem o tempo em que os caracteres ficam com a cor 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”.



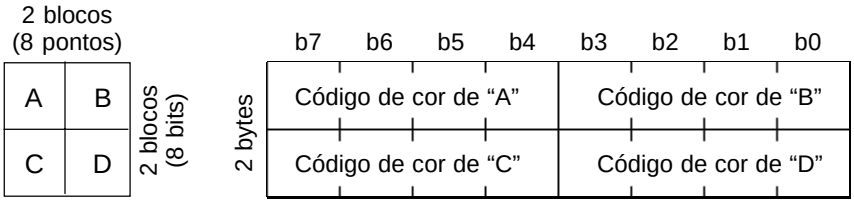
3.3 - MODO MULTICOR

- 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 00000H.

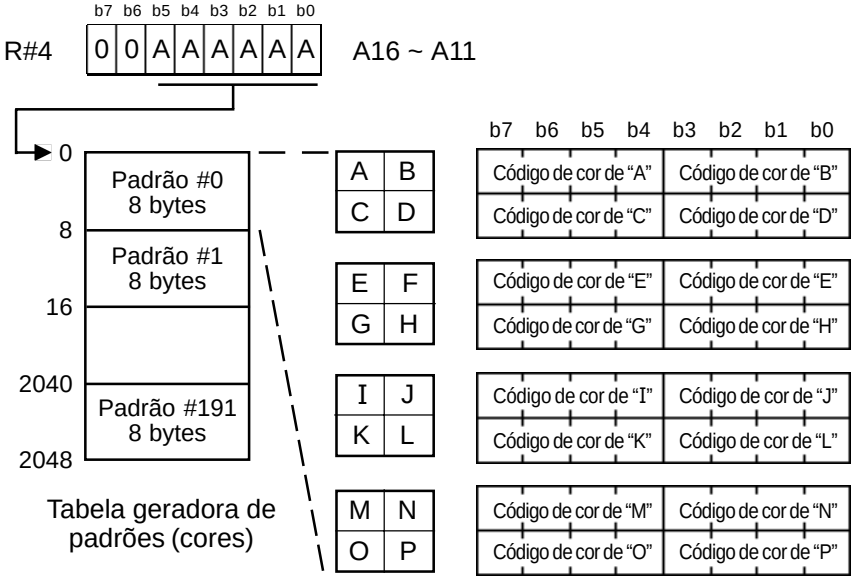
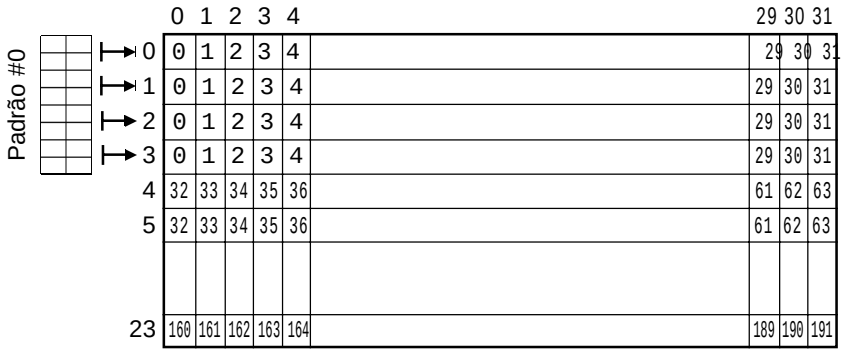


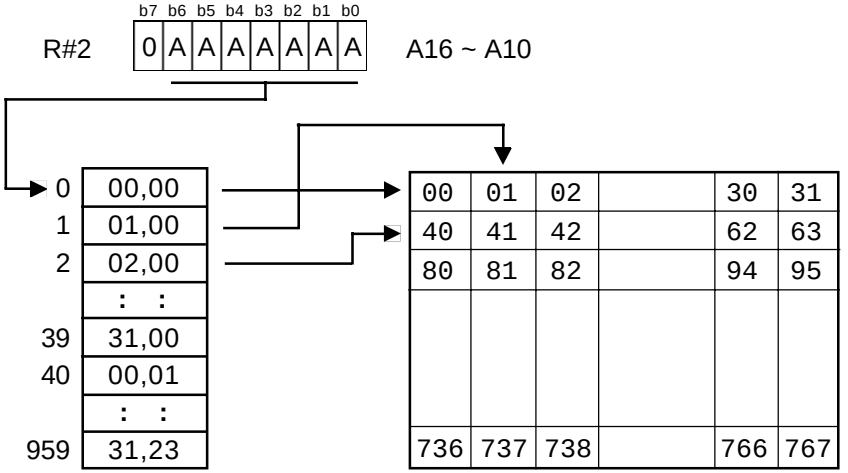
Tabela ABCD: coordenadas verticais em 0, 4, 8, 12, 16, 20
Tabela EFGH: coordenadas verticais em 1, 5, 9, 13, 17, 21
Tabela I J K L : coordenadas verticais em 2, 6, 10, 14, 18, 22
Tabela MNOP: coordenadas verticais em 3, 7, 11, 15, 19, 23

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

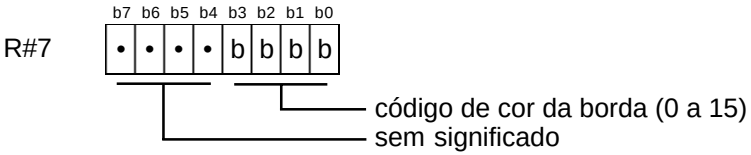
A tabela de nomes especifica as coordenadas de tela onde os caracteres (2 x 2 blocos) serão exibidos. Ela também é organizada de 4 em 4 de forma intercalada, conforme ilustrado abaixo.



O endereço inicial da tabela de nomes dos padrões é especificado em R#2. Apenas os 7 bits mais altos são especificados; por isso, a tabela de nomes sempre inicia em um múltiplo de 1 Kbyte a partir de 00000H.



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



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 ilustração da página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0
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 00000H.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#4	0	0	A	A	A	A	A	A	A16 ~ A11

endereço inicial da tabela geradora de padrões

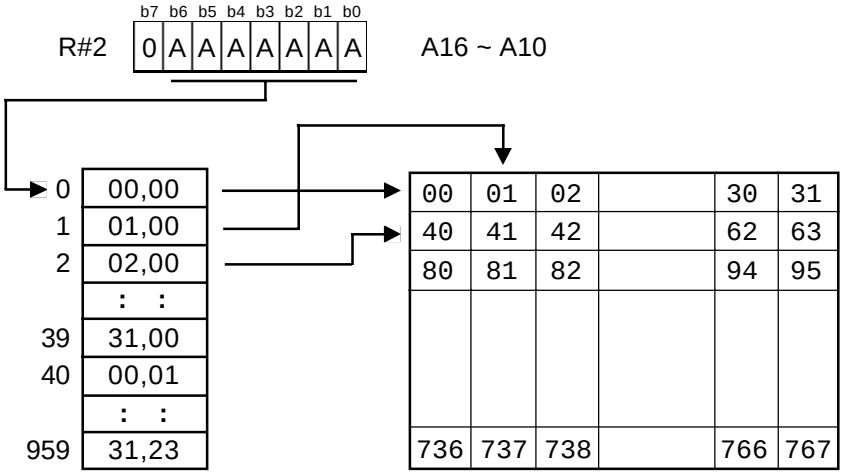
A tabela de cores especifica uma cor para cada 8 padrões consecutivos na tabela de padrões. O endereço inicial da tabela de cores é especificado em R#3 e R#10. Somente os 11 bits mais altos são especificados (A16 a A6); por isso essa tabela sempre inicia num múltiplo de 64 bytes a partir de 0000H.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#10	0	0	0	0	0	A	A	A	
R#3	A	A	A	A	A	A	A	A	A16 ~ A6

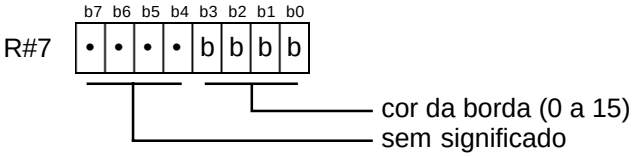
	b7	b6	b5	b4	b3	b2	b1	b0	
0	3	2	1	0	3	2	1	0	← padrões de 0 a 7
1	3	2	1	0	3	2	1	0	← padrões de 8 a 15
31	3	2	1	0	3	2	1	0	← padrões de 248 a 255

cor de frente do padrão
cor de fundo do padrão

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 na página seguinte.



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



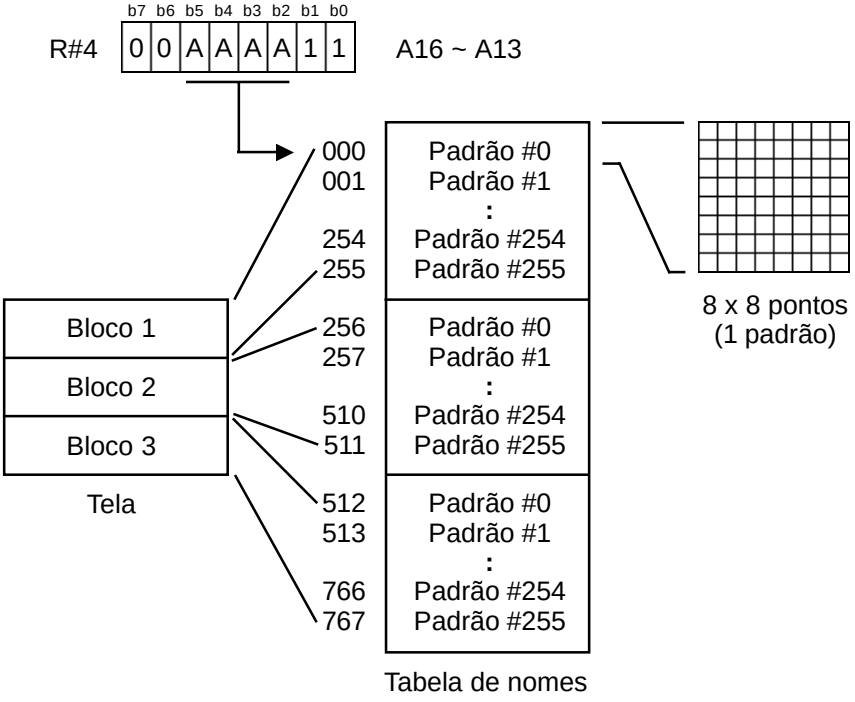
3.5 - MODOS GRÁFICOS 2 E 3

- 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 gráfico 2 e gráfico 3 são selecionados por R#0 e R#1 conforme ilustração da página seguinte. Eles são exatamente iguais, exceto que o modo gráfico 2 usa sprites modo 1 e o gráfico 3 usa sprites modo 2.

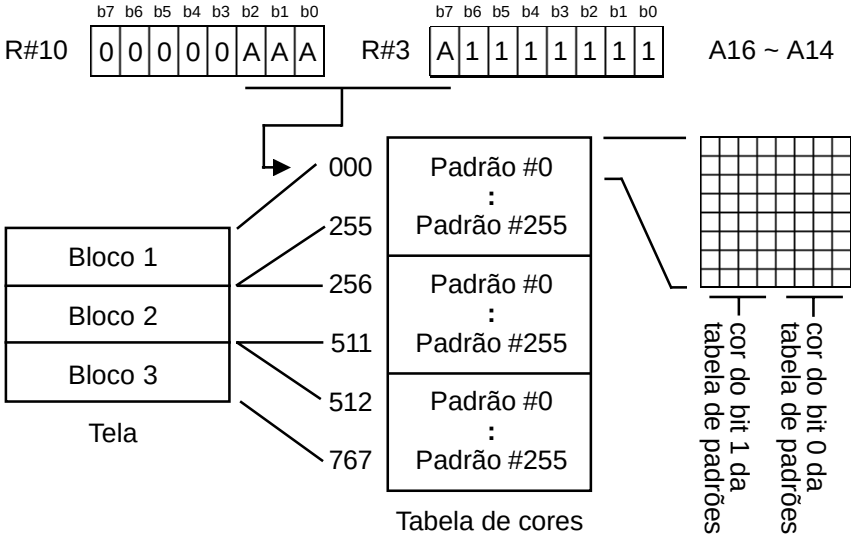
		b7	b6	b5	b4	b3	b2	b1	b0
Gráfico 2 -	R#0	•	•	•	•	0	0	1	•
	R#1	•	•	•	0	0	•	•	•
Gráfico 3 -	R#0	•	•	•	•	0	1	0	•
	R#1	•	•	•	0	0	•	•	•

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 endereço inicial da tabela geradora de padrões é especificado em R#4. Apenas os 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.

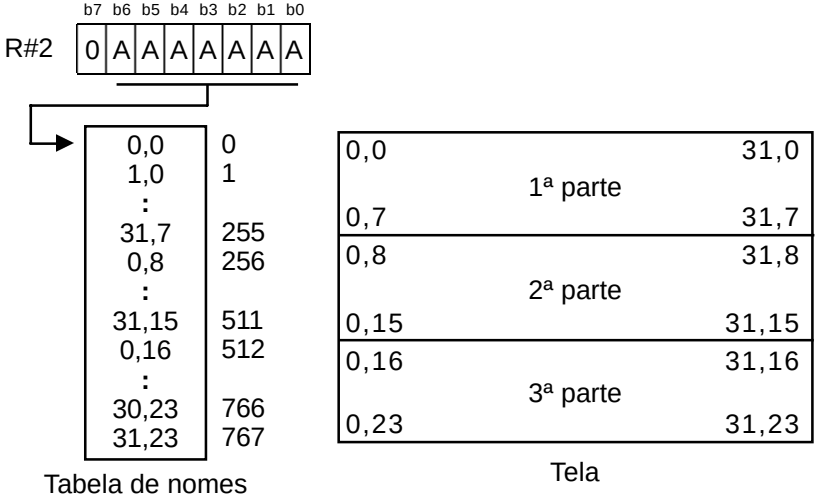


O tamanho da tabela de cores é o mesmo do da tabela geradora de padrões, e as cores podem ser especificadas para cada bit 0 ou 1 de

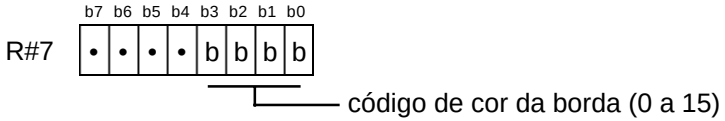
cada linha horizontal de cada padrão. O endereço inicial da tabela de cores é especificado em R#3 e R#10, mas apenas os quatro bits mais altos são especificados; por isso, a tabela de cores sempre inicia num múltiplo de 8 Kbytes a partir de 00000H.



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.



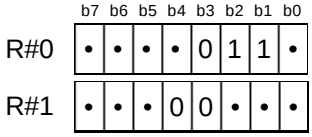
A cor da borda nos modos gráficos 2 e 3 é especificada nos 4 bits mais baixos de R#7.



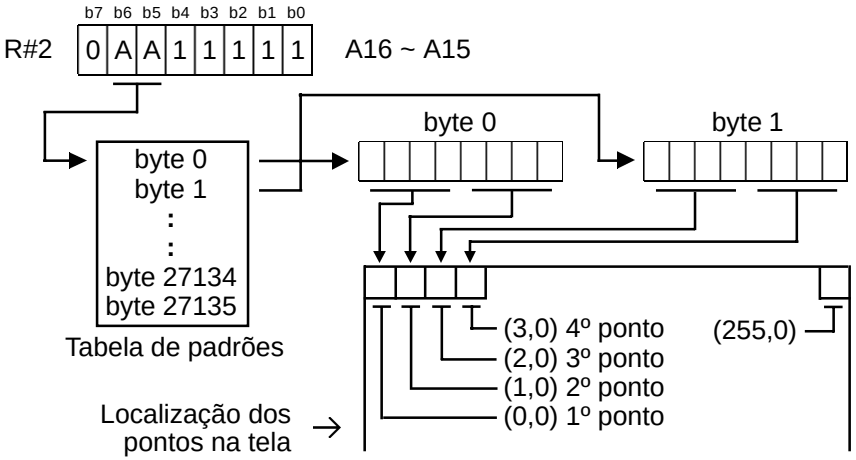
3.6 - MODO GRÁFICO 4

- 256 (norizontal) 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 abaixo:



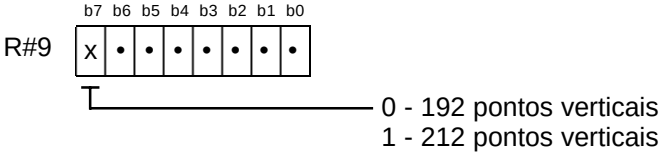
No modo gráfico 4, um byte na tabela geradora de padrões corresponde a dois pontos na tela. Cada ponto é representado por 4 bits; portanto até 16 cores podem ser especificadas para cada ponto. O endereço inicial da tabela geradora de padrões é especificado em R#2. Apenas os dois bits mais altos são especificados;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



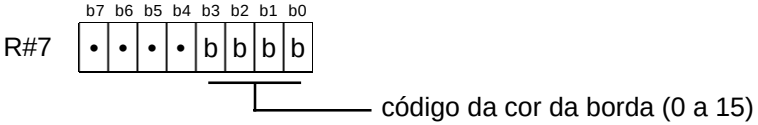
Nesse modo, cada byte corresponde a dois pontos horizontais em seqüência na tela e como a resolução horizontal é de 256 pontos, 128 bytes são necessários para cada linha de tela. Nesse modo não há necessidade da tabela de nomes. O endereço de cada ponto pode ser calculado pela seguinte expressão:

ENDEREÇO = X/2 + Y*128 + 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. O número de pontos verticais deve ser especificado em R#9, conforme ilustração abaixo.



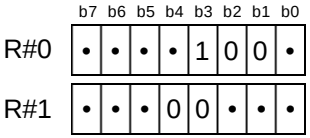
A cor da borda no modo gráfico 4 deve ser especificada nos 4 bits mais baixos de R#7, conforme ilustração abaixo.



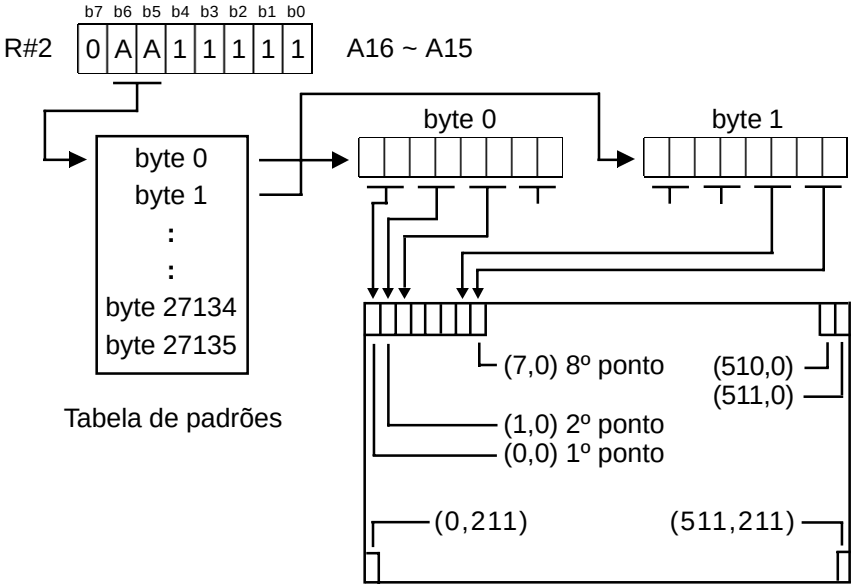
3.7 - MODO GRÁFICO 5

- 512 (norizontal) 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 podem 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 00000H.



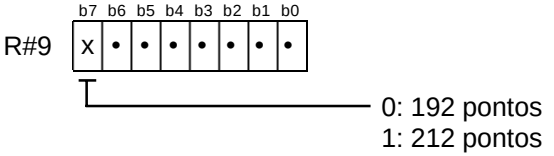
Cada byte representa quatro pontos em seqüência hozizontal na tela e como há 512 pontos de resolução horizontal, são necessários 128 bytes para representar cada linha da tela. O endereço na VRAM de cada ponto pode ser calculado pela seguinte expressão:

ENDEREÇO = X/4 + Y*128 + ENDEREÇO INICIAL

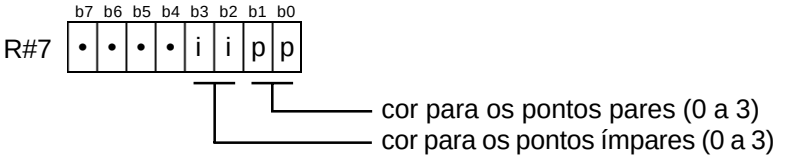
Onde X é a coordenada horizontal e Y a coordenada vertical. Como cada byte representa quatro pontos é necessária mais uma operação para determinar qual par de bits representa cada ponto.

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

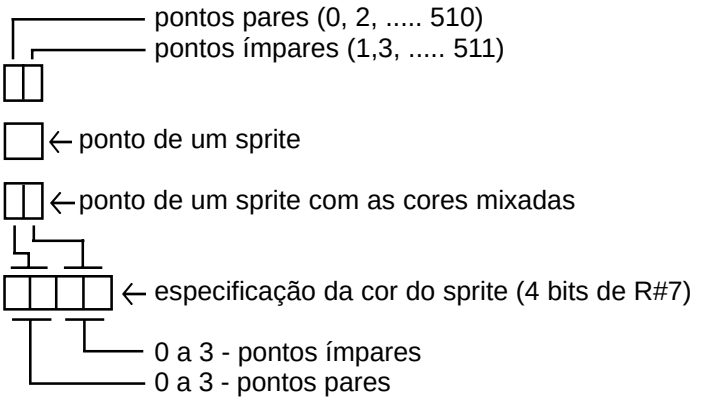
O número de pontos verticais deve ser especificado em R#9, conforme a ilustração da página seguinte.



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:



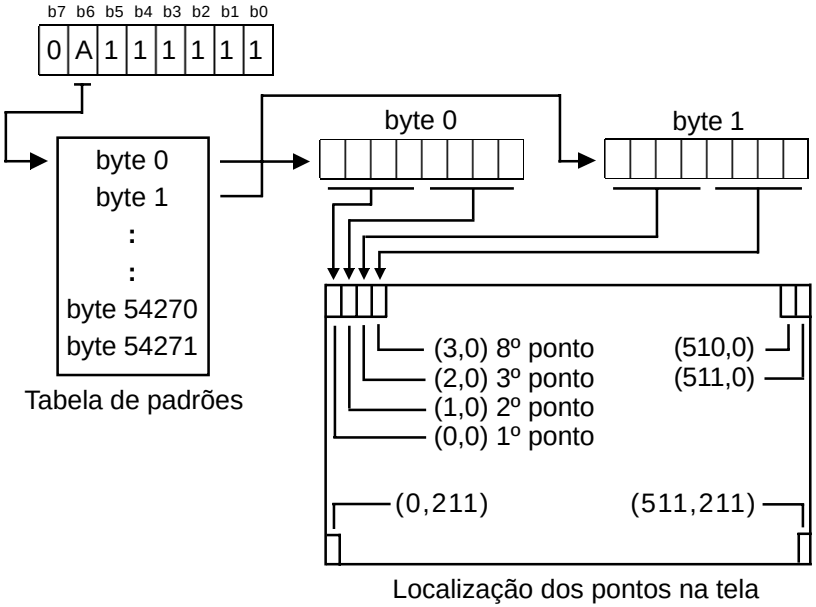
3.8 - MODO GRÁFICO 6

- 512 (norizontal) 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 da página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0
R#0	•	•	•	•	1	0	1	•
R#1	•	•	•	0	0	•	•	•

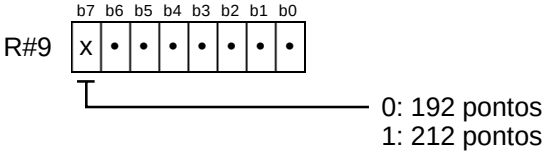
No modo gráfico 6, um byte na tabela de padrões corresponde a dois pontos na tela. Cada ponto é representado por 4 bits; portanto 16 cores podem ser especificadas. O endereço de início da tabela de padrões é especificado por um único bit em R#2; por isso, a tabela geradora de padrões sempre inicia em 00000H ou 10000H.



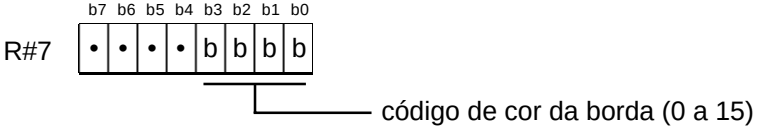
Como há 512 pontos em cada linha horizontal e cada byte representa dois pontos, são necessários 256 bytes para cada linha de tela. O endereço de cada ponto na tabela pode ser calculado pela seguinte expressão:

ENDEREÇO = X/2 + Y*256 + ENDEREÇO INICIAL

Onde X é a coordenada horizontal do ponto e Y é a coordenada 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. O número de pontos verticais deve ser especificado em R#9, conforme ilustração da página seguinte.



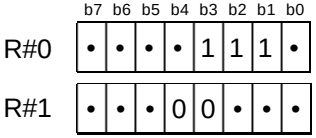
A cor da borda é especificada pelos 4 bits mais baixos de R#7.



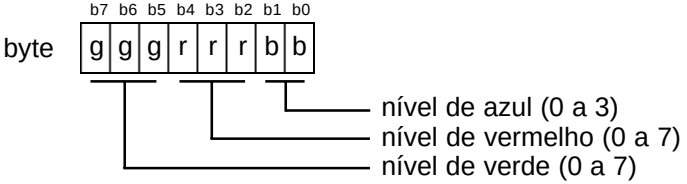
3.9 - MODO GRÁFICO 7

- 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:

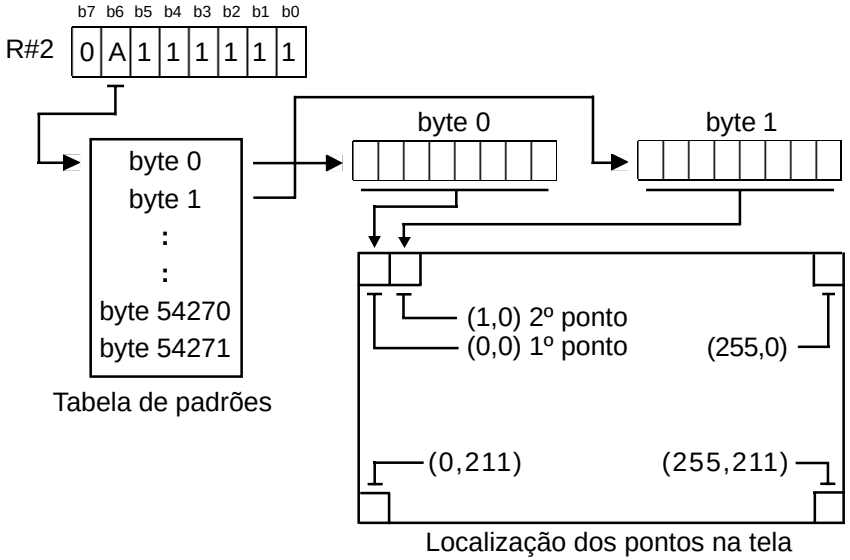


A configuração do modo gráfico 7 é a mais simples de todas; um ponto na tela corresponde a um byte na tabela de padrões, podendo apresentar até 256 cores simultâneas. Nesse modo não é usada a paleta de cores, sendo que cada byte de dados reserva 3 bits de intensidade para o verde, 3 bits para o vermelho e 2 bits para o azul.



O endereço inicial da tabela de padrões é especificado em R#2

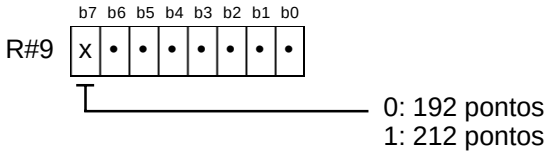
por um único bit; por isso, a tabela de padrões sempre inicia em 00000H ou 10000H.



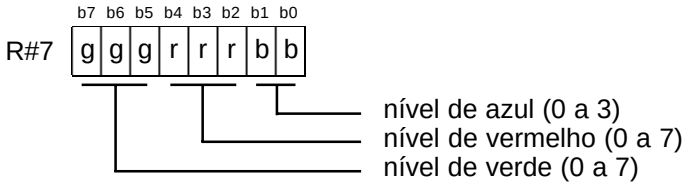
O endereço de cada ponto na tela pode ser calculado pela seguinte expressão:

ENDEREÇO = X + Y*256 + ENDEREÇO INICIAL

Onde X é a coordenada horizontal e Y a vertical. O número de pontos verticais deve ser especificado em R#9.



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.



3.10 - MODO GRÁFICO 8

- 256 (norizontal) por 212 (vertical) pontos;
- apresenta até 12499 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;
- mapeamente 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.

	b7	b6	b5	b4	b3	b2	b1	b0
R#0	•	•	•	•	1	1	1	•
R#1	•	•	•	0	0	•	•	•
R#25	•	•	•	1	1	•	•	•

O endereço de início da tabela geradora de padrões é especificado por um único bit de R#2; por isso ela pode iniciar em 00000H ou 10000H.

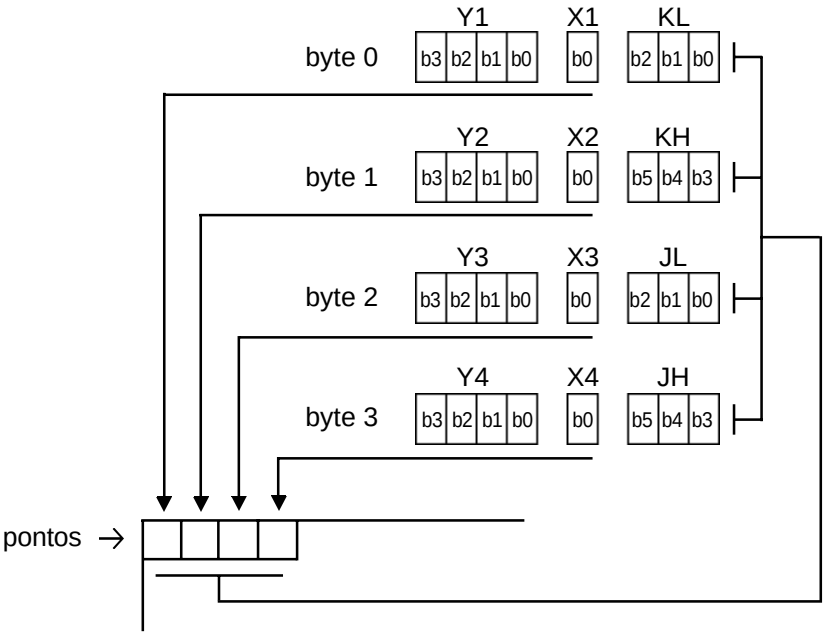
	b7	b6	b5	b4	b3	b2	b1	b0
R#2	0	A	1	1	1	1	1	1

0: inicia em 00000H
1: inicia em 10000H

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.

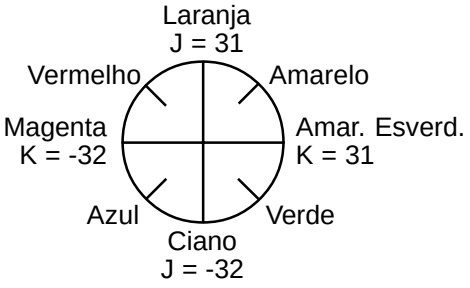
Nesse modo, os pontos estão organizados de quatro em quatro na horizontal. Cada grupo de 4 pontos pode ter uma única cor, escolhidas de 4096, com até 16 níveis de saturação para cada ponto individual, desde o branco até a cor saturada. Ou então cada ponto pode ter até 16 cores escolhidas de uma paleta 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 na página seguinte.



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. Por isso, nesse caso, os pontos são totalmente independentes.

Quando os bits Xn 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 representado como as cores são representadas por esses vetores.



J=100000 = -32 K=011111 = 31	Verde
J=011111 = 31 K=100000 = -32	Vermelho
J=100000 = -32 K=100000 = -32	Azul
J=011111 = 31 K=011111 = 31	Amarelo

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 saturação de 16 níveis, representada pelos bits Yn, desde o branco até a cor saturada. Se seu valor for 1111B, o ponto será branco. Se for 0000B, o ponto terá a cor saturada.

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 primárias do sistema YJK: verde, vermelho, azul e amarelo. O uso de quatro 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 YJK para o RGB e vice-versa pode ser feita através das 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 - 1/2 J - 1/4 K

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 = 65536 cores possíveis. Acontece que nesse modo as cores não são totalmente independentes (além de características técnicas do V9958 que não vêm ao caso), 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:

ENDEREÇO = X + Y*256 + ENDEREÇO INICIAL

Onde X é a coordenada horizontal e Y a vertical.

O número de pontos verticais deve ser especificado em R#9, da forma já descrita. A cor da borda deve ser especificada em R#7, obedecendo a paleta de cores, como no modo gráfico 4.

A diferença entre a Screen 10 e a Screen 11 está no tratamento dado a elas pelo BIOS e 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). Essa flag está descrita na seção “VARIÁVEIS DE SISTEMA DOS MODOS DE TELA”, mais adiante.

3.11 - MODO GRÁFICO 9

- 256 (norizontal) por 212 (vertical) pontos;
- apresenta até 19268 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;
- mapeamente gráfico YJK;
- compatível com Screens 12.

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

	b7	b6	b5	b4	b3	b2	b1	b0
R#0	•	•	•	•	1	1	1	•
R#1	•	•	•	0	0	•	•	•
R#25	•	•	•	0	1	•	•	•

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 00000H ou 10000H.

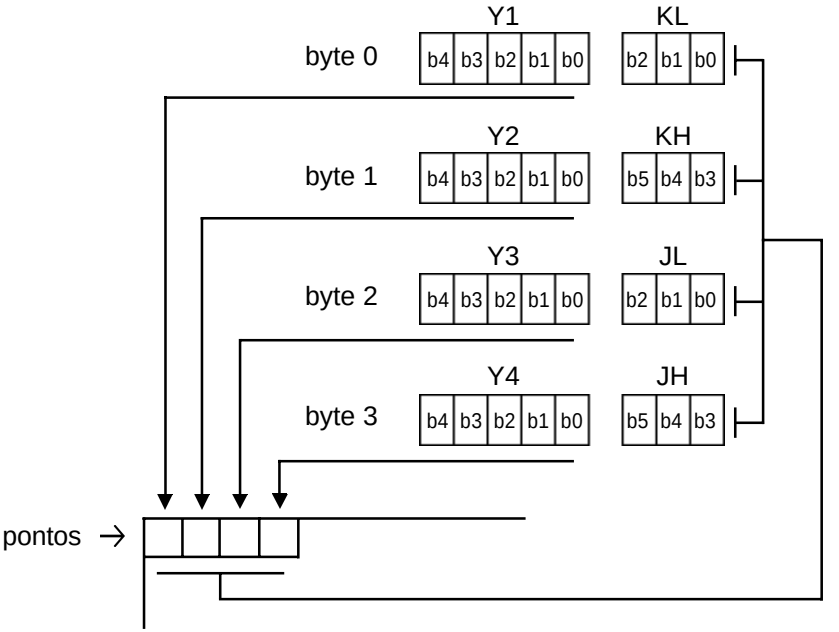
	b7	b6	b5	b4	b3	b2	b1	b0
R#2	0	A	1	1	1	1	1	1

0: endereço 00000H
1: endereço 10000H

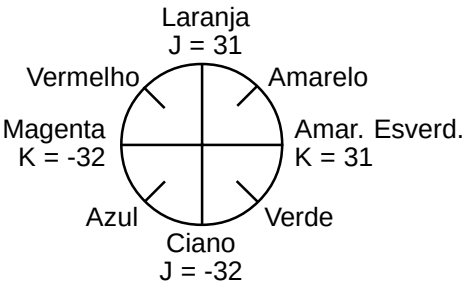
No modo gráfico 9 é usado o sistema YJK puro. Os pontos estão organizados de quatro em quatro na horizontal, sendo que cada grupo de 4 pontos pode ter uma única cor, escolhida de 4096, com até 32 níveis de saturação para cada ponto individual, do branco até a cor saturada.

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 de saturação, pode variar de 11111B (branco) até 00000B (cor saturada). Como há 4096 cores e 32 níveis de saturação para cada ponto, na verdade existem 131072 cores possíveis (32 * 4096). Entretanto, por motivos já explicados no modo gráfico 8, há uma redução do número de cores que podem ser apresentadas simultaneamente para 19268.

A organização do modo gráfico 8 está ilustrada abaixo.



As cores são excolhidas pelos vetores J e K, conforme ilustrado abaixo.



J=100000 = -32 K=011111 = 31	Verde
J=011111 = 31 K=100000 = -32	Vermelho
J=100000 = -32 K=100000 = -32	Azul
J=011111 = 31 K=011111 = 31	Amarelo

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

ENDEREÇO = X + Y*256 + ENDEREÇO INICIAL

Onde X é a coordenada horizontal e Y a vertical.

O número de pontos verticais deve ser especificado em R#9, con-

forme já descrito. A cor da borda deve ser especificada em R#7, obedecendo à paleta de cores, tal qual no modo gráfico 4.

3.12 - VARIÁVEIS DE SISTEMA DOS MODOS DE TELA

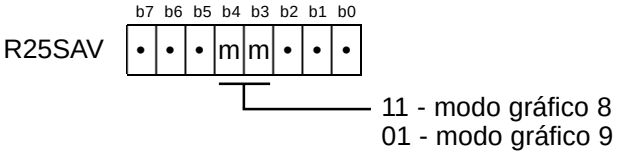
As seguintes variáveis de sistema são usadas pelo BIOS e pelo BASIC para diferenciar os modos de tela:

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

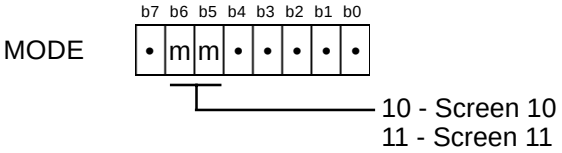
Nos modos multicor e gráficos 1 a 8, não há segredo; o valor da Screen em uso é salvo na variável SCRMOD com o mesmo valor usado pelo interpretador na instrução SCREEN.

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, é feito um malabarismo: a variável SCRMOD sempre conterá o valor 8 (como na Screen 8). Para diferenciar um do outro, é lido o valor da variável R25SAV, conforme abaixo:



Para diferenciar a Screen 10 da Screen 11, é feita mais uma operação; essa diferença está armazenada na variável MODE.



4 - SPRITES

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.

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.

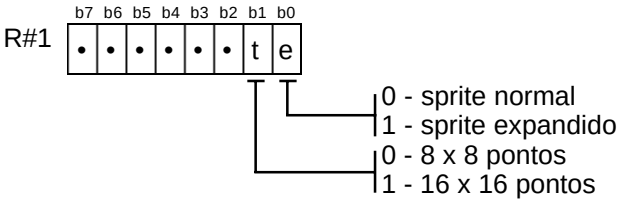
4.1 - SPRITES MODO 1

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

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

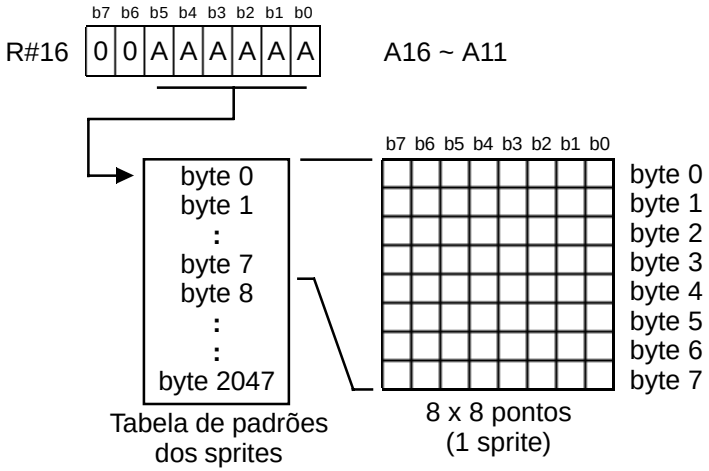
Os sprites também podem ser expandidos para o dobro do tamanho, na vertical e na horizontal. Nesse caso, um ponto do sprite corresponde a quatro pontos na tela (exceto nos modos gráficos 5 e 6, onde corresponderá e 8 pontos). Essa função é controlada pelo bit 0 de R#1.



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 seqüência ilustrada abaixo:

01	03
02	04

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 00000H. 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 00000H. 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 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.

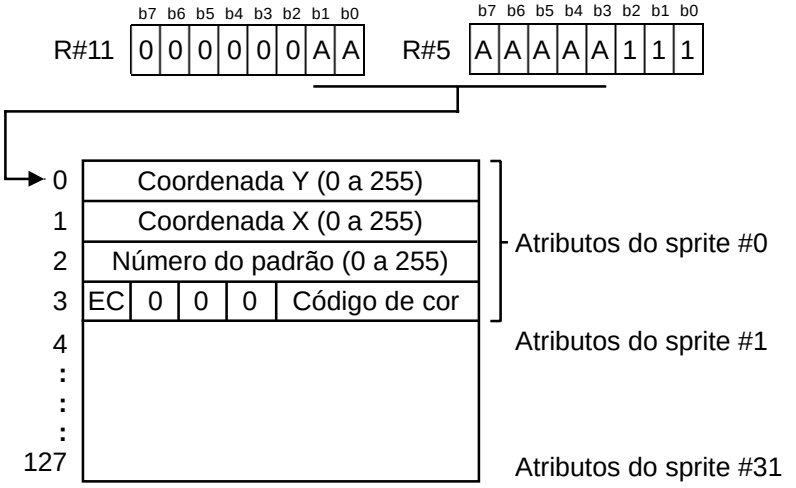
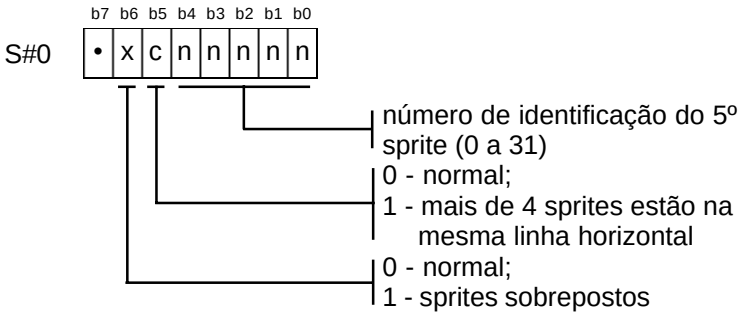


Tabela de atributos dos sprites

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.

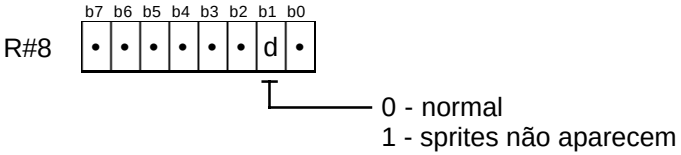


4.2 - SPRITES MODO 2

Os sprites modo 2 foram adicionados ao VDP V9938 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.
Coodenada 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:

Código de cor: Cor da linha respectiva, podendo variar de 0 a 15 obedecendo a 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.

IC: Quando esse bit for 1, a linha respectiva não causará conflito quando ocorrer sobreposição com outros sprites.

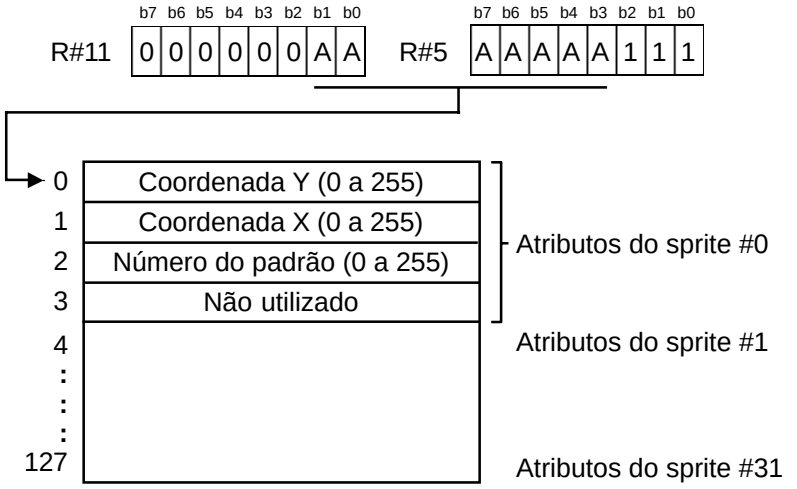


Tabela de atributos dos sprites

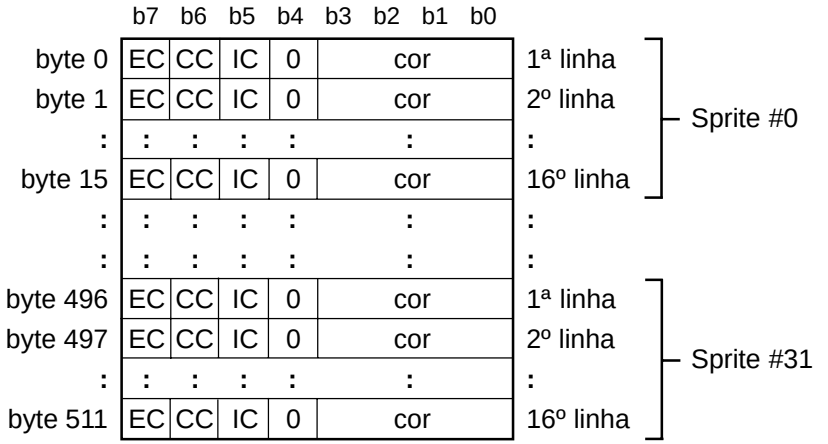
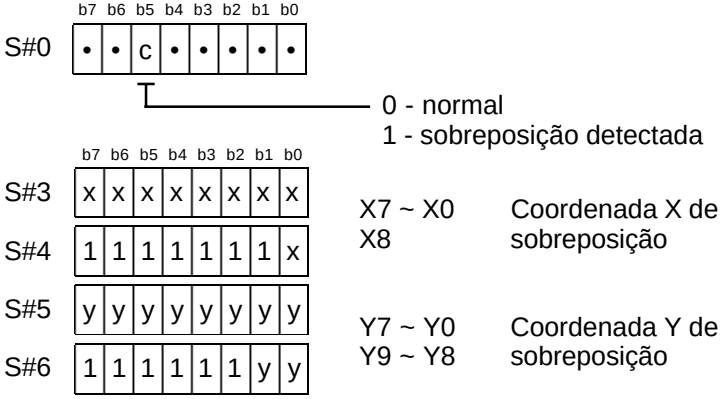


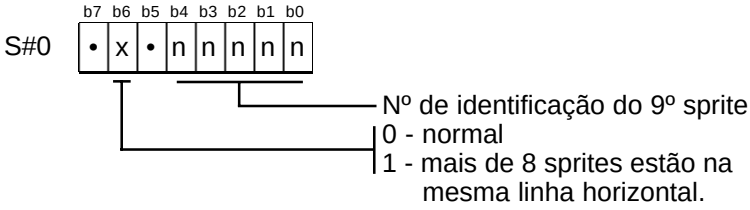
Tabela de cores dos sprites

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 atual ocorreu. Para obter as coordenadas exatas, a seguinte expressão deve ser usada:

Coordenada X = (S#3 e S#4) - 12
Coordenada Y = (S#5 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.



5 - COMANDOS DO VDP

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 na página seguinte.

GRÁFICO 4 (SCREEN5) **ENDEREÇO** GRÁFICO 5 (SCREEN6)

(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)		0FFFFH	(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)		1FFFFH	(0,1023) (511,1023)	

GRÁFICOS 7~9 (SCREEN 8~12) GRÁFICO 6 (SCREEN 7)

(0,0) (255,0)	00000H	(0,0) (511,0)	
Página 0		Página 0	
(0,255) (255,255)	0FFFFH	(0,255) (511,255)	
(0,256) (255,256)		(0,256) (511,256)	
Página 1		Página 1	
(0,511) (255,511)	1FFFFH	(0,511) (511,511)	

5.1 - DESCRIÇÃO DOS COMANDOS DO VDP

Existem 12 comandos disponíveis no MSX-VIDEO:

NOME COMANDO	ORIGEM	DESTINO	UNIDADE	MEMÔNICO	R#46-4msb
MOVIMENTOS RÁPIDOS	CPU	VRAM	bytes	HMMC	1 1 1 1
	VRAM	VRAM	bytes	YMMM	1 1 1 0
	VRAM	VRAM	bytes	HMMM	1 1 0 1
	VDP	VRAM	bytes	HMMV	1 1 0 0
MOVIMENTOS LÓGICOS	CPU	VRAM	pontos	LMMC	1 0 1 1
	VRAM	CPU	pontos	LMCM	1 0 1 0
	VRAM	VRAM	pontos	LMMM	1 0 0 1
	VDP	VRAM	pontos	LMMV	1 0 0 0
LINHA	VDP	VRAM	pontos	LINE	0 1 1 1
PROCURA	VDP	VRAM	pontos	SRCH	0 1 1 0
PSET	VDP	VRAM	pontos	PSET	0 1 0 1
POINT	VRAM	VDP	pontos	POINT	0 1 0 0

Os valores 0011B, 0010B e 0001B são reservados e o valor 0000B significa que o VDP não está executando comando algum.

Quando um dado é escrito em R#46 (registrador de comando), o VDP começa a executar o comando e seta o bit 0 (CE / command execute) do registrador de status S#2. Os parâmetros necessários devem 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 comando 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.

5.2 - OPERAÇÕES LÓGICAS

Quando os comandos são executados, várias operações lógicas 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	0 0 0 0
AND	DC = SC * DC	0 0 0 1
OR	DC = $\overline{SC}$ + DC	0 0 1 0
XOR	DC = $\overline{SC}$ * DC + SC * $\overline{DC}$	0 0 1 1
NOT	DC = $\overline{SC}$	0 1 0 0
TIMP	Se SC=0, DC=DC senão DC = SC	1 0 0 0
TAND	Se SC=0, DC=DC senão DC = SC * DC	1 0 0 1
TOR	Se SC=0, DC=DC senão DC = $\overline{SC}$ + DC	1 0 1 0
TXOR	Se SC=0, DC=DC senão DC = $\overline{SC}$*DC + SC*$\overline{DC}$	1 0 1 1
TNOT	Se SC=0, DC=DC senão DC = $\overline{SC}$	1 1 0 0

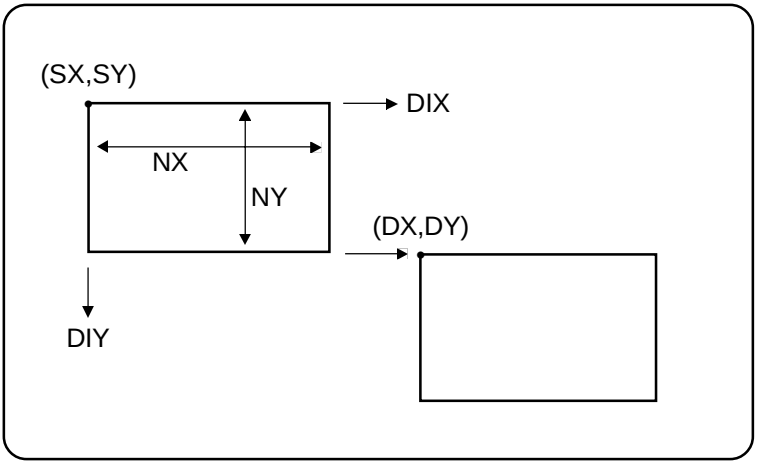
Na tabela, SC representa o código de cor da origem e DC o código de cor do destino. IMP, AND, OR, XOR e NOT são as operações lógicas possíveis. Nas operações com nomes precedidos por “T”, os pontos de origem que tiverem a cor 0 não serão objeto de operação lógica no destino. Usando esse recurso, somente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

5.3 - ESPECIFICAÇÃO DE ÁREAS

Os comandos de movimentação de áreas transferem os dados dentro de uma área especificada por um retângulo. A área a ser tranferida 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 se-

rão 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

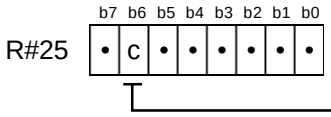
5.4 - USANDO OS COMANDOS DO VDP

Os comandos do VDP são classificados em três tipos: comandos de tranferência 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) e também deve ser aguardado que o VDP termine de executar o comando para enviar o próximo, o que deve ser feito lendo o bit CE do registrador de status 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
```

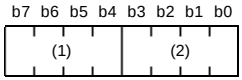
No caso de usar comandos no V9958, há uma peculiaridade. Deve ser observado o bit 6 de R#25 (CMD). Se esse bit for 0, os comandos somente funcionarão nos modos gráficos 4 a 7. É o valor default. Para que os comandos funcionem nos modos gráficos 8 e 9 é necessário setar esse bit em 1. Nesse caso, os comandos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



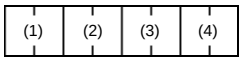
0 - comandos funcionam somente nos modos gráficos 4 a 7.
1 - comandos funcionam em todos os modos gráficos; nos modos 8 e 9 funcionam da mesma forma que no modo 7.

5.4.1 - HMMC (Tranferência rápida - CPU → VRAM)

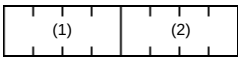
Nesse comando, os dados são tranferidos da CPU para uma área especificada na VRAM. Operações lógicas não são possíveis; os dados são transferidos em bytes em alta velocidade. O bit mais baixo da coordenada X não é referenciado para nos modos gráficos 4 e 6. Os dois bits mais baixos não são referenciados para o modo gráfico 5.



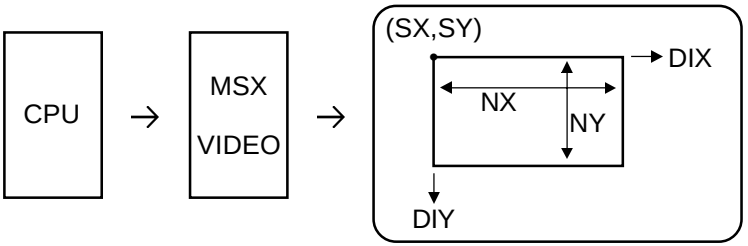
Como um byte da VRAM representa 2 pontos, o bit mais baixo da coordenada X não é representado (1).



Como um byte da VRAM representa 4 pontos, os dois bits mais baixos da coordenada X não são representados (1) e (2).



Como um byte da VRAM representa 2 pontos, o bit mais baixo da coordenada X não é representado (1).



Os parâmetros mostrados na tabela abaixo seguinte devem ser colocados nos registradores apropriados. Nesse ponto, deve ser escrito, em R#44, o primeiro byte de dados a ser tranferido 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 tranferidos.

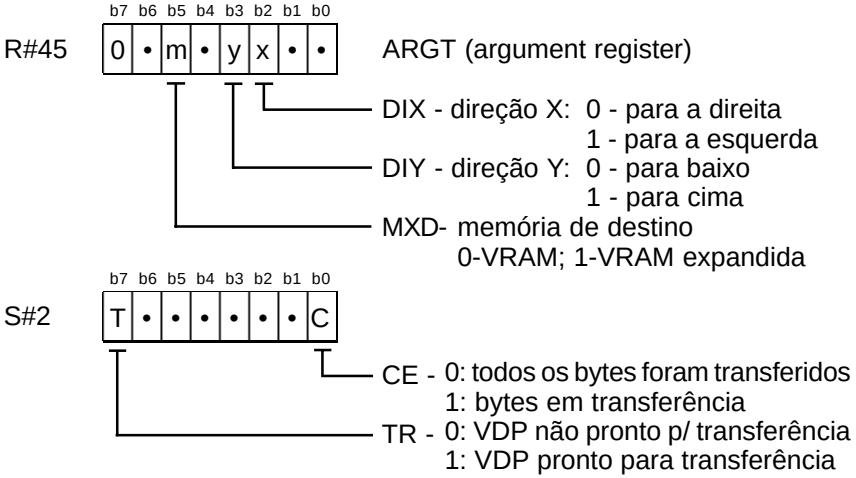
	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal da tela para onde os bytes de dados serão transferidos (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical da tela para onde os bytes de dados serão transferidos (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Número de pontos a tranferir na direção horizontal (0 a 511).
R#41	0	0	0	0	0	0	0	x	X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Número de pontos a transferir na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		

R#44

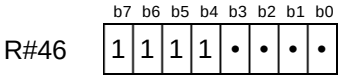
x=2n	x=2n+1		
x=4n	x=4n+1	x=4n+2	x=4n+3
1 byte por ponto			

CLR (Gráficos 4, 6) (n = 0 a 127)
CLR (Gráfico 5) (n = 0 a 127)
CLR (Gráficos 7, 8, 9)

formato dos bytes de dados a serem transferidos

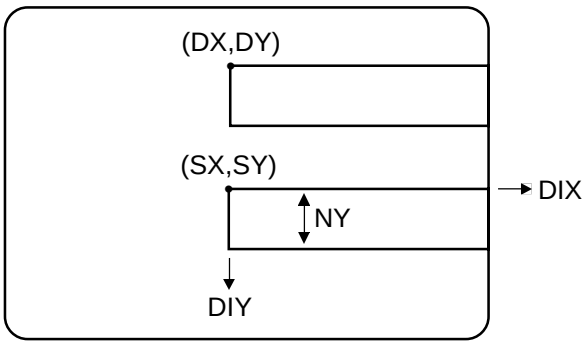


Para executar o comando HMMC, basta escrever o valor F0H em R#46:



5.4.2 - YMMM (Transferência rápida - VRAM na 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 tabela da página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0			
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical de origem (0 a 1023).
R#35	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de origem e de destino[11] (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a transferir na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		

	b7	b6	b5	b4	b3	b2	b1	b0	
R#45	0	•	m	•	y	x	•	•	ARGT (argument register)

DIX - direção X: 0 - à direita
1 - à esquerda

DIY - direção Y: 0 - para baixo
1 - para cima

MXD - memória de destino
0-VRAM; 1-VRAM expandida

	b7	b6	b5	b4	b3	b2	b1	b0
S#2	T	•	•	•	•	•	•	C

CE - 0: todos os bytes foram transferidos
1: bytes em transferência

TR - 0: VDP não pronto p/ transferência
1: VDP pronto para transferência

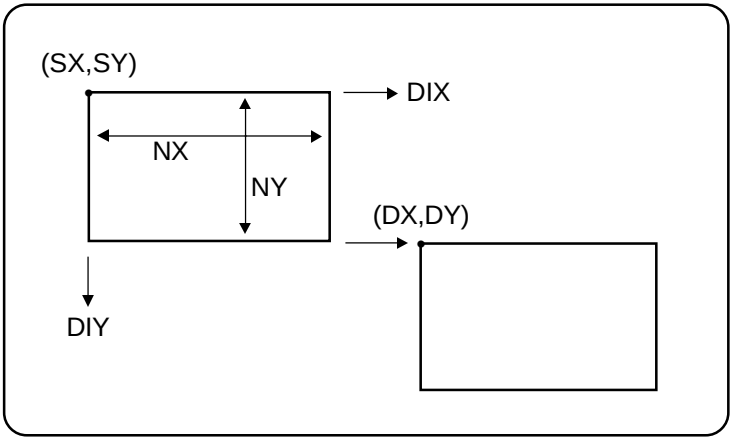
Para executar o comando HMMC, basta escrever o valor E0H em R#46:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#46	1	1	1	0	•	•	•	•	CMR

Nota 11: para os modos gráficos 4 a 6, o bit mais baixo em DX é ignorado e para o modo gráfico 5, são ignorados os dois bits mais baixos.

5.4.3 - HMMM (Transferência rápida - VRAM → VRAM)

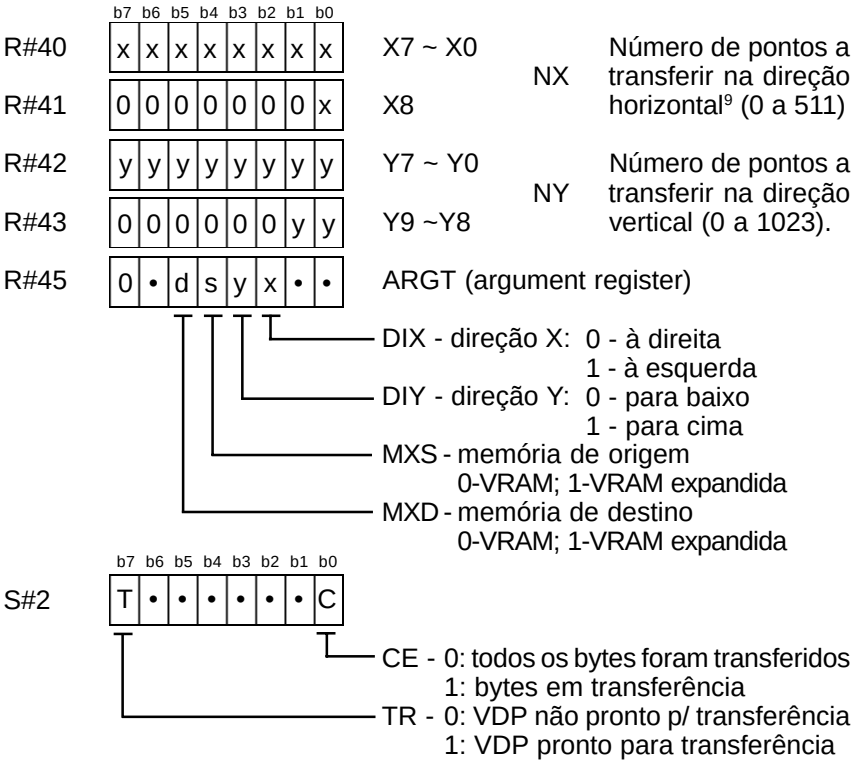
Nesse comando, os dados são transferidos de uma área da VRAM para outra. Os dados são transferidos em áreas retangulares, conforme ilustração abaixo:



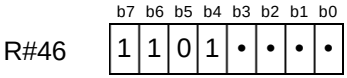
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal de origem[12] (0 a 511).
R#33	0	0	0	0	0	0	0	x	X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical de origem (0 a 1023).
R#35	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino[12] (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~Y8		

Nota 12: para os modos gráficos 4 e 6, o bit mais baixo em DX, SX e NX é ignorado e para o modo gráfico 5 são ignorados os dois bits mais baixos.

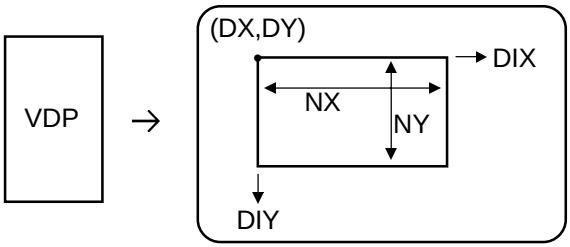


Para executar o comando HMMM, basta escrever o valor D0H em R#46:

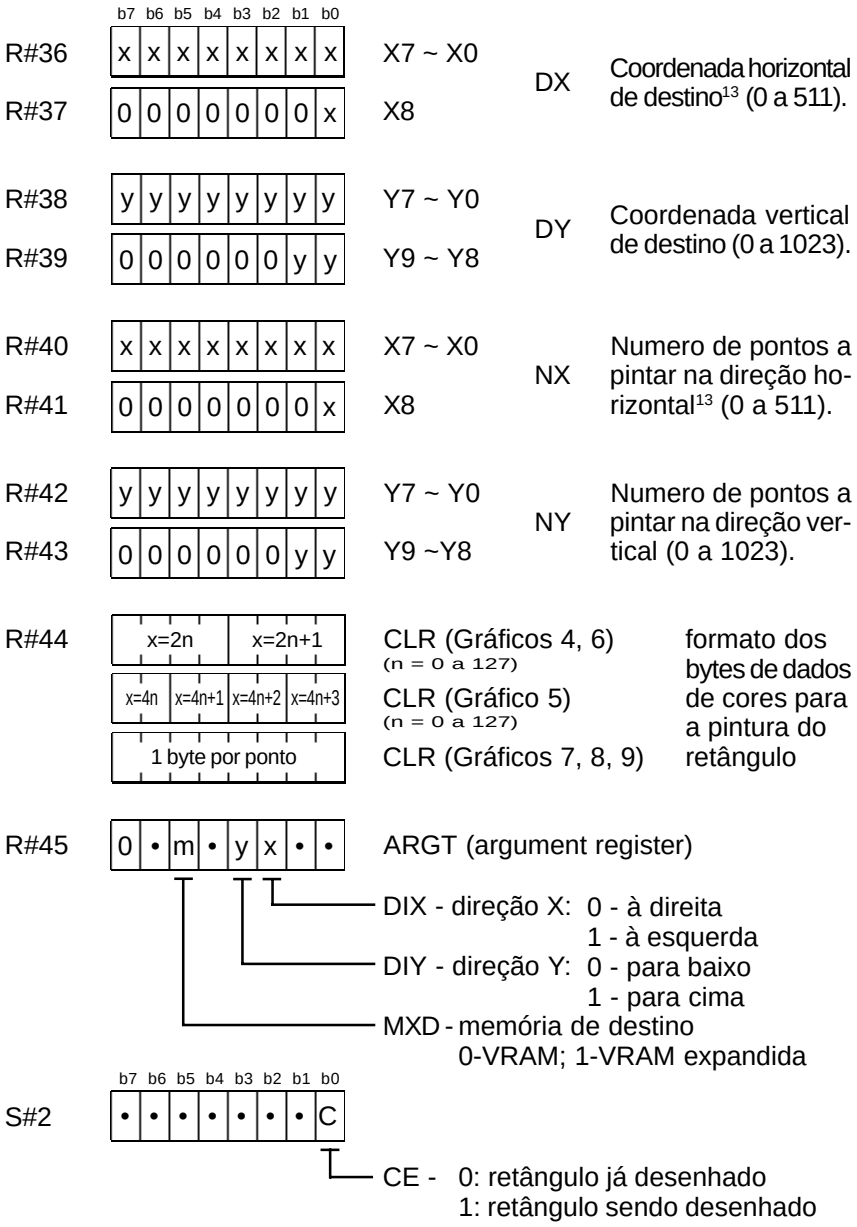


5.4.4 - HMMV (Desenha retângulo am alta velocidade)

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:



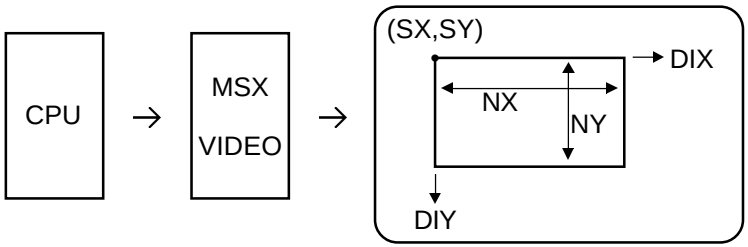
Nota 13: para os modos gráficos 4 e 6, o bit mais baixo em DX e NX é ignorado e para o modo gráfico 5 são ignorados os dois bits mais baixos.

Para executar o comando HMMC, basta escrever o valor C0H em R#46:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#46	1	1	0	0	•	•	•	•	CMR

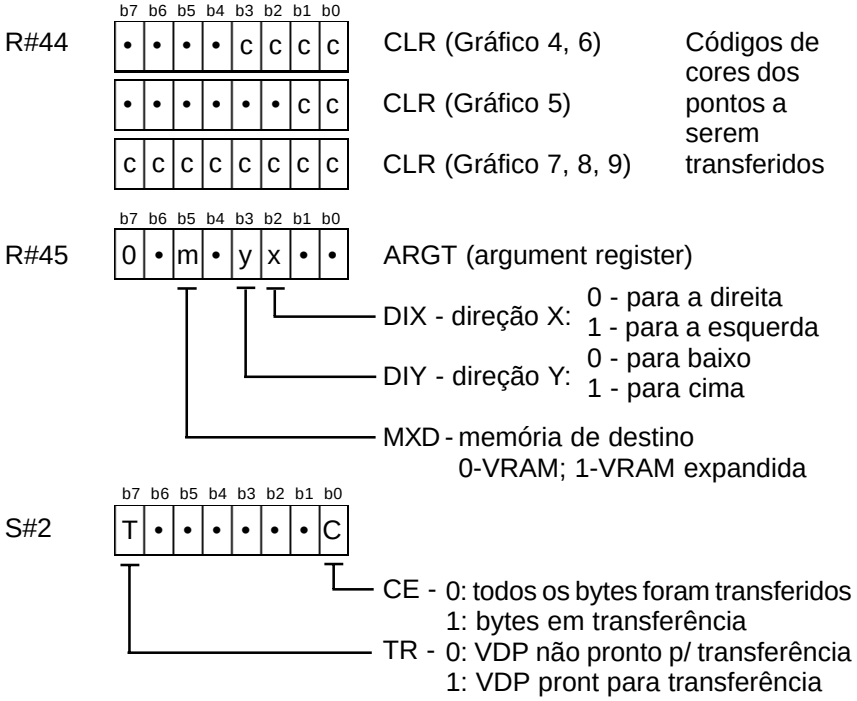
5.4.5 - LMMC (Transferência lógica - CPU → VRAM)

Nesse comando, os bytes de dados são transferidos da CPU para uma área específica da VRAM em pontos. Operações lógicas durante a tranferência são possíveis. Nos comandos de tranferência lógica, como o LMMC, os dados são tranferidos em pontos e um byte é requerido para cada ponto em todos os modos de tela. O código da operação lógica deve ser especifidado nos 4 bits mais baixos do registrador R#46. Os dados são transferidos tendo como referência os bits TR e CE de S#2, como nos comandos de transferência rápida.

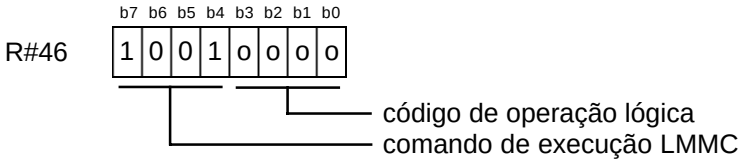


Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino[10] (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Numero de pontos a transferir na direção horizontal[10] (0 a 511).
R#41	0	0	0	0	0	0	0	x	X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Numero de pontos a transferir na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		



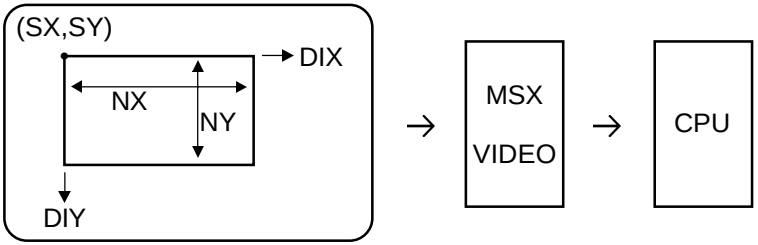
Para executar o comando LMMC, deve ser escrito o valor 1001B nos quatro bits mais altos de R#46 e o código de operação lógica nos quatro bits mais baixos.



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:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal de origem (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical de origem (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Numero de pontos a transferir na direção horizontal (0 a 511).
R#41	0	0	0	0	0	0	0	x	X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Numero de pontos a transferir na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		

	b7	b6	b5	b4	b3	b2	b1	b0	
R#45	0	•	•	m	y	x	•	•	ARGT (argument register)

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
0-VRAM; 1-VRAM expandida

	b7	b6	b5	b4	b3	b2	b1	b0
S#2	T	•	•	•	•	•	•	C

CE - 0: todos os bytes foram transferidos
1: bytes em transferência
TR - 0: VDP não pronto p/ transferência
1: VDP pronto para transferência

Antes de executar o comando LMCM, é aconselhável ler o registrador S#7 para garantir que o bit TR esteja resetado. Depois, basta escrever o valor A0H em R#46 para executar o comando.

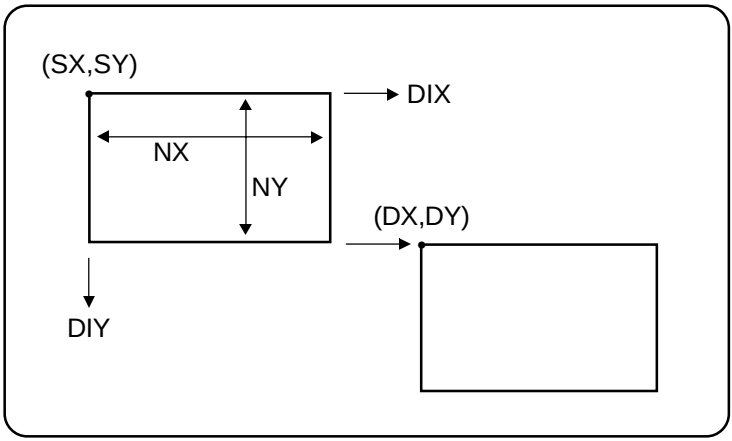
	b7	b6	b5	b4	b3	b2	b1	b0	
R#46	1	0	1	0	•	•	•	•	CMR

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 será 0.

	b7	b6	b5	b4	b3	b2	b1	b0		
S#7	•	•	•	•	c	c	c	c	CLR (Gráfico 4, 6)	Códigos de cores dos pontos lidos
	•	•	•	•	•	•	c	c	CLR (Gráfico 5)	
	c	c	c	c	c	c	c	c	CLR (Gráfico 7, 8, 9)	

5.4.7 - LMMM (Transferência 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 registradores descritos na página seguinte deverão ser carregados antes da execução do comando.

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal de origem (0 a 511).
R#33	0	0	0	0	0	0	0	x	X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical de origem (0 a 1023).
R#35	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos a transferir na direção horizontal (0 a 511)
R#41	0	0	0	0	0	0	0	x	X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a transferir na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		

R#45	0	•	d	s	y	x	•	•	ARGT (argument register)

DIX - direção X: 0 - para a direita
1 - para a esquerda

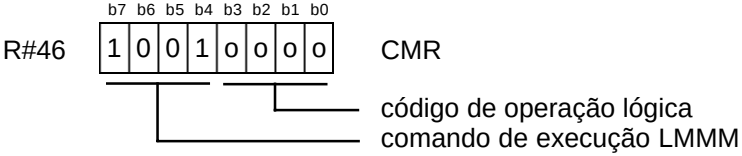
DIY - direção Y: 0 - para baixo
1 - para cima

MXS - memória de origem
0-VRAM; 1-VRAM expandida

MXD - memória de destino
0-VRAM; 1-VRAM expandida

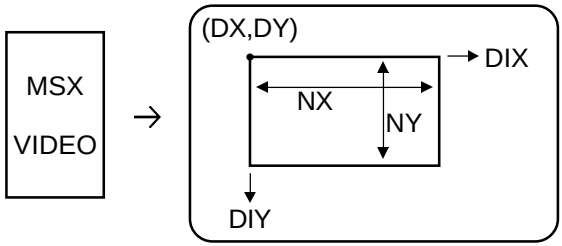
	b7	b6	b5	b4	b3	b2	b1	b0	
S#2	•	•	•	•	•	•	•	C	CE - 0: transferência da área completada 1: transferência em progresso

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.



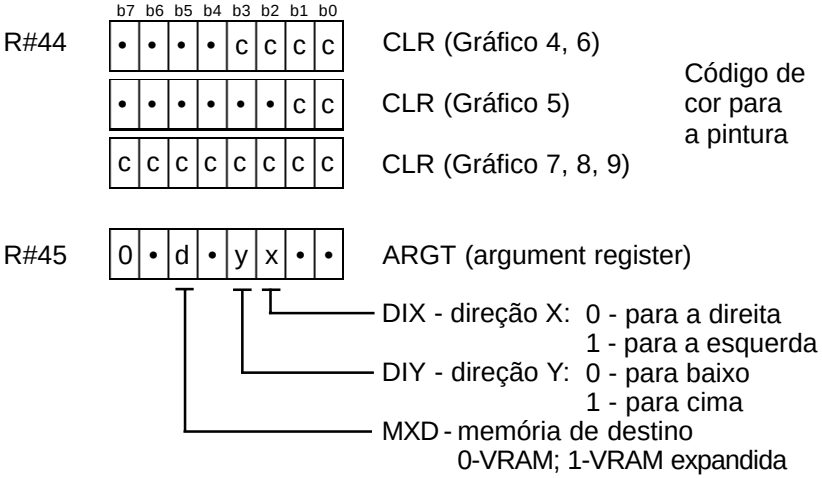
5.4.8 - LMMV (Pintura lógica da VRAM)

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.

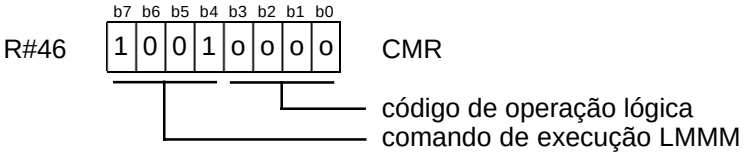


Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de início da pintura (0 a 511).
R#37	0	0	0	0	0	0	0	x	X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de início da pintura (0 a 1023).
R#39	0	0	0	0	0	0	y	y	Y9 ~ Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Numero de pontos a pintar na direção horizontal (0 a 511).
R#41	0	0	0	0	0	0	0	x	X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Numero de pontos a pintar na direção vertical (0 a 1023).
R#43	0	0	0	0	0	0	y	y	Y9 ~Y8		

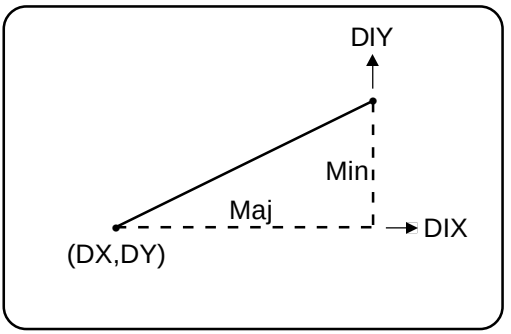


Para executar o comando LMMV deve-se escrever o valor 1000B nos quatro bits mais altos de R#46 sendo que os quatro bits mais baixos devem conter o código de operação lógica.

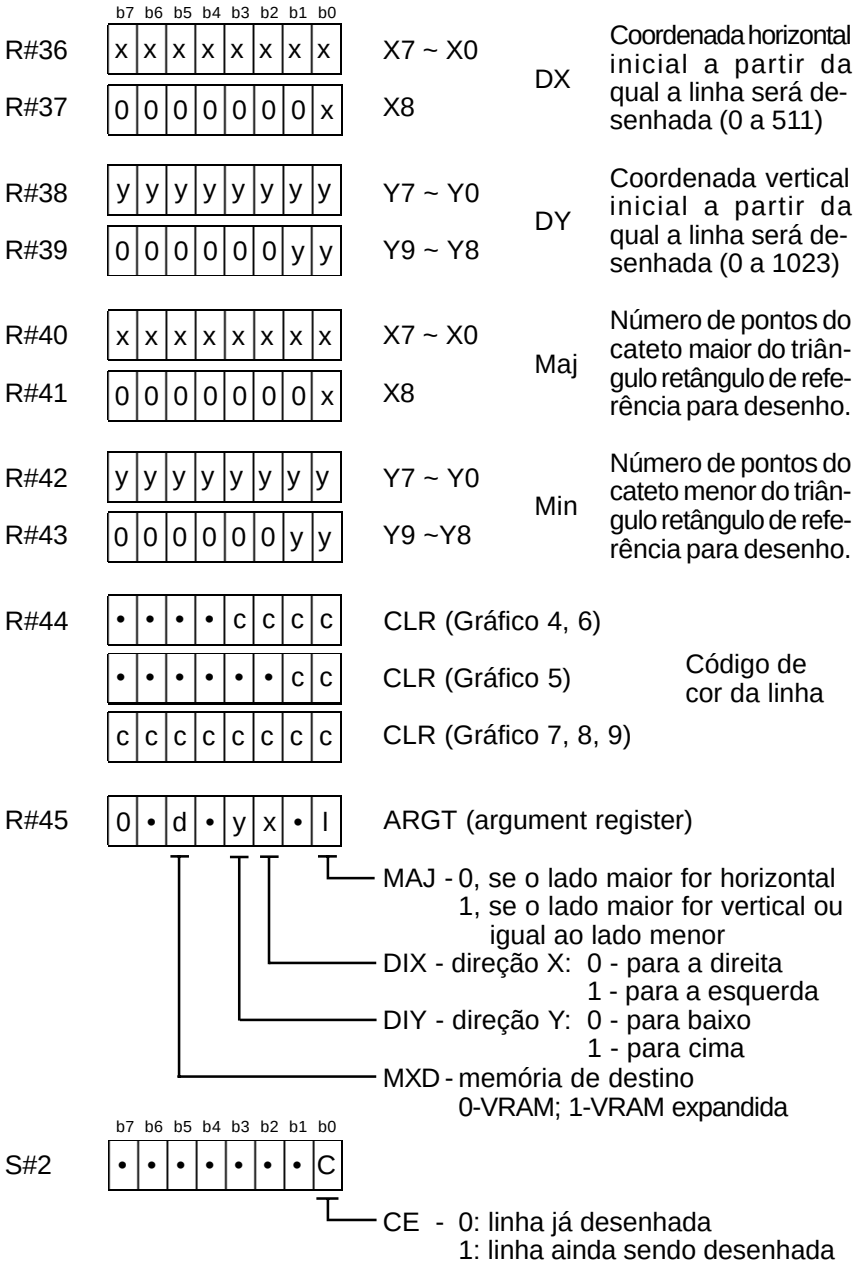


5.4.9 - LINE (Desenha uma linha)

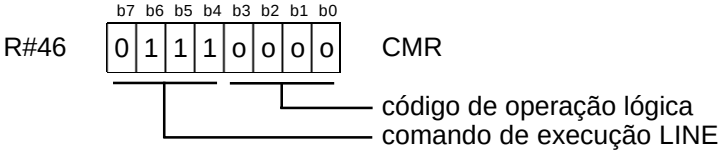
Esse comando desenha uma linha entre coordenadas da tela. Os parâmetros são especificados incluindo a coordenada (X,Y) de início da linha e o comprimento horizontal e vertical até o ponto final, conforme a ilustração abaixo:



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



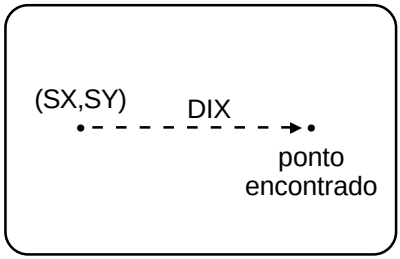
Para executar o comando LINE deve-se escrever o valor 0001B nos quatro bits mais altos de R#46 sendo que os quatro bits mais baixos devem conter o código de operação lógica.



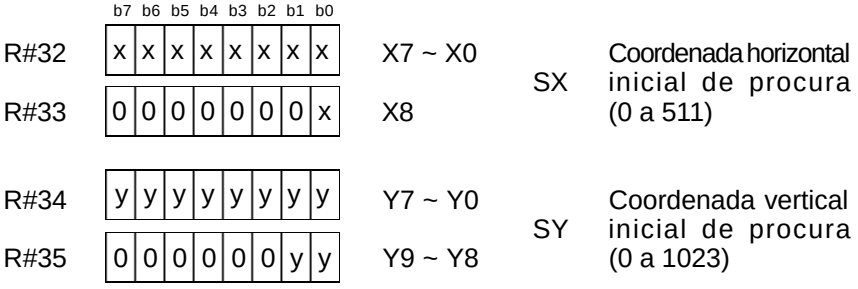
5.4.10 - SRCH (Procura código de cor)

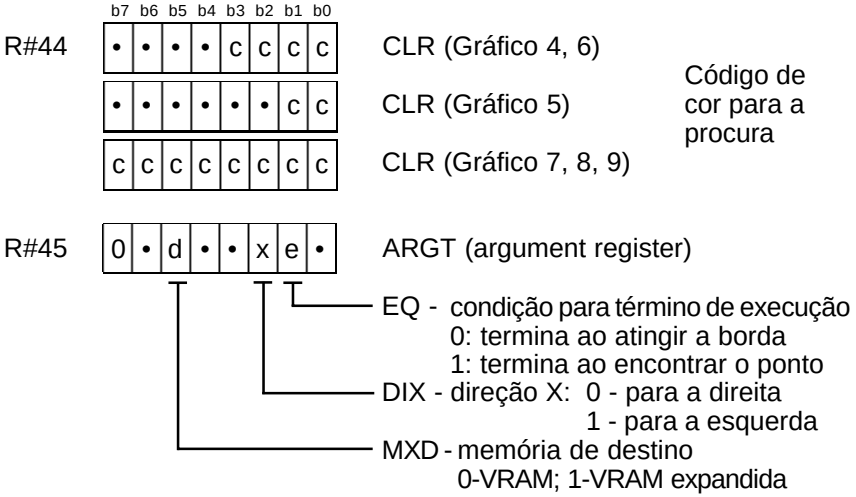
Esse comando procura a existência de um ponto com uma cor específica a partir de uma coordenada na VRAM, sempre na horizontal, para a esquerda ou para a direita. É um comando útil para rotinas de pintura ou preenchimento.

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.

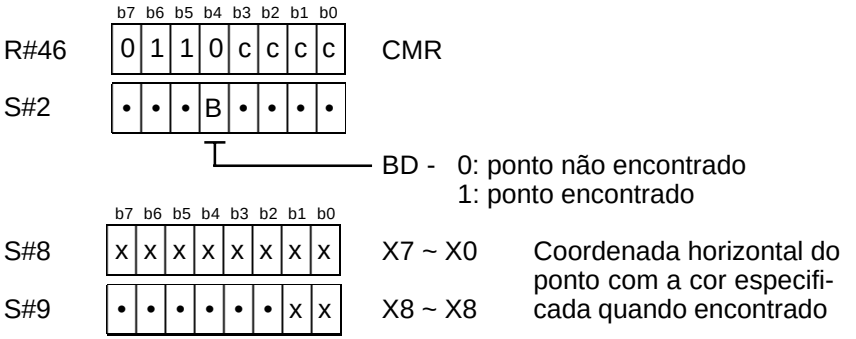


Antes da execução do comando SRCH, os seguintes registradores devem ser carregados:



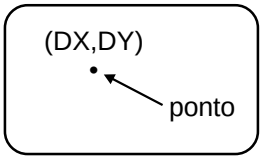


Para executar o comando SRCH, basta escrever o valor 60H no registrador R#46. Ao terminar a execução, se o bit BD se S#2 for 1, o ponto com a cor especificada foi encontrado e sua coordenada horizontal está armazenada em S#8 e S#9. Se o ponto não foi encontrado, o bit BD de S#2 será 0.

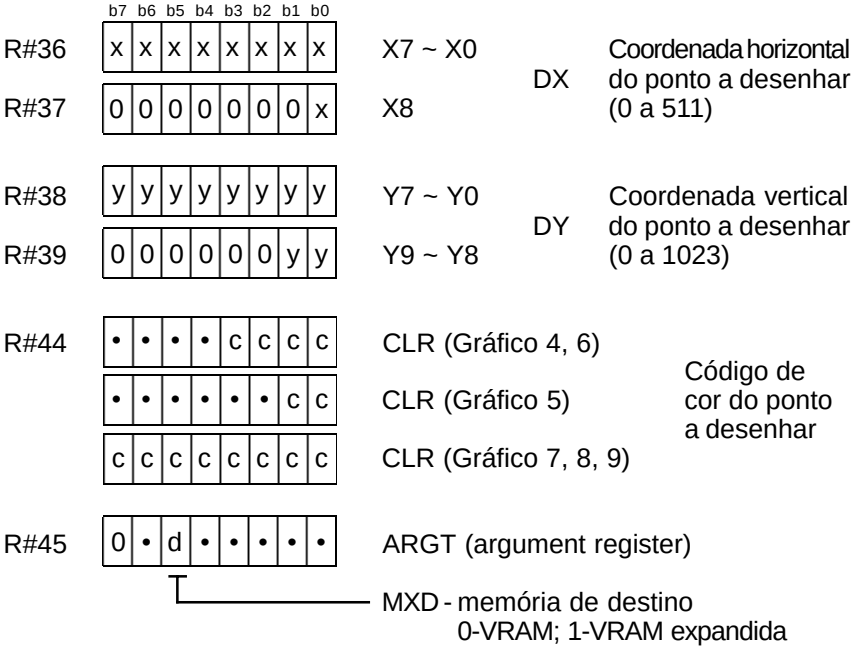


5.4.11 - PSET (Desenha um ponto)

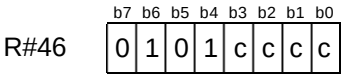
Usando esse comando, pode-se desenhar um ponto em qualquer coordenada da VRAM. Operações lógicas no destino são possíveis.



Os seguintes registradores devem ser carregados:

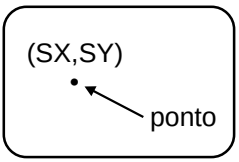


Para executar o comando PSET, basta escrever o valor 0101B nos quatro bits mais altos de R#46 e o código de operação lógica nos quatro bits mais baixos.

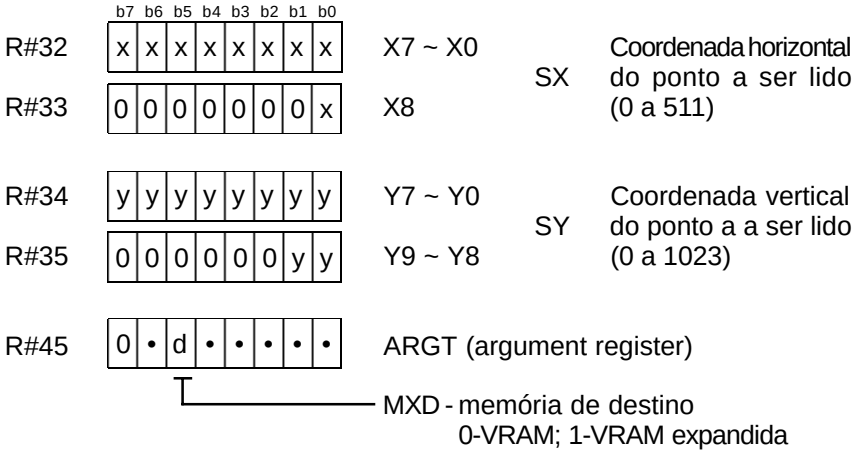


5.4.12 - POINT (Lê código de cor de um ponto)

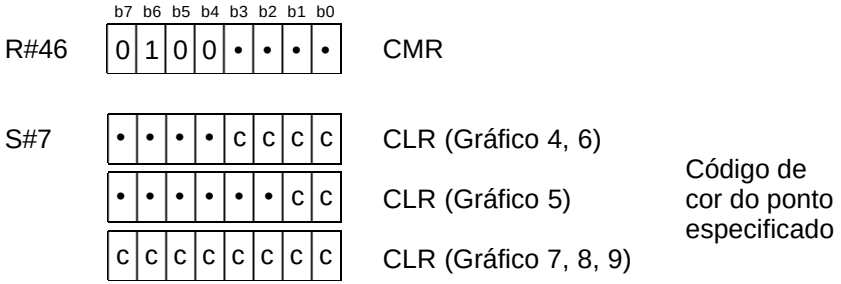
O comando POINT lê um código de cor de um ponto em qualquer coordenada da VRAM.



Antes da execução do comando, devem ser carregados todos os registradores ilustrados na página seguinte.



Para executar o comando POINT, basta escrever o valor 40H em R#46. O código de cor retornará em S#7.

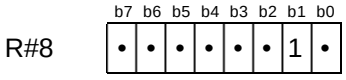


5.5 - TORNANDO OS COMANDOS MAIS RÁPIDOS

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.

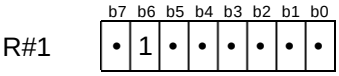
• INIBIÇÃO DE APRESENTAÇÃO DOS SPRITES

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 só deve ser utilizado no caso de inicialização da tela, uma vez que, quando inibida, a tela toda fica com uma só cor. Para tanto, basta setar o bit 6 de R#1 em 1.

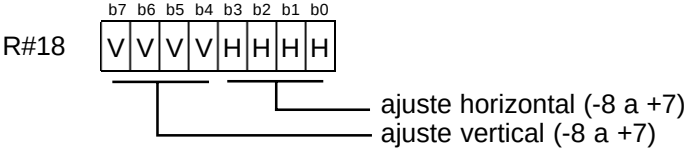


6 - MISCELÂNEA DE FUNÇÕES DO VDP

Essa seção descreve várias funções adicionais do VDP.

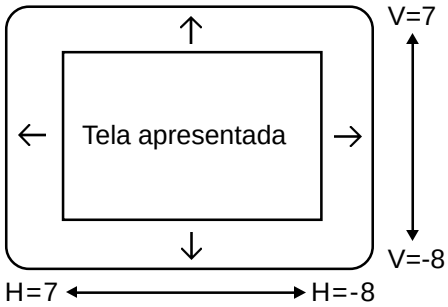
6.1 - AJUSTE DA LOCALIZAÇÃO DA TELA

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



H=7 ..... H=1, H=0, H=15 ..... H=8
(esquerda) (centro) (direita)

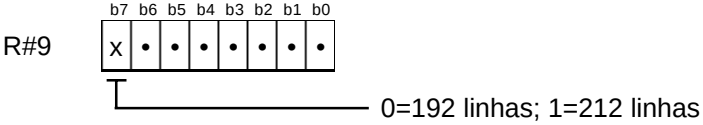
V=8 ..... V=15, V=0, V=1 ..... V=7
(abaixo) (centro) (acima)



6.2 - NÚMERO DE PONTOS NA DIREÇÃO VERTICAL

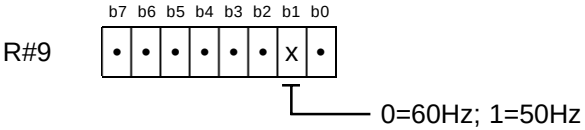
O número de pontos na direção vertical pode ser escolhido entre 192 ou 212, através do bit 7 de R#9. Essa função só é válida para o

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.



6.3 - FREQUÊNCIA DE INTERRUPÇÃO (PAL/NTSC)

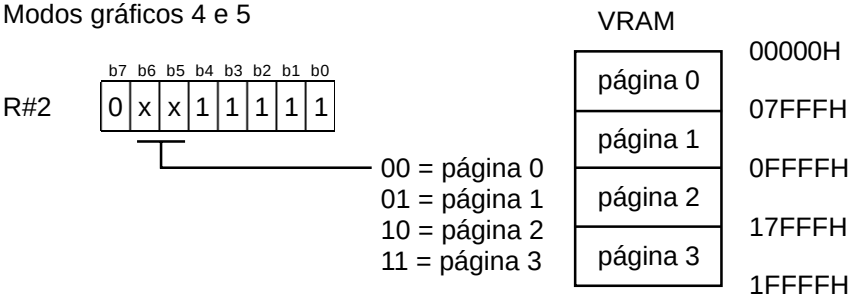
A freqüência de interrupção no MSX é controlada pelo VDP e pode ser de 50Hz ou 60Hz. A freqüência de 60Hz é usada no sistema NTSC do Japão e no sistema PAL-M brasileiro. A de 50Hz é usada para o sistema PAL-N europeu.



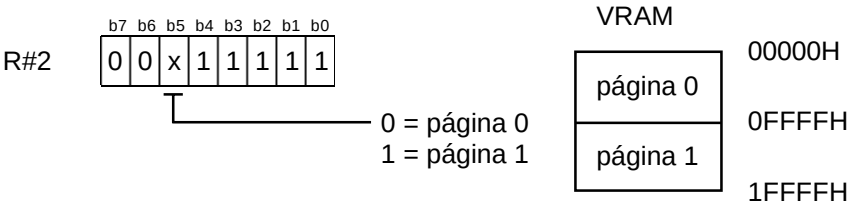
6.4 - TROCA DAS PÁGINAS DE VÍDEO

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

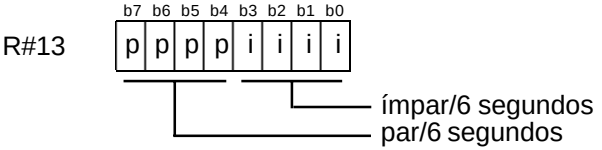


Modos gráficos 6, 7, 8 e 9



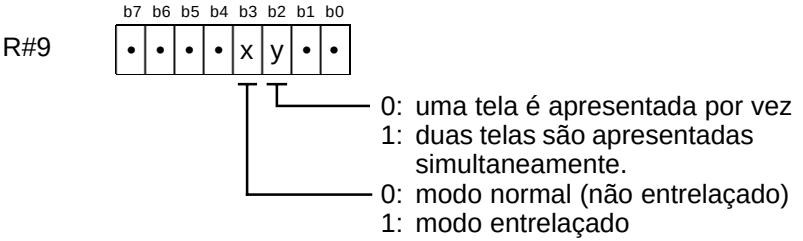
6.5 - TROCA AUTOMÁTICA DE TELA

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, apenas a página ímpar será apresentada.

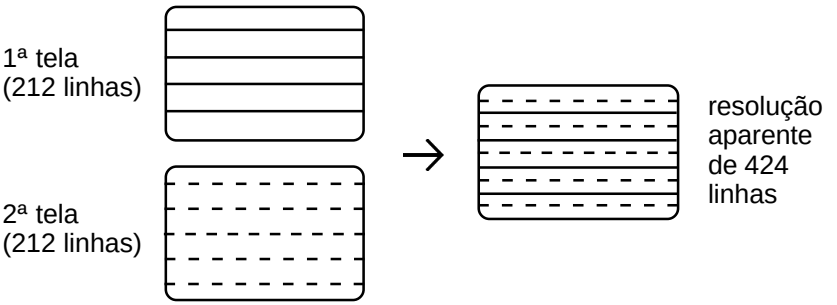


6.6 MODO ENTRELAÇADO

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.



6.7 - SCROLL VERTICAL

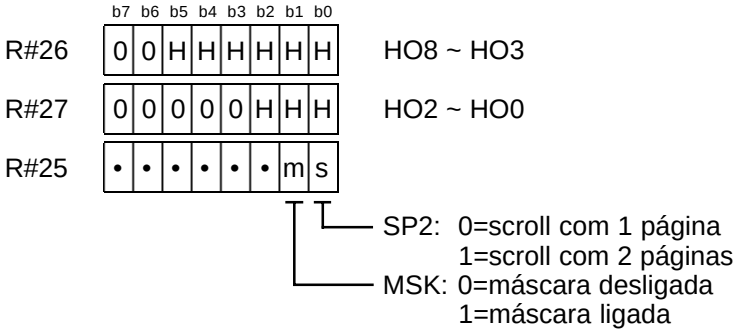
O registrador R#23 é usado para indicar a linha inicial a tela. Trocando o valor desse registrador, pode-se fazer um scroll vertical muito suave. Como o scroll é feito para 256 linhas, a tabela de sprites poderá aparecer e ser movida para outra página.

6.8 - SCROLL HORIZONTAL (V9958 somente)

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.

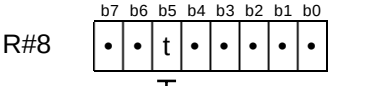
O registrador R#26 pode variar de 0 a 31 (scroll com uma página de vídeo) ou de 0 a 63 (scroll com duas páginas de vídeo). Cada incremento corresponde ao deslocamento de 8 pontos na tela (16 nos modos gráficos 5 e 6). Já R#27 pode variar de 7 a 0 sendo que cada decremento corresponde ao deslocamento 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.



6.9 - CÓDIGO DE COR 0

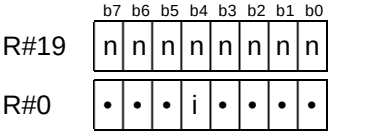
Das 16 cores da paleta, a cor 0 é transparente, ou seja, não pode ser definida uma cor para ela e qualquer objeto desenhado com ela não será visto. Entretanto, setando o bit 5 de R#8, a função de transparente será desativada e a cor 0 poderá ser definida por P#0.



0: código de cor transparente ativo
1: código de cor transparente desativado

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.

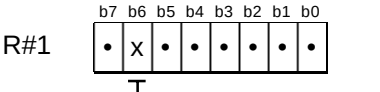


Número de linha para gerar interrupção

0: interrupção de linha desativada
1: interrupção de linha ativa

6.11 - LIGA/DESLIGA A TELA

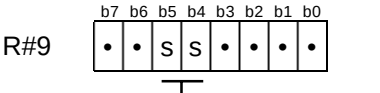
A função de ligar e desligar a apresentação na tela é controlada pelo bit 6 de R#1. Quando estiver desligada, a tela inteira fica com a cor especificada pelos quatro bits mais baixos de R#7 (8 bits no modo gráfico 7). Os comandos de hardware do VDP ficam mais rápidos quando a tela estiver desligada.



0=tela desligada
1=tela ligada

6.12 - MODOS DE SINCRONIZAÇÃO

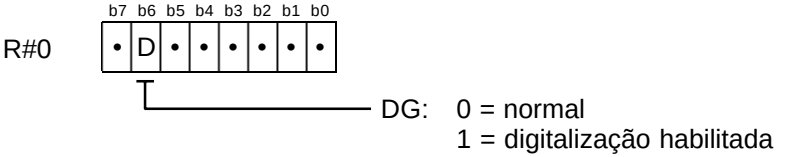
Os modos de sincronização são setados por R#9.



00 - sincronização interna
01 - sincronização mixada
10 - sincronização externa (digitalização)
11 - sem sincronização

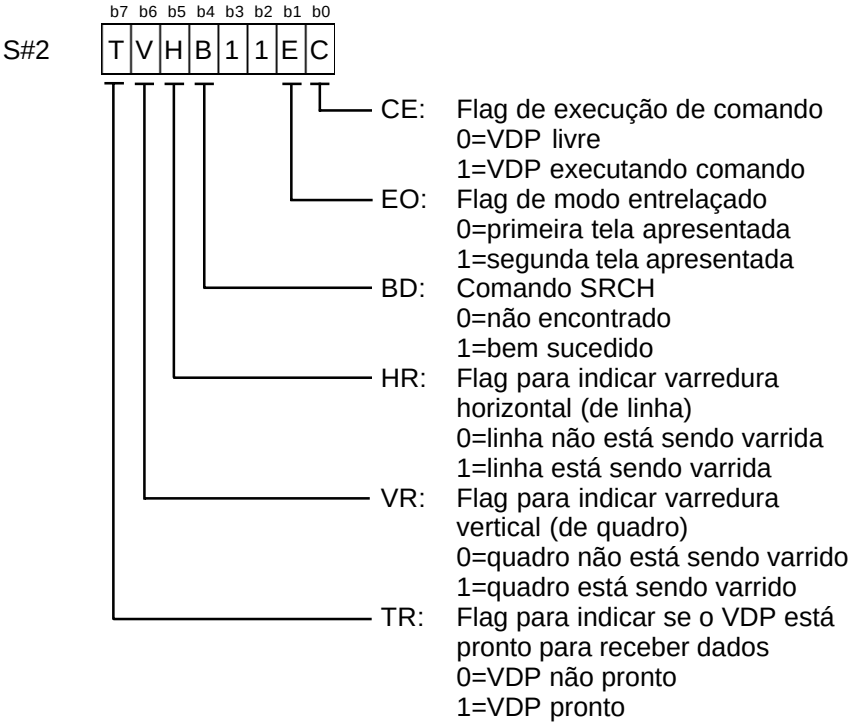
6.13 - DIGITALIZAÇÃO

A função de digitalização é controlada pelo bit 6 de R#0. Ele determina a gravação ou não na VRAM e seta o Color Bus adequadamente.



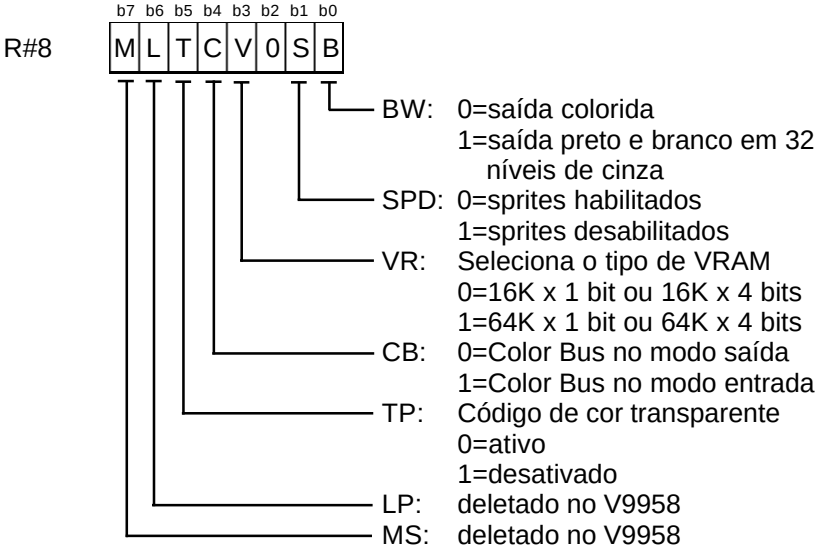
6.14 - O REGISTRADOR DE INFORMAÇÃO E CONTROLE

O registrador de status S#2 é chamado de Registrador de Informação e Controle. Sua organização é a seguinte:



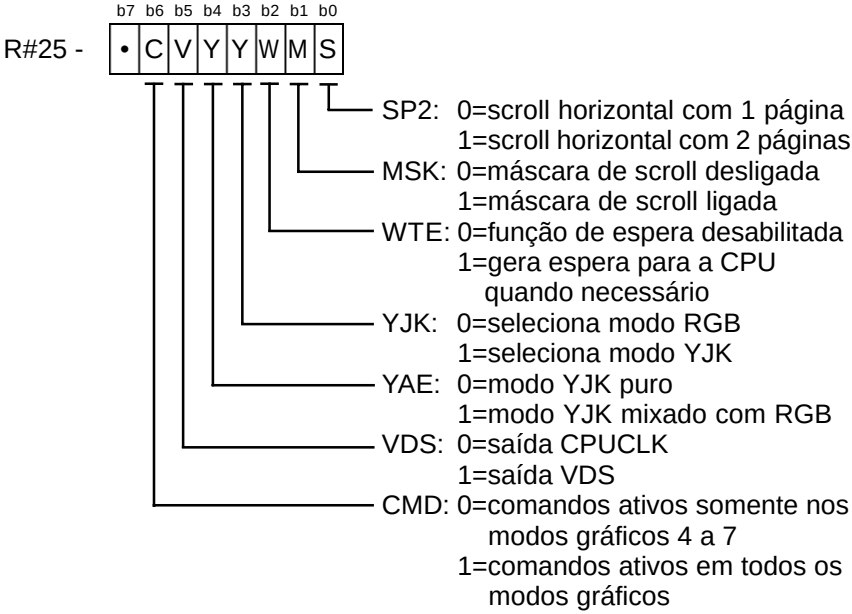
6.15 - O REGISTRADOR DE MODO #2

O registrador R#8 é o registrador de modo 2 e define várias funções especiais do VDP.



6.16 - O REGISTRADOR DE MODO #4

O registrador R#25 foi adicionado ao V9958 e é também conhecido como registrador de modo 4.



7 - O VDP V9990

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. 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 e muito poderosos, além de ser muito mais rápido.

7.1 - 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 todas 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)

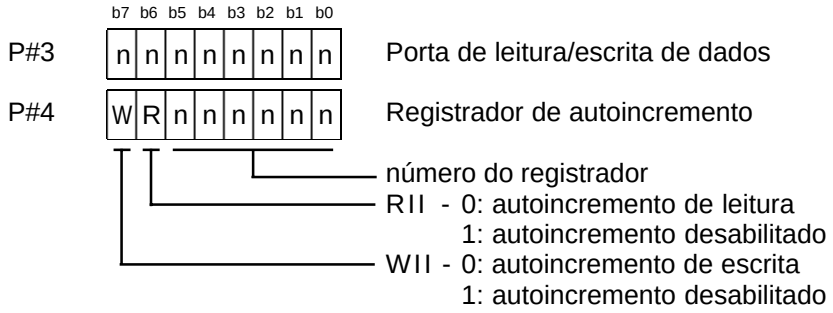
7.2 - ACESSO AO V9990

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 status
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

7.2.1 - ACESSO AOS REGISTRADORES

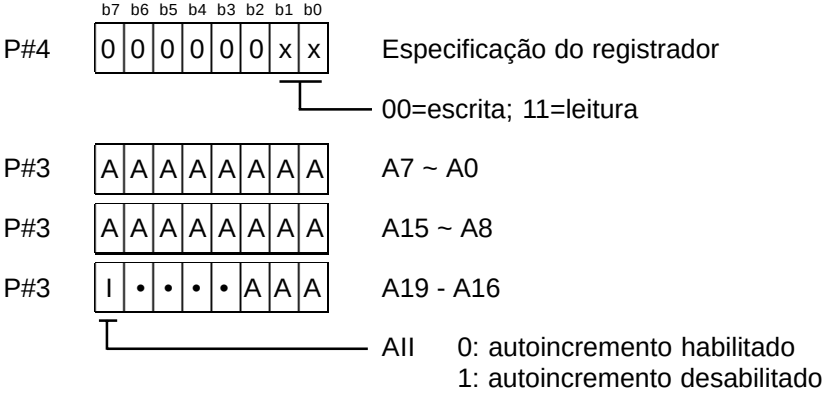
Para escrever um valor num registrador, primeiro é necessário escrever o número do registrador na porta P#4 (64H) e depois o byte de dados na porta P#3 (63H). Para ler um valor de um registrador, basta escrever o número do registrador na porta P#4 (64H) e depois ler o valor respectivo na porta P#3 (63H).



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 registradores subseqüentes. Se forem 1, os bytes serão sempre enviados para o mesmo registrador, indefinidamente. O número do registrador a ser acessado deve ser especificado nos 6 bits mais baixos da porta P#4.

7.2.2 - 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 registradores 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).



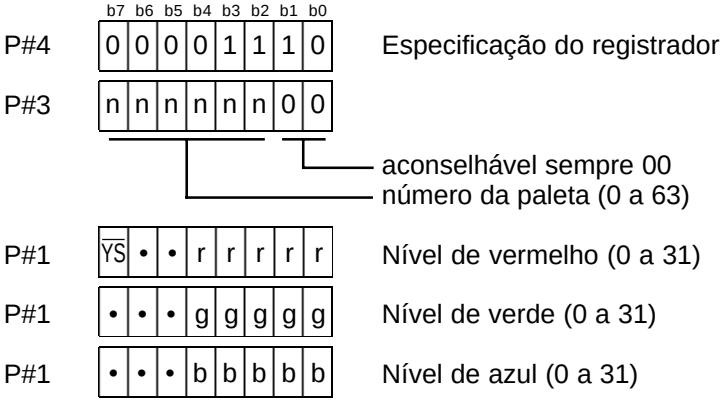
O bit 7 de R#2 ou R#5 habilita ou desabilita autoincremento durante a escrita ou leitura da VRAM. Se for 0, ao ser lido ou escrito um byte pela porta P#0 (60H), o endereço será automaticamente incrementado em 1 e o próximo acesso será no endereço seguinte. Se esse bit for 1, a função de autoincremento será desativada.

7.2.3 - ACESSO À PALETA

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 32768.

O número da 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.

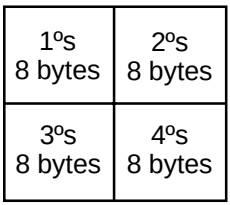
A paleta que será atualmente apresentada também depende do Registrador de Controle de Paleta (R#13).



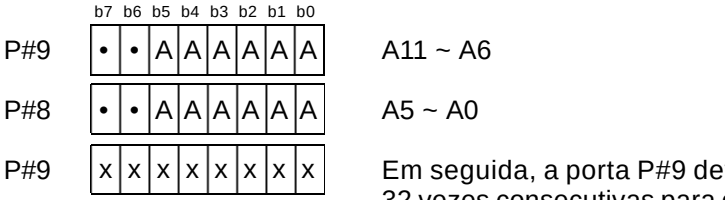
7.2.4 - ACESSO À KANJI ROM

A Kanji-ROM pode ser acessada pelas 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 o acesso, os seis bits mais altos do Kanji respectivo devem ser enviados pela porta P#9/B, seguidos dos seis bits mais baixos pela porta P#8/A. Em seguida, a porta P#9/B deve ser lida 32 vezes para obter os 32 bytes que compõem o padrão do Kanji.



Os 32 bytes que compõem o padrão são obtidos em 4 seqüências de 8 bytes, conforme ilustração ao lado, de forma a compor uma célula de 16 x 16 pontos.



A fonte primária (P#8 e P#9) é conhecida como JIS1 e a fonte secundária (P#A e P#B) é conhecida como JIS2.

7.3 - 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. Existem dois sistemas de screens: Pattern Mode (P1 e P2) e Bit Map Mode (B1 a B6).

MODOS POR APRESENTAÇÃO DE PADRÕES

Nome do Modo		P1	P2
Freqüência Horizontal		15,75 KHz (NTSC)	15,75 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 32768	4 paletas de 16 cores escolhidas de 32768
Área de Imagem		64 x 64 padrões	128 x 64 padrões
Padrões selecionados		16384 máximo	16384 máximo
Gerador de Padrões	(128 Kbytes)	1535 unidades[14]	3071 unidades
	(256 Kbytes)	3583 unidades[14]	7167 unidades
	(512 Kbytes)	7679 unidades[14]	15359 unidades

MODOS BIT MAP (VRAM 128 Kbytes)

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B1	15,75 KHz (NTSC)	256 x 212 (256 x 424)	16	256 x 256
			8	256 x 512 512 x 256
			4	256 x 1024 512 x 512 1024 x 256
			2	256 x 2048 512 x 1024 1024 x 512 2048 x 256

Nota 14: valores relativos a cada screen independente.

MODOS BIT MAP (VRAM 128 Kbytes) - Continuação

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B2	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	8	512 x 256
			4	512 x 512 1024 x 256
			2	512 x 1024 1024 x 512 2048 x 256
B3	15,75 KHz (NTSC)	512 x 212 (512 x 424)	8	512 x 256
			4	512 x 512 1024 x 256
			2	512 x 1024 1024 x 512 2048 x 256
B4	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	4	1024 x 256
			2	1024 x 512 2048 x 256
B5	25,3 KHz	640 x 400	2	1024 x 512
B6	31,5 KHz	640 x 480	2	1024 x 512

MODOS BIT MAP (VRAM 256 Kbytes)

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B1	15,75 KHz (NTSC)	256 x 212 (256 x 424)	16	256 x 512 512 x 256
			8	256 x 1024 512 x 512 1024 x 256
			4	256 x 2048 512 x 1024 1024 x 512 2048 x 256
			2	256 x 4096 512 x 2048 1024 x 1024 2048 x 512

MODOS BIT MAP (VRAM 256 Kbytes) - Continuação

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B2	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	16	512 x 256
			8	512 x 512 1024 x 256
			4	512 x 1024 1024 x 512 2048 x 256
			2	512 x 2048 1024 x 1024 2048 x 512
B3	15,75 KHz (NTSC)	512 x 212 (512 x 424)	16	512 x 256
			8	512 x 512 1024 x 256
			4	512 x 1024 1024 x 512 2048 x 256
			2	512 x 2048 1024 x 1024 2048 x 512
B4	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	4	1024 x 512 2048 x 256
			2	1024 x 1024 2048 x 512
B5	25,3 KHz	640 x 400	4	1024 x 512
			2	1024 x 1024 2048 x 512
B6	31,5 KHz	640 x 480	4	1024 x 512
			2	1024 x 1024 2048 x 512

MODOS BIT MAP (VRAM 512 Kbytes)

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B1	15,75 KHz (NTSC)	256 x 212 (256 x 424)	16	256 x 1024 512 x 512 1024 x 512
			8	256 x 2048 512 x 1024 1024 x 512 2048 x 256
			4	256 x 4096 512 x 2048 1024 x 1024 2048 x 512
			2	256 x 8192 512 x 4096 1024 x 2048 2048 x 1024
B2	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	16	512 x 512 1024 x 256
			8	512 x 1024 1024 x 512 2048 x 256
			4	512 x 2048 1024 x 1024 2048 x 512
			2	512 x 4096 1024 x 2048 2048 x 1024
B3	15,75 KHz (NTSC)	512 x 212 (512 x 424)	16	512 x 512 1024 x 256
			8	512 x 1024 1024 x 512 2048 x 256
			4	512 x 2048 1024 x 1024 2048 x 512
			2	512 x 4096 1024 x 2048 2048 x 1024

MODOS BIT MAP (VRAM 512 Kbytes) - Continuação

Modo	Freqüência Horizontal	Resolução	Número de bits por ponto	Tamanho da Imagem
B4	15,75 KHz (NTSC)	384 x 240 (384 x 480) Overscan	4	1024 x 1024 2048 x 512
			2	1024 x 2048 2048 x 1024
B5	25,3 KHz	640 x 400	4	1024 x 1024 2048 x 512
			2	1024 x 2048 2048 x 1024
B6	31,5 KHz	640 x 480	4	1024 x 1024 2048 x 512
			2	1024 x 2048 2048 x 1024

ESPECIFICAÇÕES DAS PALETAS PARA OS MODOS BIT MAP

	b7/b6 de R#13	Conversão RGB	Número de cores apresentadas
16 bits / ponto	0	RGB direto (YS=1bit; G=5bit; R=5 bit; B=5bit)	32768 cores simultâneas
8 bits / ponto	0	Paleta de cores	64 cores de 32768
	1	RGB direto (G=3bit; R=3bit; B=2bit)	256 cores simultâneas
	2	Codificação YJK	19268 cores simultâneas
	3	Codificação YUV	19268 cores simutâneas
4 bits / ponto	0	Paleta de cores	16 cores de 32768
2 bits / ponto	0	Paleta de cores	4 cores de 32768

7.3.1 - MODO P1

O modo P1 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	0	Controle de sistema
R#6	0	0	0	0	0	1	0	1	Registrador de modo screen #0
R#7	0	•	•	•	•	•	•	0	Registrador de modo screen #1
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

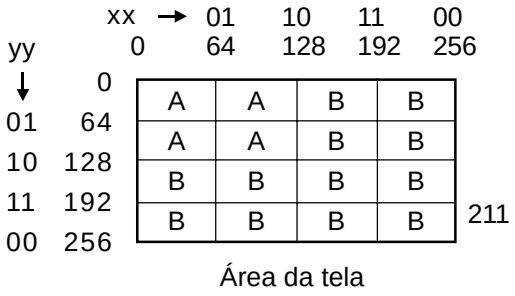
Esse modo tem uma resolução de 256 pontos horizontais por 212 verticais e é mapeado por padrões. Uma característica interessante é que ele tem 2 telas independentes que podem ser sobrepostas (A e B). A área total de imagem, na verdade, é de 512 x 512 pontos cada tela, mas apenas 256 x 212 são apresentados. 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#27	0	0	0	0	y	y	x	x	Registrador de prioridade

Coordenada horizontal a partir da qual a tela "B" será a de frente e a "A" a de fundo.

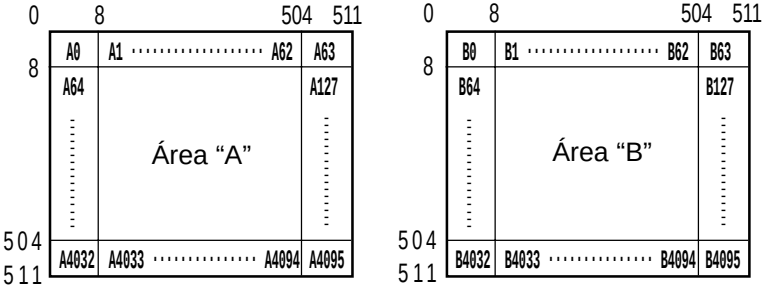
Coordenada vertical a partir da qual a tela "B" será a de frente e a "A" a de fundo.

Quando xx e yy 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 yy=10B e xx=10B.



Área da tela

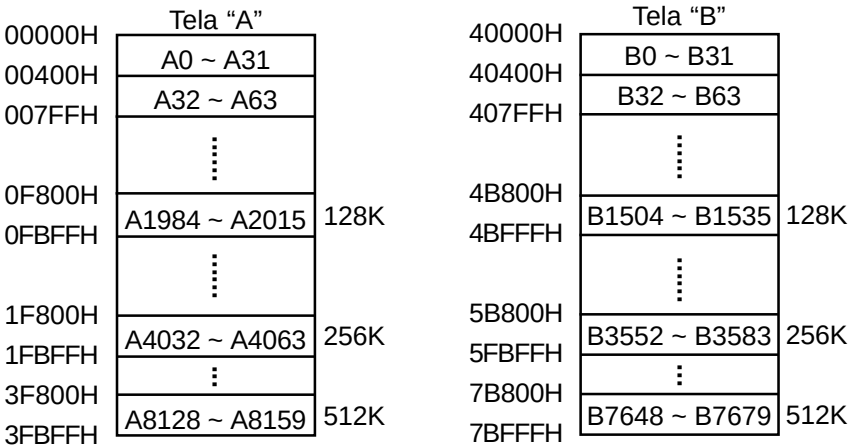
Cada espaço de imagem nas telas “A” e “B” consistem em 4096 (64 x 64) padrões de 8 x 8 pontos, com 16 cores para cada ponto. Para cada um desses padrões, são reservados 2 bytes na Tabela de Nomes para localizá-los na área de imagem.



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	Screen “A”	Screen “B”
128K	2015	1535
256K	4063	3583
512K	8159	7679

A tabela geradora de padrões começa em 00000H e vai até 3FBFFH para a área “A” e de 40000H até 7BFFFH para a área “B”, conforme ilustração abaixo.



A tabela geradora de padrões do modo P1 é organizada como bit map, baseada em 256 pontos hozizontais, reservando 4 bits por ponto.

00000H	Linha 0	A0~A31
00080H	Linha 1	A0~A31
00100H		
00380H	Linha 7	A0~A31
003FFH	Linha 0	A32~A63

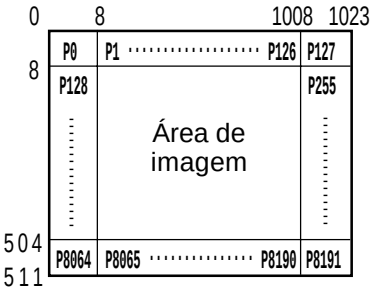
	b7 b6 b5 b4	b3 b2 b1 b0	
00000H	PC0	PC1	A0 Linha 0
00001H	PC2	PC3	A0 Linha 0
00002H	PC4	PC5	A0 Linha 0
00003H	PC6	PC7	A0 Linha0
00004H	PC0	PC1	A1 Linha0
0007FH	PC6	PC7	A31 Linha 0
00080H	PC0	PC1	A0 Linha 1

7.3.2 - MODO P2

O modo P2 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	0	Controle do sistema
R#6	0	1	0	1	1	0	0	1	Registrador de modo screen #0
R#7	0	•	•	•	•	•	•	0	Registrador de modo screen #1
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

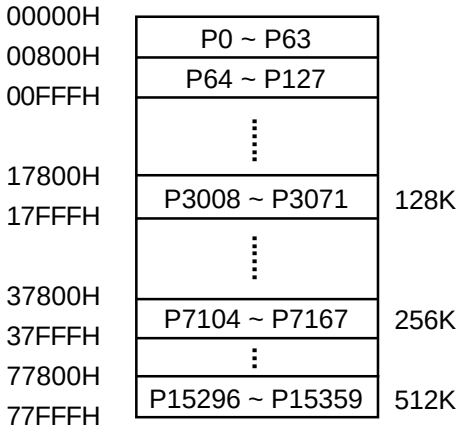
Esse modo tem uma resolução de 512 pontos horizontais por 212 verticais e é mapeado por padrões. Esse modo tem apenas uma tela, e não duas, como no modo P1. A área total de imagem é de 1024 x 512 pontos embora apenas 512 x 212 sejam apresentados. O ponto superior esquerdo da tela pode ser localizado 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.



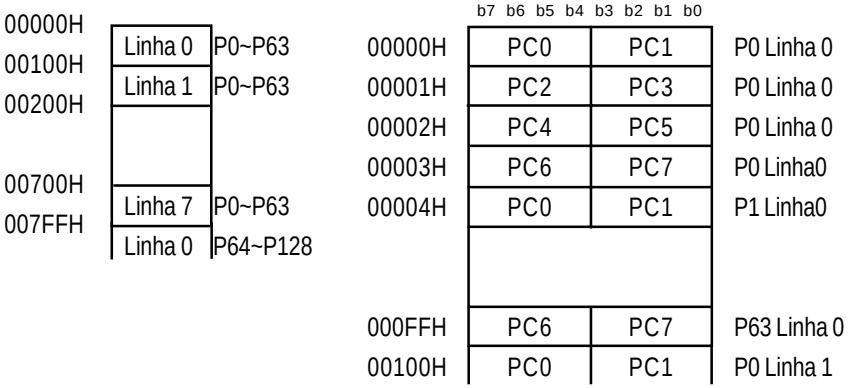
A tabela de nomes começa no endereço 7C000H e vai até o endereço 7FFFFH. Assim, os endereços 7C000H/7C001H correspondem ao padrão P0, 7C000H/7C001H ao padrão P1 assim sucessivamente, na forma low/high. O número máximo de padrões definíveis depende do tamanho da VRAM, conforme tabela abaixo:

VRAM	Padrões
128K	3071
256K	7167
512K	15359

A tabela geradora de padrões começa em 00000H e vai até 77FFFH, conforme ilustração abaixo.



A tabela geradora de padrões do modo P2 é organizada como bit map, baseada em 512 pontos hozizontais, reservando 4 bits por ponto.



7.3.3 - MODO B1

O modo B1 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	0	Controle do sistema
R#6	1	0	0	0	•	•	•	•	Registrador de modo screen #0
R#7	0	•	•	•	•	•	•	0	Registrador de modo screen #1

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 tama-nho 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	x	c	c	Registrador de modo screen #0

CLRM: 00 = 2 bits por ponto
01 = 4 bits por ponto
10 = 8 bits por ponto
11 = 16 bits por ponto

XIMM: 00 = 256 pontos horizontais
01 = 512 pontos horizontais
10 = 1024 pontos horizontais
11 = 2048 pontos horizontais

A descrição do mapa de memória dos modos B1~B6 será descrita adiante, na seção 7.4.

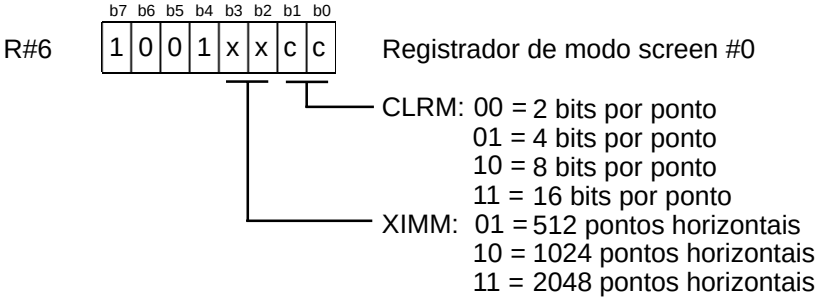
7.3.4 - MODO B2

O modo B2 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	1	Controle do sistema
R#6	1	0	0	1	•	•	•	•	Registrador de modo screen #0
R#7	0	•	•	•	•	•	•	0	Registrador de modo screen #1

O modo B2 tem uma resolução de 384 x 240 (60 Hz) ou 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 tama-

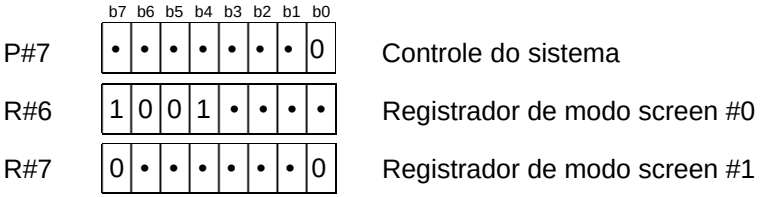
nho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no início dessa seção. O número de bits usados por ponto apresentado e o tamanho horizontal da área de imagem são especificados em R#6. O número de pontos verticais é automaticamente setado.



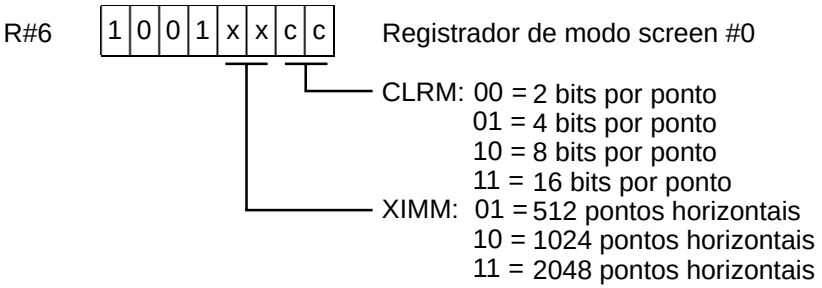
A descrição do mapa de memória dos modos B1~B6 será descrita adiante, na seção 7.4.

7.3.5 - MODO B3

O modo B3 é selecionado pelos seguintes registradores:



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.



A descrição do mapa de memória dos modos B1~B6 será descrita adiante, na seção 7.4.

7.3.6 - MODO B4

O modo B4 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	1	Controle do sistema
R#6	1	0	1	0	•	•	•	•	Registrador de modo screen #0
R#7	0	•	•	•	•	•	•	0	Registrador de modo screen #1

O modo B4 tem uma resolução de 768 x 240 (60 Hz) ou 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	1	0	1	0	x	x	c	c	Registrador de modo screen #0

CLRM: 00 = 2 bits por ponto
01 = 4 bits por ponto
10 = 8 bits por ponto
11 = 16 bits por ponto

XIMM: 10 = 1024 pontos horizontais
11 = 2048 pontos horizontais

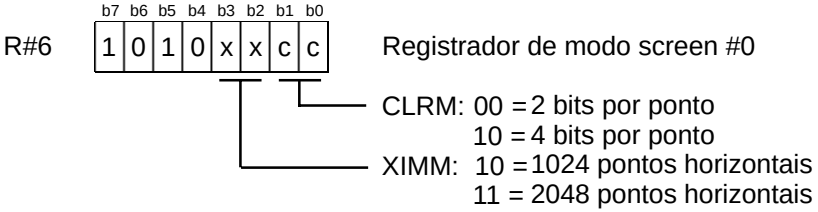
A descrição do mapa de memória dos modos B1~B6 será descrita adiante, na seção 7.4.

7.3.7 - MODO B5

O modo B5 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
P#7	•	•	•	•	•	•	•	0	Controle do 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

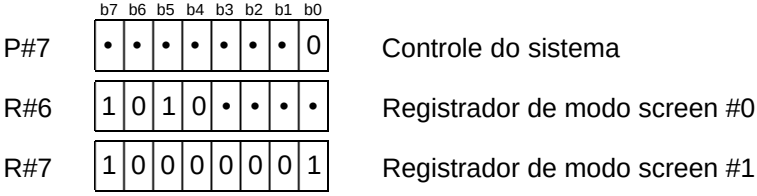
O modo B5 tem 640 x 400 pontos e é um modo de alta resolução, só funcionando em monitores tipo VGA (fH=24,8 KHz). A área de imagem pode variar entre 1024 e 2048 pontos horizontais e 512 e 2048 pontos verticais. dependendo do tamanho da VRAM e do tipo de paleta usada. Esses tamanhos foram descritos no 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.



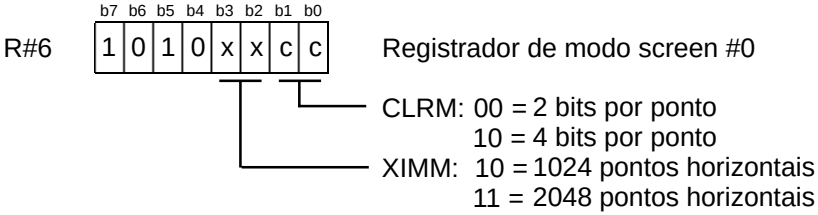
A descrição do mapa de memória dos modos B1~B6 será descrita adiante, na seção 7.4.

7.3.8 - MODO B6

O modo B6 é selecionado pelos seguintes registradores:



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.



A descrição do mapa de memória dos modos B1~B6 está descrita logo abaixo (seção 7.4).

7.4 - MAPA DE MEMÓRIA DOS MODOS B1~B6

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	001FFH
1	00200H				003FFH
	⋮				⋮
511	3FE00H	3FE01H	·····	3FFFEH	3FFFFH

Área de imagem

Os pontos são distribuídos na memória de acordo com a paleta usada, conforme ilustrado abaixo.

2 bits por ponto

	b7 b6	b5 b4	b3 b2	b1 b0
00000H	0,0	1,0	2,0	3,0
00001H	4,0	5,0	6,0	7,0
00002H	8,0	9,0	10,0	11,0

4 bits por ponto

	b7 b6 b5 b4	b3 b2 b1 b0
00000H	0,0	1,0
00001H	2,0	3,0
00002H	4,0	5,0

8 bits por ponto

	b7 b6 b5 b4 b3 b2 b1 b0
00000H	0,0
00001H	1,0
00002H	2,0

16 bits por ponto

	b7 b6 b5 b4 b3 b2 b1 b0
00000H	0,0 low
00001H	0,0 high
00002H	1,0 low

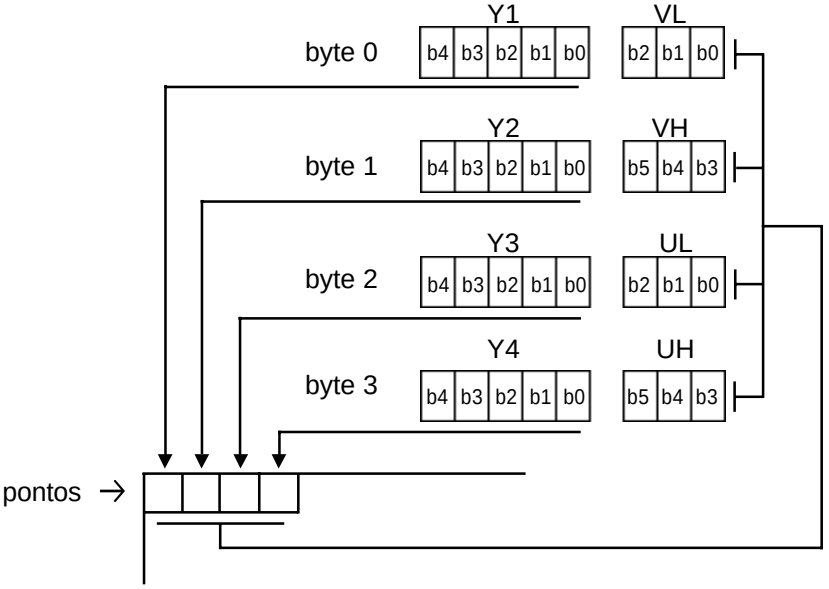
7.5 - ESPECIFICAÇÕES DE CORES PARA OS MODOS B1~B6

Como já descrito, cada ponto pode ocupar 2, 4, 8 ou 16 bits nos modos B1 a B6. Entretanto, existem dez tipos de representações diferentes que podem ser selecionadas. Esses tipos são os seguintes:

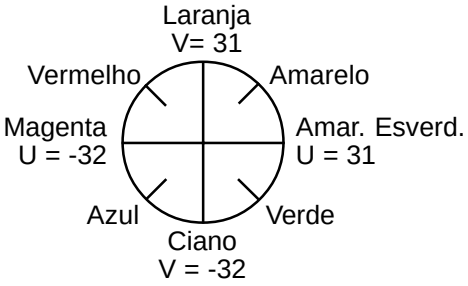
1. BYUV - Modo YUV puro nos modos B1 ~ B6.
2. BYUVP - Modo YUV mixado com paleta nos modos B1 ~B6.
3. BYJK - Modo YJK puro nos modos B1 ~ B6.
4. BYJKP - Modo YJK mixado com paleta nos modos B1 ~B6.
5. BD16 - Apresenta 32768 cores simultâneas sem uso de paleta.
6. BD8 - Apresenta 256 cores simultâneas sem uso de paleta.
7. BP6 - Apresenta 64 cores simultâneas de uma paleta de 32768.
8. BP4 - Apresenta 16 cores simultâneas de uma paleta de 32768.
9. BP2 - Apresenta 4 cores simultâneas de uma paleta de 32768.
10. PP - Usada exclusivamente nos modos P1 e P2.

7.5.1 - MODO BYUV

O modo BYUV apresenta até 19268 cores simutâneas 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 na página seguinte.



U=100000 = -32 V=011111 = 31	Verde
U=011111 = 31 V=100000 = -32	Vermelho
U=100000 = -32 V=100000 = -32	Azul
U=011111 = 31 V=011111 = 31	Amarelo

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 saturação de 32 níveis, representada pelos bits Yn, desde o branco até a cor saturada. Se seu valor for 11111B, o ponto será branco. Se for 00000B, o ponto terá a cor saturada.

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 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	R = Y + U
U = R - Y	G = 5/4 Y - 1/2 U - 1/4 V
V = G - Y	B = Y + V

Um detalhe importante é quanto ao número de cores. Como há 4096 cores e 32 níveis de saturação para cada uma, na verdade são 32 * 4096 = 131072 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 19268.

O endereço na VRAM de cada ponto da área de imagem pode ser calculado pela seguinte expressão:

ENDEREÇO = X + Y*(Tamanho horizontal da área em pontos)

Onde X é a coordenada horizontal e Y a vertical.

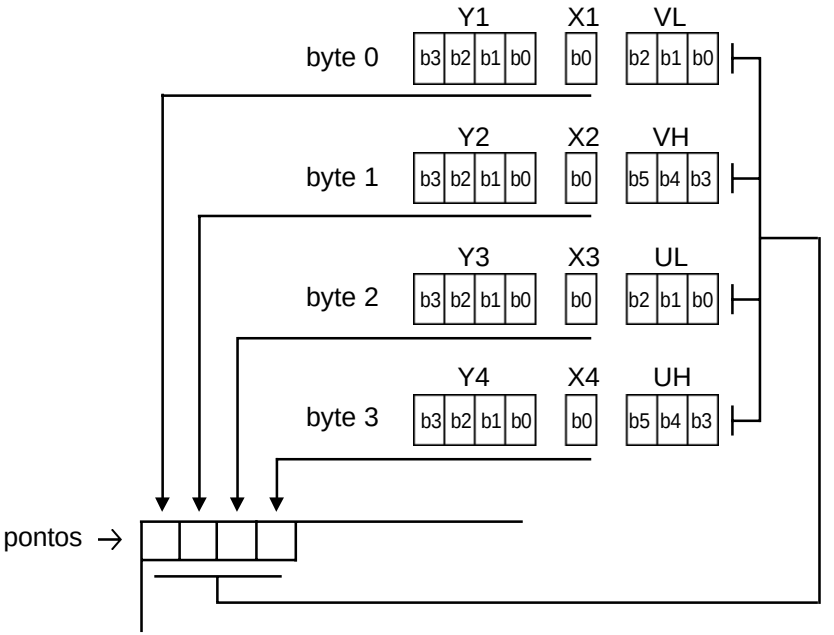
O modo BYUV é selecionado por R#6 e R#13, conforme ilustração abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	1	1	0	•	•	•	•	•	Registrador de paleta

7.5.2 - MODO BYUVP

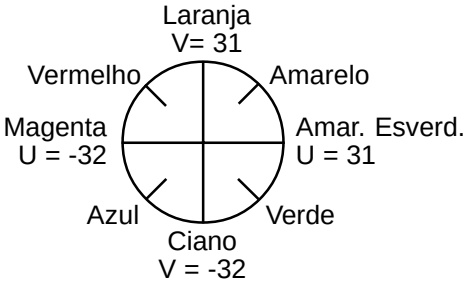
O modo BYUVP é um modo misto, podendo apresentar até 12499 cores simultâneas, através do sistema YUV, ou então usar a paleta.

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 saturação para cada ponto individual, desde o branco até a cor saturada. Ou então cada ponto pode ter até 16 cores escolhidas de uma paleta de 32768, tal qual o modo BYUV. A organização desse modo está ilustrada abaixo.



Quando os bits Xn 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 Yn só pode variar de 0 a 15. Já se os bits

Xn forem 1, a cor será escolhida da paleta. Podem ser escolhidas até 16 cores de 32768. Não é obrigatório que todos os bits Xn 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.



U=100000 = -32 V=011111 = 31	Verde
U=011111 = 31 V=100000 = -32	Vermelho
U=100000 = -32 V=100000 = -32	Azul
U=011111 = 31 V=011111 = 31	Amarelo

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 saturação de apenas 16 níveis, e não 32 como no modo BYUV, representada pelos bits Yn, desde o branco até a cor saturada. Se seu valor for 1111B, o ponto será branco. Se for 0000B, o ponto terá a cor saturada.

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 = 65536 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 12499.

O endereço na VRAM de cada ponto da área de imagem pode ser calculado pela seguinte expressão:

ENDEREÇO = X + Y*(Tamanho horizontal da área em pontos)

O modo BYUV é selecionado por R#6 e R#13, conforme ilustração abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	1	1	1	•	•	•	•	•	Registrador de paleta

7.5.3 - MODO BYJK

O modo BYJK é exatamente igual ao modo BYUV, exceto pela proporção dos vetores de cor (JK ou UV). 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 ente 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	0	1	0	•	•	•	•	•	Registrador de paleta

7.5.4 - MODO BYJKP

O modo BYJKP é exatamente igual ao modo BYUVP, exceto pela proporção dos vetores de cor (JK ou UV), que podem ser calculada pelas fórmulas apresentadas acima. Esse modo é selecionado por R#6 e R#13, conforme ilustração abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	0	1	1	•	•	•	•	•	Registrador de paleta

7.5.5 - MODO BD16

Nesse modo, podem ser apresentadas até 32768 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 $\overline{YS}$ (superimpose). Esses bits são organizados na VRAM de acordo com a ilustração da página seguinte.

	b15	b14	b13	b12	b11	b10	b9	b8
byte 1	$\overline{YS}$	G	G	G	G	G	R	R

	b7	b6	b5	b4	b3	b2	b1	b0
byte 0	R	R	R	B	B	B	B	B

Onde G é a intensidade de verde (00000B a 11111B), R a do vermelho e B a do azul. $\overline{YS}$ é uma flag para indicar superimpose para cada ponto individual. Quando $\overline{YS}$ 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:

ENDEREÇO = X*2 + Y*2*(Tamanho horizontal da área em pontos)

O modo BD16 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	1	Registrador de modo screen #0
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

7.5.6 - 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.

	b7	b6	b5	b4	b3	b2	b1	b0
byte	G	G	G	R	R	R	B	B

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:

ENDEREÇO = X + Y*(Tamanho horizontal da área em pontos)

Esse modo é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	0	1	0	•	•	•	•	•	Registrador de paleta

Nesse modo, não há como fazer superimpose para cada ponto individual.

7.5.7 - MODO BP6

Nesse modo, cada ponto pode ter até 64 cores escolhidas de uma paleta de 32768. Cada ponto é representado por um byte, sendo que apenas os seis bits mais baixos são válidos, conforme ilustração abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0
byte	0	0	p	p	p	p	p	p

Onde p é o número da paleta a ser apresentada (0 a 63). Esse modo permite superimpose seletivo, desde essa função esteja selecionada para a cor respecitva 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:

ENDEREÇO = X + Y*(Tamanho horizontal da área em pontos)

Esse modo é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	1	0	Registrador de modo screen #0
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

7.5.8 - MODO BP4

Nesse modo, cada ponto pode ter até 16 cores escolhidas de uma paleta de 32768. A vantagem sobre o modo BP6 é que cada que um byte é usado para representar dois pontos, ocupando metade da memória do modo BP6, conforme ilustração abaixo.

	b7	b6	b5	b4	b3	b2	b1	b0
byte	p0	p0	p0	p0	p1	p1	p1	p1

Onde p0 representa a paleta do primeiro ponto (0 a 15) e p1 representa a paleta do ponto seguinte no sentido horizontal (0 a 15). Esse modo permite 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 endereço de cada ponto na área de imagem pode ser calculado pela seguinte expressão:

ENDEREÇO = X/2 + Y*(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:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	0	1	Registrador de modo screen #0
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

7.5.9 - MODO BP2

Nesse modo, cada ponto pode ter até 4 cores escolhidas de uma paleta de 32768. 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.

	b7	b6	b5	b4	b3	b2	b1	b0
byte	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. 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:

ENDEREÇO = X/4 + Y*(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 BP4 é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	0	0	Registrador de modo screen #0
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

7.6 - ESPECIFICAÇÃO DE CORES PARA OS MODOS P1~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 32768. Duas delas podem ser usadas simultaneamente, de acordo com a tabela da página seguinte.

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 b3 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 $\overline{YS}$ setado em 1). Nesse caso, todos os pontos com a mesma cor estarão selecionados para superimpose. O modo PP é selecionado pelos seguintes registradores:

	b7	b6	b5	b4	b3	b2	b1	b0	
R#6	•	•	•	•	•	•	0	1	Registrador de modo screen #0
R#13	0	0	0	•	•	•	•	•	Registrador de paleta

7.7 - 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.

7.7.1 - 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 32768 (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 fomato 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#25	0	0	0	0	A	A	A	0	A17 ~ A15 (modo P1)
R#25	0	0	0	0	A	A	A	A	A18 ~ A15 (modo P2)

A tabela geradora de sprites só pode começar em múltiplos de 16 Kbytes a partir de 00000H. Para o modo P1, ela tem a seguinte estrutura:

Offset		
00000H	Linha 0	S0~S15
00080H	Linha 1	S0~S15
00100H	⋮	
00380H	Linha 15	S0~S15
003FFH	Linha 0	S16~S31

Offset	b7 b6 b5 b4 b3 b2 b1 b0		
00000H	SC0	SC1	S0 Linha 0
	⋮		
00007H	SC14	SC15	S0 Linha0
00008H	SC0	SC1	S1 Linha0
	⋮		
0007FH	SC14	SC15	S15 Linha 0
00080H	SC0	SC1	S0 Linha 1

Para o modo P2, há uma ligeira mudança nos endereços.

Offset		
00000H	Linha 0	S0~S31
00100H	Linha 1	S0~S31
00200H	⋮	
00F00H	Linha 15	S0~S31
01000H	Linha 0	S32~S63

Offset	b7 b6 b5 b4 b3 b2 b1 b0		
00000H	SC0	SC1	S0 Linha 0
	⋮		
00007H	SC14	SC15	S0 Linha0
00008H	SC0	SC1	S1 Linha0
	⋮		
000FFH	SC14	SC15	S31 Linha 0
00100H	SC0	SC1	S0 Linha 1

A 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.

3FE00H	Spr #0
3FE04H	Spr #1
3FE08H	⋮
3FFF0H	Spr #125
3FFF3H	

	b7	b6	b5	b4	b3	b2	b1	b0	
3FE00	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Coord. Vertical
3FE01	P7	P6	P5	P4	P3	P2	P1	P0	Número do Padrão
3FE02	X7	X6	X5	X4	X3	X2	X1	X0	Coord. Horizontal
3FE03	C5	C4	P1	P0			X9	X8	Inform. Adicional

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 da página anterior, toda a paleta pode ser selecionada para os 125 sprites, podendo ter estes até 64 cores de 32768. Entretanto, são possíveis apenas 16 cores por sprite individual. Na verdade, são 15 cores, pois a cor 0 é 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
1	0	A > SP > B > BD
-	1[15]	A > B > BD

SP: plano dos sprites
A: plano de frente
B: plano de trás
BD: plano de fundo

Já a prioridade de apresentação no modo P2 segue a tabela abaixo.

P1	P0	Ordem de prioridade
0	0	SP > IP > BD
1	0	IP > SP > BD
-	1[13]	IP > BD

SP: plano dos sprites
IP: plano de imagem
BD: plano de fundo

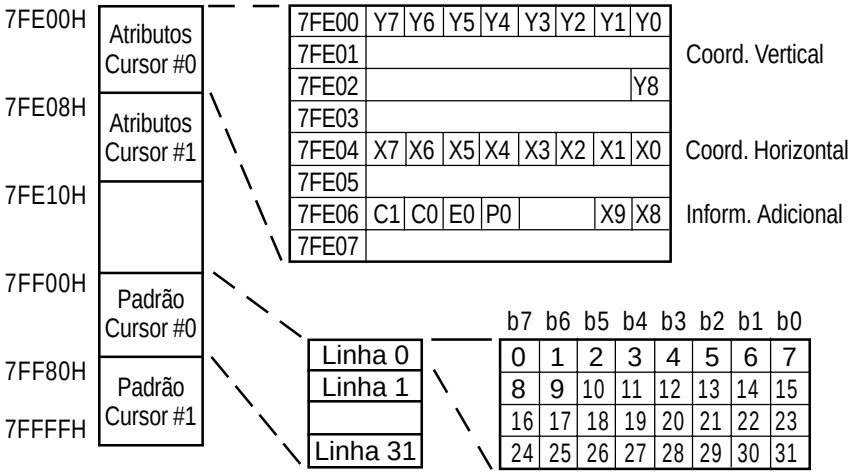
Em qualquer dos casos, o sprite de número menor tem prioriodade de apresentação maior em relação ao de número maior, da mesma forma que para o VDP V9958.

7.7.2 - CURSORES PARA OS MODOS B1 ~ B6

Para os modos B1 a B6, existe uma função denominada “cursor”. Trata-se de sprites com bem menos recursos. Apenas 2 cursores 32 x 32

Nota 15: Os sprites não são exibidos quando P0=1.

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.



A área em que o cursor pode ser apresentado é a área apresentada na 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.

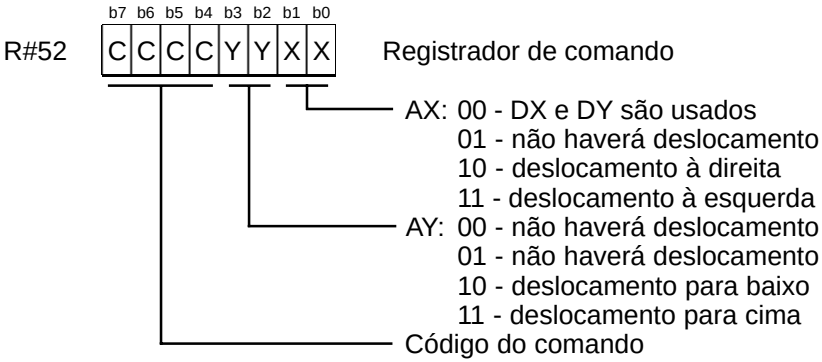
7.8 - COMANDOS DO VDP V9990

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 comando. A tabela na página seguinte descreve resumidamente todos os comandos.

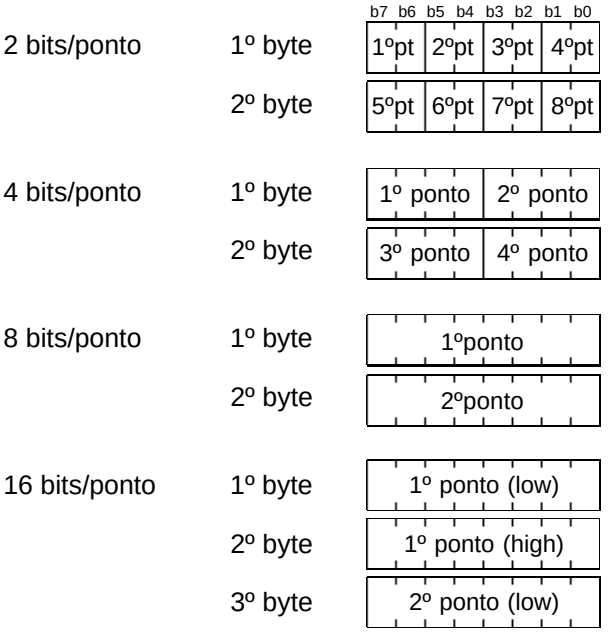
0 0 0 0	STOP	Comando de parada
0 0 0 1	LMMC	Tranferência da CPU → VRAM (coordenadas)
0 0 1 0	LMMV	Pinta retângulo na VRAM
0 0 1 1	LMCM	Tranferência da VRAM → CPU (coordenadas)
0 1 0 0	LMMM	Transferência VRAM ↔ VRAM (coordenadas)
0 1 0 1	CMMC	Transfere caractere da CPU para a VRAM
0 1 1 0	CMMK	Transfere 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):

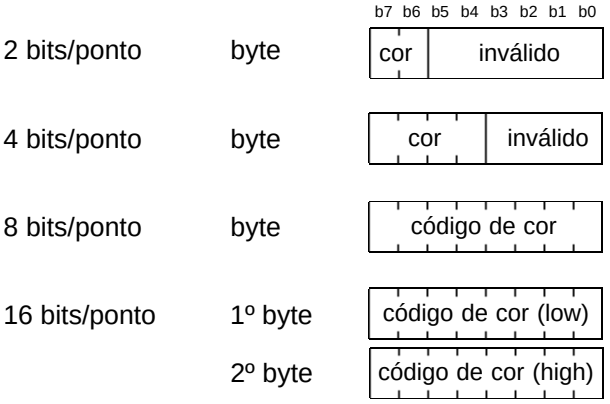


7.8.1 - FORMATO DOS DADOS PARA OS COMANDOS

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 na página seguinte.



Para o comando POINT, o formato para os dados é o seguinte:



7.8.2 - PARÂMETROS PARA OS COMANDOS

Os parâmetros para a execução de cada um dos comandos devem ser setados nos registradores R#32 a R#51 antes de enviar o comando respectivo ao registrador R#52. Esses parâmetros estão descritos na página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0
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						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					SY11	SY10	SY9	SY8
						SA18	SA17	SA16
							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.

1- Comandos LMCM, LMMM, BMLX, SRCH e POINT
   SX0 ~ 10: Especifica a coordenada horizontal inicial. 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 SX9=0 e o plano “B” será selecionado quando SX9=1.
   SY0 ~ 11: Especifica a coordenada vertical inicial. Será levada a 0 quando a coordenada especificada for maior que a altura da área de imagem.
2- Comandos CMMM, BMXL e BMLL
   SA0 ~ 18: Especifica o endereço inicial da VRAM.
3- Comando CMMK
   KA0 ~ 17: Especifica o endereço da Kanji ROM.

	b7	b6	b5	b4	b3	b2	b1	b0
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
R#37						DX10	DX9	DX8

	b7	b6	b5	b4	b3	b2	b1	b0
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8

R#39					DY11	DY10	DY9	DY8
						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.

1- 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.
2- Comandos BMLX e BMLL
   DA0 ~ 18: Especifica o endereço final da VRAM.

	b7	b6	b5	b4	b3	b2	b1	b0
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						NX10	NX9	NX8
					MJ11	MJ10	MJ9	MJ8

R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
	NA15	NA14	NA13	NA12	NA11	NA10	NA9	NA8
	M17	M16	M15	M14	M13	M12	M11	M10

R#43					NY11	NY10	NY9	NY8
						NA18	NA17	NA16
					MI11	MI10	MI9	MI8

Os registradores da página anterior 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.

1- 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).

2- 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).

3- 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.
   - M10 ~ 11: Tamanho do lado menor do triângulo retângulo de refe-rência em pontos. Será levada a 0 quando seu valor exceder o tamanho da área de imagem.

	b7	b6	b5	b4	b3	b2	b1	b0
R#44	•	•	•	•	DIY	DIX	NEQ	MAJ

Esse é o registrador de argumento.

- 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.
- 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.
- 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).

	b7	b6	b5	b4	b3	b2	b1	b0
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	

	b7	b6	b5	b4	b3	b2	b1	b0
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 (ou Plano “A” para modo P1) e R#47 para VRAM1 (ou Plano “B” para modo P1). 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.

	b7	b6	b5	b4	b3	b2	b1	b0
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

7.8.3 - EXECUTANDO OS COMANDOS

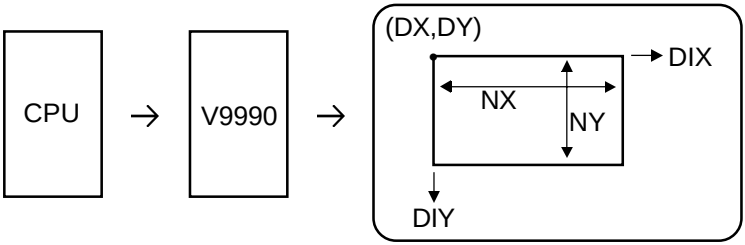
Primeiro, é necessário setar todos os valores nos registradores adequados. Depois, basta escrever o código do comando em R#52, juntamente com os dados para deslocamento de pontos para os comandos PSET e ADVN. Para interromper o comando, é só escrever o comando de parada em R#52 (00H).

Enquando o comando estiver sendo executado, o bit CE da porta P#5 ficará setado em 1.

	b7	b6	b5	b4	b3	b2	b1	b0
P#5	•	•	•	•	•	•	•	CE

7.8.4 - LMMC (Tranferência lógica CPU → VRAM)

Nesse comando, os dados são transferidos da CPU para uma área retangular na VRAM.



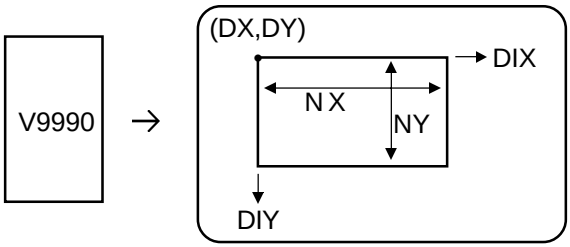
Os registradores devem ser carregados de acordo com a ilustração da página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos a transferir na direção horizontal
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a transferir na direção vertical
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX	Direção de transferência	
R#45	0	0	0	t	l	l	l	l	LOP	Código de Operação Lógica	
R#46	w	w	w	w	w	w	w	w	WM7~WM0	Máscara de escrita	
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#52	0	0	0	1	0	0	0	0	OP-CODE	Código de comando LMMC	

Ao executar o comando, o número necessário de bytes a transferir será enviado pela porta de comando (P#2).

7.8.5 - LMMV (Desenha retângulo)

Esse comando desenha um retângulo na área de imagem.

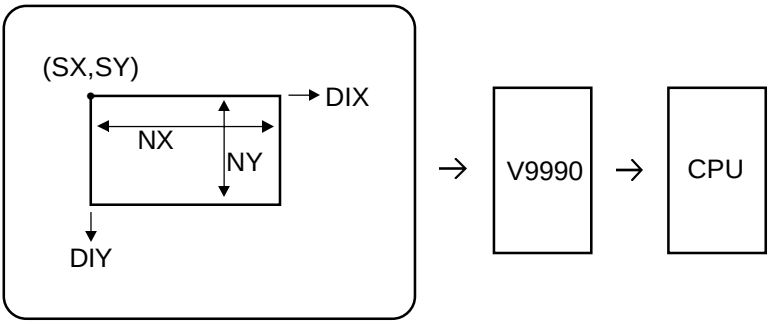


Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal do ponto inicial para a pintura
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical do ponto inicial para a pintura
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos a pintar na direção horizontal
R#41	0	0	0	0	0	x	x	x	X10 ~ X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a pintar na direção vertical
R#43	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de pintura
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#48	f	f	f	f	f	f	f	f	FC7~ C0		Código de cor para a pintura
R#49	f	f	f	f	f	f	f	f	FC15~FC8		
R#52	0	0	1	0	0	0	0	0	OP-CODE		Código de comando LMMV

7.8.6 - LMCM (Tranferência lógica VRAM → CPU)

Nesse comando, os dados de uma área retangular na VRAM são tranferidos para a CPU. O comando LMCM está ilustrado na página seguinte.



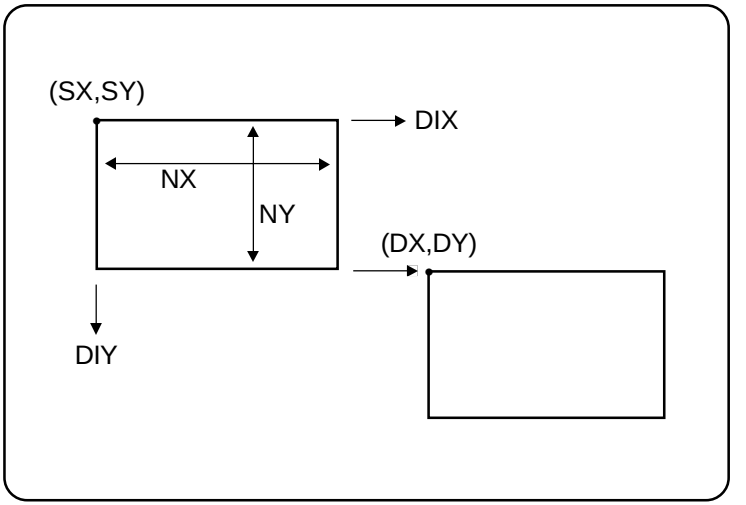
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal inicial para transferência
R#33	0	0	0	0	0	x	x	x	X10 ~X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical inicial para transferência
R#35	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos a transferir na direção horizontal
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a transferir na direção vertical
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX	Direção de transferência	
R#45	0	0	0	t	l	l	l	l	LOP	Código de Operação Lógica	
R#52	0	0	1	1	0	0	0	0	OP-CODE	Código do comando LMCM	

Os dados para o número necessário de bytes a serem transferidos devem ser entrados pela porta de comando (P#2).

7.8.7 - LMMM (Tranferência lógica VRAM → VRAM)

Nesse comando, uma área retangular da VRAM é tranferida para outra posição na VRAM. Operações lógicas no destino são possíveis.



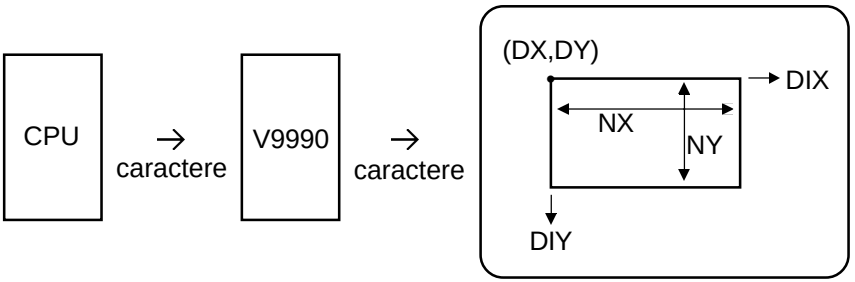
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal inicial para transferência
R#33	0	0	0	0	0	x	x	x	X10 ~X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical inicial para transferência
R#35	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino para transferência
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino para transferência
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		

	b7	b6	b5	b4	b3	b2	b1	b0			
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos a transferir na direção horizontal
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos a transferir na direção vertical
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#52	0	1	0	0	0	0	0	0	OP-CODE		Código do comando LMMM

7.8.8 - CMMC (Tranferência de caractere CPU → VRAM)

Nesse comando, caracteres são tranferidos da CPU para uma área retangular da VRAM. Operações lógicas no destino são possíveis.

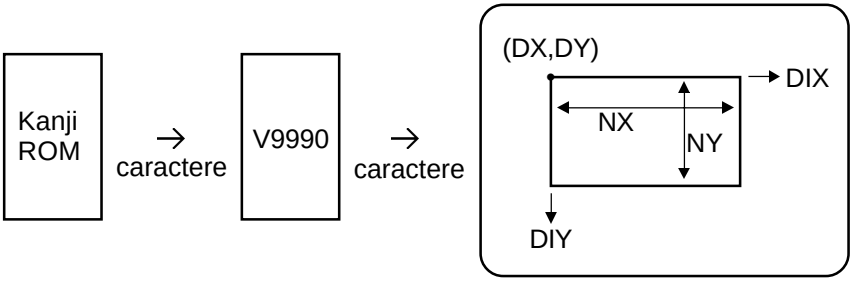


Os registradores a serem carregados estão ilustrados na página seguinte. Ao executar o comando, os dados para o número necessário de bytes são enviados para a porta de comando (P#2).

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino para transferência
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino para transferência
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Largura da célula do caractere (número de pontos horizontais)
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Altura da célula do caractere (número de pontos verticais)
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#48	f	f	f	f	f	f	f	f	FC7~FC0		Código de cor do caractere a ser desenhado para a fonte #1
R#49	f	f	f	f	f	f	f	f	FC15~FC8		
R#50	b	b	b	b	b	b	b	b	BC7~BC0		Código de cor do caractere a ser desenhado para a fonte #0
R#51	b	b	b	b	b	b	b	b	BC15~BC8		
R#52	0	1	0	1	0	0	0	0	OP-CODE		Código do comando CMMC

7.8.9- CMMK (Tranferência de caractere Kanji ROM → VRAM)

Nesse comando, os caracteres são tranferidos da Kanji ROM conectada diretamente ao V9990 para uma área retangular na VRAM. Operações lógicas no destino são possíveis.



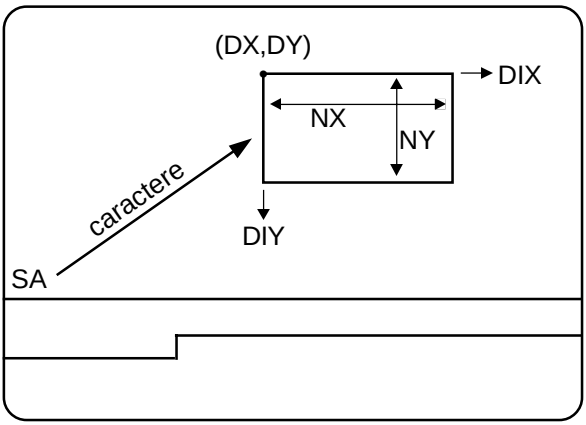
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	k	k	k	k	k	k	k	k	KA7~KA0		Endereço do caractere Kanji a ser transferido para a VRAM
R#34	k	k	k	k	k	k	k	k	KA15~KA8		
R#35	0	0	0	0	0	0	k	k	KA17~KA16		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino para transferência
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino para transferência
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Largura da célula do caractere (número de pontos horizontais)
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Altura da célula do caractere (número de pontos verticais)
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		

	b7	b6	b5	b4	b3	b2	b1	b0		
R#44	0	0	0	0	y	x	0	0	DIY, DIX	Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP	Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0	Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8	
R#48	f	f	f	f	f	f	f	f	FC7~FC0	Código de cor do caractere Kanji a ser transferido para a fonte #1
R#49	f	f	f	f	f	f	f	f	FC15~FC8	
R#50	b	b	b	b	b	b	b	b	BC7~BC0	Código de cor do caractere Kanji a ser transferido para a fonte #0
R#51	b	b	b	b	b	b	b	b	BC15~BC8	
R#52	0	1	1	0	0	0	0	0	OP-CODE	Código do comando CMMK

7.8.10 - CMMM (Tranferência 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.



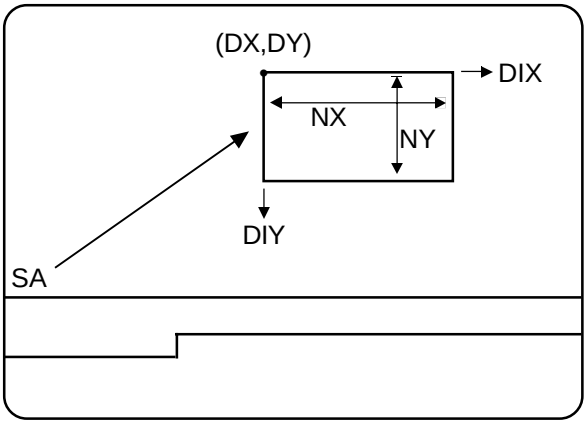
Para o comando CMMM, os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	s	s	s	s	s	s	s	s	SA7~SA0		
R#34	s	s	s	s	s	s	s	s	SA15~SA8		Endereço na VRAM do caractere a ser transferido para as coordenadas
R#35	0	0	0	0	0	s	s	s	SA17~SA16		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino para transferência
R#37	0	0	0	0	0	x	x	x	X10 ~X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino para transferência
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Largura da célula do caractere (número de pontos horizontais)
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Altura da célula do caractere (número de pontos verticais)
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#48	f	f	f	f	f	f	f	f	FC7~FC0		Código de cor do caractere a ser transferido para a fonte #1
R#49	f	f	f	f	f	f	f	f	FC15~FC8		

	b7	b6	b5	b4	b3	b2	b1	b0		
R#50	b	b	b	b	b	b	b	b	BC7~BC0	Código de cor do caractere a ser transferido para a fonte #0
R#51	b	b	b	b	b	b	b	b	BC15~BC8	
R#52	0	1	1	1	0	0	0	0	OP-CODE	Código do comando CMMM

7.8.11 - BMXL (Tranferência de bytes - linear → coordenadas)

Nesse comando, bytes de dados são tranferidos de uma área linear da VRAM para uma área retangular na VRAM. Operações lógicas no destino são possíveis.

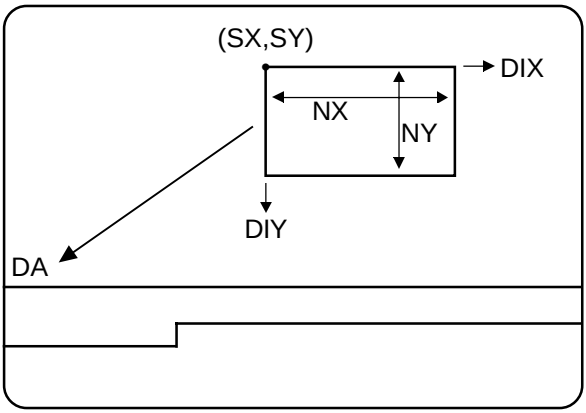


Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	s	s	s	s	s	s	s	s	SA7~SA0		
R#34	s	s	s	s	s	s	s	s	SA15~SA8	Endereço linear da VRAM	
R#35	0	0	0	0	0	s	s	s	SA18~SA16		
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal de destino para transferência
R#37	0	0	0	0	0	x	x	x	X10 ~X8		

	b7	b6	b5	b4	b3	b2	b1	b0			
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical de destino para transferência
R#39	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos na direção horizontal a transferir
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos na direção vertical a transferir
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#52	1	0	0	0	0	0	0	0	OP-CODE		Código do comando BMXL

7.8.12 - BMLX (Tranferência de bytes - coordenadas → linear)

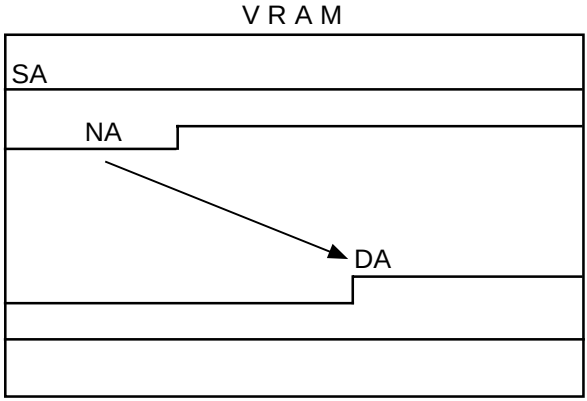


No comando BMLX, bytes de dados são transferidos de uma área retangular da VRAM para uma área linear na VRAM. Operações lógicas no destino são possíveis. Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal de origem para transferência
R#33	0	0	0	0	0	x	x	x	X10 ~X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SX	Coordenada vertical de origem para transferência
R#35	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#36	d	d	d	d	d	d	d	d	DA7~DA0		
R#38	d	d	d	d	d	d	d	d	DA15~DA8		Endereço linear da VRAM
R#39	0	0	0	0	0	d	d	d	DA18~DA16		
R#40	x	x	x	x	x	x	x	x	X7 ~ X0	NX	Número de pontos na direção horizontal a transferir
R#41	0	0	0	0	0	x	x	x	X10 ~X8		
R#42	y	y	y	y	y	y	y	y	Y7 ~ Y0	NY	Número de pontos na direção vertical a transferir
R#43	0	0	0	0	y	y	y	y	Y11 ~Y8		
R#44	0	0	0	0	y	x	0	0	DIY, DIX		Direção de transferência
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#52	1	0	0	1	0	0	0	0	OP-CODE		Código do comando BMLX

7.8.13 - BMLL (Tranferência 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.



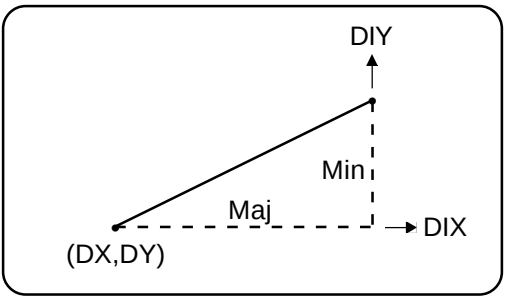
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0		
R#32	s	s	s	s	s	s	s	s	SA7~SA0	Endereço linear de origem na VRAM para a transferência
R#34	s	s	s	s	s	s	s	s	SA15~SA8	
R#35	0	0	0	0	0	s	s	s	SA18~SA16	
R#36	d	d	d	d	d	d	d	d	DA7~DA0	Endereço linear de destino na VRAM para a transferência
R#38	d	d	d	d	d	d	d	d	DA15~DA8	
R#39	0	0	0	0	0	d	d	d	DA18~DA16	
R#40	n	n	n	n	n	n	n	n	NA7~NA0	Número de bytes a tranferir
R#42	n	n	n	n	n	n	n	n	NA15~NA8	
R#43	0	0	0	0	0	n	n	n	NA18~NA16	
R#44	0	0	0	0	y	x	0	0	DIY, DIX	Direção de transferência

	b7	b6	b5	b4	b3	b2	b1	b0		
R#45	0	0	0	t	l	l	l	l	LOP	Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0	Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8	
R#52	1	0	1	0	0	0	0	0	OP-CODE	Código do comando BMLL

7.8.14 - LINE (Desenha uma linha)

Esse 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:



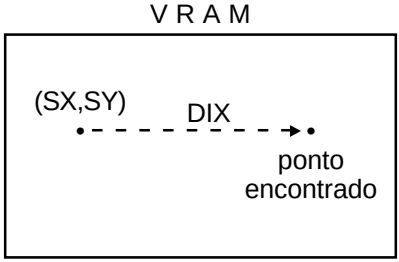
Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal inicial a partir da qual a linha será desenhada.
R#37	0	0	0	0	0	x	x	x	X10 ~ X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical inicial a partir da qual a linha será desenhada.
R#39	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#40	j	j	j	j	j	j	j	j	MJ7 ~ MJ0	Maj	Número de pontos do cateto maior do triângulo retângulo de referência para desenho.
R#41	0	0	0	0	j	j	j	j	MJ11 ~ MJ8		

	b7	b6	b5	b4	b3	b2	b1	b0			
R#42	i	i	i	i	i	i	i	i	MI7 ~ MI0	Min	Número de pontos do cateto menor do triângulo retângulo de referência para desenho.
R#43	0	0	0	0	i	i	i	i	MI9 ~MI8		
R#44	0	0	0	0	y	x	0	m	DIY, DIX, MAJ[16]		Direção de desenho
R#45	0	0	0	t	l	l	l	l	LOP		Código de Operação Lógica
R#46	w	w	w	w	w	w	w	w	WM7~WM0		Máscara de escrita
R#47	w	w	w	w	w	w	w	w	WM15~WM8		
R#48	f	f	f	f	f	f	f	f	FC7~FC0		Código de cor da linha a ser desenhada
R#49	f	f	f	f	f	f	f	f	FC15~FC8		
R#52	1	0	1	1	0	0	0	0	OP-CODE		Código do comando LINE

7.8.15 - 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.



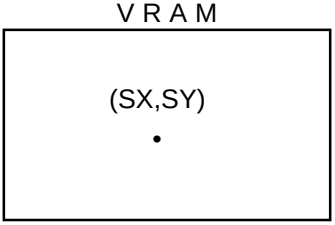
Nota 16: para MAJ=0, o lado maior do triângulo retângulo de referência é paralelo ao eixo X (horizontal) e para MAJ=1, o lado maior é paralelo ao eixo Y (vertical)

Os seguintes registradores devem ser carregados:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal para ínicio da procura
R#33	0	0	0	0	0	x	x	x	X10 ~X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical para ínicio da procura
R#35	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#44	0	0	0	0	0	x	n	0	DIX, NEQ[17]		Direção de procura e especificação de cor
R#48	f	f	f	f	f	f	f	f	FC7~FC0		Código de cor do ponto a ser detectado
R#49	f	f	f	f	f	f	f	f	FC15~FC8		
R#52	1	1	0	0	0	0	0	0	OP-CODE		Código do comando SRCH
R#53	x	x	x	x	x	x	x	x	X7 ~ X0	BX	Coordenada horizontal do ponto, se encontrado
R#54	0	0	0	0	0	x	x	x	X10 ~X8		

7.8.16 - 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.



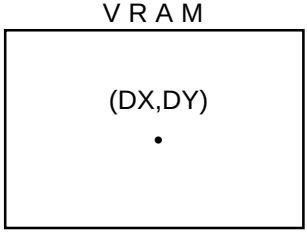
Nota 17: para NEQ=0, a cor para detecção é especificada; para NEQ=1 a cor para detecção não é especificada.

Os seguintes registradores devem ser carregados para a execução do comando POINT:

	b7	b6	b5	b4	b3	b2	b1	b0			
R#32	x	x	x	x	x	x	x	x	X7 ~ X0	SX	Coordenada horizontal do ponto
R#33	0	0	0	0	0	x	x	x	X10 ~X8		
R#34	y	y	y	y	y	y	y	y	Y7 ~ Y0	SY	Coordenada vertical do ponto
R#35	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#52	1	1	0	1	0	0	0	0	OP-CODE	Código do comando POINT	

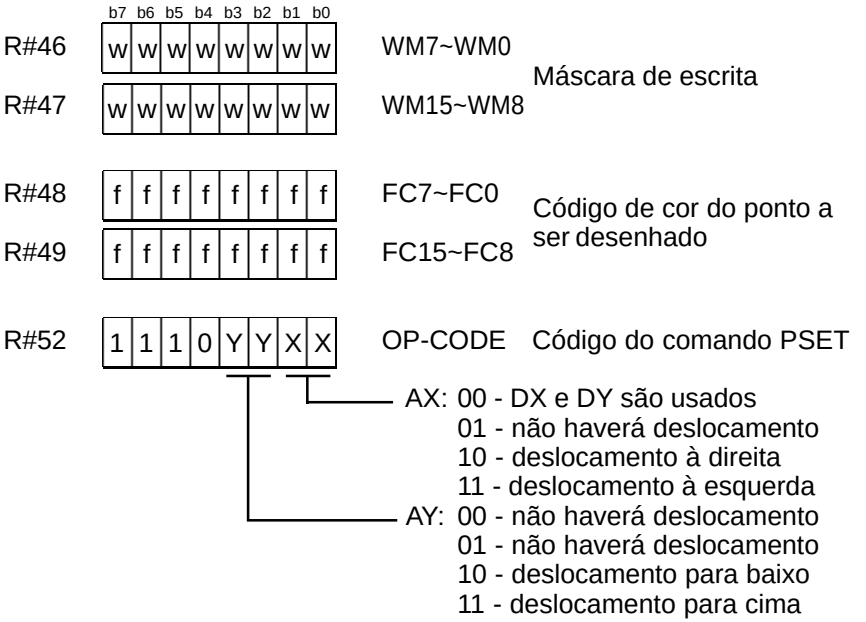
7.8.17 - PSET (Desenha um ponto e avança)

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:

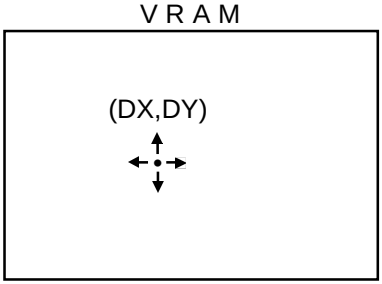
	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal onde o ponto será desenhado
R#37	0	0	0	0	0	x	x	x	X10 ~ X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical onde o ponto será desenhado
R#39	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#45	0	0	0	t	l	l	l	l	LOP	Código de operação lógica	



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 YN, YE, XM e XE e o próximo ponto poderá ser desenhado nessa posição.

7.8.18 - ADVN (Avança coordenadas)

Esse comando simplesmente avança coordenadas na área de imagem sem desenhar.



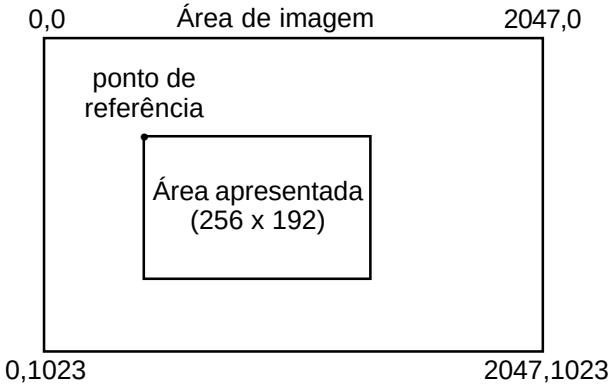
Os registradores a serem carregados estão ilustrados na página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0			
R#36	x	x	x	x	x	x	x	x	X7 ~ X0	DX	Coordenada horizontal a partir da qual ocorrerá o avanço
R#37	0	0	0	0	0	x	x	x	X10 ~ X8		
R#38	y	y	y	y	y	y	y	y	Y7 ~ Y0	DY	Coordenada vertical a partir da qual ocorrerá o avanço
R#39	0	0	0	0	y	y	y	y	Y11 ~ Y8		
R#52	1	1	1	1	Y	Y	X	X	OP-CODE		Código do comando ADVN

Os valores de deslocamento (X e Y) são os mesmos usados para o comando PSET, descritos na página anterior. 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.

7.9 - 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 212 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 na página seguinte.

	b7	b6	b5	b4	b3	b2	b1	b0	
R#17	y	y	y	y	y	y	y	y	SCAY7~SCAY0
R#18	0	0	0	y	y	y	y	y	SCAY12~SCAY8
R#19	0	0	0	0	0	x	x	x	SCAX2~SCAX0
R#20	x	x	x	x	x	x	x	x	SCAX10~SCAX3
R#21	y	y	y	y	y	y	y	y	SCBY7~SCBY0
R#22	0	0	0	0	0	0	0	y	SCBY8
R#23	0	0	0	0	0	x	x	x	SCBX2~SCBX0
R#24	0	0	x	x	x	x	x	x	SCBX8~SCBX3

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.

	b7	b6	b5	b4	b3	b2	b1	b0
R#18	R	R	0	•	•	•	•	•

00: scroll em toda a área de imagem
01: scroll em 256 linhas
10: scroll em 512 linhas
11: sem significado

7.10 - FUNÇÕES ADICIONAIS DO V9990

Existem muitas funções adicionais que foram implementadas no V9990 e que servem de complemento às funções descritas até agora.

As funções adicionais estão descritas a partir da página seguinte.

7.10.1 - O REGISTRADOR DE MODO #1

O registrador R#7 (registrador de modo #1) possui várias funções especiais além da seleção de modos de tela. Essas funções são as seguintes:

	b7	b6	b5	b4	b3	b2	b1	b0
R#7	0	•	SM1	SM	PAL	EO	IL	•

SM1: número de linhas verticais (não interlace, NTSC)
1: 263 linhas (em combinação com SM, a fase da subportadora de cor é invertida em cada quadro)
0: 262 linhas
SM: freqüência horizontal (inválido nos modos B5 e B6)
1: 1H = fsc / 227,5 (a fase da subportadora de cor é invertida para cada linha)
0: 1H = fsc / 228
PAL: modo PAL ou NTSC (inválido nos modos B5 e B6)
1: sistema PAL (50 Hz)
0: sistema NTSC (60 Hz)
EO: resolução vertical para modo entrelaçado (inválido nos modos B5 e B6)
1: resolução vertical dobrada em relação ao modo não entrelaçado
0: resolução vertical igual ao modo não entrelaçado
IL: seleção de modo entrelaçado
1: modo entrelaçado
0: modo não entrelaçado

7.10.2 - O REGISTRADOR DE CONTROLE

	b7	b6	b5	b4	b3	b2	b1	b0
R#8	DISP	SPD	YSE	VMTE	VWM	DMAE	VSL1	VSL0

DISP: habilita/desabilita apresentação de tela
1: apresentação de tela normal
0: a tela inteira apresenta a cor de fundo
SPD: habilita/desabilita apresentação dos sprites ou cursores
1: sprites e cursores não são apresentados
0: sprites e cursores são apresentados normalmente
YSE: habilita/desabilita sinal $\overline{YS}$ (superimpose)
1: sinal $\overline{YS}$ habilitado
0: sinal $\overline{YS}$ desabilitado

VMTE: controle do barramento da VRAM para digitalização
1: transferência para escrita (dummy) é executada durante o intervalo de retraço horizontal (barramento de dados da VRAM no modo entrada)
0: transferência para leitura é executada durante o intervalo de retraço horizontal (barramento de dados da VRAM no modo saída)

VWM: controle de escrita na VRAM para digitalização
1: escrita é executada durante o intervalo de retraço horizontal
0: escrita de dados desabilitada

DMAE: habilita/desabilita sinal $\overline{\text{DREQ}}$ (requisição de dados)
1: o sinal é sincronizado com o bit TR para comandos do VDP
0: sinal desabilitado

VSL1 e VSL0: configuração da VRAM
00: 64K x 4 bits, 4 unidades (128K total)
01: 128K x 8 bits, 2 unidades (256K total)
10: 256K x 4 bits, 4 unidades (512K total)

7.10.3 - CONTROLE DE INTERRUPÇÃO

	b7	b6	b5	b4	b3	b2	b1	b0
R#9	0	0	0	0	0	IECE	IEH	IEV
R#10	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
R#11	IEHM	0	0	0	0	0	IL9	IL8
R#12	0	0	0	0	IX3	IX2	IX1	IX0

IECE: habilita/desabilita interrupção de fim de comando
1: gera interrupção quando o bit CE de P#6 for 1
0: não gera interrupção de acordo com o bit CE de P#6

IEV: habilita/desabilita interrupção de quadro
1: interrupção de quadro ativa
0: interrupção de quadro desabilitada

IEH: habilita/desabilita interrupção de linha (IL0~9, IX0~3, IEHM)
1: interrupção de linha ativa
0: interrupção de linha desabilitada

IL0~9: número de linha que vai gerar a interrupção

IX0~3: posição horizontal que vai gerar a interrupção (especificada em incrementos de 64 pontos)

IEHM: seleção de linha para interrupção
1: interrupção gerada em em todas as linhas (IL0~9 ignorados)
0: interrupção de linha de acordo com IL0~9.

7.10.4 - ESPECIFICAÇÃO DA COR DE FUNDO

A cor de fundo é a cor que é apresentada quando for usada a cor transparente na pintura da tela. Quando a apresentação de tela for desabilitada, fica inteiramente com a cor de fundo.

	b7	b6	b5	b4	b3	b2	b1	b0
R#15	0	0	BDC5	BDC4	BDC3	BDC2	BDC1	BDC0

BDC5~0: número da cor de fundo na paleta (0 ~ 63)

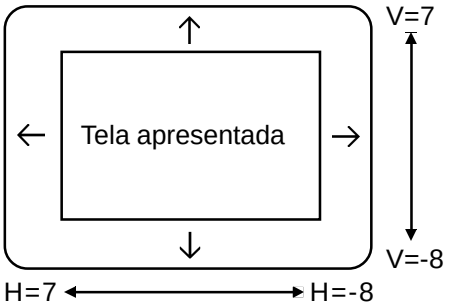
7.10.5 - AJUSTE DE TELA

O registrador R#16 é usado para ajustar a localização da tela. Corresponde à instrução SET ADJUST do BASIC.

	b7	b6	b5	b4	b3	b2	b1	b0
R#16	ADJV3	ADJV2	ADJV1	ADJV0	ADJH3	ADJH2	ADJH1	ADJH0

ADJH=7 ..... ADJH=1, ADJH=0, ADJH=15 ..... ADJH=8
(esquerda) (centro) (direita)

ADJV=8 ..... ADJV=15, ADJV=0, ADJV=1 ..... ADJV=7
(abaixo) (centro) (acima)



Para os modos P1 e B1, o deslocamento é feito em unidades de 1 ponto de tela; para os modos P2, B2 e B3, em unidades de 2 pontos de tela e para os modos B4, B5 e B6, em unidades de 4 pontos de tela.

Capítulo 6
GERADORES DE ÁUDIO

Os micros MSX têm várias opções para a geração de sons, incluindo desde geradores de AM simples até digitalizadores sofisticados. 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)

1 - O PSG

PSG significa *"Programmable Sound Generator"*, ou seja, *Gerador 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.

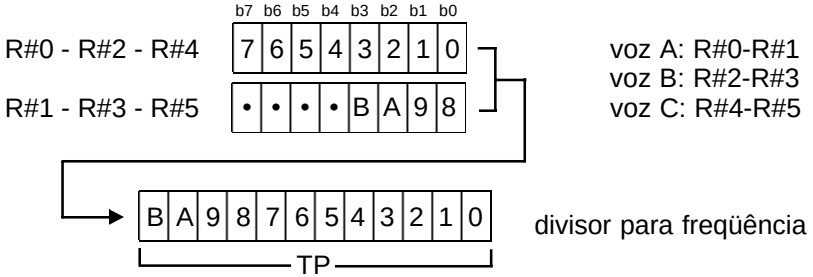
R#0 R#1	a a a a a a a a • • • • a a a a	freqüência da voz A
R#2 R#3	b b b b b b b b • • • • b b b b	freqüência da voz B
R#4 R#5	c c c c c c c c • • • • c c c c	freqüência da voz C
R#6	• • • r r r r r	freqüência do ruído branco
R#7	i o r r r t t t	habilita / desabilita sons
R#8 R#9 R#10	• • • m v v v v • • • m v v v v • • • m v v v v	volume da voz A volume da voz B volume da voz C
R#11 R#12	f f f f f f f f f f f f f f f f	freqüência da envoltória
R#13	• • • • e e e e	forma da envoltória
R#14 R#15	i i i i i i i i o o o o o o o o	porta A de I/O porta B de I/O

1.1 - DESCRIÇÃO DOS REGISTRADORES

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.

1.1.1 - ESPECIFICAÇÃO DA FREQÜÊNCIA

A freqüência central usada pelo PSG para comandar o divisor de freqüências é de 111860,78 Hz. Assim, para obter a freqüência 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.

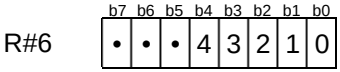


Os valores de cada registro TP para as 8 oitavas dos três geradores de tom com a nota LÁ central de 440 Hz estão listados abaixo.

Cifrado		1	2	3	4	5	6	7	8
Dó	C	D5D	6AF	357	1AC	0D6	06B	035	01B
	C#	C9C	64E	327	194	0CA	085	032	019
Ré	D	BE7	5F4	2FA	17D	0BE	05F	030	018
	D#	B3C	59E	2CF	168	0B4	05A	02D	016
Mi	E	A9B	54E	2A7	153	0AA	055	02A	015
Fá	F	A02	501	281	140	0A0	050	028	014
	F#	973	4BA	25D	12E	097	04C	026	013
Sol	G	8EB	476	23B	11D	08F	047	024	012
	G#	86B	436	21B	10D	087	043	022	011
Lá	A	7F2	3F9	1FD	0FE	07F	040	020	010
	A#	780	3C0	1E0	0F0	078	03C	01E	00F
Si	B	714	38A	1C5	0E3	071	039	01C	00E

1.1.2 - GERADOR DE RUÍDO BRANCO

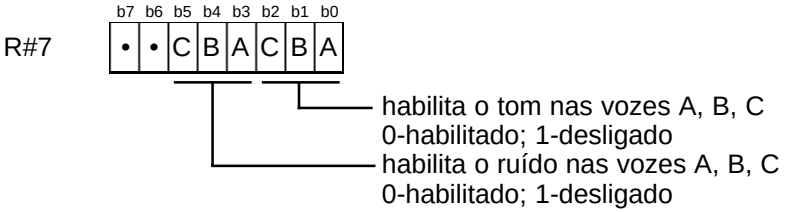
O gerador de ruído branco (chiado) é útil para gerar sons de explosões e outros. O PSG gera o chiado através de uma das três vozes de tom e sua freqüência é especificada no registrador R#6.



A freqüência 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 freqüência do ruído varia de 3,6 KHz a 111,8 KHz (divisão de 11186,78 pelo valor contido em R#6).

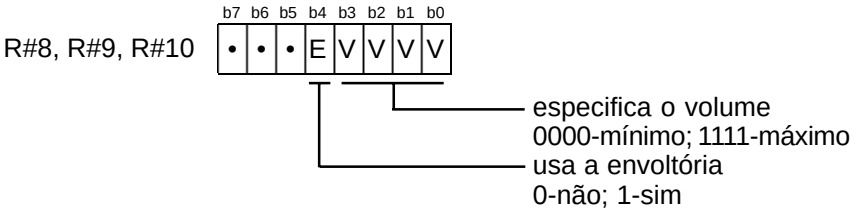
1.1.3 - MIXANDO OS SONS

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.



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).

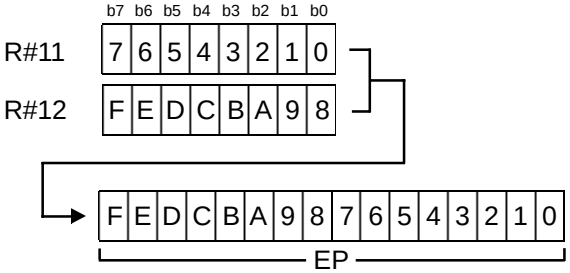


Quando os bit E for 0, o volume é especificado pelos bits V. Quando o bit E for 1, o volume é controlado pelo gerador de envoltória e os bits V são solenemente ignorados.

1.1.5 - FREQÜÊNCIA DA ENVOLTÓRIA

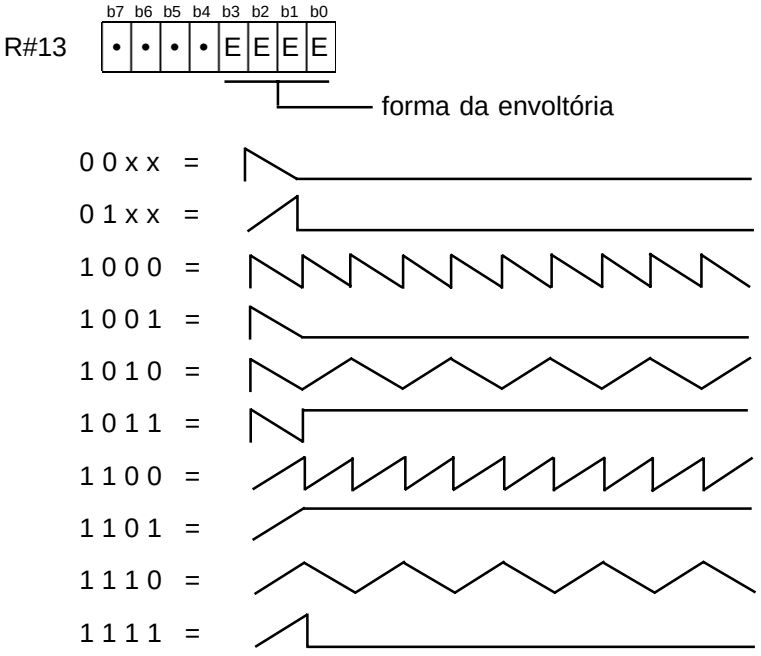
Os registradores R#11 e R#12 são usados como divisor de freqüência para o gerador de envoltória. Todos os bits são válidos. A freqüência central usada pelo gerador de envoltória para comandar o divisor de freqüências é

de 6983,3 Hz; portanto, a freqüência da envoltória pode variar de 6983,3 Hz a 0,107 Hz.



1.1.6 - FORMA DA ENVOLTÓRIA

A forma da envoltória é especificada nos quatro primeiros bits de R#13, conforme ilustração abaixo.



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 do BIOS, evitando assim problemas de sincronização.

As rotinas do BIOS destinadas ao acesso ao PSG são as seguintes:

WRTPSG (0093H/Main)
Função: escreve um byte de dados em um registrador do PSG
Entrada: A - número do registrador do PSG a ser escrito
E - byte de dados a ser escrito
Saída: nenhuma

RDPSG (0096H/Main)
Função: lê um byte de dados de um registrador do PSG
Entrada: A - número do registrador do PSG a ser lido
Saída: A - byte de dados lido

É possível o acesso direto também. Existem três portas destinadas ao acesso ao PSG. Essas portas são:

Porta A0H: porta de endereço
Porta A1H: porta de escrita de dados
Porta A2H: porta de leitura de dados

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).

2 - GERAÇÃO DE SONS PELA PORTA 1-bit

O padrão MSX dispõe de outro método padrão para a geração de sons. Estes são gerados ligando e desligando repetidamente uma porta de I/O de 1 bit. Esse bit é o responsável pelo “click” das teclas. O acesso a esse bit é feito através de uma rotina do BIOS:

CHGSND (0135H/Main)
Entrada: A - 0, desliga o bit; outro valor, liga o bit
Saída: nenhuma

Aqui também é possível o acesso direto, tomando-se cuidado com a sincronização. Ela é acessada pelo bit b7 da porta C da PPI (porta de I/O AAH). Os outros bits dessa porta não devem ser modificados.

AAH -

b7	b6	b5	b4	b3	b2	b1	b0
C	•	•	•	•	•	•	•

Porta C da PPI

Ligando e desligando repetidamente esse bit, podem ser gerados diversos efeitos sonoros, inclusive reprodução grosseira da voz humana.

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.

3.1 - DESCRIÇÃO DA SÍNTESE FM

O FM-OPLL usa faz uso de harmônicas geradas por modulação para sintetizar sons musicais, chamado por "síntese FM". Esse tipo de síntese é expressado por 3 parâmetros:

1. $F = A \sin(\omega c t + I \sin \omega m t)$

Onde A é a amplitude de saída, I é o índice de modulação, ωc e ωm as freqüências angulares da portadora e da moduladora, respectivamente. A eqüação 1 pode ser expressada alternativamente como abaixo:

2. $A [J_0 (I) \sin \omega c t + J_1 (I) (\sin(\omega c + \omega m)t - \sin(\omega c - \omega m)t) + J_2 (I) (\sin(\omega c + 2\omega m)t + \sin(\omega c - 2\omega m)t + ...]$

Onde Jn (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 médodo de "feedback" ou realimentação resolve o problema. Ele é caracterizado pela seguinte equação:

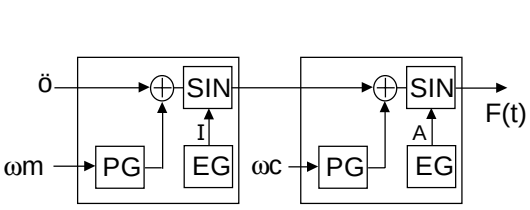
3. $F = A \sin(\omega c t + \beta F)$

Onde β 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 ωt;
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 da página seguinte, em células que combinam as funções dos três blocos. Só é necessário definir os parâmetros da freqüência e da envoltória.



a. F(t) = A sin (ωct + I sin ωmt) b. F(t) = A sin (ωt + βF(t))

3.2 - MAPA DOS REGISTRADORES DO OPLL

Reg.	7	6	5	4	3	2	1	0	Descrição resumida
$00H	AM	VIB	EGT	KSR	Múltiplo				(m)
$01H	AM	VIB	EGT	KSR	Múltiplo				(c)
$02H	KSL(m)		Nível total de modul. (c)						Registradores para definição do instrumento do usuário
$03H	KSL(c)		•	DC	DM	Feedback			
$04H	Attack (m)				Decay (m)				
$05H	Attack (c)				Decay (c)				
$06H	Sustain (m)				Release(m)				(m) = onda moduladora
$07H	Sustain (c)				Release(c)				(c) = onda portadora
$0EH	•	•	R	BD	SD	TOM	TCY	HH	Controle de peças de bateria
$0FH	Modo do OPLL								Teste do OPLL
$10H $18H	Freqüência LSB (8 bits)								Registradores usados para a seleção de freqüências do gerador de tons
$20H $28H	•	•	Sustain	Key	Oitava			Freqüência	Freqüência MSB 1 bit Oitava Key on/off Sustain on/off
$30H $38H	Instrumentos				Volume				Registradores usados para seleção de volume e de instrumentos

Mapa dos registradores para o modo bateria ($0EH, b5=1):

$36H	•	•	•	•	BD-vol	Registradores de volume das peças da bateria
$37H	HH-vol				SD-vol	
$38H	TOM-vol				TCY-vol	

Reg.	Bit	Conteúdo
00: (m) 01: (c)	b7 b6 b5 b4 b0~b3	Liga/desliga a modulação de amplitude (trêmolo) Liga/desliga a modulação de freqüência (vibrato) 0-tom percussivo; 1-tom constante Razão da “Key Scale” Controle multi-sample e harmônicos
02, 03	b6~b7	Nível da “Key Scale” - $02(m); $03(c)
02:	b0~b5	Nível total de modulação
03: (c)	b4(c) b3(m) b0~b2	Distorção da onda portadora Distorção da onda moduladora Constante de realimentação FM (m)
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
10~18	b0~b7	Freqüência (LSB 8 bits)
20~28	b5 b4 b1~b3 b0	Liga/desliga o “sustain” Liga/desliga a “key” Seleciona a oitava Freqüência (MSB 1 bit)
30~38	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	

3.3 - DESCRIÇÃO DOS REGISTRADORES

Essa seção descreve detalhadamente os diversos registradores do YM2413 e seu funcionamento.

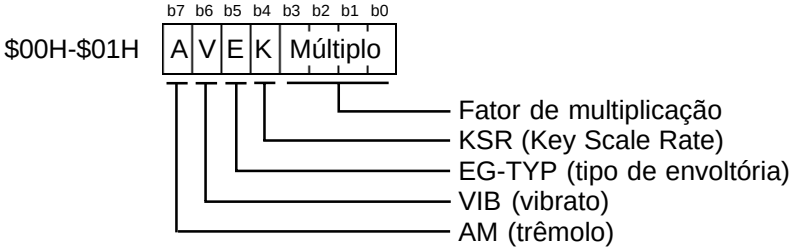
3.3.1 - REGISTRADOR DE TESTE

O registrador $0FH é o registrador de teste. Ele é estabelecido somente para teste do OPLL. Normalmente seu valor é 0.

3.3.2 - REGISTRADORES PARA DEFINIÇÃO DE INSTRUMENTO

• AM/VIB/EG-TYP/KSR/MÚLTIPLO ($00H e $01H)

Esses registradores especificam o fator de multiplicação para as freqüências do modulador ($00H) e da portadora ($01H) com seus repectivos componentes, como a envoltória e demais.



MÚLTIPLO (b0~b3)

As freqüências 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:	½	1	2	3	4	5	6	7	8	9	10	10	12	12	15	15

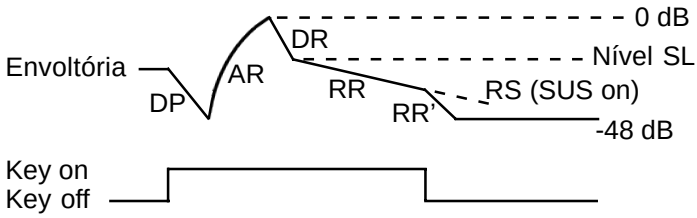
KSR (b4)

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 seu níveis alterados. Se KSR for igual a 0, o nível será o mesmo para todas as freqüências. Se KSR for igual a 1, haverá atenuação do som conforme a freqüência; quanto mais alta a freqüência gerada, maior será o nível de atenuação. Esse nível é especificado nos bits KSL.

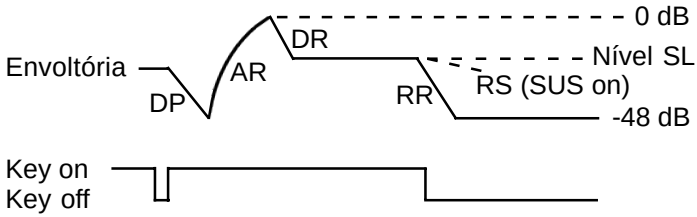
EG-TYP (b5)

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 freqüência do vibrato é de 6,4 Hz.

AM (b7)

Flag usada para ativar ou desativar a modulação de amplitude ou trêmolo. Se for 1, a modulação de amplitude estará ativa e se for 0 estará desligada. A freqüência para a modulação de amplitude é de 3,7 Hz.

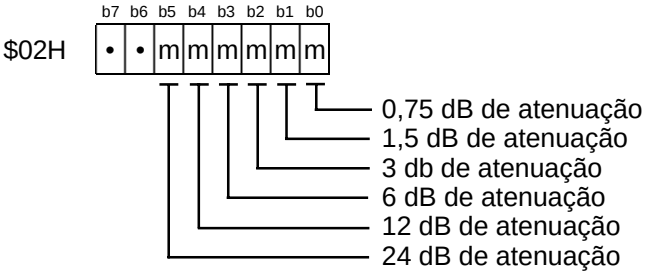
• KSL/NÍVEL TOTAL/DISTORÇÃO/NÍVEL DE REALIMENTAÇÃO ($02H,$03H)

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		Nível total					
$03H	KSL		•	DC	DM	Realimentação		

NÍVEL TOTAL (b0~b5)

Esse valor permite controlar o nível de modulação através da atenuação do mesmo (envoltória). Com o valor 000000, não haverá atenuação e a modulação será máxima. Já com o valor 111111, 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)

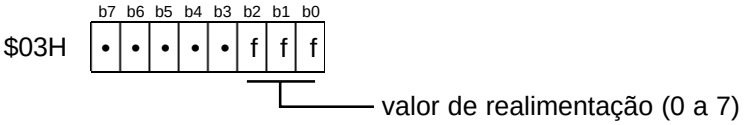
Quando esse bit for igual a 1, a onda moduladora é retificada para meia onda.

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:	0	1	2	3	4	5	6	7
Nível de realimentação:	0	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	π	2π	4π

• 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” e/ou “decay”. A variação de tempo obedece, aproximadamente, a uma progressão geométrica.

	mínimo	máximo
Decay (0dB a 48dB)	1,27 ms	20926 ms
Decay (10% a 90%)	0,52 ms	8403 ms
Attack (0dB a 48dB)	0,14 ms[18]	1730 ms
Attack (10% a 90%)	0,10 ms[18]	1112 ms

	attack (AR)				decay (DR)				
$04H	b7	b6	b5	b4	b3	b2	b1	b0	onda moduladora
$05H	b7	b6	b5	b4	b3	b2	b1	b0	onda portadora

• 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”.

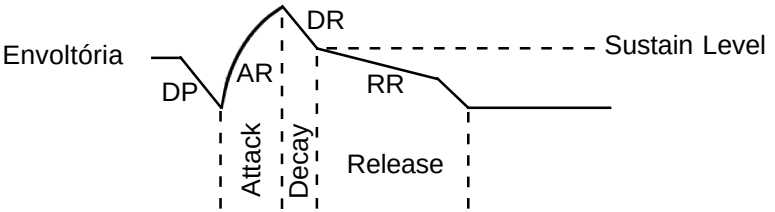
“Release rate” é a relação de desaparecimento do som após a “key off”. Para o tom percussivo, é expressada pela atenuação após o “sustain level”. Quanto maior o valor do registrador, menor será a duração do “release rate”.

	sustain (SL)				release (RR)				
$06H	24dB	12dB	6dB	3dB	b3	b2	b1	b0	moduladora
$07H	24dB	12dB	6dB	3dB	b3	b2	b1	b0	portadora

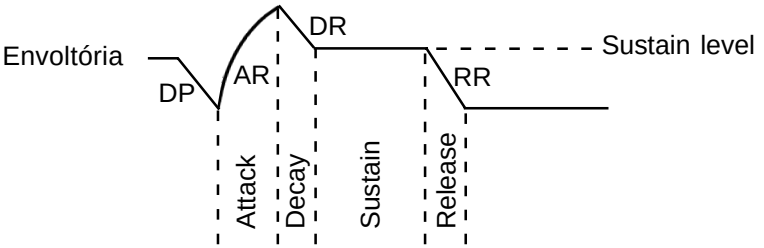
Na página seguinte, há uma ilustração dos valores de “attack”, “decay”, “sustain level” e “release rate” na forma de onda.

Nota 18: Tempo para AR=14. Para AR=15, o tempo será 0 ms.

Tom percussivo



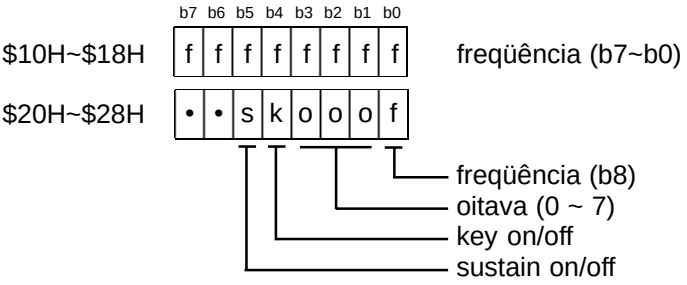
Tom Constante



3.3.3- REGISTRADORES DE SELEÇÃO

• OITAVA/FREQÜÊNCIA/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 freqüência de cada uma das 9 vozes que podem ser geradas pelo OPLL.



FREQÜÊNCIA ($1xH e bit 0 de $2xH)

Esses 9 bits definem um escala de freqüências 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		Freqüência	Decimal	$2xH,b0	$1xH
Dó	C#	277,2 Hz	181	0	10110101
Ré	D	293,7 Hz	192	0	11000000
	D#	311,1 Hz	204	0	11001100
Mi	E	329,6 Hz	216	0	11011000
Fá	F	349,2 Hz	229	0	11110010
	F#	370,0 Hz	242	0	11110010
Sol	G	392,0 Hz	257	1	00000001
	G#	415,3 Hz	272	1	00010000
Lá	A	440,0 Hz	288	1	00100000
	A#	466,2 Hz	305	1	00110001
Si	B	493,9 Hz	323	1	01000011
Dó	C	523,3 Hz	343	1	01010111

Os valores das freqüências 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 freqüência 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)

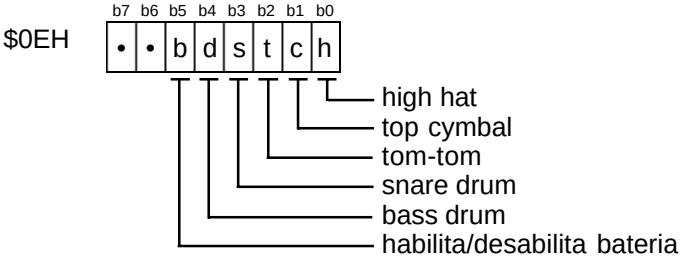
Esse bit deve ser setado em 1 para que o som de cada uma das nove vozes seja habilitado. Quando for 0, o som da voz respectiva estará desligado (key off).

SUSTAIN ($2xH, b5)

Quando esse bit estiver setado em 1, o valor de “release rate - RR” decairá gradativamente quando o bit “key” respectivo for desligado; caso contrário, o som será cortado abruptamente.

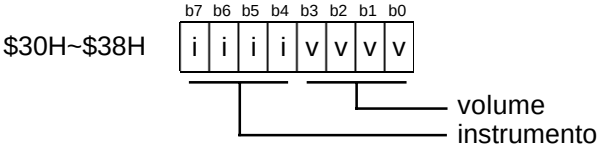
• CONTROLE DA BATERIA ($0EH)

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, de acordo com a tabela abaixo:

b0 = 3 dB b1 = 6 dB b2 = 12dB b3 = 24 dB

Os bits b7~b4 selecionam o instrumento, sendo que o valor 0000B seleciona o instrumento definido pelo usuário. Os 15 instrumentos possíveis 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
$36H	•	•	•	•	bass drum				Registradores
$37H	high hat				snare drum				de volume das peças
$38H	tom-tom				top cymbal				da bateria

3.4 - O 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 do FM-BIOS não podem ser mixados uns com os outros, já que o OPLL aceita a definição externa de apenas um instrumento. O 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 do FM-BIOS

	$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 - Pipe organ	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 1	dados do OPLL (11)							
15 - Harpsicode 2	01	10	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 - Sybthesizer	dados do OPLL (10)							
25 - Synthesizer percussion	01	03	0B	07	BA	D9	25	06
26 - Synthesizer rhytm	40	00	00	07	FA	D9	37	04
27 - Harm drum	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.

3.5 - O FM ESTÉREO

Embora não previsto oficialmente para o padrão MSX, o FM estéreo acabou sendo padronizado pelo mercado devido à 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.

3.6 - ACESSO AO OPLL

O acesso ao OPLL é feito diretamente por duas portas de I/O, a 7CH e a 7DH. A porta 7CH seleciona os registradores e a porta 7DH 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)	3,4 µS	12 ciclos T (3,58 MHz)
Escrita de dados (7DH)	23,5 µS	84 ciclos T (3,58 MHz)

Recomenda-se o uso de 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:

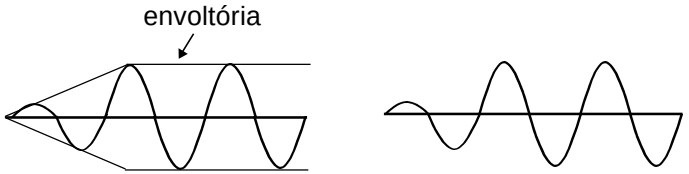
```
LD   A,REG    ;número do registrador em A
OUT  (07CH),A ;seleciona o registrador
NOP           ;pausa
```

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"[19] (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.

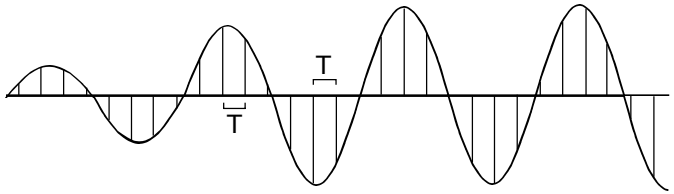
```
LD   A,REG    ;número do registrador em A
OUT  (07CH),A ;seleciona o registrador
NOP           ;pausa
LD   A,DADO   ;dado a ser escrito no registrador
OUT  (07DH),A ;escreve o dado no registrador
EX   (SP),HL  ;pausa[19]
EX   (SP),HL  ;pausa[19]
EX   (SP),HL  ;pausa[19]
EX   (SP),HL  ;pausa[19]
```

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 PSG, no caso uma onda senoidal, está ilustrado abaixo:



A forma de atuação do PCM para reproduzir essa mesma onda está ilustrada abaixo:



Nota 19: Ao ser usada em pausas, a instrução EX (SP),HL deve sempre vir em duplas, para evitar que o conteúdo da pilha seja alterado.

A cada período T, o PCM faz uma coleta do nível de som. O período T é fixo e a freqüência 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 freqüência de amostragem (sampling rate), melhor a qualidade do som reproduzido.

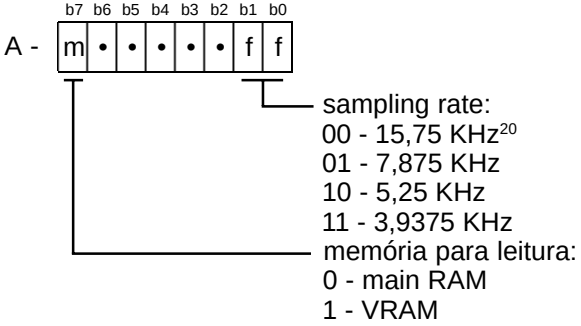
Outra característica do PCM é a resolução. No caso do MSX, a resolução é de 8 bits, sou seja, cada coleta tem 8 bits e por isso a amplitude da onda sonora coletada tem uma variação de 256 níveis. Cada amostra ocupa, portanto, um byte de memória.

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 armazenameto 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.

4.1 - ACESSO AO PCM

O PCM pode ser acessado tanto pelo BIOS quanto diretamente. No caso de acesso pelo BIOS, há duas rotinas disponíveis:

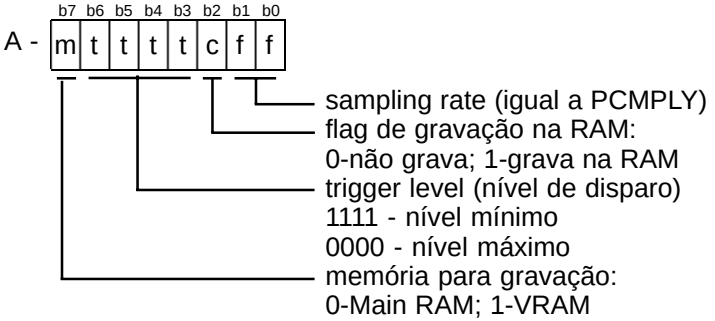
PCMPLY (0186H/Main)
Função: reproduzir o som pelo PCM
Entrada: HL - endereço de início para leitura
BC - quantidade de bytes a reproduzir



Saída: Flag CY: 0 - parou
1 - parou porque tem erro
A: 0 - término normal de execução
1 - tem erro na freqüência
2 - foi pressionada a tecla STOP

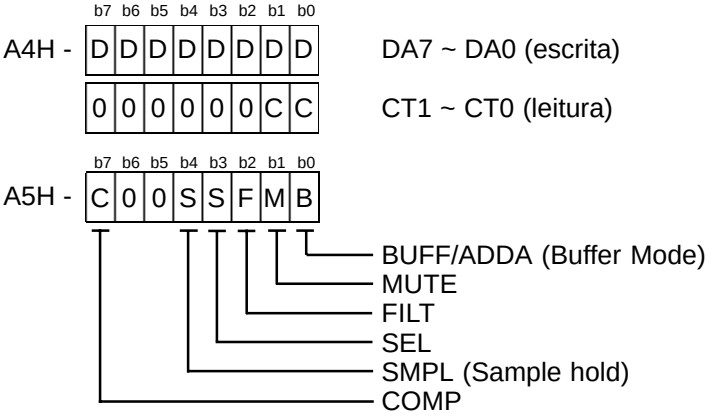
Nota 20: O sampling rate de 15,75 KHz só pode ser usado no modo R800 DRAM.

PCMREC (0189H/Main)
Função: digitalizar os sons através do PCM
Entrada: igual a PCMPLY, exceto para o registrador A:



Saída: mesma de PCMPLY.

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 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.

SMPL (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 μS, o contador é incrementado. Como esse período corresponde à freqüência 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  (PMCNTL),A    ; Modo A/D
          DI
          XOR  A
          OUT  (SYSTML),A
REC1:     IN   A,(SYSTML)
          CP   E
          JR   C,REC1
          XOR  A
          OUT  (SYSTML),A
          PUSH BC
          LD   A,00011100B
          OUT  (PMCNTL),A    ; Segura dado
          LD   A,080H
          LD   C,PMSTAT
          OUT  (PMDAC),A     ; Leitura do bit 7
          DEFB 0EDH,070H     ; Instrução IN F,(C)
```

```
        JP   M,RECAD0
        AND  01111111B
RECAD0: OR   01000000B
        OUT  (PMDAC),A      ; Leitura do bit 6
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD1
        AND  10111111B
RECAD1: OR   00100000B
        OUT  (PMDAC),A      ; Leitura do bit 5
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD2
        AND  11011111B
RECAD2: OR   00010000B
        OUT  (PMDAC),A      ; Leitura do bit 4
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD3
        AND  11101111B
RECAD3: OR   00001000B
        OUT  (PMDAC),A      ; Leitura do bit 3
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD4
        AND  11110111B
RECAD4: OR   00000100B
        OUT  (PMDAC),A      ; Leitura do bit 2
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD5
        AND  11111011B
RECAD5: OR   00000010B
        OUT  (PMDAC),A      ; Leitura do bit 1
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD6
        AND  11111101B
RECAD6: OR   00000001B
        OUT  (PMDAC),A      ; Leitura do bit 1
        DEFB 0EDH,070H      ; Instrução IN F,(C)
        JP   M,RECAD7
        AND  11111110B
RECAD7: OR   00000000B
        LD   (HL),A         ; Armazena o byte lido
        LD   A,00001100B
        OUT  (PMCNTL),A
        POP  BC
        INC  HL
        DEC  BC
        LD   A,C
        OR   B
        JR   NZ,REC1
        LD   A,00000011B
        OUT  (PMCNTL),A     ; Modo D/A
        EI
        RET
```

Para reproduzir os sons, basta 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
        PMCNTL  EQU 0A5H
        PMSTAT  EQU 0A5H
        SYSTML  EQU 0E6H

PLAY:   LD   A,00000011B
        OUT  (PMCNTL),A      ; Modo D/A
        DI
        XOR A
        OUT (SYSTML),A
PLAY1:  IN   A,(SYSTML)
        CP   E
        JR   C,PLAY1
        XOR  A
        OUT  (SYSTML),A
        LD   A,(HL)          ; Lê byte de dados
        OUT  (PMDAC),A       ; Reprodução dos dados
        INC  HL
        DEC  BC
        LD   A,C
        OR   B
        JR   NZ,PLAY1
        EI
        RET
```

5 - O MSX-AUDIO

O MSX-Audio foi criado juntamente com o MSX2 em 1985 como periférico opcional padronizado, mas, provavelmente por seu preço mais 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 Pulsos por Adaptação Diferencial), que funciona de forma semelhante ao PCM, mas é mais elaborado, ocupando menos memória que o PCM simples. Além disso, também pode ser conectado a 256 Kbytes de memória externa, liberando a Main RAM e a CPU. Pode também ser conectado 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.

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 subseqüente. 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 (Xn);

2. O resultado é multiplicado por 256 para convertê-lo em 16 bits e é então comparado com o dado subseqüente (Xn + 1) para obter a diferença (dn);

3. Quando a diferença resultar num valor positivo, o bit L4 do dado ADPCM será 0; quando negativo, será 1. Ao mesmo tempo, o valor absoluto da diferença é calculado (|dn|).

4. Então, os três bits restantes são setados de acordo com a tabela abaixo (Δ_n é a variação relativa calculada).

L_4		L_3	L_2	L_1	Valor analisado
$dn \geq 0$	$dn < 0$				
0	1	0	0	0	$\|dn\| < \Delta_n / 4$
		0	0	1	$\Delta_n / 4 \leq \|dn\| < \Delta_n / 2$
		0	1	0	$\Delta_n / 2 \leq \|dn\| < \Delta_n * 3/4$
		0	1	1	$\Delta_n * 3/4 \leq \|dn\| < \Delta_n$
		1	0	0	$\Delta_n \leq \|dn\| < \Delta_n * 5/4$
		1	0	1	$\Delta_n * 5/4 \leq \|dn\| < \Delta_n * 3/2$
		1	1	0	$\Delta_n * 3/2 \leq \|dn\| < \Delta_n * 7/4$
		1	1	1	$\Delta_n * 7/4 \leq \|dn\|$

A conversão da amostra de som em dados ADPCM está completa.

5. Depois que o dado ADPCM for obtido, novo dado subseqüente (X_{n+2}) e nova variação (Δ_{n+1}) são obtidos.

$$X_{n+2} = (1 - 2 * L_4) * (L_3 + L_2 / 2 + L_1 / 4 + 1/8) * \Delta_n + X_{n+1}$$
$$\Delta_{n+1} = f(L_3, L_2, L_1) * \Delta_n$$

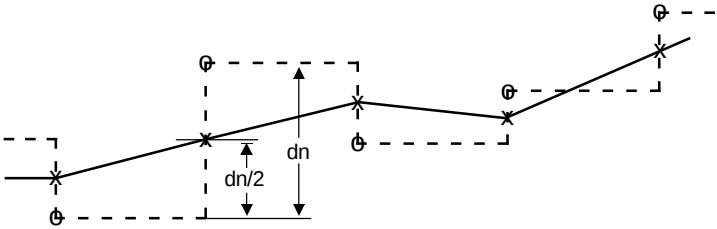
A tabela da página seguinte apresenta os fatores de multiplicação usados para ajustar a variação relativa (Δ_n).

L_3	L_2	L_1	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 o 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 freqüência. Depois é feita uma interpolação linear e os dados digitalizados são repetidos a uma freqüência 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

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 da página seguinte.

Reg	7	6	5	4	3	2	1	0	
$01H	TESTE								Registrador de teste
$02H $03H	TIMER 1 TIMER 2								Registradores de tempo
$04H	IRQ	T1M	T2M	EOS	BR	•	ST2	ST1	Registrador de flags
$05H $06H	Teclado (entrada) Teclado (saída)								Registradores para teclado externo
$07H $08H	STA CSM	REC SEL	MEM •	REP •	OFF SAM	• DAD	• 64K	RST ROM	Registradores de controle
$09H $0AH $0BH $0CH	Endereço inicial (low) Endereço inicial (high) Endereço final (low) Endereço final (high)								Endereços inicial e final para acesso pela CPU e ADPCM
$0DH $0EH	Sampling rate (low) Sampling rate (high)								Freqüência para o ADPCM
$0FH	Dados para o ADPCM								Registrador de dados
$10H $11H	Fator de interpolação (low) Fator de interpolação (high)								Fator de interpolação para o ADPCM
$12H	Volume do ADPCM								Volume do ADPCM
$15H	Dados do DAC (high)								Dados digitais para a conversão D/A
$16H	DAC-low		•	•	•	•	•	•	
$17H	•	•	•	•	•	SH2	SH1	SH0	
$18H $19H	• •	• •	• •	• •	Controle I/O Dados I/O				Controle das portas I/O de 4 bits
$1AH	Dados para o ADPCM								Registrador de dados
$20H $35H	AM	VIB	EGT	KSR	Múltiplo				Registradores para a definição de instrumentos para o gerador FM
$40H $55H	KSL		Nível total						
$60H $75H	Attack Rate (AR)				Decay Rate (DR)				
$80H $95H	Sustain Level (SL)				Release Rate (RR)				
$A0H $A8H	Freqüência (lsb)								Freqüência das 9 vozes do gerador FM (lsb)

$B0H $B8H	•	•	KEY	Oitava			Freq. msb 2 bits		Freq. msb 2 bits (FM) Oitava (FM) Key on/off (FM)
$BDH	AM	VIB	BAT	BD	SD	TOM	TCY	HH	Controle da bateria
$C0H $C8H	•	•	•	•	Feedback			CON	Fator de realimentação e conexão
STAT	INT	T1	T2	EOS	BUF	•	•	PCM	Registrador de status

5.3 - DESCRIÇÃO DOS REGISTRADORES

Essa seção descreve detalhadamente os diversos registradores do Y8950 e seu funcionamento.

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.

5.3.2 - REGISTRADORES DE TEMPO

Existem dois registradores de tempo: $02H, com resolução de 80 μs e $03H com resolução de 320 μs. 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 também controlar a modulação senoidal composta do gerador FM. Esses registradores podem ser carregados com qualquer valor entre 0 e 255. Quando a contagem exceder 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 de cada registrador pode ser calculado pelas seguintes fórmulas:

T0 (ms) = (256 - ($02H)) * 0,08
T1 (ms) = (256 - ($03H)) * 0,32

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 na página seguinte.

$04H - | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

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 mascarados 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.

5.3.4 - CONTROLE DE TECLADO, MEMÓRIA E ADPCM

	b7	b6	b5	b4	b3	b2	b1	b0
$05H -								
$06H -								

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.

	b7	b6	b5	b4	b3	b2	b1	b0
$07H -								

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) e realiza uma operação diferente, conforme descrito na página seguinte.

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.

$08H -

b7	b6	b5	b4	b3	b2	b1	b0

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 (NOTESEL): 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 freqüência. Quando for 1, o ponto de separação é especificado pelo bit MSB de freqüência (registradores $B0H a $B8H), confome 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	1	0	1	0	1	0	1	F-num 2º MSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Nº sep. teclado

b6 = 0

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
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	F-num 2º MSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Nº sep. teclado

b7 (CSM): Esse 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.

5.3.5 - ENDEREÇOS DE ACESSO

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).

Os registradores $09H e $0AH especificam o endereço inicial, como ilustrado abaixo.

64K DRAM

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

256K DRAM

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

64K ROM

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 -										
*	*	*	b4	b3	b2	b1	b0	b7 b6 b5 b4 b3 b2 b1 b0	0	0	0	0	0			

No caso de acesso à ROM, os bits b7~b5 de $0AH devem ser iguais aos mesmos bits de $0CH.

Já os registradores $0BH e $0CH especificam o endereço final para acesso à memória de áudio.

64K DRAM

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 -						- $0BH -										
b7	b6	b5	0	b3	b2	b1	b0	b7 b6 b5 b4 0 b2 b1 b0	1	1	1	1	1			

256K DRAM

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 -						- $0BH -										
b7	b6	b5	b4	b3	b2	b1	b0	b7 b6 b5 b4 b3 b2 b1 b0	1	1	1	1	1			

64K ROM

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 -						- $0BH -										
*	*	*	b4	b3	b2	b1	b0	b7 b6 b5 b4 b3 b2 b1 b0	1	1	1	1	1			

No caso de acesso à ROM, os bits b7~b5 de $0CH devem ser iguais aos mesmos bits de $0AH.

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.

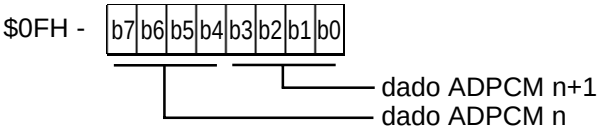
Taxa (KHz) = 3580 / NPRE

O valor NPRE é o valor contido em 11 bits dos registradores $0DH e $0EH, conforme ilustrado abaixo.

$0DH -	b7	b6	b5	b4	b3	b2	b1	b0	low
$0EH -	0	0	0	0	0	b10	b9	b8	high

16 KHz → $0DH = E0H e $0EH = 00H
1,8 KHz → $0DH = C4H e $0EH = 07H

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 freqüência 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. Para calcular o valor desses registradores, pode ser usada a seguinte fórmula simplificada:

Δ_n = 1310,72 * Taxa de amostragem (KHz)

$10H -	b7	b6	b5	b4	b3	b2	b1	b0	Δ_n (low)
$11H -	b15	b14	b13	b12	b11	b10	b9	b8	Δ_n (high)

16 KHz → $10H = ECH e $11H = 51H
1,8 KHz → $10H = 37H 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.

	b7	b6	b5	b4	b3	b2	b1	b0	
$12H -									volume do ADPCM

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:

$$SAÍDA = \frac{Vcc}{2} + \frac{Vcc}{4} * (-1 + F_9 + F_8 * 2^{-1} + \cdots\cdots + F_1 * 2^{-8} + F_0 * 2^{-9} + 2^{-10}) * 2^{-E}$$

$$E = S_2 * 2^2 + S_1 * 2^1 + S_0 * 2^0 \qquad @ \quad S_0 + S_1 + S_2 > 0$$

	b7	b6	b5	b4	b3	b2	b1	b0
$15H -	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2
$16H -	F_1	F_0	•	•	•	•	•	•
$17H -	•	•	•	•	•	S_2	S_1	S_0

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 pelo porta I/O 4 bits.

					b3	b2	b1	b0	
$18H -	•	•	•	•					controle de I/O
$19H -	•	•	•	•					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.

	b7	b6	b5	b4	b3	b2	b1	b0
$1AH -								

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.

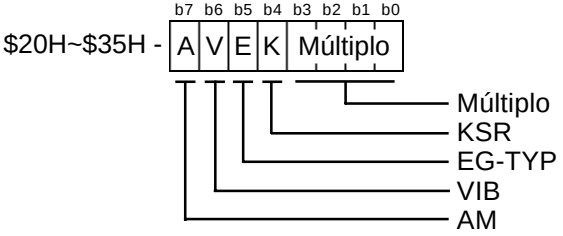
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	*1
1	2	3	1	2	3	4	5	6	4	5	6	7	8	9	7	8	9	*2
20	21	22	23	24	25	28	29	2A	2B	2C	2D	30	31	32	33	34	35	*3
A0	A1	A2	A0	A1	A2	A3	A4	A5	A3	A4	A5	A6	A7	A8	A6	A7	A8	*4

*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 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	12	15	15

Por exemplo, se F-num for ωf, o fator para a onda portadora for 1 e para a onda moduladora for 7, F(t) será calculado pela seguinte fórmula:

$$F(t) = E \sin(\omega f t + 1 \sin(7\omega f t))$$

KSR (b4)

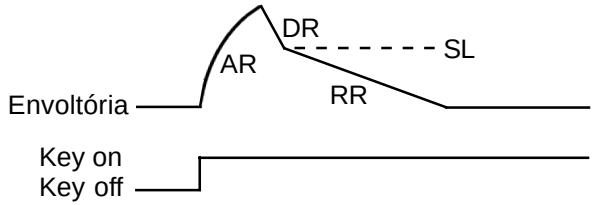
KSR significa “key scale rate”. Ele especifica as razões de “attack” e “decay”. A “key scale” é usada para fazer com que o som gerado pelo FM se aproxime do dos instrumentos musicais acústicos. O significado desse bit está ilustrado na tabela abaixo:

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

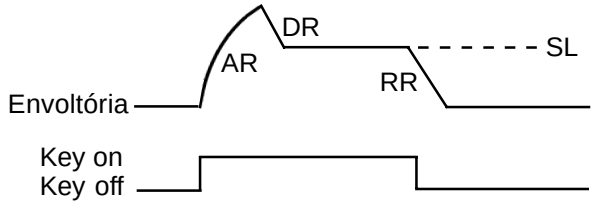
EG-TYP (b5)

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

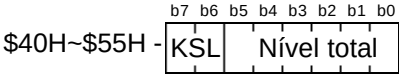
VIB (b6)

Esse bit liga ou desliga o vibrato. Se for 1, o vibrato estará ligado e ser for 0 estará desligado. A freqüência do vibrato é de 6,4 Hz. O grau do vibrato é setado pelo bit VIB-DEPHT do registrador $BDH.

AM (b7)

Esse bit liga ou desliga a modulação de amplitude (trêmolo). Se for 1, a modulação de amplitude estará ligada e se for 0 estará desligada. A freqüência do trêmolo é de 3,7 Hz. O grau do trêmolo é setado pelo bit AM-DEPTH do registrador $BDH.

• 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

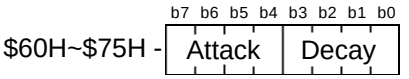
Para obter o valor correto, deve-se somar os graus quando o bit respectivo for 1. Assim, a resolução máxima de “decay” é de 0,75 dB e o nível de saída pode ser reduzido até 47,25 dB.

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 freqüência 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

• RELAÇÃO ATTACK/DECAY

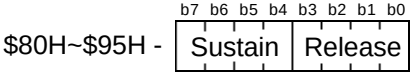


As razões de “attack” e “decay” são definidas pelos registradores $60H a $75H, sendo que os bits b0~b3 definem o nível de “decay” e os bits b4~b7 o de attack. Quanto maior o valor, menor o tempo de “attack” e

"decay". A variação de tempo obedece, aproximadamente, a uma progressão geométrica, e seus valores extremos estão listados abaixo.

	mínimo	máximo
Decay (0dB a 96dB)	2,40 ms	39280 ms
Decay (10% a 90%)	0,51 ms	8212 ms
Attack (0dB a 96dB)	0,20 ms[21]	2826 ms
Attack (10% a 90%)	0,11 ms[21]	1482 ms

• SUSTAIN LEVEL / RELEASE RATE



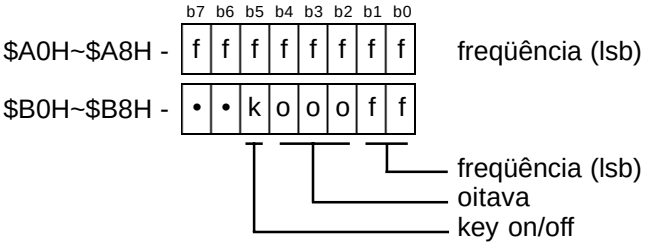
"Sustain level" especifica o nível em que o som ficará depois do "decay rate". Para o tom percussivo, especifica o ponto de troca do modo "decay" para o modo "release". Quanto maior o valor desse registrador, menor será o nível de "sustain". A tabela abaixo mostra os níveis possíveis de "sustain". Para obter o valor correto, para somar os graus quando o bit respectivo for 1.

b7	b6	b5	b4
24dB	12dB	6dB	3dB

"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"

• OITAVA/FREQÜÊNCIA

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.



Nota 21: Tempo para AR=14. Para AR=15, o tempo será 0 ms.

FREQÜÊNCIA ($AxH e bits 0-1 de $BxH)

Esses 10 bits definem uma escala de freqüências 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		Freqüência	Decimal	$BxH,b1-b0	$AxH
Dó	C#	277,2 Hz	363	0 1	01101011
Ré	D	293,7 Hz	385	0 1	10000001
	D#	311,1 Hz	408	0 1	10011000
Mi	E	329,6 Hz	432	0 1	10110000
Fá	F	349,2 Hz	458	0 1	11001010
	F#	370,0 Hz	485	0 1	11100101
Sol	G	392,0 Hz	514	1 0	00000010
	G#	415,3 Hz	544	1 0	00100000
Lá	A	440,0 Hz	577	1 0	01000001
	A#	466,2 Hz	611	1 0	01100011
Si	B	493,9 Hz	647	1 0	10000111
Dó	C	523,3 Hz	686	1 0	10101110

Os valores das freqüências 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 freqüência gerada dentro da escala musical. Os valores dos registradores também guardam entre si a mesma relação.

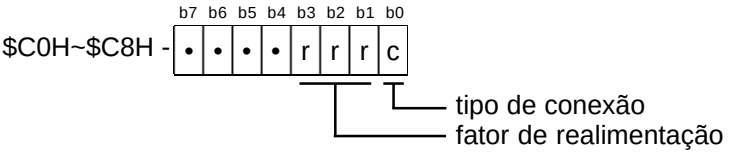
OITAVA ($BxH, b2~b4)

Esses três bits definem a oitava. Podem ser definidas até 8 oitavas, de 000 a 111, sendo que a quarta oitava é a de número 011.

KEY ($BxH, b5)

Esse bit deve ser setado em 1 para que o som de cada uma das nove vozes seja iniciado. Quando for 1, provocará a “key on” e quando for 0 provocará a “key off”.

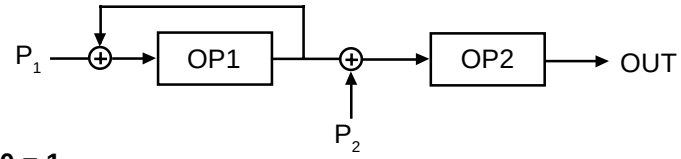
• REALIMENTAÇÃO/CONEXÃO



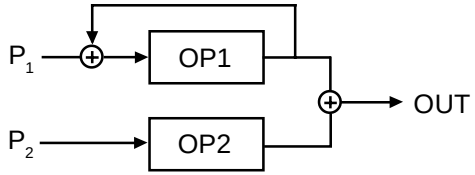
CONEXÃO ($CxH, b0)

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 senoidal composta (em paralelo). Esses modos estão ilustrados abaixo.

b0 = 0



b0 = 1

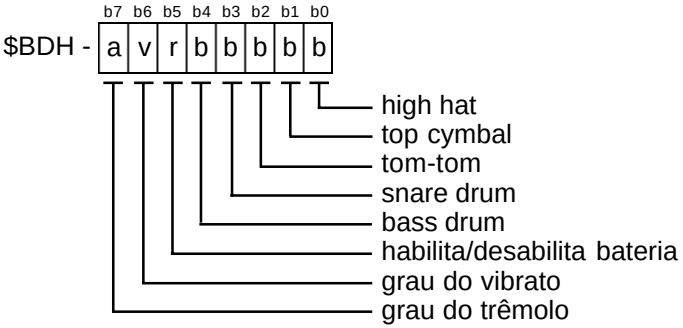


REALIMENTAÇÃO ($CxH, b1~b3)

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:	0	1	2	3	4	5	6	7
Fator de realimentação:	0	π/16	π/8	π/4	π/2	π	2π	4π

• 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.

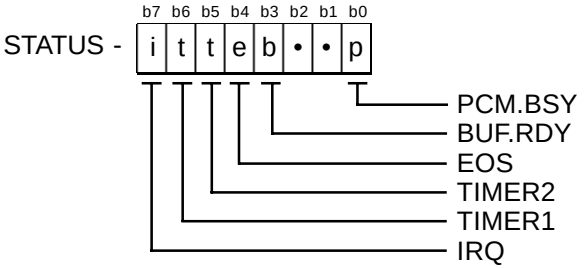
VIB-DEPTH (b6)

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 (trêmolo). Se for 0, o grau será de 1 dB e se for 1, de 4,8 dB.

5.3.8 - O REGISTRADOR DE STATUS



O MSX-Audio possui um registrador de status 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 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): Esse bit será setado em 1 quando um ou mais dos bits b3 a b6 for ou forem 1.

5.4 - PROTOCOLOS PARA ACESSO À MEMÓRIA DE ÁUDIO E ADPCM

Os protocolos recomendados para acesso à memória de áudio e ao ADPCM estão descritos em seguida.

5.4.1 - ANÁLISE DE SOM (MSX-Audio → CPU)

Reg.	Dado	R/W	Comentários
			• Inicialização
$04H	00H	W	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	
$0DH	C2H	W	"Sampling rate" = 8 KHz (NPRE = 450)
$0EH	01H	W	
			• Inicia a análise
$0FH	-	R	Inicia com a leitura do "dummy"
			• Análise
$0FH	-	R	Quando BUF.RDY for 1, $0FH é lido, o dado é armaze-
($04H	-	R)	nado e a flag resetada. Quando BUF.RDY for 0, espera.
			• Fim da análise
$07H	48H	W	A análise pelo ADPCM foi completada
$07H	00H	W	O registrador $07H é resetado

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 (Δ_n = 10486)
$11H	28H	W	
$12H	xxH	W	Especificar o volume de saída
			• Início da síntese
$0FH	xxH	W	Escreve dado para o ADPCM em $0FH

			• Síntese
$0FH	xxH	W	Quando BUF.RDY for 1, o dado de síntese é escrito em
($04H	80H	W)	$0FH e a flag resetada. Quando BUF.RDY for 0, esperar.
			• Final da síntese
$07H	00H	W	A síntese pelo ADPCM foi completada.

5.4.3 - ANÁLISE DE SOM (MSX-Audio → Memória de Áudio)

Reg.	Dado	R/W	Comentários
			• Inicialização
$04H	08H	W	Somente a máscara 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	
$0BH	xxH	W	Endereço final da memória de áudio
$0CH	xxH	W	
$0DH	E1H	W	"Sampling rate" = 16 KHz (NPRE = 225)
$0EH	00H	W	
			• Início da análise
$07H	E8H	W	Iniciar quando o bit b0 de $07H for 1.
			• Análise
			A flag EOS fica setada em 1 até o final da análise
			• Fim da análise
$07H	68H	W	A análise pelo ADPCM foi completada
$07H	00H	W	O registrador $07H é resetado

5.4.4 - SÍNTESE DE SOM (Memória de Áudio → MSX-Audio)

Reg.	Dado	R/W	Comentários
			• Inicialização
$04H	08H	W	Somente a máscara 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	xxH	W	Endereço final da memória de áudio
$0CH	xxH	W	
$10H	ECH	W	"Sampling rate" = 16 KHz (Δ_n = 20992)
$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

5.4.5 - ESCRITA NA RAM DE ÁUDIO (CPU → Memória de Áudio)

Reg.	Dado	R/W	Comentários
			• Inicialização
$04H	00H	W	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	xxH	W	Endereço final da memória de áudio
$0CH	xxH	W	
			• Escrita na memória
$0FH	xxH	W	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

5.4.6 - LEITURA DA RAM/ROM DE ÁUDIO (Memória de Áudio → CPU)

Reg.	Dado	R/W	Comentários
			• Inicialização
$04H	00H	W	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	xxH	W	Endereço final da memória de áudio
$0CH	xxH	W	

			• Leitura da memória
$0FH	-	R	Iniciar após ler o “dummy” duas vezes
$0FH	-	R	(Necessário para checar a flag)
$0FH	xxH	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

5.5 - ACESSO AO MSX-AUDIO

O acesso ao MSX-Audio é feito diretamente 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 status e a porta C1H lê ou escreve os dados nos outros registradores. Entretando, 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 status não há necessidade de especificar endereços. Para escrever um byte de dados nos registradores $00H a $1AH, deve-se proceder da forma ilustrada na página seguinte.

```
LD   A,REG    ;número 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"[22] (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    ;número 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[22]
EX   (SP),HL  ;pausa[22]
EX   (SP),HL  ;pausa[22]
EX   (SP),HL  ;pausa[22]
```

Os registradores $00H a $1AH e o registrador de status 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 status
NOP           ;pausa
```

6 - O SCC

O SCC (Sound Chip Custom) é um gerador de áudio criado pela softhouse japonesa Konami para equipar seus cartuchos de jogos megarom. Existem dois tipos de SCC, o SCC "simples" e o SCC+.

O SCC gera sons mediante gravação da forma de onda em sua memória interna e sua reprodução de dá tal qual o PCM. A diferença é que a

Nota 22: Ao ser usada em pausas, a instrução EX (SP),HL deve sempre vir em duplas, para evitar que o conteúdo da pilha seja alterado.

memória reservada para os sons é muito limitada, de apenas 32 bytes por voz, que devem ser repetidos continuamente durante a síntese.

6.1 - O SCC "simples"

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 divisão para cada voz.

1- 32 bytes -> forma de onda
2- 12 bits -> freqüência de reprodução
3- 4 bits -> volume de saída
4- 1 bit -> liga/desliga a voz

A distribuição na memória dos dados referentes a cada uma das cinco vozes está ilustrada na tabela abaixo.

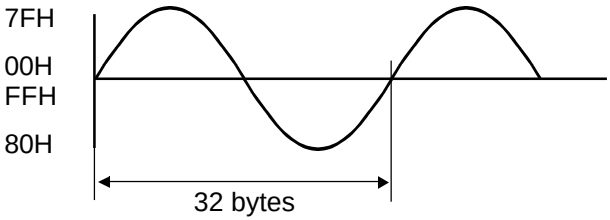
Endereço	Função
9800H a 981FH	Forma de onda da voz 1
9820H a 983FH	Forma de onda da voz 2
9840H a 985FH	Forma de onda da voz 3
9860H a 987FH	Forma de onda das vozes 4 e 5
9880H a 9881H	Freqüência da voz 1
9882H a 9883H	Freqüência da voz 2
9884H a 9885H	Freqüência da voz 3
9886H a 9887H	Freqüência da voz 4
9888H a 9889H	Freqüência 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	Byte de chaves liga/desliga
9890H a 989FH	Espelho de 9880H a 988FH
98A0H a 98DFH	Sem função
98E0H a 98FFH	Registrador de deformação

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 seqüência de bytes a serem armazenados para uma forma de onda senoidal está ilustrada na tabela da página seguinte.

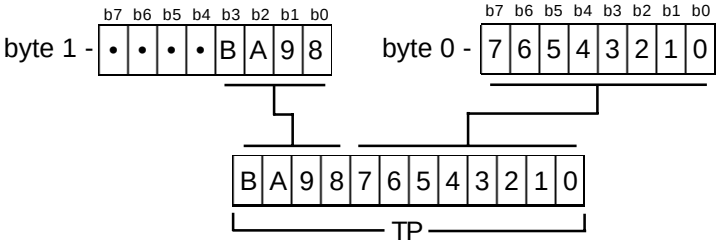
byte	valor	byte	valor	byte	valor	byte	valor
0	00H	8	7FH	16	00H	24	80H
1	18H	9	7DH	17	E7H	25	82H
2	30H	10	76H	18	CFH	26	89H
3	47H	11	6AH	19	B8H	27	95H
4	5AH	12	5AH	20	A5H	28	A5H
5	6AH	13	47H	21	95H	29	B8H
6	76H	14	30H	22	89H	30	CFH
7	7DH	15	18H	23	82H	31	E7H

A forma de onda resultante será a seguinte:



6.1.2 - AJUSTE DA FREQÜÊNCIA

A freqüência da onda é armazenada 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 freqüência. A fórmula usada para o cálculo da freqüência é a seguinte:

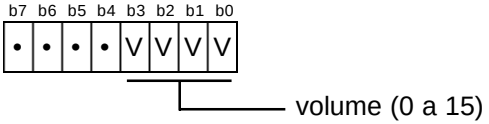
$$F_{tone} = \frac{F_{clock}}{32 * (TP + 1)}$$

Onde: F_{clock} é o clock presente no barramento interno (normalmente 3,579545 MHz);
F_{tone} é a freqüência gerada na respectiva voz;
TP é o valor armazenado nos registradores de freqüência.

Cabe aqui ressaltar que o ciclo da freqüência da 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.

6.1.3 - AJUSTE DO VOLUME

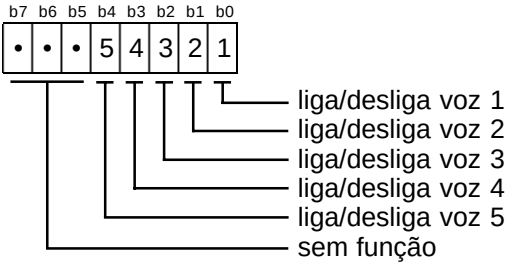
O volume também é armazenado no mesmo formato que para o PSG, conforme ilustrado abaixo.



Quando esse registrador for 0, o som será ausente; quando for 15, o volume será máximo.

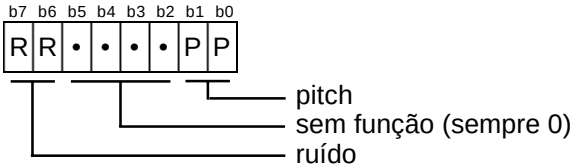
6.1.4 - REGISTRADOR DE CHAVES

O registrador de chaves é mapeado como ilustrado abaixo (se o bit respectivo for 0, desliga a voz; se for 1, liga a voz).



6.1.5 - REGISTRADOR DE DEFORMAÇÃO

Todos os endereços de 98E0H até 98FFH se referem ao mesmo registrador. Sua estrutura é a seguinte:



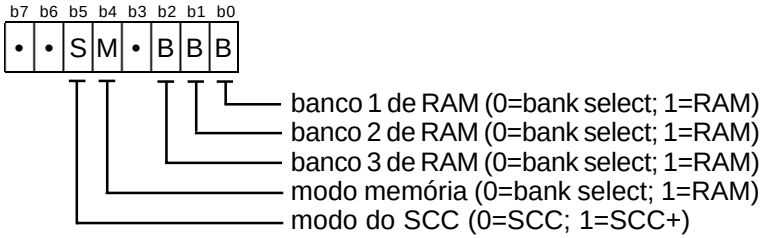
Os bits pitch alteram a freqüência de todas as vozes. Se forem

11B ou 10B, as freqüências serão multiplicadas por 16. Se for 01B, as freqüências serão multiplicadas por 256. Se forem 00B, as freqüências 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).

6.2 - O SCC+

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 00111111B 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	Freqüência da voz 1
9882H a 9883H	Freqüência da voz 2
9884H a 9885H	Freqüência da voz 3
9886H a 9887H	Freqüência da voz 4
9888H a 9889H	Freqüência 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	Byte 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 10000000B (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	Freqüência da voz 1
B8A2H a B8A3H	Freqüência da voz 2
B8A4H a B8A5H	Freqüência da voz 3
B8A6H a B8A7H	Freqüência da voz 4
B8A8H a B8A9H	Freqüência 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	Byte 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 "sim-

ples". 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+.

6.3 - 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.

7 - O OPL4

Em termos de reprodução sonora, o OPL4 é o mais perfeito, tendo qualidade de CD de áudio. Porém ele não é padrão do MSX. Apenas um cartucho de som o utilizou: o Moonsound.

O chip responsável é o YMF278B. 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).

7.1 - DESCRIÇÃO DOS REGISTRADORES PARA SÍNTESE WAVE

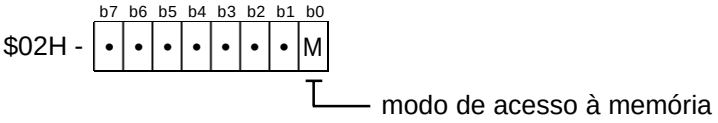
Reg	7	6	5	4	3	2	1	0	
$00H~$01H	Teste								Registradores de teste
$02H	ID dispositivo			Cabeçalho Wave			MemT	MemM	Funçoes especiais
$03H	•	•	Endereço de memória (A21~A16)						Endereço da memória de áudio externa
$04H	Endereço de memória (A15~A8)								
$05H	Endereço de memória (A7~A0)								
$06H	Dados da memoria								Registrador de dados
$07H	Não usado								

$08H $1FH	Número da tabela “wave” (n7~n0)							24 registradores com o número lsb (n7~n0) da tabela “wave” reproduzida
$20H $37H	F_number (f6~f0)						Tab. wave n8	Freqüência gerada (7 bits lsb) Número da tabela wave (n8)
$38H $4FH	Oitava o3~o0			Pseudo reverb.	F_number (f9-f7)			Oitava (-7 a +7) Pseudo reverberação Freqüência (3 bits msb)
$50H $67H	Nível total (l6~l0)						Nível Direto	
$68H $7FH	Key on	Damp	LFO RST	CH	Panpot			Funções diversas Balanceamente estéreo (Panpot)
$80H $97H	Não usado		LFO (s2~s0)		VIB (v2~v0)			Freqüência do trêmolo e do vibrato (LFO) Grau do vibrato (VIB)
$98H $AFH	Attack Rate				Decay Rate(1)			Registradores definidores da forma de onda
$B0H $C7H	Decay Level				Decay Rate (2)			
$C8H $DFH	Rate Correction				Release Rate			
$E0H $F7H	Não usado				AM (a2~a0)			Grau do trêmolo
$F8H	•	•	Mixagem (FM_R)		Mixagem (FM_L)			Controle de Mixagem
$F9H	•	•	Mixagem (PCM_R)		Mixagem (PCM_L)			

Os registradores para a síntese wave (Wave Table Synthesis) estão descritos na página seguinte. No final dessa seção, será descrito o modelo da síntese wave. Para a síntese FM, os registradores diferem um pouco. Serão descritos mais adiante.

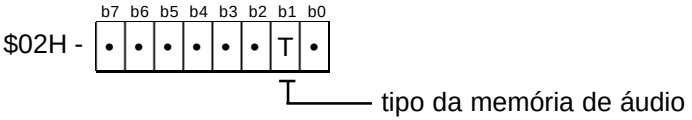
Os registradores $00H e $01H são usados apenas para teste do YMF278B. Devem sempre ser setados em 00H.

7.1.1 - ACESSO À MEMÓRIA DE ÁUDIO



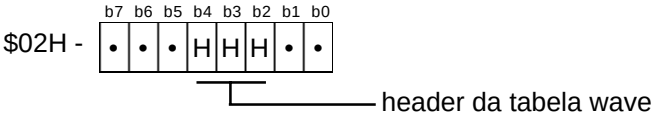
Quando esse bit for 0, há geração normal de sons. Quando for 1, a CPU poderá ler ou escrever dados na memória de áudio e não haverá geração de sons.

TIPO DE MEMÓRIA DE ÁUDIO



Quando esse bit for 0, somente ROM pode ser conectada. Quando for 1, podem ser conectadas SRAM mais ROM.

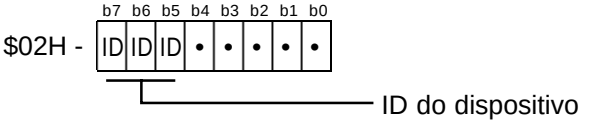
HEADER DA TABELA WAVE



O header deve ser setado a partir do endereço 000000H 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.

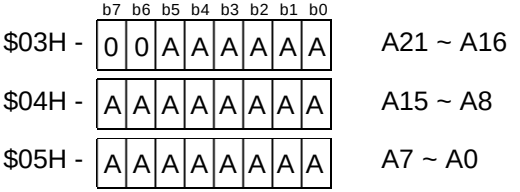
Header b4	Header b3	Header b2	Área de memória de áudio
0	0	0	waves número 0 a 511 setados em 000000H
0	0	1	waves número 384 a 511 setados em 080000H
0	1	0	waves número 384 a 511 setados em 100000H
0	1	1	waves número 384 a 511 setados em 180000H
1	0	0	waves número 384 a 511 setados em 200000H
1	0	1	waves número 384 a 511 setados em 280000H
1	1	0	waves número 384 a 511 setados em 300000H
1	1	1	waves número 384 a 511 setados em 380000H

ID DO DISPOSITIVO



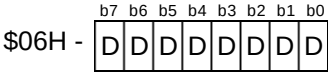
Registrador de indentificação do OPL4. Sempre retorna o valor 001B.

ENDEREÇOS DA MEMÓRIA DE ÁUDIO



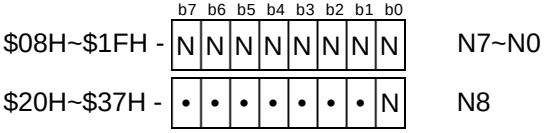
Esses registradores especificam o endereço da memória de áudio a ser escrito ou lido. O endereço é setado ao escrever o valor em $05H; por isso, é necessário sempre preencher os registradores a partir de $03H. Esses registradores são incrementados a cada acesso à memória de áudio.

REGISTRADOR DE DADOS DA MEMÓRIA



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.

7.1.2 - ACESSO AO MODO WAVE



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 header 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 header demora cerca de 300μs após a escrita de N8. O bit b1 do registrador de status indica quando um header está sendo carregado.

FREQÜÊNCIA E OITAVA

	b7	b6	b5	b4	b3	b2	b1	b0	
$20H~$37H -	f	f	f	f	f	f	f	f	F_number (f6~f0)
$38H~$4FH -	•	•	•	•	•	f	f	f	F_number (f9~f8)
$38H~$4FH -	o	o	o	o	•	•	•	•	Oitava (o3 ~ o0)

Esses registradores são usados para controlar o “pitch” 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:

$$F(¢) = 1200 * (\text{oitava} - 1) + 1200 * \log_2 \frac{1024 + F\_number}{1024}$$

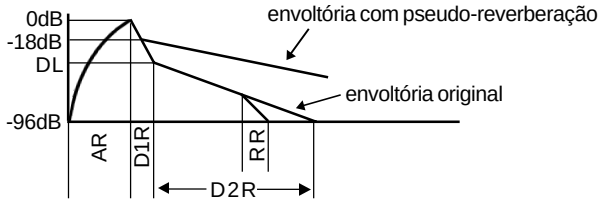
(1 oitava = 1200¢)

PSEUDO REVERBERAÇÃO

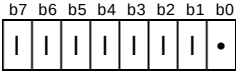
	b7	b6	b5	b4	b3	b2	b1	b0
$38H~$4FH -	•	•	•	•	r	•	•	•

pseudo-reverberação

Quando esse bit for 0, a pseudo-reverberação estará desligada; quando for 1, estará ligada.



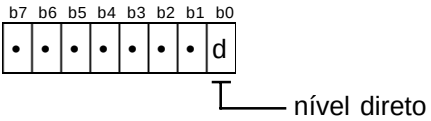
NÍVEL TOTAL



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.

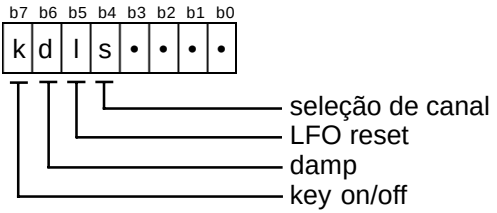
I6 = -24 dB	I2 = -1,5 dB
I5 = -12 dB	I1 = -0,75 dB
I4 = -6 dB	I0 = -0,375 dB
I3 = -3 dB	

NÍVEL DIRETO



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 do máximo para o mínimo será de 156,4 ms.

KEY ON, DAMP, LFO RESET E SELEÇÃO DE CANAL



O canal de saída pode ser selecionado de acordo com o bit b4 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 Freqüência) que é usado para os efeitos de vibrato e trêmolo. 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. 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”.

PANPOT

	b7	b6	b5	b4	b3	b2	b1	b0	
$68H~$7FH -	•	•	•	•	p	p	p	p	p3 ~ p0

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

LFO (Low Frequency Oscillator)

	b7	b6	b5	b4	b3	b2	b1	b0	
$80H~$97H -	•	•	s	s	s	•	•	•	s2 ~ s0

Esse registrador determina a freqüência do trêmolo e do vibrato, conforme a tabela abaixo:

Registrador:	0	1	2	3	4	5	6	7
Freqüência (Hz):	0,168	2,019	3,196	4,206	5,215	5,888	6,224	7,066

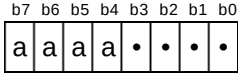
VIBRATO

	b7	b6	b5	b4	b3	b2	b1	b0	
$80H~$97H -	•	•	•	•	•	v	v	v	v2 ~ v0

Esse registrador determina o grau do vibrato, conforme a tabela abaixo.

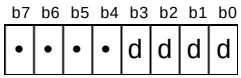
Registrador:	0	1	2	3	4	5	6	7
Grau (¢):	off	3,378	5,065	6,760	10,11	20,17	40,11	79,31

ATTACK RATE



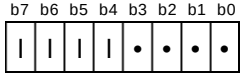
Esse registrador define o “attack rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

DECAY 1 RATE



Esse registrador define o “decay 1 rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

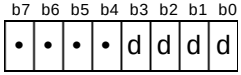
DECAY LEVEL



Esse registrador define 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. Quando todos os bits forem um, o “decay level” será levado a -93 dB, e não a -45 dB.

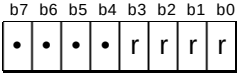
bit	dl3	dl2	dl1	dl0
nível (dB)	-24	-12	-6	-3

DECAY 2 RATE



Esse registrador define o “decay 2 rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

RELEASE RATE



Esse registrador define o “release rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

RATE CORRECTION

	b7	b6	b5	b4	b3	b2	b1	b0	
$C8H~$DFH -	c	c	c	c	•	•	•	•	c3 ~c0

Esse registrador determina o grau de correção da escala “rate”. Maiores detalhes podem ser vistos na seção “CALCULANDO OS “RATES””.

AM DEPTH (TRÊMOLO)

	b7	b6	b5	b4	b3	b2	b1	b0	
$E0H~$F7H -	•	•	•	•	•	a	a	a	am2 ~am0

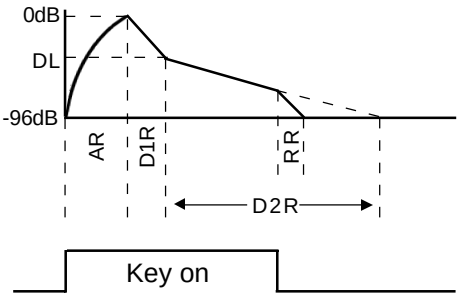
Esse registrador determina o grau do trêmolo, de acordo com a tabela abaixo:

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

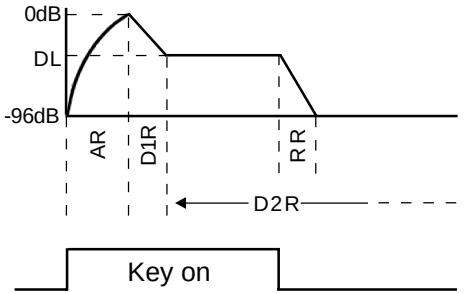
FORMA DE ONDA DA ENVOLTÓRIA

A envoltória pode ter duas formas distintas, designadas por tom percussivo e tom constante. Elas estão ilustradas abaixo e na página seguinte.

Tom percussivo (D2R > 0)



Tom constante (D2R = 0)



CALCULANDO OS "RATES"

O "rate" atual pode ser calculado pela seguinte fórmula:

RATE = (OCT + rate correction) * 2 + f9 + 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". Na tabela abaixo, estão descritos seus valores extremos. A variação de tempo obedece, aproximadamente, a uma progressão geométrica entre esses valores.

	mínimo	máximo
Attack (0dB a 96dB)	0,45 mS[23]	6223 mS[24]
Attack (10% a 90%)	0,23 mS[22]	3715 mS[24]
Decay (0dB a 96dB)	5,44 mS	89164 mS[24]
Decay (10% a 90%)	1,18 mS	19040 mS[24]
Release (0dB a 96dB)	5,44 mS	89164 mS[24]
Release (10% a 90%)	1,18 mS	19040 mS[24]

7.1.3 - FORMATO DA "WAVE TABLE SYNTHESIS"

A "Wave Table Synthesis" é uma maneira de reprodução do PCM que utiliza samples, geralmente curtos, reproduz a parte inicial e repete continuamente a parte seguinte. Para isso, dispõe de um header que é gravado na memória externa (de áudio). A estrutura do header 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) Endereço inicial (s21~s0)
01H	s15	s14	s13	s12	s11	s10	s9	s8	
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	Freqüência 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 (trêmolo)

Todos os headers ficam em seqüência na memória de aúdio, antes da área de dados (samples), de 0 a 383 ou de 384 a 511, ainda que não utilizados.

DATA BIT LENGTH

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
	d11	d10	d9	d8	d7	d6	d5	d4	+02H
8 bits	d7	d6	d5	d4	d3	d2	d1	d0	+00H

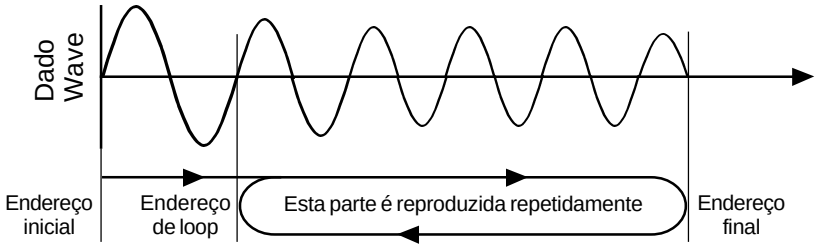
Nota 23: Valor para RATE=62. Para RATE=63, o tempo será nulo.
Nota 24: Valor para RATE=4. Para RATE=0~3, o tempo será infinito.

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

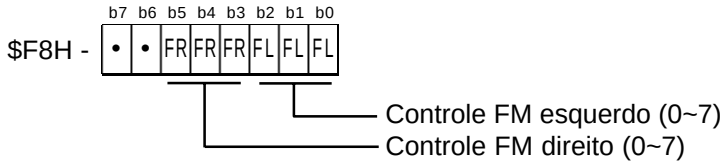
O endereço de loop especifica o endereço a partir do qual os dados serão repetidos. Ele é relativo ao endereço inicial.

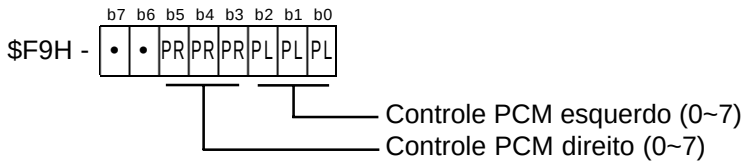


ENDEREÇO FINAL

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.

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 da página seguinte.

Registrador:	0	1	2	3	4	5	6	7
Nível mix (dB):	0	-3	-6	-9	-12	-15	-18	-∞

7.2 - DESCRIÇÃO DOS REGISTRADORES PARA O GERADOR FM

Register Array 0 (A1 = "1")									
Reg	7	6	5	4	3	2	1	0	
	Teste								Registradores de teste
$02H	Timer 1								Registradores de tempo
$03H	Timer 2								
$04H	RST	MT1	MT2	Não usado			ST2	ST1	Registrador de flags
$08H	•	NTS	Não usado						Configuração de teclado
$20H $35H	AM	VIB	EGT	KSR	Múltiplo				Registradores modificadores da forma de onda e freqüência definidas
$40H $55H	KSL		Nível Total						Nível da forma de onda e "damping"
$60H $75H	Attack Rate				Decay Rate				Registradores definidores da forma de onda
$80H $95H	Sustain Level				Release Rate				
$A0H $A8H	Freqüência (low)								Freqüência da onda gerada (8 bits lsb)
$B0H $B8H	Não usado		KON	Oitava			Freq. (high)		Freqüência (2 bits msb) Oitava Key on/off
$BDH	DAM	DVB	RYT	BD	SD	TOM	TC	HH	Controle da bateria
$C0H $C8H	CHD	CHC	CHB	CHA	Feedback			CNT	Tipo de conexão Realimentação Seleção de canal
$E0H $F5H	Não usado				Wave Select				Seleção de forma de onda

Register Array 1 (A1 = "H")									
Reg	7	6	5	4	3	2	1	0	
	Teste								Registradores de teste
$04H			Connection SEL						Seleção de modo 4-oper.
$05H	Não usado						NEW2	NEW	Registrador de expansão
$20H $35H	AM	VIB	EGT	KSR	Múltiplo				Registradores modificadores da forma de onda e freqüência definidas
$40H $55H	KSL		Nível Total						Nível da forma de onda e "damping"
$60H $75H	Attack Rate				Decay Rate				Registradores definidores da forma de onda
$80H $95H	Sustain Level				Release Rate				
$A0H $A8H	Freqüência (low)								Freqüência da onda gerada (8 bits lsb)
$B0H $B8H	Não usado		KON	Oitava			Freq. (high)		Freqüência (2 bits msb) Oitava Key on/off
$C0H $C8H	CHD	CHC	CHB	CHA	Feedback			CNT	Tipo de conexão Realimentação Seleção de canal
$E0H $F5H	Não usado					Wave Select			Seleção de forma de onda

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".

7.2.1 - TIMERS

	b7	b6	b5	b4	b3	b2	b1	b0	
$02H -	d7	d6	d5	d4	d3	d2	d1	d0	Timer 1 (80,8 μs) - Reg. Array 0
$03H -	d7	d6	d5	d4	d3	d2	d1	d0	Timer 2 (323,1 μs) - Reg. Array 0
$04H -				•	•	•	T2	T1	Controle dos timers - Reg. Array 0

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 são as seguintes:

t1(ms) = (256 - n1) * 0,08 (timer 1)
t2(ms) = (256 - n2) * 0,32 (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 setado em 1, os bits d5, d6 e d7 do registrador de status serão resetados. Depois, RS retornará automaticamente a 0.

7.2.2 - ACESSO AO MODO FM

SELEÇÃO DE MODO 4 OPERADORES

	b7	b6	b5	b4	b3	b2	b1	b0	
$04H -	•	•	6	5	4	3	2	1	(Register Array 1)

Quando qualquer dos bits 1~6 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

	b7	b6	b5	b4	b3	b2	b1	b0	
$05H -	•	•	•	•	•	•	N2	N	(Register Array 1)

Esses registradores permitem a expansão dos modos OPL2 e OPL3 para OPL4. Se os dois bits forem 0, será 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 setados em 1 para ativar o modo OPL4 antes da utilização do array 1 ou do PCM.

SELEÇÃO DE SEPARAÇÃO DE TECLADO

	b7	b6	b5	b4	b3	b2	b1	b0	
$08H -	•	S	•	•	•	•	•	•	(Register array 0)

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	0	1	0	1	0	1	0	1
Key Scale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

NTS = 0

Oitava	0		1		2		3		4		5		6		7	
F_num msb	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
F_num 2º	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Key Scale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

TRÊMOLO E VIBRATO

	b7	b6	b5	b4	b3	b2	b1	b0	
$20H~$35H -	T	V	•	•	•	•	•	•	(Register array 0 e 1)
$BDH -	DA	DV	•	•	•	•	•	•	(Register array 0)

Quando o bit b7 de $20H~$35H for 1, o trêmolo para a voz respectiva será ativado. A freqüência do trêmolo é de 3,7 Hz e o grau é determinado pelo bit DA (DA=0, 1db; DA=1, 4,8 dB).

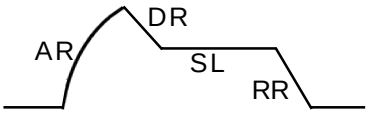
O bit b6 de $20H~$35H liga ou desliga o vibrato para a voz respectiva; se for 0, estará desligado; se for 1estará ligado. A freqüência do vibrato é de 6 Hz e seu grau é determinado pelo bit DV (DV=0, 7%, DV=1, 14%).

FORMA DE ONDA DA ENVOLTÓRIA

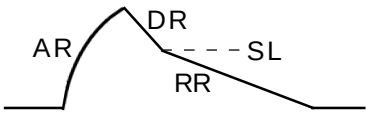
	b7	b6	b5	b4	b3	b2	b1	b0	
$20H~$35H -	•	•	E	•	•	•	•	•	(Register array 0 e 1)

Esse bit determina a forma de onda da envoltória, conforme abaixo.

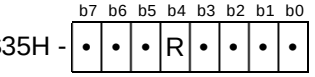
E=0 (tom constante)



E=1 (tom percussivo)



KSR (KEY SCALE RATE)



(Register array 0 e 1)

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	3
	KSR=1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

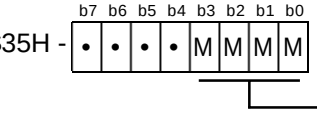
O RATE pode ser calculado pela sequinte expressão:

RATE = (Rate value) * 4 + 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. A variação de tempo obedece, aproximadamente, a uma progressão geométrica entre esses valores.

	mínimo	máximo
Attack (0dB a 96dB)	0,20 mS[25]	2826 mS[26]
Attack (10% a 90%)	0,11 mS[25]	1482 mS[26]
Decay (0dB a 96dB)	2,40 mS	39280 mS[26]
Decay (10% a 90%)	0,51 mS	8212 mS[26]
Release (0dB a 96dB)	2,40 mS	39280 mS[26]
Release (10% a 90%)	0,51 mS	8212 mS[26]

MÚLTIPLO



(Register array 0 e 1)

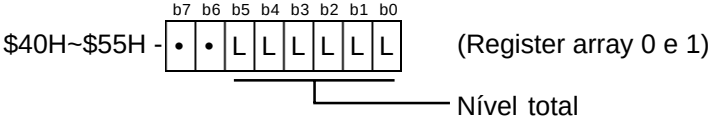
Esse registrador especifica o multiplicador para as freqüências especificadas por BLOCK e F_number. Os fatores de multiplicação estão mostrados na página seguinte.

Nota 25: Valor para RATE=59. Para RATE=60~63, o tempo será nulo.

Nota 26: Valor para RATE=4. Para RATE=0~3, o tempo será infinito.

Valor do registrador:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fator de multipicação:	½	1	2	3	4	5	6	7	8	9	10	10	12	12	15	15

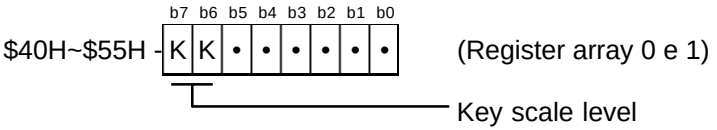
NÍVEL TOTAL



O nível total especifica o volume e a taxa de modulação da onda gerada. Seu valor corresponde à soma dos valores listados na tabela abaixo quando o bit respectivo for 1.

b5 = -24 dB	b2 = -3 dB
b4 = -12 dB	b1 = -1,5 dB
b3 = -6 dB	b0 = -0,75 dB

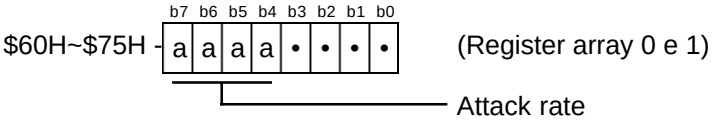
KSL (KEY SCALE LEVEL)



Esse 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.

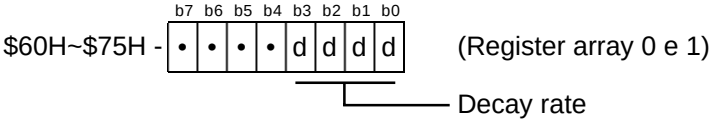
KSL	0	1	2	3
Atenuação	0 dB/oitava	3 dB/oitava	1,5 dB/oitava	6 dB/oitava

ATTACK RATE (AR)



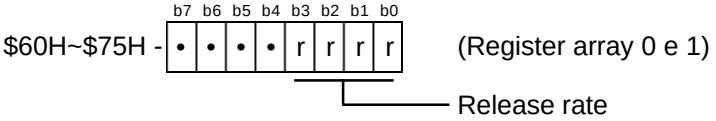
Esse registrador define o “attack rate”. Maiores detalhes podem ser vistos no item “KSR (KEY SCALE RATE)”.

DECAY RATE (DR)



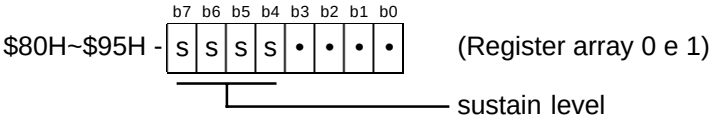
Esse registrador define o “decay rate”. Maiores detalhes podem ser vistos no item “KSR (KEY SCALE RATE)”.

RELEASE RATE (RR)



Esse registrador define o “release rate”. Maiores detalhes podem ser vistos no item “KSR (KEY SCALE RATE)”.

SUSTAIN LEVEL (SL)

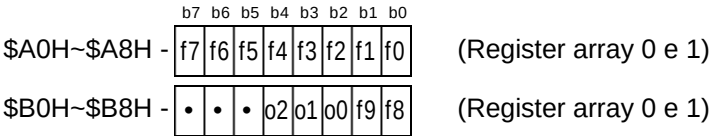


Esse registrador especifica o nível em que a envoltória permanece após a “decay rate”. Para tom percussivo, especifica o ponto de transição da “decay rate” para a “release rate”. Seu valor corresponde à soma dos valores listados abaixo, quando o bit respectivo for 1.

b7	b6	b5	b4
-24dB	-12dB	-6dB	-3 dB

Quando todos os bits forem 1, o valor de SL será setado em -93dB.

AJUSTE DA FREQÜÊNCIA



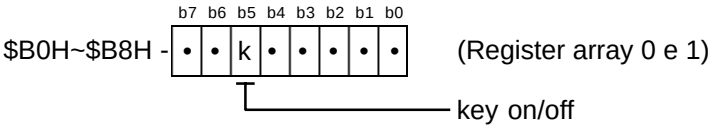
Os valores f0~f9 são chamados de F_number e especificam a freqüência gerada e os valores o0~o2 especificam a oitava. F_number pode

variar de 0 a 1023 e a oitava de 0 a 7. Os valores para a oitava de número 4 (na escala 0 a 7) estão listados abaixo.

Cifrado		Freqüência	Decimal	$BxH,b1-b0	$AxH
Dó	C	261,6 Hz	346	0 1	01011010
	C#	277,2 Hz	367	0 1	01101111
Ré	D	293,7 Hz	389	0 1	10000101
	D#	311,1 Hz	412	0 1	10011100
Mi	E	329,6 Hz	436	0 1	10110100
Fá	F	349,2 Hz	462	0 1	11001110
	F#	370,0 Hz	490	0 1	11101010
Sol	G	392,0 Hz	519	1 0	00000111
	G#	415,3 Hz	550	1 0	00100110
Lá	A	440,0 Hz	582	1 0	01000110
	A#	466,2 Hz	617	1 0	01101001
Si	B	493,9 Hz	654	1 0	10001110
Dó	C	523,3 Hz	693	1 0	10110101

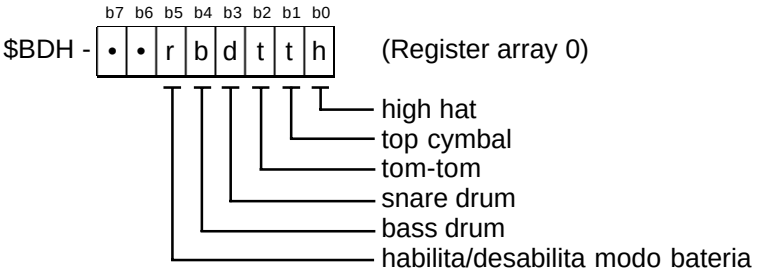
Os valores das freqüências guardam entre si uma relação geométrica igual à 12ª raiz de 2, que vale 1,0594630943592. Pode-se usar esse número para alterar os valores dos registradores a fim de aumentar ou diminuir a freqüência gerada dentro da escala musical. Os valores dos registradores também guardam entre si a mesma relação.

KEY ON/OFF



Esse bit controla a geração de som.

MODO BATERIA

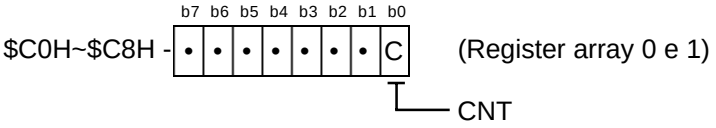


Esse registrador controla o modo bateria. Quando o bit r for 1, esse modo estará ativo e as três últimas vozes do gerador FM estarão indisponíveis. Entretanto, poderão ser geradas até 5 peças de bateria, conforme descrito na página anterior. Os slots usados são os seguintes:

Bass drum (BD)	13,16
Snare drum (SD)	17
Tom-tom	15
Top cymbal	18
High hat	14

Os valores rate etc. podem ser setados para manipular o som das peças de bateria. Quando em modo bateria, o bit “key” deve ser setado em 0 para os slots 13 a 18.

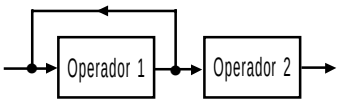
CNT (CONNECTION)



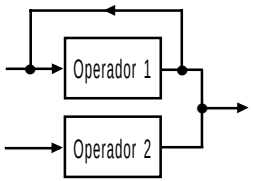
• MODO 2 OPERADORES

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)



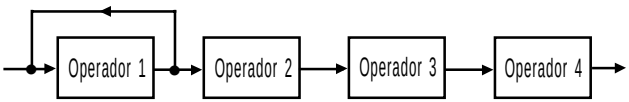
• MODO 4 OPERADORES

Para selecionar o modo 4 operadores, é necessário setar os registradores $04H e $05H nesse modo, e depois usar os dois bits CNT disponíveis para aplicar os algoritmos. Os dois bits CNT responsáveis pela seleção dos algoritmos estão ilustrados na tabela da página seguinte.

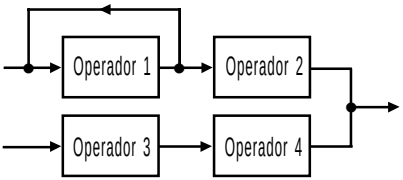
A1	Canal	CNT_n	CNT_{n+3}
0	1	C0H	C3H
	2	C1H	C4H
	3	C2H	C5H
1	4	C0H	C3H
	5	C1H	C4H
	6	C2H	C5H

Os 4 algoritmos possíveis, usando os bits CNT_n e CNT_{n+3} estão ilustrados abaixo.

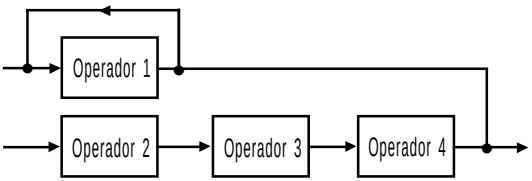
Algoritmo 1 (CNT_n=0, CNT_{n+3}=0)



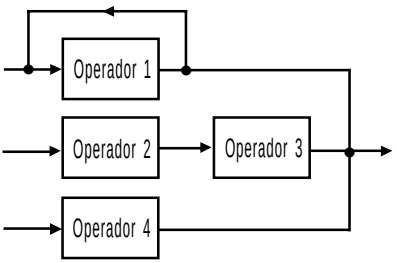
Algoritmo 2 (CNT_n=0, CNT_{n+3}=1)



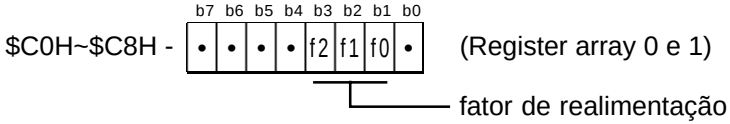
Algoritmo 3 (CNT_n=1, CNT_{n+3}=0)



Algoritmo 4 (CNT_n=1, CNT_{n+3}=1)



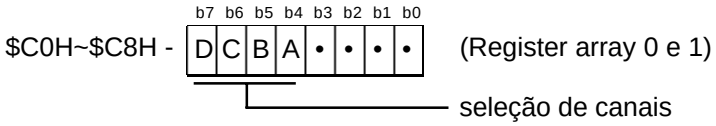
REALIMENTAÇÃO (FEEDBACK)



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
Fator de realimentação:	0	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	π	2π	4π

SELEÇÃO DE CANAIS DE SAÍDA

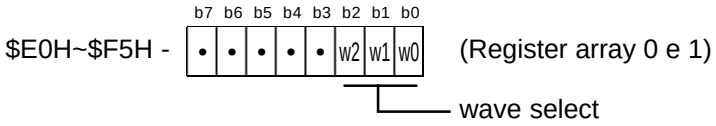


Até 4 canais de saída são disponíveis para o gerador FM. A saída estará habilitata quando o bit respectivo for 1.

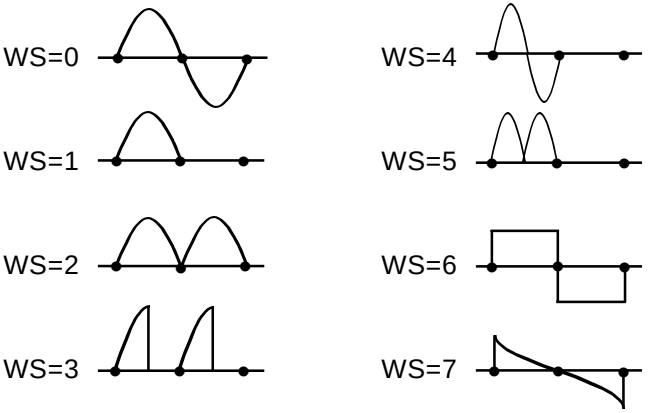
Canais A e B: a saída do gerador FM é mixada digitalmente com a Wave Table Synthetizer, quando o bit respectivo dos registradores $68H a $7FH for setado em 0, e o sinal será enviado para o pino DO2. O canal A é mixado com a saída esquerda do PCM e o canal B é mixado com a saída direita.

Canais C e D: A saída será pelo pino DO0. Como o gerador FM envia o sinal para o pino DO0 e o PCM para o pino DO1, vários efeitos sonoros podem ser aplicados usando o chip YSS225, que é conectado aos pinos DO0 e DO1. Um conversor D/A também pode ser conectado.

SELEÇÃO DA FORMA DE ONDA



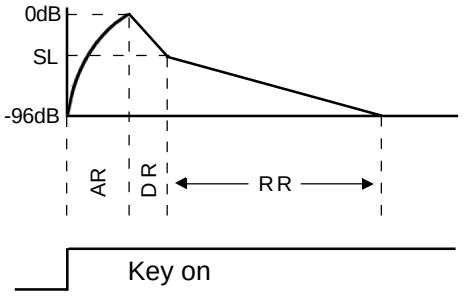
Esse registrador seleciona a forma de onda a ser usada em cada slot. As formas de onda possíveis estão ilustradas na página seguinte. Quando em modo OPL2, somente WS0 a WS3 estarão disponíveis.



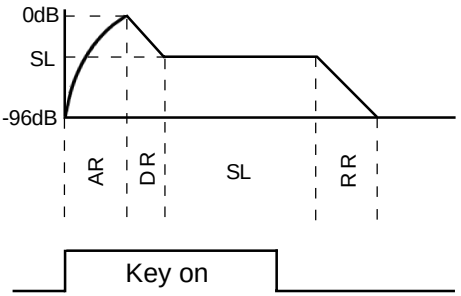
DETALHES DA FORMA DA ENVOLTÓRIA

A envoltória pode ter duas formas distintas, designadas por tom percussivo e tom constante. Elas estão ilustradas abaixo.

Tom percussivo (EGT = 1)



Tom constante (EGT = 0)



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. As portas usadas pelo cartucho Moonsound são as seguintes:

C4H FM register array 0 (banco 1) e registrador de status
C5H FM (dados)
C6H FM register array 1 (banco 2)
C7H Espelho de C5H (o acesso por C5H é preferido)
7EH Registradores PCM (wave)
7FH Dados PCM (wave)

O 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 modelo MSX turbo R; os acessos podem ser feitos seqüencialmente sem problemas[27].

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
```

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 “sampling rates” padronizados; o “sampling rate” deve ser determinado por temporização através de software.

8.1 - ACESSO AO COVOX

Para acessar o Covox, basta enviar os bytes de dados seqüencialmente 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.

Nota 27: Mesmo no caso do registrador $06H, que precisa de até 38 ciclos T de pausa, não é necessário se preocupar, pois a pausa requerida refere ao clock do OPL4, que é de 33,8688 MHz, equivalendo a uma pausa de apenas 4 ciclos T numa máquina MSX padrão a 3,58 MHz. No caso do MSX turbo R, são gerados wait states para os slots externos a fim de compatibilizar a temporização com o MSX padrão a 3,58 MHz.

Capítulo 7
OS SISTEMAS DE DISCO

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 em disco (disk-drive, hard disks, ZIP drive, CD-ROM, etc). Esses periféricos são normalmente acionados por rotinas do BDOS (Basic Disk Operating System). 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 do BIOS, através das rotinas PHYDIO e FORMAT.

Atualmente, há três sistemas de disco disponíveis para o MSX: o MSXDOS, o MSXDOS2 e o UZIX. 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 256 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 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.

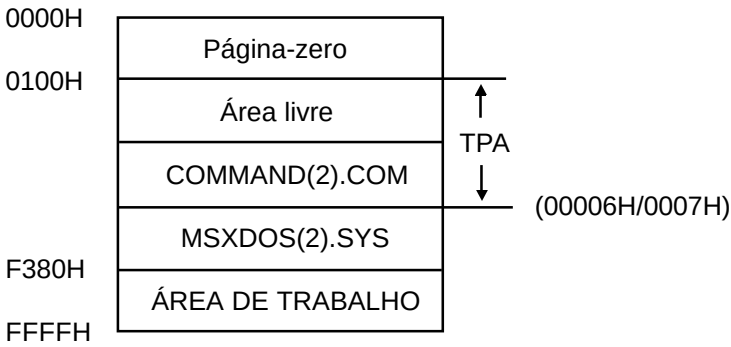
O padrão de formatação física dos dispositivos de disco do MSX é igual para todos os sistemas. Eles estão ilustrados abaixo.

	1DD3½	2DD3½	1DD5¼	2DD5¼	HD
ID mídia	F8H	F9H	FCH	FDH	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

1 - MSXDOS E MSXDOS2

O MSXDOS (1 ou 2) consiste nos seguintes módulos: interface de disco com BDOS 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.

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, seqüencialmente (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.

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.

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.

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.

1.4.1 - SETORES

Cada tipo de disco tem um determinado número de trilhas; assim, os disquetes de 5¼" 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.

1.4.2 - CLUSTERS (AGLOMERADOS)

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.

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

<table>
<tr><td rowspan="4">↑
disco
inteiro
↓</td><td>Setor de boot</td><td>setor #0</td></tr>
<tr><td>FAT</td><td rowspan="2">Os setores de início e o tamanho dessas áreas devem ser obtidos no DPB</td></tr>
<tr><td>Diretório</td></tr>
<tr><td>Área de dados</td><td>último setor</td></tr>
</table>

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 copiadas do setor de boot do disco durante a inicialização, embora alguns dados sejam diferentes entre o setor de boot e o DPB.

Na tabela da página seguinte 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.

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.

Na tabela da página seguinte está descrito o conteúdo do FIB.

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 distingüir o FIB de uma string pathname. Os dados do FIB são armazenados no mesmo formato dos dados do diretório. Eles estão detalhados na seção “DIRETÓRIO”, mais adiante.

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 aquivo 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: a FAT12 e a FAT16.

FAT12

A estrutura da FAT12 está ilustrada na página seguinte. 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.

	4-bit	4-bit	
Endereço inicial	F	9	FAT ID (80 trillhas, 9 setores)
	F	F	dummy
	F	F	dummy
	0	3	Entrada FAT 2 - link = 003H
	4	0	
	0	0	Entrada FAT 3 - link = 004H
	F	F	Entrada FAT 4 - link = FFFH (fim)
	6	F	
	0	0	Entrada FAT 5 - 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 FFFH, 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.

2	1	link = 321H
4	3	
6	5	link = 654H

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 FFFH 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 FFFH	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 65536 setores no disco. Assim, como cada setor tem 512 bytes, pode-se fazer 65536 * 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-bit	4-bit	
Endereço inicial	F	0	FAT ID
	F	F	dummy
	F	F	dummy
	1	2	
	3	4	Entrada FAT 2 - link = 1234H
	1	2	
	3	5	Entrada FAT 3 - link = 1235H
	F	F	
	F	F	Entrada FAT 4 - 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

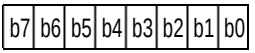
A FAT16 no MSXDOS usa o mesmo esquema da FAT16 no PC; por isso, uma partição FAT16 pode ter até 2 Gbytes. Para partições desse tamanho, entretanto, os clusters são enormes (32 Kbytes), o que propicia um grande desperdício de espaço no disco.

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.

1.4.7 - O DIRETÓRIO

A FAT, descrita acima, armazena a localização dos dados de um arquivo no disco, mas não contém qualquer 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 aquivo (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
+26/+27	primeiro cluster do arquivo
+28/+31	tamanho do arquivo em bytes

Byte de atributos - 

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, aparacerá 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 - 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	s4	s3	s2	s1	s0
hora (0~23)					minuto (0~59)						segundo (0~29)[28]				

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	d4	d3	d2	d1	d0
ano (0~99)[29]							mês (1~12)				dia (1~31)				

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 pode 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.

Nota 28: para obter o valor correto dos segundos, multiplicar o valor do registrador por 2.
Nota 29: para obter o ano correto, somar 1980 ao valor do registrador (1980 até 2079).

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 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 seqüencial 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 seqüencial (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 seqüencial atual (20H)
Contém o número ao registro atual para acesso seqüencial (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 o 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).

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.

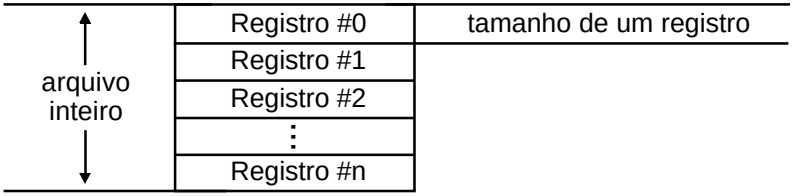
Ao abrir um arquivo, o número do drive no FCB é convertido para drive real (1 a 6 para MSXDOS1 ou 1 a 8 para MSXDOS2) e os outros campos do FCB são preenchidos (função 0FH do BDOS).

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).

1.5.3 - ACESSO SEQÜENCIAL E ALEATÓRIO

No acesso aleatório, os registros que compõem o arquivo podem ser acessados livremente, sem qualquer padrão estabelecido. Já no acesso seqüencial, como o próprio nome diz, 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 seqüencial extremo) ou de apenas um byte (acesso aleatório extremo). O valor default para o tamanho do registro é 128 bytes. Abaixo está ilustrado um arquivo com seus respectivos registros.



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

Esses 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 do 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.

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 alocada em uma página lógica (16K) fora da área da TPA, não reduzindo, portanto, 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)

1.6 - DESCRIÇÃO DAS 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 da página seguinte 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
```

As funções do BDOS estão descritas conforme a seguinte notação:

FUNÇÃO (xxH) ← xxH = número da função
Função: resumo da função que o BDOS realiza
Entrada: valores a colocar nos registradores ou na memória antes de chamar a função
Retorno: valores de retorno na memória ou registradores após a função ser executada

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 deve ser evitado, já que cada fabricante pode usar o controlador que melhor lhe convier e os programas podem não funcionar em interfaces diferentes.

1.6.1 - MANIPULAÇÃO DE I/O

(01H)
CONIN Entrada de um caractere pelo teclado
Função: Nenhum
Entrada: A - código ASCII do caratere
Retorno: Essa função espera uma tecla ser pressionada e imprime o
Nota: caractere na tela. As seguintes seqüências de controle são checadas por essa 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 também na impressora.
CTRL+N - Desliga o eco para a impressora.
CTRL+S - Causa uma parada de apresentação dos caracteres até que uma tecla seja pressionada.

CONOUT (02H)
Função: Saída de caractere para o monitor
Entrada: E - código do caractere
Retorno: Nenhum
Nota: Essa função apresenta na tela o caractere cujo código ASCII está no registrador E. As quatro seqüências de controle descritas anteriormente também são checadas.

AUXIN (03H)
Função: Entrada externa auxiliar.
Entrada: Nenhum.
Retorno: A - código ASCII do caractere do dispositivo auxiliar.
Nota: O dispositivo auxiliar pode ser qualquer um (modem, por exemplo). Entretanto, essa função só funciona em dispositivos que seguem o padrão MSX, e existem muitos dispositivos que não seguem o padrão, incluindo modems. As quatro seqüências de controle também são checadas.

AUXOUT (04H)
Função: Saída para dispositivo auxiliar.
Entrada: E - código ASCII do caractere a enviar.
Retorno: Nenhum.
Nota: Essa função também checa as quatro seqüências de controle.

LSTOUT (05H)
Função: Saída de caractere para a impressora.
Entrada: E - código ASCII do caractere a enviar.
Retorno: Nenhum.
Nota: Essa função também checa as quatro seqüências de controle.

DIRIO (06H)
Função: Entrada ou saída de string.
Entrada: E - código ASCII do caractere a ser impresso na tela. Se for FFH, o caractere será recebido.
Retorno: Quando o registrador E contiver o valor FFH na entrada, o código da tecla pressionada retornará em A. Se A retornar o valor 00H, não foi pressionada nenhuma tecla.
Nota: Essa função não suporta caracteres de contole, mas checa as quatro seqüências de controle.

DIRIN (07H)
Função: Leitura do teclado com espera (I)
Entrada: Nenhuma.
Retorno: A - código ASCII do caractere lido.
Nota: O caractere lido é impresso na tela. Essa função não suporta caracteres de controle.

INNOE (08H)
Função: Leitura de teclado com espera (II)
Entrada: Nenhum
Retorno: A - código ASCII do caractere lido.
Nota: Essa função é idêntica à anterior, exceto que o caractere lido não é enviado para a tela.

STROUT (09H)
Função: Saída de string para a tela.
Entrada: DE - endereço inicial da string a ser enviada.
Retorno: Nenhum.
Nota: O caractere ASCII 24H ($) marca o final da string a ser enviada e não será impresso na tela. Essa função checa as quatro seqüências de controle.

BUFIN (0AH)
Função: Entrada de string.
Entrada: DE - deve apontar para um buffer com a seguinte estrutura:
DE+0 - número de caracteres a ler
DE+1 - número de caracteres efetivamente lidos
DE+2 em diante: códigos ASCII 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 armazenados os códigos ASCII dos caracteres lidos.
Nota: 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. Essa função checa as quatro seqüências de controle.

CONST (0BH)
Função: Checagem do status do teclado.
Entrada: Nennhum.
Retorno: Se alguma tecla foi pressionada, o registrador A retorna com o valor FFH, caso contrário retorna com o valor 00H.
Nota: Essa função checa as quatro seqüências de controle.

1.6.2 - DEFINIÇÃO E LEITURA DE PARÂMETROS

TERM0 (00H)
Função: Reset do sistema.
Entrada: Nenhum.
Retorno: Nenhum.
Nota: Quando essa função for chamada bob o DOS, promove a recarga do MSXDOS. Quando for chamada sob o Disk-BASIC, provoca um reset total no sistema.

CPMVER (0CH)
Função: Leitura da versão do sistema.
Entrada: Nenhum.
Retorno: HL - 0022H
Nota: Essa função retorna em HL a versão do DOS instalado. No caso do MSX, sempre retornará o valor 0022H, indicando compatibilidade com o CP/M 2.2.

DSKRST (0DH)
Função: Reset do disco.
Entrada: Nenhum.
Retorno: Nenhum.
Nota: Essa função atualiza todos os dados sobre o disco contidos nos buffers do MSXDOS. Todos os buffers são apagados (FCB, DPB, etc.), o drive default será o A: e a DTA será setada em 0080H.

SELDSK (0EH)
Função: Selecionar o drive default.
Entrada: E - número do drive (A:=00H, B:=01H, etc.)
Retorno: A - número de drives lógicos conectados (1 a 8)
Nota: Essa função muda o número do drive default, ou seja, o drive que será acessado quando não houver especificação de drive. O número do drive corrente será armazenado em 0004H.

LOGIN (18H)
Função: Leitura de drives conectados.
Entrada: Nenhum.
Retorno: HL - drives conectados.

[H]								[L]							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0	H:	G:	F:	E:	D:	C:	B:	A:

Nota: Essa função retorna em HL os drives que estão conectados, até um máximo de oito. O bit conterá 0 se o drive não estiver conectado e 1 se estiver. Se B: contiver 1 e A: contiver 0: (b1=1 e b0=0), significa que há apenas um drive físico funcionando como A: e B:. O registrador H: sempre retornará 00H.

CURDRV (19H)
Função: Leitura do drive corrente (default).
Entrada: Nenhum.
Retorno: A - número do drive default (A:=00H, B:=01H, etc.).
Nota: Essa função retorna em A: o número do drive atual (0 A 7).

SETDTA (1AH)
Função: Seta o endereço para transferência de dados.
Entrada: DE - endereço inicial da DTA (Disk Transfer Area).
Retorno: Nenhum.
Nota: No reset do sistema, a DTA é setada em 0080H, mas pode ser deslocada para qualquer área da memória com essa função. A DTA também é conhecida como DMA (Disk Memory Area).

ALLOC
Função: (1BH)
Entrada: Leitura de informações sobre o disco.
Retorno: E - número do drive desejado (0=default, 1=A:, etc.)
A - FFH se a especificação de drive for inválida,
caso contrário:
A - número de setores lógicos por cluster;
BC - tamanho do setor em bytes;
DE - número total de clusters no disco;
HL - número de clusters livres;
IX - endereço inicial do DPB na RAM;
IY - endereço inicial da FAT na RAM.

GDATE
Função: (2AH)
Entrada: Leitura da data do sistema.
Retorno: Nenhuma
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
Função: (2BH)
Entrada: Modificar a data do sistema.
HL - ano (1980 a 2079)
D - mês (1=janeiro, 2=fevereiro, etc.)
Retorno: E - dia do mês (1 a 31)
A - 00H se a especificação de data foi válida;
FFH se a especificação foi inválida.

GTIME
Função: (2CH)
Entrada: Leitura da hora do sistema.
Retorno: Nenhuma.
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 foi válida;
FFH se a especificação foi inválida.

VERIFY (2EH)
Função: Verificação de escrita no disco.
Entrada: E - igual a 0 para desativar o modo de verificação de escrita.
Qualquer outro valor ativa a verificação de escrita no disco.
Retorno: Nenhum.
Nota: Quando a verificação de escrita estiver ativa, logo após uma gravação no disco o sistema automaticamente fará uma checagem para verificar se a gravação foi bem sucedida. Na carga do sistema, a função de verificação é desativada. Infelizmente, essa função pode variar de interface para interface, tornando-a incompatível com o padrão MSX.

1.6.3 - 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 32 Mb: 0 a 65535

As funções do BDOS descritas abaixo acessam diretamente os setores lógicos do disco.

RDABS: (2FH)
Função: Leitura de setores lógicos do disco.
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 - se contiver 0, a leitura foi bem sucedida; outro valor será o código de erro.
Nota: Essa função lê os setores continuamente até atingir o total especificado no registrrador H ou detectar algum erro. Os setores lidos são colocados a partir da DTA.

WRABS (30H)
Função: Escrita de setores lógicos no disco.
Entrada: DE - número do primeiro setor lógico a ser escrito;
H - número de setores a ler;
L - número do drive (0=A:, 1=B:, etc.)
Retorno: A - se contiver 0, a escrita foi bem sucedida; outro valor será o código de erro.
Nota: Os dados a serem escritos no disco serão lidos na RAM a partir da DTA.

1.6.4 - ACESSO AOS ARQUIVOS USANDO O FCB

Acessar os arquivos do disco usando as funções do BDOS descritas até agora é um processo muito complicado. As funções do BDOS que acessam o disco usando o FCB tornam essas operações mais simples.

Existem três categorias de acesso a arquivos usando o FCB: acesso seqüencial, 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).

FOPEN (0FH)
Função: Abrir um arquivo.
Entrada: DE - endereço inicial de um FCB não aberto.
Retorno: A - se contiver 0, a operação foi bem sucedida; se contiver FFH, houve algum problema.
Nota: Quando o arquivo é aberto, todos os campos do FCB (exceto o tamanho do registro, registro atual e registro aleatório) são pre-enchidos com os dados contidos no diretório.

FCLOSE (10H)
Função: Fechar um arquivo.
Entrada: DE - endereço inicial de um FCB aberto.
Retorno: A - se contiver 0, a operação foi bem sucedida; se contiver FFH, houve algum problema.
Nota: Essa função transfere os dados contidos no FCB para o diretório. É absolutamente necessário chamar essa função após a gravação de novos registros em um arquivo, caso contrário as entradas no diretório não serão atualizadas, com a conseqüente perda de dados do arquivo.

SFIRST (11H)
Função: Procurar o primeiro arquivo.
Entrada: DE - endereço inicial de um FCB não aberto.
Retorno: A - se contiver 00H, o arquivo foi encontrado; se contiver FFH, o arquivo não foi encontrado.
Nota: Caso o arquivo tenha sido encontrado, a entrada respectiva no diretório é copiada na DTA e o número do drive do FCB é setado (33 bytes são usados). O FCB poderá ser aberto na própria DTA. Caracteres coringa podem ser usados (? e *), caso em que o primeiro arquivo encontrado terá seus dados transferidos para a DTA.

SNEXT (12H)
Função: Procurar o próximo arquivo.
Entrada: Nenhum.
Retorno: A - se contiver 00H, o arquivo foi encontrado; se contiver FFH, o arquivo não foi encontrado.
Nota: Essa função procura o próximo arquivo ao que foi encontrado com a função anterior (SFIRST). Caso seja encontrado mais algum arquivo, a entrada respectiva do diretório é copiada na DTA e o número do drive no FCB é setado. Essa função foi criada especialmente para uso com caracteres coringa (? e *), pois cada vez que é chamada procura no diretório o próximo arquivo que coincida com a especificação dada.

FDEL (13H)
Função: Deletar arquivos.
Entrada: DE - endereços inicial de um FCB aberto.
Retorno: A - se contiver 0, a operação foi bem sucedida; se contiver FFH, houve algum problema.
Nota: Essa função aceita caracteres coringa (? e *) na especificação do FCB para deletar mais de um arquivo simultaneamente.

RDSEQ (14H)
Função: Leitura seqüencial.
Entrada: DE ← endereço inicial de um FCB aberto.
Bloco atual no FCB ← bloco inicial a ser lido.
Registro inicial no FCB ← registro inicial a ser lido.
Retorno: A - se contiver 00H, a leitura foi bem sucedida; se contiver 01H, houve erro durante a leitura.
Nota: Quando a leitura for bem sucedida, o registro lido será colocado na DTA. Além disso, o bloco e registro atuais do FCB são automaticamente incrementados para facilitar a próxima leitura. O tamanho de cada registro é fixado em 128 bytes.

WRSEQ (14H)
Função: Escrita seqüencial.
Entrada: DE ← endereço inicial de um FCB aberto.
Bloco atual no FCB ← bloco inicial a ser escrito.
Registro atual no FCB ← registro inicial a ser escrito.
128 bytes iniciais da DTA ← dados a serem escritos.
Retorno: A - se contiver 00H, a escrita foi bem sucedida; se contiver 01H, houve erro durante a escrita.
Nota: O bloco e o registro atuais são automaticamente incrementados após a escrita para facilitar a escrita seqüencial.

FMAKE (16H)
Função: Criar arquivos.
Entrada: DE - endereço inicial de um FCB não aberto.
Retorno: A - se contiver 00H, a operação foi bem sucedida. Se contiver FFH, houve erro na criação do arquivo.
Nota: O tamanho do registro, o bloco e registro atuais e o registro aleatório do FCB podem ser setados após executar essa função.

FREN (17H)
Função: Renomear arquivos.
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 byte deve ser colocado o novo nome do arquivo.
Retorno: A - se contiver 00H, a renomeação foi executada com sucesso; se contiver FFH, houve erro na renomeação.
Nota: O caractere coringa "?" pode ser usado pelo atual e pelo novo nome do arquivo, para renomear vários arquivos simultaneamente. Por exemplo, especificando "????????.MAC" para os arquivos a renomear e "????????.OBJ" para o novo nome do arquivo, todos os arquivos com a extensão ".MAC" serão renomeados com a extensão ".OBJ".

RDRND (21H)
Função: Leitura aleatória.
Entrada: DE ← endereço inicial de um FCB aberto.
Registro aleatório no FCB ← número do registro a ler.
Retorno: A - se contiver 00H, a leitura foi bem sucedida; se contiver 01H, houve erro durante a leitura.
Nota: O registro lido será colocado na DTA. O tamanho do registro é fixado em 128 bytes.

WRRND (22H)
Função: Escrita aleatória.
Entrada: DE ← endereço inicial de um FCB aberto.
Registro aleatório no FCB ← número de registro a escrever.
128 bytes a partir da DTA ← dados a serem escritos.
Retorno: A - se contiver 00H, a escrita foi bem sucedida; se contiver 01H, houve erro durante a escrita.

FSIZE (23H)
Função: Ler o tamanho do arquivo.
Entrada: DE - endereço inicial de um FCB aberto.
Retorno: A - se contiver 00H, a operação foi bem sucedida; se contiver FFH, houve erro durante a execução da função.
Nota: O tamanho do arquivo é especificado nos três primeiros bytes no campo de tamanho do arquivo aleatório no FCB em incrementos de 128 bytes. Assim, se um arquivo conter de 1 a 128 bytes, essa função retornará 1, se conter de 129 a 256 bytes retornará 2; se conter de 257 a 384 bytes retornará 3 e assim por diante.

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 desejada para o registro atual, calculada a partir do FCB calculada a partir do registro e bloco contidos no FCB, é colocada no campo de registro aleatório. Apenas os três primentos bytes do registro aleatório são preenchidos.

WRBLK (26H)
Função: Escrita aleatória em bloco.
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 - se contiver 00H, a escrita foi bem sucedida; se contiver 01H, houve erro durante a escrita.
Nota: Após a escrita, o número do registro aleatório é automaticamente incrementado para facilicar eventuais escritas posteriores. O tamanho do registro pode ser qualquer um, desde 1 byte até 65535 bytes, bastando setar o campo respectivo do FCB.

RDBLK (27H)
Função: Leitura aleatória em bloco.
Entrada: DE ← endereço inicial de um FCB aberto.
HL ← número de registros a serem lidos.
DTA ← 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 - se contiver 00H, a leitura foi bem sucedida; se contiver 01H, houve erro durante a leitura.
HL - número de registros efetivamente lidos, caso o fim do arquivo seja atingido antes do número de registros especificado ser completado.

WRZER (28H)
Função: Escrita aleatória com bytes 00H.
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 - se contiver 00H, e escrita foi bem sucedida; se contiver 01H, houve erro durante a escrita.
Nota: O tamanho dos registros é fixado em 128 bytes. Essa função é igual à WRRND (22H), exceto pelo fato de preencher os registros restantes do arquivo com bytes 00H, se o registro especificado não for o último do arquivo.

1.6.5 - FUNÇÕES ADICIONADAS PARA O MSXDOS2

As funções do BDOS que serão descritas a seguir foram adicionadas para o MSXDOS2 e não estão implementadas para o MSXDOS1.

DPARM
Função: Lê os parâmetros do disco.
Entrada: DE - endereço inicial de um buffer de 32 bytes.
L - número do drive (0=default; 1=A:, 2=B:, etc)
Retorno: A - código de erro (se for 0, não houve erro).
DE - endereço inicial do buffer de parâmetros.
Nota: Essa função retorna uma série de parâmetros do disco especificado em um buffer. O formato desse buffer está descrito na página seguinte.

offset	*descrição resumida*
DE+0	número do drive físico
DE+1~2	tamanho de um setor em bytes
DE+3	número de setores por cluster
DE+4~5	número de setores reservados
DE+6	número de FAT´s (normalmente 2)
DE+7~8	número de entradas do diretório
DE+9~10	número total de setores lógicos
DE+11	ID do disco
DE+12	número de setores por FAT
DE+13~14	primeiro setor do diretório
DE+15~16	primeiro setor da área de dados
DE+17~18	número máximo de clusters
DE+19	dirty disk flag
DE+20~23	volume ID (-1 = sem ID de volume)
DE+24~31	reservado

FFIRST (40H)

Função: Procura a primeira entrada.

Entrada: DE - endereço inicial do FIB ou de uma string ASCII “drive/path/arquivo”.
HL - endereço inicial do nome do 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 00H, não houve erro).
IX - endereço inicial do novo FIB preenchido.

Nota: O bit “somente leitura” do byte de atributos é ignorado para a procura. O nome do arquivo pode conter os caracteres coringa (? e *), para procurar mais de um arquivo que tenham partes de seus nomes iguais.

FNEXT

Função: (41H)

Entrada: Procura a próxima entrada.

Retorno: IX - Endereço inicial do FIB.
A - código de erro (se for 00H, não houve erro).

Nota: IX - endereço inicial do novo FIB preenchido.
Essa função só deve ser chamada após a função 40H. Ela foi criada especialmente para uso com caracteres coringa (? e *). Difere da anterior pelo fato de procurar todos os arquivos que tenham partes de seus nomes iguais, especificado através dos caracteres coringa, um após o outro.

FNEW (42H)
Função: Procura nova entrada.
Entrada: DE - endereço inicial do FIB ou de uma string ASCII "drive/path/arquivo".
HL - endereço inicial do nome do arquivo (somente quando DE apontar para o FIB).
B - atributos para procura (igual ao do diretório, exceto que, se b7 estiver setado, cria nova flag).
IX - endereço inicial de um novo FIB contendo o nome de arquivo padrão.
Retorno: A - código de erro (se for 00H, não houve erro).
IX - endereço inicial do novo FIB preenchido com a nova entrada.
Nota: Essa função é parecida com a função 40H, mas ao invés de procurar uma entrada no diretório, ela cria uma nova entrada como o mesmo nome. O FIB apontado por IX será preenchido com as informações da nova entrada. Se houver caracteres coringa no nome de arquivo, eles serão trocados por caracteres apropriados pelo "nome de arquivo padrão". Se o bit "diretório" estiver setado na entrada (registrador B), será criado um subdiretório. Os outros bits serão copiados.

OPEN (43H)
Função: Abre arquivo handle.
Entrada: DE - endereço inicial do FIB ou de uma string ASCII "drive/path/arquivo".
A - Modo de abertura: b0=1 - não escrita
b1=1 - não leitura
b2=1 - inheritable (herdado)
b3~b7 - sempre 0
Retorno: A - código de erro (se for 0, não houve erro)
B - novo arquivo handle.
Nota: O FIB ou a string drive/path/arquivo preferencialmente referem-se a um subdiretório ou a um nome de volume. O arquivo especificado é aberto para escrita e/ou leitura (dependendo do valor do registrador A) e o novo arquivo handle retorna no registrador B. Se o bit "inheritable" de A estiver setado, o arquivo handle será aberto por outro processo (função 60H).

CREATE (44H)
Função: Criar arquivo handle.
Entrada: DE - endereço inicial de uma string ASCII "drive/path/arquivo".
A - Modo de abertura: b0=1 - não escrita
b1=1 - não leitura
b2=1 - inheritable (herdado)
b3~b7 - sempre 0
B - b0~b6 = atributos; b7 = cria nova flag

Retorno: A - código de erro (se for 0, não houve erro).
B - novo arquivo handle.
Nota: O arquivo criado por essa função será automaticamente aberto (função 43H). Se o arquivo for um subdiretório, este não será aberto. Caso o registrador B retorne com o valor FFH, o arquivo handle criado não é válido.

CLOSE (45H)
Função: Fechar arquivo handle.
Entrada: B - arquivo handle a ser fechado.
Retorno: A - código de erro (se for 0, não houve erro).

ENSURE (46H)
Função: Proteger arquivo handle.
Entrada: B - arquivo handle a ser protegido.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Se o arquivo handle estiver protegido, o apontador do arquivo corrente não poderá ser modificado, mas se este for escrito, os campos hora, data, atributos e dados serão transferidos para o disco.

DUP (47H)
Função: Duplicar um arquivo handle.
Entrada: B - arquivo handle a ser duplicado.
Retorno: A - código de erro (se for 0, não houve erro).
B - novo arquivo handle.
Nota: Essa função cria uma cópia do arquivo handle especificado. O novo arquivo handle referirá ao mesmo arquivo que o original. Se um dos arquivos handle for fechado ou tiver o apontador de arquivo modificado, o outro também o terá.

READ (48H)
Função: Ler através de um arquivo handle.
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).
Nota: O número de bytes especificado é lido do arquivo corrente e copiado para o buffer indicado por DE. Se o fim de arquivo for detectado antes do término da leitura, o número de bytes lidos retornará em HL e não será gerado erro. As quatro seqüências de controle (Ctrl+P, Ctrl+N, Ctrl+S e Ctrl+C) são checadas por essa função.

WRITE (49H)
Função: Escrever por um arquivo handle.
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.
Nota: Essa função é parecida com a anterior, mas escreve os dados ao invés de ler. Se o arquivo handle foi aberto com as flags de "não escrita" ou "não leitura", a função retornará com erro. Se o fim de arquivo for encontrado, ele será extendido até o valor necessário. Os dados a escrever são retirados do buffer apontado por DE.

SEEK (4AH)
Função: Mover o apontador do arquivo handle.
Entrada: B - arquivo handle.
A - código do método.
DE:HL - sinalização de offset.
Retorno: A - código de erro (se for 0, não houve erro).
DE:HL - novo apontador de arquivo.
Nota: O apontador de arquivo associado com o arquivo handle será alterado de acordo com o código do método como descrito:
A = 0 - relativo ao início do arquivo;
A = 1 - relativo à posição corrente;
A = 2 - relativo ao final do arquivo.
Se houver mais de um arquivo handle criado pela função 47H, todos serão alterados da mesma forma.

IOCTL (4BH)
Função: Controla 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 o dispositivo está pronto para entrada;
03H - testa se o dispositivo está pronto para saída;
04H - calcula o tamanho da screen.
DE - outros parâmetros.
Retorno: A - código de erro (se for 0, não houve erro).
DE - outros valores de retorno.
Nota: Essa função retorna vários aspectos dos arquivos handle, principalmente se este refere a um arquivo ou a um dispositivo. 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;
b5=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 for igual 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 o dispositivo 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=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 para o FIB ou para string ASCII “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.

Nota: Essa função testa se o arquivo handle especificado em B se refere ao arquivo apontado por DE. Se se referir ao mesmo arquivo, B retornará com o valor FFH; caso contrário, retornará com 00H.

DELETE (4DH)

Função: Apagar arquivo ou subdiretório.

Entrada: DE - apontador para o FIB ou para string ASCII “drive/path/arquivo”.

Retorno: A - código de erro (se for 0, não houve erro).

Nota: Um subdiretório só poderá ser apagado se não contiver nenhum nome de arquivo (deve estar vazio). Se um nome de arquivo for especificado, não retornará erro, mas, é claro, o dispositivo não será apagado.

RENAME (4EH)
Função: Renomear arquivo ou subdiretório.
Entrada: DE - apontador para o FIB ou para string ASCII "drive/path/arquivo".
HL - apontador para o novo nome (em ASCII).
Retorno: A - código de erro (se for 0, não houve erro).
Nota: O novo nome apontado por HL não deverá conter a especificação de drive e/ou diretório path. Se um nome de dispositivo for especificado, não retornará código de erro, mas o nome de dispositivo não será modificado. O FIB não será modificado.

MOVE (4FH)
Função: Mover arquivo ou subdiretório.
Entrada: DE - apontador para o FIB ou para string ASCII "drive/path/arquivo".
HL - apontador para o novo nome (em ASCII).
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função move o arquivo ou subdiretório apontado por DE para o diretório especificado pela string path apontada por HL. A string path não deve conter especificação de drive. Se um subdiretório for movido, todas as suas entradas com os respectivos arquivos serão movidos junto. Um arquivo não poderá ser movido se o arquivo handle respectivo estiver aberto. O FIB do arquivo movido não é atualizado.

ATTR (50H)
Função: Ler ou setar os atributos de um arquivo.
Entrada: DE - apontador para o FIB ou para string ASCII "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.
Nota: Se A=0, o byte de atributos do arquivo ou subdiretório retornará no registrador L. Os atributos de um arquivo não podem ser modificados se o arquivo handle respectivo estiver aberto.

FTIME (51H)
Função: Ler ou setar data e hora de um arquivo.
Entrada: DE - apontador para o FIB ou para string ASCII "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.
Nota: Se A=1, a data e a hora do arquivo serão modificadas de acordo com o valor dos registradores IX e HL. Se A=0, a data e a hora do arquivo apontado por DE retornarão em DE e HL. O formato da data e da hora é igual ao do diretório.

HDELET (52H)
Função: Apagar um arquivo handle.
Entrada: B - aquivo handle.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função apaga um arquivo handle. Se houver outro arquivo handle aberto para o mesmo arquivo, então esse não poderá ser apagado.

HRENAM (53H)
Função: Renomear por um arquivo handle.
Entrada: B - arquivo handle.
HL - apontador para o novo nome do arquivo em ASCII.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função renomeia o arquivo associado com o arquivo handle especificado. O arquivo não poderá ser renomeado se houver outro arquivo handle aberto para o mesmo arquivo. Essa função é idêntica à função RENAME (4EH), exceto pelo fato do registrador HL não poder apontar para um FIB.

HMOVE (54H)
Função: Mover por um arquivo handle.
Entrada: B - arquivo handle.
HL - apontador para uma nova path em ASCII.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função move o arquivo associado ao arquivo handle especificado para o diretório especificado pela nova string path apontada por HL. O arquivo não poderá ser movido se houver outro arquivo handle aberto para o mesmo arquivo. Essa função é idêntica à função MOVE (4FH), exceto pelo fato do registrador HL não poder apontar para um FIB.

HATTR (55H)
Função: Ler ou setar atributos por um arquivo handle.
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.

Nota: Essa função lê ou seta os atributos do arquivo associado ao arquivo handle especificado. O byte de atributos não poderá ser setado se houver outro arquivo handle aberto para o mesmo arquivo.

HFTIME (56H)
Função: Ler ou setar hora e data por um arquivo handle.
Entrada: B - arquivo handle.
A - 0=ler data de 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.
Nota: Essa função lê ou seta a data e a hora do arquivo associado ao arquivo handle especificado. Se houver outro arquivo handle aberto para o mesmo arquivo, a data e a hora não poderão ser modificadas. Essa função é idêntica à função FTIME (51H) exceto pelo fato de não haver apontador; somente o arquivo handle.

GETDTA (57H)
Função: Obter o endereço inicial da DTA (Disk Transfer Area).
Entrada: Nenhuma.
Retorno: DE - endereço inicial da DTA.

GETVFY (58H)
Função: Ler flag de verificação de escrita.
Entrada: Nenhuma.
Retorno: B - 0 = verificação de escrita desativada;
1 = verificação de escrita ativa.

GETCD (59H)
Função: Ler diretório ou subdiretório corrente.
Entrada: B - número do drive (0=default, 1=A:, etc)
DE - endereço inicial deum buffer de 64 bytes.
Retorno: A - código de erro (se for 0, não houve erro).
DE - preenchido de acordo com a path corrente.
Nota: Essa função retorna no buffer apontado por DE o nome do diretório corrente em ASCII. 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).

Nota: Essa função troca o diretório ou subdiretório corrente pelo apontado pelo registrador DE.

PARSE (5BH)
Função: Analisa pathname (nome do caminho).
Entrada: B - flag do nome do volume (bit 4);
DE - string ASCII a ser analisada;
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;
C - drive lógico (1=A:, 2=B:, etc).
Nota: O bit 4 do registrador B na entrada deve estar setado para string "drive/volume" ou resetado (00H) para string "drive/path/arquivo". O valor retornado em HL apontará para o primeiro caractere do último item da string. Por exemplo, para uma string tipo "A:\XYX\P.Q /F", DE apontará para o espaço em branco antes do "/F" e HL apontará para "P". As flags retornadas no registrador B são as seguintes:
b0=1 - se algum caractere apontar para outro nome de drive;
b1=1 - se algum diretório path for especificado;
b2=1 - se um 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 "..".

PFILE (5CH)
Função: Analisar nome de arquivo.
Entrada: DE - string ASCII a ser analisada.
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.
Nota: A string ASCII apontada por DE deve ser um nome de arquivo simples, sem especificação de drive. Podem ser usados caracteres coringa (? e *). Os significados das flags do registrador B são idênticos aos da função PARSE (5BH), exceto que os bits 0, 1 e 2 sempre serão 0.

CHKCHR (5DH)
Função: Checa caractere.

Entrada: D - flags do caractere
E - caractere a ser checado.
Retorno: A - sempre 00H.
D - flags atualizadas do caractere.
E - caractere checado.
Nota: Essa função também checa caracteres de 16 bits e manipula nomes de arquivos. As flags do caractere são as seguintes:
b0=1 para suprimir o caractere;
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 de volume ou preferencialmente nome de arquivo;
b4=1 caractere de arquivo/volume não válido;
b5~b7 - reservados (sempre 0).
Se o bit 0 for 1, o caractere retornado em E será sempre o mesmo; se for 0, poderá ser trocado de acordo com a língua setada na máquina. Para analisar um caractere de dois bytes, deve-se enviar o primeiro byte e depois o segundo, setando a flag correspondente. O bit 4 será setado no retorno se o caractere for um terminador de nome de arquivo ou volume.

WPATH (5EH)
Função: Ler string path completa.
Entrada: DE - apontador para um buffer de 64 bytes.
Retorno: A - código de erro (se for 0, não houve erro).
DE - início do buffer preenchido com a string path completa.
HL - apontador para o início di último item.
Nota: Essa função copia a string path ASCII corrente para o buffer apontado por DE. A string retornada não contém a especificação de drive e o caractere "\" inicial. O registrador HL aponta para o primeiro caractere do último item, exatamente como na função PARSE (5BH). Para maior confiabilidade, pode-se primeiro chamar a função 40H ou 41H e depois chamar WPATH duas vezes, já que outras funções podem alterar o dados.

FLUSH (5FH)
Função: Descarregar todos os buffers para o disco.
Entrada: B - especificação de drive (0=default, 1=A:, etc.).
D - 00H = somente descarregar;
FFH = descarregar e invalidar.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função todos os buffers para o drive especificado ou para todos os drives se B=FFH na entrada. Se o registrador D for FFH, todos os buffers do drive especificado serão também invalidados.

FORK (60H)
Função Duplicar arquivos handle.
Entrada: Nenhuma.
Retorno: A - código de erro (se for 0, não houve erro).
B - ID do processo de duplicação.
Nota: Novos arquivos handle são criados e os arquivos handle correntes que estão abertos no modo "inheritable" (ver função 43H) são copiados para os novos arquivos handle. Os arquivos handle standard (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 Restaurar arquivo handle duplicado.
Entrada: B - ID do processo de duplicação (ou 00H).
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: Essa função retorna para o arquivo handle original o arquivo handle copiado pela função anterior. 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 essa função for chamada pelo endereço F37DH, os registradores B e C não retornarão código de erro (ver função 62H).

TERM (62H)
Função Finalizar com código de erro.
Entrada: B - código de erro para finalização.
Retorno: Nenhum.
Nota: Essa função termina o programa com o código de erro especificado. A operação dessa 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 DEFAB (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 jump para o endereço 0000H, provocando uma partida a quente do DOS. O interpretador de comandos somente imprimirá a mensagem de erro na tela se essa estiver entre 20H e FFH, mas não abaixo de 20H. Se essa 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 abortagem (saída).
Entrada: DE - endereço inicial da rotina de abortagem (o endereço default é 0000H).
Retorno: A - sempre 00H.
Nota: Essa rotina somente estará disponível se for chamada por 0005H; ela não deve ser chamada por F37DH. A rotina apontada por DE também será chamada no caso do sistema detectar as teclas Ctrl-C ou Ctrl-STOP pressionadas juntas ou se houver erro de disco abortado.

DEFER (64H)
Função: Definir rotina para erro de disco.
Entrada: DE - endereço inicial da rotina (o endereço default é 0000H).
Retorno: A - sempre 00H.
Nota: Essa função especifica o endereço de uma rotina criada pelo usuário caso ocorra algum erro de disco. Deve ser usada com muita cautela. Os parâmetros e resultados dessa rotina estão especificados abaixo.
Parâmetros: A - código de erro;
B - número do drive físico;
C - b0=1 se for erro de escrita;
b1=1 se ignorar 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 o código de erro antecipadamente.
Entrada: Nenhum.
Retorno: A - sempre 00H.
B - código de erro da função.
Nota: Essa função pode ser chamada para prevenir o tipo de erro que poderá ocorrer na próxima chamada de função.

EXPLN (66H)
Função: Pegar 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 - início do buffer preenchido com a mensagem de erro.

Nota: Essa função retorna no buffer apontado por DE a mensagem ASCII de erro. Se a mensagem de erro for do tipo “System error 194” ou “User error 45”, o registrador B retornará com o valor 00H.

FORMAT (67H)
Função: Formatar um disco.
Entrada: B - número do drive (0=default, 1=A:, 2=B:, etc.)
A - 00H = retorna mensagem de escolha;
01H~09H = formata com esta escolha;
0AH~0DH = ilegal;
FEH~FFH = novo setor de boot.
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).
Nota: Essa função é usada para formatar um disco e tem três diferentes opções de acordo com o valor passado em A. Se A=0, os registradores B e HL retornarão com o número do slot e endereço da mensagem ASCII interna do DOS.Se A for igual a 01H~09H, o sistema formatará o disco sem apresentar mensagem e, nesse caso, os registradores HL e DE devem especificar o buffer usado pelo disk-drive. Se A=FFH, o disco não será formatado, mas será atualizado para MSXDOS2. Se A=FEH, o disco também não será formatado e somente os parâmetros do disco serão atualizados para MSXDOS2.

RAMD (68H)
Função: Criar ou desativar a RamDisk.
Entrada: B - 00H = desativa a RamDisk;
01H~FEH = cria nova Ramdisk;
FFH = retorna o tamanho da RamDisk.
Retorno: A - código de erro (se for 0, não houve erro).
B - tamanho da RamDisk.
Nota: Se o registrador B for FFH na entrada, ele somente retornará com o número de segmentos de 16K (número de páginas lógicas) alocadas para a RamDisk. Se for 00H, desativará a RamDisk. Se contiver entre 01H e FEH, será criada uma RamDisk usando o número de páginas lógicas (segmentos de 16K) especificado em B. A RamDisk sempre será o drive “H:”.

BUFFER (69H)
Função: Alocar buffers.
Entrada: B - 00H = retorna o número de buffers alocados.
01H~A5H = aloca o número especificado de buffers.

Retorno: A - código de erro (se for 0, não houve erro).
B - número corrente de buffers
Nota: Se o registrador B for 00H na entrada, essa função retornará em B o número atual de buffers. Se B contiver de 1 a 20, o sistema alocará o número de buffers especificado; caso a memória seja insuficiente, será alocado o número possível de buffers retornando o número em B. Não será gerado código de erro. O número máximo de buffers alocados é de 20. Cada buffer ocupa uma página lógica (16K) fora do segmento normal de 64K, não afetando o tamanho da TPA.

ASSIGN (6AH)
Função: Atribuir drive lógico.
Entrada: B - número do drive lógico (1=A:, etc);
D - número do drive físico (1=A:, etc).
Retorno: A - código de erro (se for 0, não houve erro).
D - número do drive físico.
Nota: Essa função atribui o drive lógico ao drive físico especificado. Se B e D variarem de 1 a 7, então uma nova atribuição será feita. Se D for 0 e B conter de 1 a 7, a atribuição do drive lógico respectivo será cancelada. Se D for FFH e B conter de 1 a 7, o número do drive lógico especificado em B simplesmente retornará em D.

GENV (6BH)
Função: Ler item externo.
Entrada: HL - apontador para o nome da string em ASCII.
DE - apontador do buffer para valor.
B - tamanho do buffer.
Retorno: A - código de erro (se for 0, não houve erro).
DE - apontador para o buffer preenchido.
Nota: Essa função lê o valor corrente do item externo cujo nome é apontado pelo registrador HL. Se o tamanho do buffer for pequeno, o valor de retorno será truncado, com o último caractere valendo 00H. Um buffer de 255 bytes sempre será suficiente.

SENV (6CH)
Função: Setar item externo.
Entrada: HL - apontador para o nome em ASCII.
DE - apontador para o valor a ser setado.
Retorno: A - código de erro (se for 0, não houve erro).
Nota: Essa função seta um novo item externo. A string de valor não pode conter mais de 255 caracteres e deve ser terminada com um byte 00H. Se a string de valor for nula, o item externo será removido.

FENV (6DH)
Função: Procurar item externo.
Entrada: DE - número o 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.
Nota: Essa função é usada para procurar o item externo cujo número está no registrador DE. O primeiro item corresponde a DE=1. O nome do item externo especificado em DE retornará no buffer apontado por HL, sendo o último caractere um byte 00H.

DSKCHK (6EH)
Função: Ativar ou desativar chegagem de disco.
Entrada: A - 00H - ler o valor de checagem do disco.
01H - setar o valor de checagem do disco.
B - 00H - ativa a chegagem (se A=01H)
01H - desativa a checagem (se A=01H)
Retorno: A - código de erro (se for 0, não houve erro).
B - valor de checagem do disco corrente.
Nota: Se A=00H, o valor de checagem do disco corrente retornará em B. Se B for 00H, a checagem do disco está ativa; se for outro valor, a checagem do disco estará inativa. O valor default é ativa. 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. Se estivar inativa, isso não ocorrerá.

DOSVER (6FH)
Função: Ler o número da versão do MSXDOS.
Entrada: Nenhuma.
Retorno: A - código de erro (se for 0, não houve erro)
BC - versão do DOS Kernel
DE - versão do MSXDOS.SYS.
Nota: 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, é necessário verificar primeiro se houve algum erro (A≠0). Se houver erro, o MSXDOS não está totalmente instalado. Se não houver erro, é necessário verificar o registrador B. Se for menor que 2, a versão é anterior à 2.00 e os valores de C e DE são indefinidos. Se B for igual ou maior que 2, os valores de BC e DE serão válidos.

REDIR (70H)
Função: Ler ou setar o estado de redirecionamento.

Entrada: A - 00H = ler estado de redirecionamento;
01H = setar estado de redirecionamento.
B - novo estado: b0 = entrada standard;
b1 = saída standard.

Nota: Essa função foi implementada primariamente para rotinas de erro de disco e outros caracteres e I/O que devem ser redirecionados. As funções 01H a 0BH normalmente referem ao console, mas elas podem ser redirecionadas para arquivos em disco. O efeito dessa função é temporário, no caso de A=01H e B=00H na entrada. Isso protegerá subseqüentes chamadas das funções 01H a 0BH, que voltarão a ser direcionadas normalmente ao console. Se necessário, as funções podem ser redirecionadas novamente.

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.

1.7.1 - ÁREA DE SISTEMA PARA O MSXDOS1

Essa seção descreve as variáveis de sistema que são usadas pelo MSXDOS1. Alguns endereços foram omitidos pelo desconhecimento da função dos mesmos.

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.

F1C9H, 24
Rotina para impressão na tela de uma string terminada por "$".
DE → endereço inicial da string.

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.

FIF4H, 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, 11
Número do dispositivo atual:
PRN = -5; LST = -4; NUL = -3; AUX = -2; CON = -1.

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.

F23FH, 4
Número de 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 default (0=A:, 1=B:, etc)

F248H, 1
Dia

F249H, 1
Mês

F24AH, 2
Ano-1980

F24CH, 2
Hora

F24EH, 1
Dia da semana

F24FH, 3
Jump da 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.

F255H, 3
Hook da rotina que repara nome de arquivo.

F258H, 3
Hook da rotina de procura de diretório.

F25BH, 3
Hook da rotina que incrementa a entrada do diretório (última entrada em A).

F25EH, 3
Hook da rotina que calcula o próximo setor do diretório.

F261H, 3
Hook da rotina que repara nome de arquivo.

F264H, 3
Hook da rotina da função ´OPEN´.

F267H, 3
Hook desconhecido.

F26AH, 3
Hook da rotina ´GETDPB´ da interface de disco.

F26DH, 3
Hook da rotina da função ´CLOSE´.

F270H, 3
Hook da rotina da função ´RDABS´ (HL=DMA, DE=setor, B=nº setores).

F273H, 3
Hook da rotina de manipulação de erro no acesso ao disco.

F276H, 3
Hook da rotina da função ´WRABS´.

F279H, 3
Hook da rotina da função ´WRABS´ (HL=DMA, DE=setor, B=nº setores).

F27CH, 3
Hook da rotina de multiplicação (HL = DE * BC).

F27FH, 3
Hook da rotina de divisão (BC = BC / DE; HL = resto).

F282H, 3
Hook da rotina padrão em 4989H na ROM da interface de disco.

F285H, 3
Hook da rotina padrão em 49B1H na ROM da interface de disco.

F288H, 3
Hook da rotina padrão em 4A36H na ROM da interface de disco.

F28BH, 3
Hook da rotina padrão em 4A46H na ROM da interface de disco.

F28EH, 3
Hook da rotina padrão em 4B56H na ROM da interface de disco.

F291H, 3
Hook da rotina padrão em 4BE2H na ROM da interface de disco.

F294H, 3
Hook da rotina padrão em 4C22H na ROM da interface de disco.

F297H, 3
Hook da rotina padrão em 4C97H na ROM da interface de disco.

F29AH, 3
Hook para o endereço 4D65H da rotina padrão em 4D05H na ROM da interface de disco.

F29DH, 3
Hook da rotina padrão em 4D8CH na ROM da interface de disco.

F2A0H, 3
Hook da rotina padrão em 4E48H na ROM da interface de disco.

F2A3H, 3
Hook da rotina padrão em 4EDBH na ROM da interface de disco.

F2A6H, 3
Hook da rotina padrão em 4F12H na ROM da interface de disco.

F2A9H, 3
Hook da rotina padrão em 4F9EH na ROM da interface de disco.

F2ACH, 3
Hook da rotina da função ´BUFIN´.

F2AFH, 3
Hook da rotina da função ´CONOUT´.

F2B2H, 3
Hook da rotina padrão em 5496H na ROM da interface de disco.

F2B5H, 3
Hook da rotina de identificação do mês de fevereiro (28/29 dias).

F2B8H, 1
Número da entrada atual do diretório.

F2B9H, 11
Nome de arquivo da última entrada do diretório lida.

F2C4H, 1
Byte de atributos do arquivo da última entrada do diretório lida.

F2CFH, 2
Hora do arquivo da última entrada do diretório lida.

F2D1H, 2
Data do arquivo da última entrada do diretório lida.

F2D3H, 2
Cluster inicial do arquivo da última entrada do diretório lida.

F2D5H, 4
Tamanho do arquivo da última entrada do diretório lida.

F2DCH, 1
Flag (0=falso).

F2E1H, 1
Drive atual para escrita e leitura absoluta de setores.

F2FEH, 2
Subcontador do contador regressivo para o drive.

F304H, 2
Armazena o valor do registrador SP (Stack Pointer).

F306H, 1
Drive default 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, 20
Área usada pelo modo Kanji.

F323H, 2
Endereço do manipulador de erro de disco.

F325H, 2
Endereço do manipulador das teclas CTRL+C.

F327H, 5
Hook da rotina ´AUXINP´ (A=byte lido do dispositivo AUX).

F32CH, 5
Hook da rotina ´AUXOUT´ (A=byte a ser enviado ao dispositivo AUX).

F331H, 5
Hook da rotina de manipulação das funções do BDOS.

F336H, 1
Status do pressionamento das teclas CTRL ou STOP.

F337H, 5
Status do pressionamento das teclas CTRL+STOP (3=ambas estão sendo pressionadas.

F338H, 1
Flag para indicar a presença de relógio interno (0=não; outro valor, sim).

F339H, 7
Rotina usada pelo relógio interno.

F340H, 1
Estado da recarga do MSXDOS.

F341H, 1
Slot da página 0 da RAM (formato igual a RDSLT - 000CH/BIOS).

F342H, 1
Slot da página 1 da RAM (formato igual a RDSLT - 000CH/BIOS).

F343H, 1
Slot da página 2 da RAM (formato igual a RDSLT - 000CH/BIOS).

F344H, 1
Slot da página 3 da RAM (formato igual a RDSLT - 000CH/BIOS).

F345H, 1
Número de buffers livres.

F346H, 1
Flag para indicar a presença do MSXDOS no disquete.
(0=não; outro valor, sim)

F347H, 1
Número total de drives lógicos no sistema.

F348H, 1
ID do slot do DOS Kernel (formato igual a RDSLT - 000CH/BIOS).

F349H, 2
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
ClearAddr MSXDOS (início para o COMMAND.COM).

F34DH, 2
Apontador para uma cópia da FAT do drive default (1,5 Kbytes).

F34FH, 2
Apontador para uma área de 512 bytes usada como DTA do Disk BASIC.

F351H, 2
Apontador para um buffer de 512 bytes usado para transferência de setores do disco.

F353H, 2
Apontador para o FCB do arquivo atual.

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:.

F365H, 3
Jump da rotina de leitura de slots primários.

F368H, 3
Jump para a rotina de troca do DOS Kernel (BDOS) na página 1 (não disponível a partir do Disk BASIC)

F36BH, 3
Jump para a rotina de troca da RAM na página 1 (não disponível a partir do Disk BASIC).

F36EH, 3
Jump para LDIR da RAM na página 1 (não disponível a partir do Disk BASIC).

F371H, 3
Jump para a rotina de entrada do dispositivo auxiliar (AUXINP).

F374H, 3
Jump para a rotina de saída do dispositivo auxiliar (AUXOUT).

F377H, 3
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.

F37AH, 3
Jump para a rotina do comando ´BSAVE´.

F37DH, 3
Jump para manipulador dos comandos do BDOS.

1.7.2 - ÁREA DE SISTEMA PARA O MSXDOS2

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.

F1C9H, 24
Rotina para impressão na tela de uma string terminada por "$".
DE → endereço inicial da string.

F1E2H, 6
Rotina para abortar o programa em caso de erro.

F1E5H, 3
Jump para o manipulador de interrupção (somente durante o processamento das funções do BDOS).

F1E8H, 3
Jump para a rotina do BIOS ´RDSLT´ (somente durante o processamento das funções do BDOS).

F1EBH, 3
Jump para a rotina do BIOS ´WRSLT´ (somente durante o processamento das funções do BDOS).

F1EEH, 3
Jump para a rotina do BIOS ´CALSLT´ (somente durante o processamento das funções do BDOS).

F1F1H, 3
Jump para a rotina do BIOS ´ENASLT´ (somente durante o processamento das funções do BDOS).

F1F4H, 3
Jump para a rotina que checa o nome arquivo (HL ← apontador para, o nome de arquivo, DE ← resultado).

F1F7H, 3
Jump para troca para o "Modo DOS" (páginas 0 e 2 para os segmentos do sistema).

F1FAH, 3
Jump para troca para o "Modo Usuário".

F1FDH, 3
Jump para a seleção de 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 ´RDSLT´.

F209H, 3
Jump para a rotina do BIOS ´WRSLT´.

F20CH, 3
Jump para a rotina do BIOS ´CALSLT´.

F20FH, 3
Jump para a rotina do BIOS ´CALLF´.

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 na página 0.

F21BH, 3
Jump para a rotina que lê segmento atual de 16 Kbytes na página 0.

F21EH, 3
Jump para a rotina que habilita segmento de 16 Kbytes na página 1.

F221H, 3
Jump para a rotina que lê segmento atual de 16 Kbytes na página 1.

F224H, 3
Jump para a rotina que habilita segmento de 16 Kbytes na página 2.

F227H, 3
Jump para a rotina que lê segmento atual de 16 Kbytes 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 na página 3.

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 default (0=A:, 1=B:, etc.).

F248H, 1
Dia

F249H, 1
Mês

F24AH, 2
Ano-1980

F24CH, 2
Hora

F24EH, 1
Dia da semana

F24FH, 3
Jump para a rotina que apresenta a mensagem “Insert disk for drive”.
A ← número dor 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.

F270H, 3
Hook da rotina de leitura direta de setores (função 2FH do BDOS).
HL ← DMA, DE ← setor inicial, B ← número de setores.

F279H, 3
Hooh da rotina de escrita direta de setores (função 30H do BDOS).
HL ← DMA, DE ← setor inicial, B ← número de setores.

F27CH, 3
Hook da rotina de multiplicação (HL = DE * BC).

F27FH, 3
Hook da rotina de divisão (BC = BC / DE; HL = resto).

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

F2B6H, 2
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.

F2C0H, 5
Segundo hook da rotina de interrupção (usado pela Disk-ROM).

F2C5H, 2
Endereço da tabela de mapeamento.

F2C7H, 1
Bloco atual da mapper na página 0.

F2C8H, 1
Bloco atual da mapper na página 1.

F2C9H, 1
Bloco atual da mapper na página 2.

F2CAH, 1
Bloco atual da mapper na página 3 (não pode ser trocado).

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).

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.

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).

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).

F30DH, 1
Flag de verificação do disco (00H=off, FFH=on).

F313H, 1
Versão do DOS2 (ex.: 22H = v2.2).

F33DH, 3
Jump para o comando BASIC ´LEN´ (acesso aleatório a arquivos).

F341H, 1
Slot da página 0 da RAM (formato igual a ´RDSLT´ - 000CH/Main).

F342H, 1
Slot da página 1 da RAM (formato igual a ´RDSLT´ - 000CH/Main).

F343H, 1
Slot da página 2 da RAM (formato igual a ´RDSLT´ - 000CH/Main).

F344H, 1
Slot da página 3 da RAM (formato igual a ´RDSLT´ - 000CH/Main).

F377H, 3
Jump para o segmento de sistema na página 0. HL ← endereço.

F37AH, 3
Jump secundário para o segmento de sistema na página 0.

F37DH, 3
Jump para o manipulador de funções do BDOS.

1.7.3 - ÁREA DE SISTEMA PÚBLICA (OFICIAL)

H.PROM (F24FH, 3)
Hook para a rotina que apresenta a mensagem “Insert disk for drive”.
A ← número do drive (41H=A:, 42H=B:, etc).

DISKVE (F323H, 2)
Endereço do manipulador de erro de disco.

BREAKV (F325H, 2)
Endereço do manipulador das teclas CTRL+C.

RAMAD0 (F341H,1)
Slot da página 0 da RAM (formato igual a RDSLT - 000CH/BIOS).

RAMAD1 (F342H,1)
Slot da página 1 da RAM (formato igual a RDSLT - 000CH/BIOS).

RAMAD2 (F343H,1)
Slot da página 2 da RAM (formato igual a RDSLT - 000CH/BIOS).

RAMAD3 (F344H,1)
Slot da página 3 da RAM (formato igual a RDSLT - 000CH/BIOS).

?????? (F346H,1)
Flag para indicar a presença do MSXDOS no disquete.
(0=não; outro valor, sim)

MASTER (F348H,1)
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).

HIMSAV (F349H,2)
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.

SECBUF (F34DH,2)
Apontador para uma cópia da FAT do drive default (1,5 Kbytes).

BUFFER (F34FH,2)
Apontador para uma área de 512 bytes usada como DTA do Disk BASIC.

DIRBUF (F351H,2)
Apontador para um buffer de 512 bytes usado para transferência de setores do disco.

FCBBASE (F353H,2)
Apontador para o FCB do arquivo atual.

DPBLIST (F355H,16)
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:

BLDCHK+1 (F378H,2)
Endereço da rotina do manipulador do comando ´BLOAD´.

DRVTBL (FB21H,8)
Tabela que contém o número de drives conectados e os slots das interfaces de disco.
FB21H,1 → número de drives lógicos conectados na primeira interface
FB22H,1 → slot da ROM da primeira interface de disco
FB23H,1 → número de drives lógicos conectados na segunda interface

FB24H,1 → slot da ROM da segunda interface de disco
FB25H,1 → número de drives lógicos conectados na terceira interface
FB26H,1 → slot da ROM da terceira interface de disco
FB27H,1 → número de drives lógicos conectados na quarta interface
FB28H,1 → slot da ROM da quarta interface de disco

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 do BIOS, que está ativa normalmente tanto sob o MSXDOS quanto sob o BASIC. A seqüência de chamada deve ser a seguinte:

```
CALSLT: EQU  0001CH    ;endereco da rotina CALSLT
HPHYD:  EQU  0FFA7H    ;end. do hook da rotina PHYDIO
CALBAS: EQU  04022H    ;endereco 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 do BIOS.

As rotinas estão listadas de acordo com o seguinte formato:

NOME (endereço)
Função: Função da rotina.
Entrada: Parâmetros para a chamada da rotina.
Saída: Parâmetros retornados pela rotina.

Todos os registradores são modificados pelas rotinas; portanto é necessário salvar na pilha os registradores que não devem ser modificados.

1.8.1 - DESCRIÇÃO DAS ROTINAS DA INTERFACE

DISKIO (4010H / Interface de disco)
Função: Leitura/escrita direta de setores.

Entrada: HL - apontador para a TPA.
DE - número o primeiro setor a ser lido ou escrito.
B - número de setores a ler ou escrever.
C - ID da formatação do disco (F0H = hard disk).
A - número do drive (00H=A:, 01H=B:, etc.).
Flag CY - resetada para efetuar leitura;
setada para efetuar escrita.

Saída: B - número de setores efetivamente transferidos.
A - código de erro (conforme listagem abaixo).
Flag CY - setada em caso de erro.
resetada se não houve erro.

Nota: *Os códigos de erro retornados em A são os seguintes:*
0 - protegido contra escrita;
2 - não pronto;
4 - erro de CRC (setor não acessível);
6 - erro de busca;
8 - cluster não encontrado;
10 - erro de escrita;
12 - erro de disco (ou drive não SCSI para MSXDOS2);
Códigos de erro adicionados para o MSXDOS2:
18 - disco não DOS;
20 - versão do MSXDOS incorreta;
22 - disco não formatado;
24 - disco trocado;
26 - erro de usuário 10;
Restantes: erro de disco.

DSKCHG (4013H / Interface de disco)

Função: Checar o estado de troca do disco.

Entrada: A - número do drive
B - ID de formatação do disco (00H para MSXDOS2)
C - ID de formatação do disco (F0H = hard disk - somente para MSXDOS2)
HL - apontador para o DPB respectivo

Saída: A - código de erro (igual à listagem acima)
B - se não houve erro: 00H - desconhecido;
01H - disco não trocado;
FFH - disco trocado.
Flag CY - setada em caso de erro;
resetada se não houve erro.

Nota: Se o disco foi ou será (desconhecido) trocado, o setor de boot (ID de formatação) deverá ser lido e o novo DPB deverá ser transferido com a rotina GETDPB (4016H).

GETDPB (4016H / Interface de disco)

Função: Ler o DPB do disco.

Entrada: A - número do drive (00H=A:, 01H=B:, etc)
B - primeiro byte da FAT (ID do disco)
C - ID de formatação do disco (F0H = hard disk - somente para MSXDOS2)
HL - apontador para o DPB respectivo
Saída: A - código de erro (igual à listagem acima)
Flag CY - setada em caso de erro;
resetada se não houve erro.

CHOICE (4019H / Interface de disco)
Função: Mensagem para formatação do disco.
Entrada: Nenhuma.
Saída: HL - endereço do byte 00H que termina a string com o texto que contém a mensagem para formatação. Se não houver escolha (somente um tipo de formatação é suportado), HL retorna com 0000H.

DSKFMT (401CH / Interface de disco)
Função: Formatar um disco.
Entrada: A - escolha especificada pelo usuário (rotina CHOICE).
D - número do drive (00H=A:, 01H=B:, etc).
HL - apontador para o início da área de trabalho.
BC - tamanho da área de trabalho.
Saída: A - código de erro (conforme listagem abaixo).
Flag CY - setada em caso de erro;
resetada se não houve erro.
Nota: Na formatação, o boot é escrito no setor 0, toda a FAT é limpa e a área do diretório é preenchida com zeros.
Códigos de erro:
0 - protegido contra escrita;
2 - não pronto;
4 - erro de CRC (setor não formata);
6 - erro de busca;
8 - cluster não encontrado;
10 - falha de escrita (ou drive não SCSI para MSXDOS2);
12 - parâmetro incorreto;
14 - memória insuficiente;
16 - outros erros.

CALBAS (4022H / Interface de disco)
Função: Chamar o interpretador BASIC.
Entrada: Nenhuma.
Saída: Nenhuma.

FORMAT (4025H / Interface de disco)
Função: Formatar um disco apresentando mensagem.

Entrada: Nenhuma.
Saída: Nenhuma.

STPDRV (4029H / Interface de disco)
Função: Parar os motores dos drives.
Entrada: Nenhuma.
Saída: Nenhuma.

SLTDOS (402DH / Interface de disco)
Função: Retorna o ID do slot do DOS Kernel.
Entrada: Nenhuma.
Saída: A - ID do slot (formato igual a RDSLT)

HIGMEM (4030H / Interface de disco)
Função: Retorna o endereço mais alto disponível da RAM.
Entrada: Nenhuma.
Saída: HL - endereço mais alto disponível da RAM.

BLKDOS (40FFH / Interface de disco - somente MSXDOS2)
Função: Retorna o bloco corrente do DOS2.
Entrada: Nenhuma.
Saída: A - número do bloco corrente.
Nota: Os 64 Kbytes da ROM do DOS Kernel 2 são divididos em 4 segmentos de 16 Kbytes. Esses segmentos podem estar ativos somente na página física 1. Portanto, eles são trocados constantemente durante o processamento. O valor retornado pode ser 0, 1, 2 ou 3.

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 do BIOS estão disponíveis nessa área. A página-zero é mapeada como descrito abaixo.

WBOOT[30] (0000H, 3)
Warm boot. Ao se chamar essa rotina, promove-se uma partida a quente do MSXDOS (o MSXDOS é recarregado sem resetar o micro).

DRIVE (0004H, 1)
Esse byte armazena o drive default (00H=A:, 01H=B:, etc)

Nota 30: As rotinas descritas devem ser chamadas exatamento como se faz para o BIOS (instrução CALL ou RST), exceto a rotina WBOOT (0000H), que deve ser chamada com um JP 0000H.

BDOS[30] (0005H, 3)
Ponto de entrada das rotinas do BDOS.

RDSLT[30] (000CH, 8)
Essa rotina lê um byte em qualquer slot. É exatamente igual à rotina RDSLT do BIOS.

WRSLT[30] (0014H, 8)
Essa rotina escreve um byte em qualquer slot. É exatamente igual à rotina WRSLT do BIOS.

CALSLT[30] (001CH, 8)
Chama uma rotina em qualquer slot. No presente caso, pode ser usada para chamar outras rotinas do BIOS. É exatamente igual à rotina CALSLT do BIOS.

ENASLT[30] (0024H, 8)
Habilita uma página em qualquer slot. É exatamente igual à rotina RDSLT do BIOS.

CALLF[30] (0030H, 8)
Chama uma rotina em qualquer slot, com parâmetos em linha. No presente caso, pode ser usada para chamar outras rotinas do BIOS. É exatamente igual à rotina CALLF do 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, ?)
Endereço inicial da DTA.

A área compreendida entre 0080H e 00FFH 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).

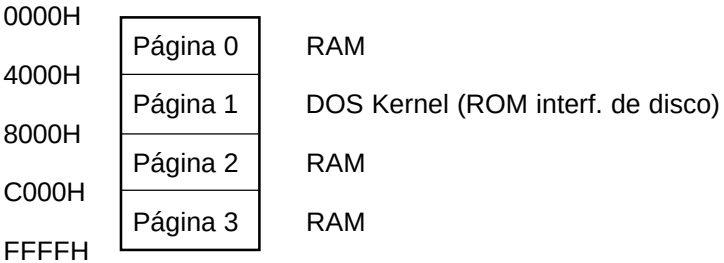
1.10 - O SETOR DE BOOT

Em todos os disquetes (e outros dispositivos de disco) existe o "setor de boot", que sempre é o setor 0 dos disco. Toda vez que o micro for resetado, 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, senão 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 na página seguinte.



1.10.1 - 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
```

2 - O UZIX

O Uzix é um novo 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.

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:

- Nome
- Conteúdo
- Outros dados de identificação, armazenados numa estrutura chamada inode (information node).

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 que identificam, além do nome, o 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.

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

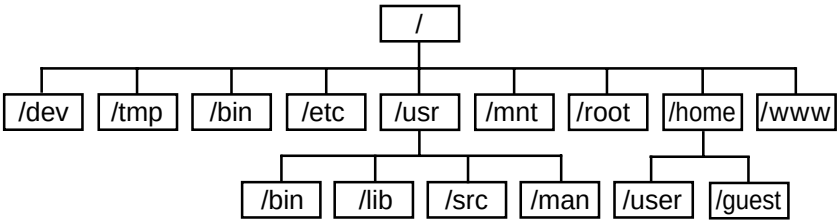
Para que se possa organizar os arquivos, existem os arquivos diretórios, ou simplesmente diretórios. No MSXDOS, o diretório é armazenado de forma bem diferente de outros arquivos, mas aqui o diretório é apenas um arquivo. Esses diretórios contém uma lista dos nomes de arquivos, que podem ser ordinários, 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.

2.1.2 - ESTRUTURA HIERÁRQUICA

No Uzix, também temos o conceito de subdiretórios. 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. 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

2.2 - 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/renomear 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".

2.3 - ESTRUTURA DOS ARQUIVOS NO DISCO

Os arquivos Uzix são estruturados no disco em 4 grupos: setor de boot, superblock, inodes e blocos de dados.

2.3.1 - 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.

2.3.2 - 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 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 de 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). O filesystem é o próprio sistema de arquivos do Uzix. Eles contêm o Kernel do Uzix e são acessados pelo bootstrap secundário para inicializar o Uzix a partir do disco.

2.3.3 - 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.

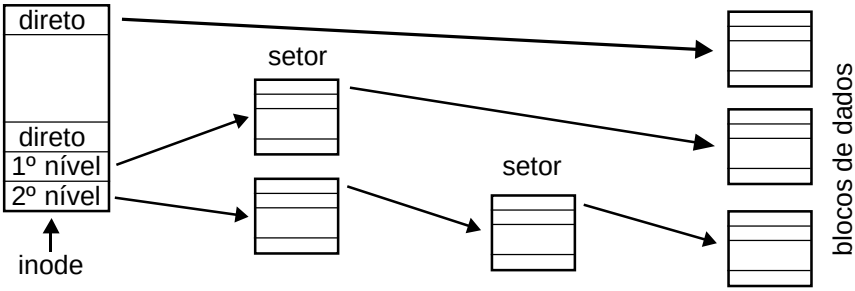
A data e a hora são armazenadas da mesma forma que para o MSXDOS. Tanto o usuário quanto o grupo e as respectivas senhas são armazenadas em um arquivo próprio. As senhas são criptografadas. Podem haver até 256 grupos cadastrados no sistema com até 256 usuários cada um.

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 * 65536). Como existem 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 Kbytes * 256 apontadores), a capacidade máxima endereçável pelo inode.

O esquema de apontadores está ilustrado na página seguinte. 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 arquivos com 135 Kbytes, são necessárias duas operações. Acima disso, até o limite de 32 Mbytes, três operações são requeridas.

2.3.4 - 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.

2.3.5 - 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.

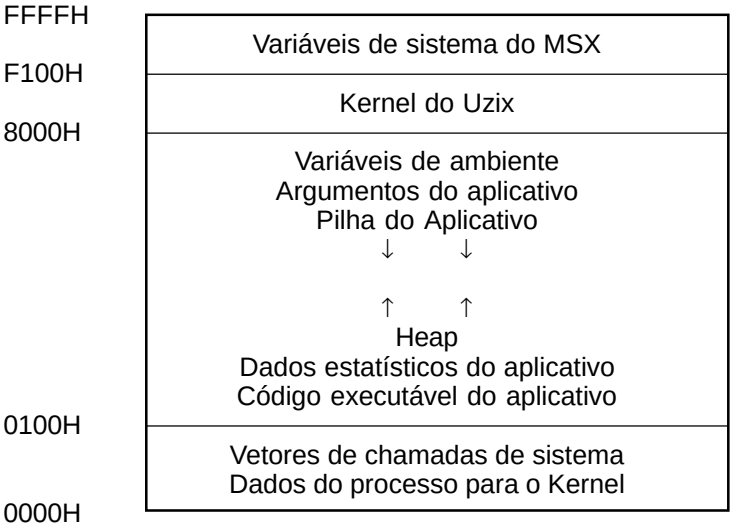
Todo dispositivo (de hardware ou software) deve sofrer o processo de montagem para que o sistema possa reconhecê-lo, muito embora isso seja processado de maneiras bem diversas de acordo com o dispositivo a ser montado. A unidade primária de disco é montada automaticamente durante a inicialização do sistema.

2.4 - MAPEAMENTO DE MEMÓRIA

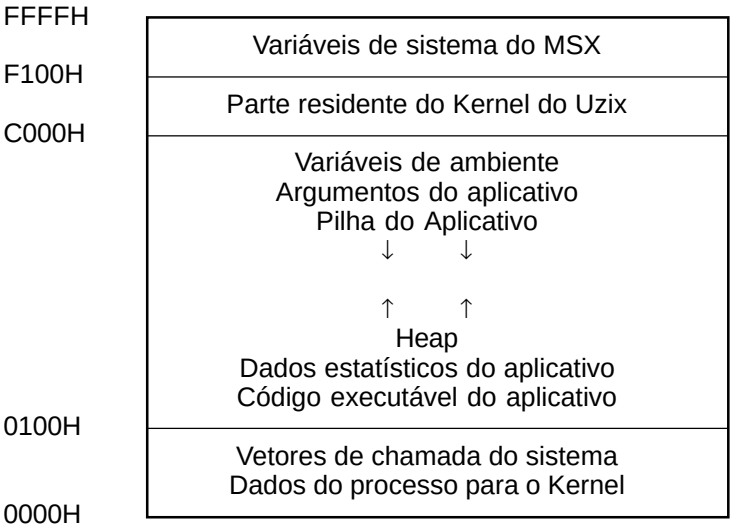
O mapeamento de memória é a maior diferença entre o Uzix 1.0 e

2.0. Na verdade, o Uzix 2.0 superou em muito o 1.0 e exige as mesmas características de hardware do 1.0; portanto é aconselhável o desenvolvimento de software somente para o 2.0

O mapeamento de memória para o Uzix 1.0 está ilustrado abaixo.



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 memoria 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.

2.5 - 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.

3 - ACESSO DIRETO AO FDC

Muito embora não seja recomendado, é possível o acesso direto ao disco, sobrepujando ao sistema operacional. 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 status
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.

3.1 - 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 da página anterior 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 reset
i2	1 = pulso de índice
i1	1 = transição pronto para não pronto
i0	1 = transição não pronto para pronto

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 e r0 = 0) para drives de disquetes de 3½". 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 status será setado.

Quando V=0, ao completar uma busca, o número da trilha não será verificado. Esse modo deve ser habilitado para disquetes não formatados. 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 status é resetado 0. Quando a CPU ler o registrador de status, o sinal de interrupção é resetado.

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 posisiona 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-Out 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.

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 status 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 status 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 ascendente 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 status é setado, e quando o campo ID é encontrado e contiver a trilha correta, setor correto, lado correto e CRC correto, 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 status 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 adress mark” encontrado no campo de dados será gravado no registrador de status (bit 5).

Comando Write Sector

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de status é setado, e quando o campo ID é encontrado e contiver a trilha correta, setor correto, lado correto e CRC correto, 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 status 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 status 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.

COMANDOS TIPO III

Os comandos tipo III são usados para acessar os headers das trilhas e setores do disco.

Comando Read Adress

Ao receber esse comando, o cabeçote é posicionado, o bit “busy” do registrador de status 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 por 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 status 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, se for desejável. Ao final do comando, uma interrupção é gerada e o bit “busy” do registrador de status é resetado.

Comando Read Track

Ao receber esse comando, o cabeçote é posicionado e o bit "busy" do registrador de status 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 status 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 status 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 seqüência 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 conseqüência, 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.

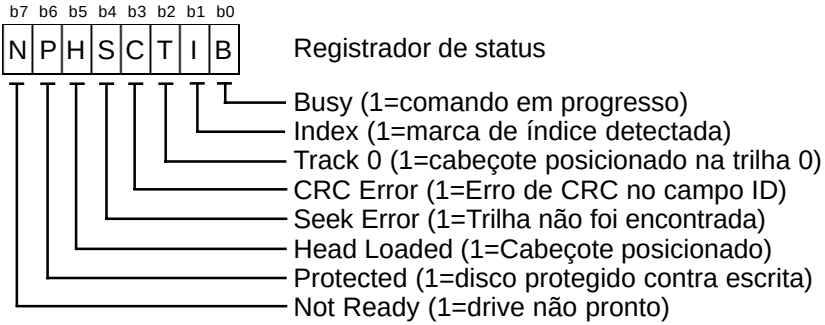
COMANDO TIPO IV

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 status será resetado.

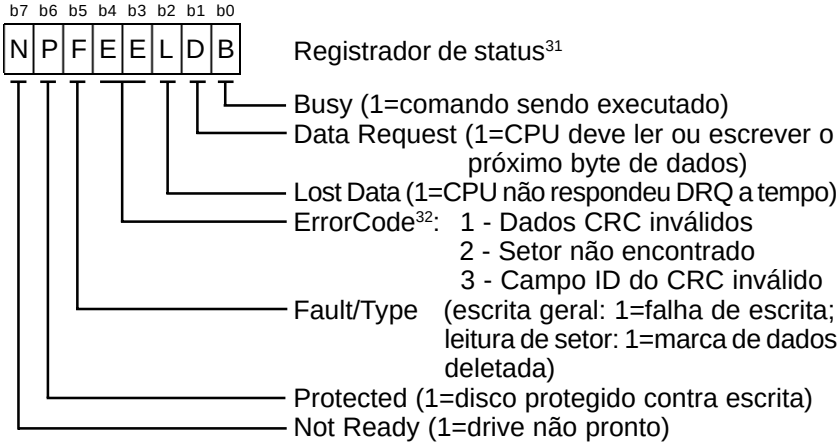
3.2 - O REGISTRADOR DE STATUS

Ao receber qualquer comando, à exceção do comando Force Interrupt, o bit "busy" é setado e os outros bits do registrador de status são atualizados ou limpos para o novo comando. O usuário tem a opção de ler o registrador de status através do programa ou usar a linha DRQ juntamente com DMA ou interrupção. Quando o registrador de dados é lido, o bit DRQ no registrador de status e a linha DRQ são automaticamente resetadas. 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.

STATUS PARA COMANDOS TIPO I



STATUS PARA COMANDOS TIPO II e III



Nota 31: Os bits 1~6 são resetados quando atualizados.
Nota 32: Códigos não válidos para escrita ou leitura de trilhas.

STATUS PARA COMANDOS TIPO IV

Se um comando "Force Interrupt" for recebido durante a execução de outro comando, o bit 0 (busy) do registrador de status é resetado 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 status é resetado e os outros bits serão atualizados ou limpos. Nesse caso, o registrador de status se comporta como nos comandos tipo I.

3.3 - FUNÇÕES 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.

3.4 - 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 da página seguinte mostra a seqüência 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 (cabeçalho de trilha)
80 x 4EH
12 x 00H
03 x F6H (escreve C2)
01 x FCH (marca de índice)
50 x 4EH
Campo ID do setor
12 x 00H
03 x F5H (escreve A1, preset CRC)
01 x FEH (marca ID de endereço)
01 x número da trilha
01 x número do lado
01 x número do setor
01 x 01 (tamanho do setor - 256 bytes)
01 x F7H (escreve 2 bytes CRC)
22 x 4EH
Campo de dados do setor
12 x 00H
03 x F5H (escreve A1, preset CRC)
01 x FBH (marca da dados de endereço)
256 x dados do setor
1 x F7H (escreve 2 bytes CRC)
54 x 4EH
Fim de trilha (preencher bytes não usados)
... x 4EH

3.5 - 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, pois elas são acessadas diretamente por portas de I/O. Esse tipo de acesso foi utilizado somente em interfaces brasileiras. O acesso padrão para o MSX é o por memória.

ENDEREÇOS PARA ACESSO POR MEMÓRIA (Padrão)

7FF8H	R	Registrador de status
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 ao invés 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 status
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 status
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 status, 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

Esse capítulo descreve alguns dispositivos que não foram descritos nos capítulos anteriores.

1 - O RELÓGIO E A SRAM

Nos micros MSX2 e superior, 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.

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”.

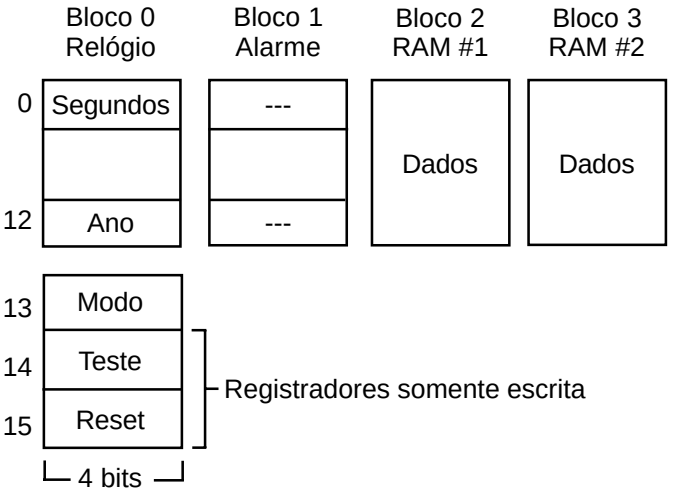
MEMÓRIA

- 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.

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 na página seguinte.



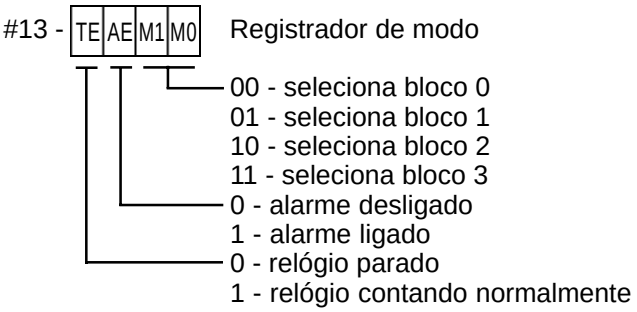
1.2.1 - O REGISTRADOR DE MODO (#13)

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.



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 16384 Hz são inseridos diretamente nos registradores de dia, hora, minuto e segundo.

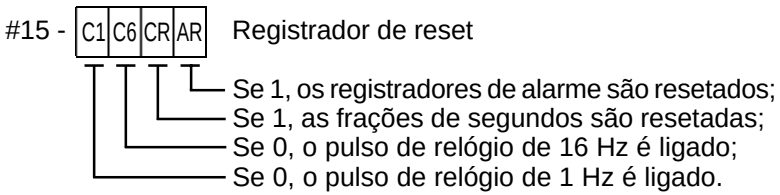
1.2.3 - O REGISTRADOR DE RESET (#15)

Esse registrador tem as seguintes funções:

Para resetar o alarme, basta setar o bit 0 em 1; todos os registradores de alarme serão resetados.

O bit 1, quanto setado em 1, causa o reset das frações do contador de segundos. Essa função é útil para acertar os segundos corretamente.

Setando o bit 2 em 0, o pulso do relógio de 16 Hz é ativado e setando o bit 3 em 0, é ativado o pulso de 1 Hz.



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.

Bloco 0 - Relógio

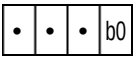
	Bits →	3	2	1	0
0	Seg. 1º dígito	x	x	x	x
1	Seg. 2º dígito	•	x	x	x
2	Min. 1º dígito	x	x	x	x
3	Min. 2º dígito	•	x	x	x
4	Hor. 1º dígito	x	x	x	x
5	Hor. 2º dígito	•	•	x	x
6	Dia da semana	•	x	x	x
7	Dia. 1º dígito	x	x	x	x
8	Dia. 2º dígito	•	•	x	x
9	Mês. 1º dígito	x	x	x	x
10	Mês. 2º dígito	•	•	x	x
11	Ano. 1º dígito	x	x	x	x
12	Ano. 2º dígito	•	•	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	Hor. 1º dígito	x	x	x	x
5	Hor. 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	Dia. 1º dígito	•	•	x	x
12	- - -	•	•	•	•

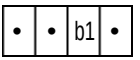
Obs.: os bits indicados com "•" devem ser sempre 0.

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.



Registrador #10 do bloco 1

0=12 horas; 1=24 horas

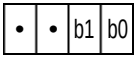


Registrador #5 do bloco 0

0=antes do meio-dia (am); 1=depois do meio dia (pm)

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.



Registrador #11 do bloco 1

bits 00 representam ano bissexto

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	bit 0
0	I D			
1	ajuste horizontal (-8 a +7)			
2	ajuste vertical (-8 a +7)			
3	———			
4	largura inicial de tela (WIDTH) - low			
5	largura inicial de tela (WIDTH) - high			
6	código da cor de fundo inicial			
7	código da cor de frente inicial			
8	código ca cor da borda inicial			
9	———	tipo impres.	click teclas	teclas funç.
10	tom do beep		volume do beep	
11	———		cor da tela inicial	
12	código nativo			

O bloco 3 pode ter três funções diferentes, dependendo do conteúdo da posição ID (registrador #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 na página seguinte.

ID = 0 → apresenta um título na tela inicial

0	0
1	1º caractere - low
2	1º caractere - high
	⋮
11	6º caractere - low
12	6º caractere - high

ID = 1 → armazena a senha (password)

0	1	
1	Uso ID = 1	
2	Uso ID = 2	
3	Uso ID = 3	
4	Senha	
5	Senha	A senha é armazenada
6	Senha	compactada em 4x4 bits
7	Senha	
8	Key cartridge flag	
9	Key cartridge value	
10	Key cartridge value	
11	Key cartridge value	
12	Key cartridge value	

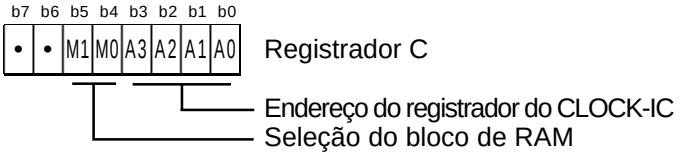
ID = 2 → armazena um novo prompt para o BASIC

0	2
1	1º caractere - low
2	1º caractere - high
	⋮
11	6º caractere - low
12	6º caractere - high

1.3 - ACESSO AO CLOCK-IC

O acesso ao relógio e à memória mantida a bateria é feito através de duas rotinas do BIOS da Sub-ROM, sendo necessário o uso de chamada inter-slot para acessá-las.

RDCLK (015FH / Sub-ROM)
Função: Ler um registrador do CLOCK-IC.
Entrada: C - endereço do CLOCK-IC

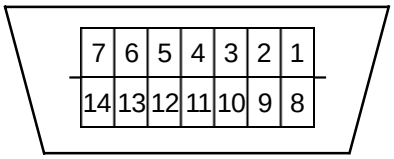


Saída: A - Dado lido. Apenas os 4 bits mais baixos são válidos.

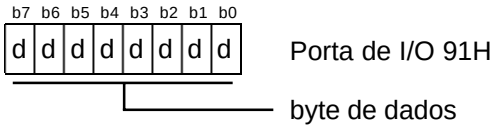
WRTCLK (01F9H / Sub-ROM)
Função: Escrever um dado em um registrador do CLOCK-IC.
Entrada: C - Endereço do CLOK-IC (igual a RDCLK).
A - Dado a ser escrito. Apenas os 4 bits mais baixos serão de fato escritos.
Saída: Nenhuma.

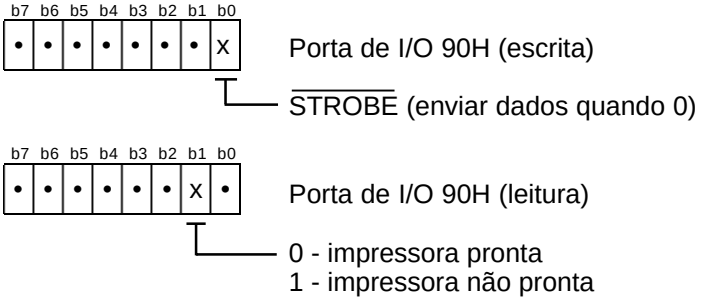
2 - INTERFACE DE IMPRESSORA

Essa seção descreve como acessar a impressora pelo BIOS e diretamente através de portas de I/O. A interface de impressora é suportada pelo BIOS, pelo BASIC e pelo DOS. O MSX usa duas portas paralelas de 8 bits 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 - $\overline{\text{STROBE}}$
2 ~ 9 - Dados (b0~b7)
11 - BUSY
14 - GND





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) - Subustitui TAB por espaços quando o conteúdo for 0; caso contrário, não substitui.

2.1 - ACESSO À IMPRESSORA

A impressora pode ser acessada tanto diretamente quanto através de rotinas do BIOS. O acesso deve ser feito preferencialmente através das rotinas do BIOS para prevenir problemas de incompatibilidade e sincronização. As rotinas do BIOS dedicadas à impressora são as seguintes:

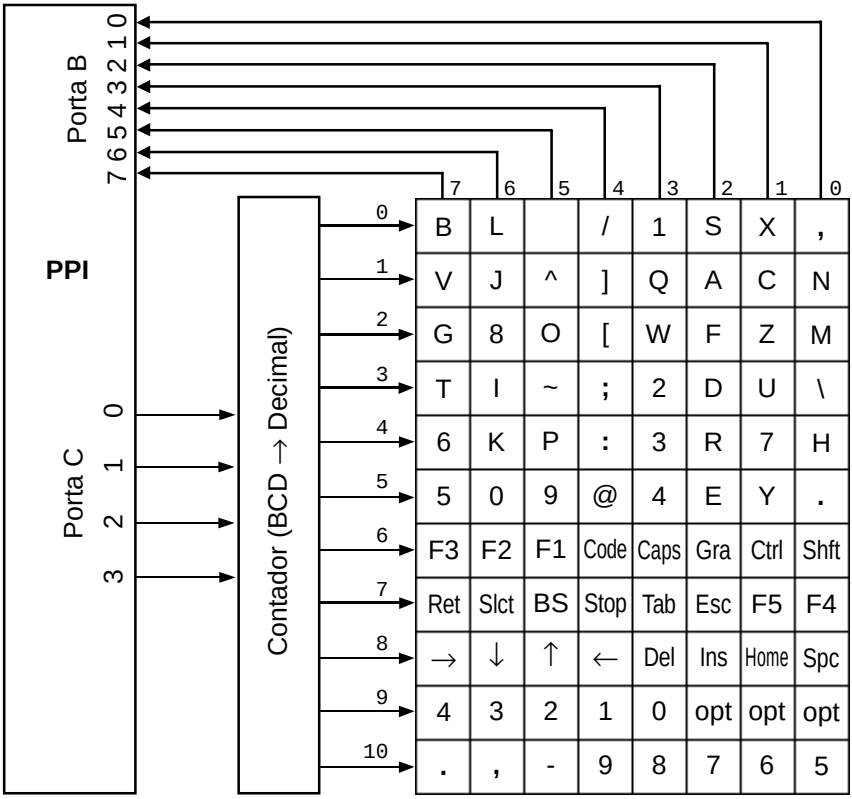
LPTOUT	(00A5H/Main)	Envia um caractere para a impressora
LPTSTT	(00A8H/Main)	Obtém o status da impressora
OUTDLP	(014DH/Main)	Envia um caractere para a impressora, com algumas diferenças em relação à LPTOUT.

A descrição detalhada dessas rotinas pode ser vista na seção “BIOS EM ROM” no capítulo 2.

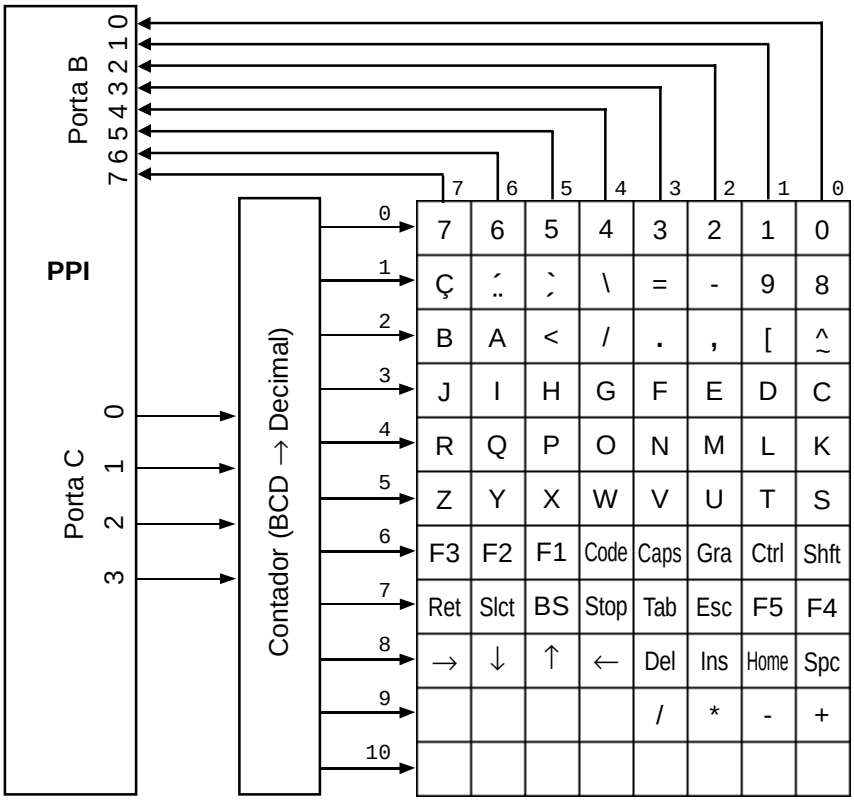
3 - INTERFACE DE TECLADO

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.



No Brasil, a matriz de teclado é diferente. Nela, só há 10 linhas e não 11, permitindo um máximo de 80 teclas. A disposição de caracteres também é bem diferente. A matriz de teclado brasileira está ilustrada na página seguinte.



A matriz apresentada é a do Hotbit 1.1 e do Expert 1.1. Para o Expert 1.0, a matriz diferia ligeiramente.

Uma observação importante é que as duas últimas linhas (9 e 10) correspondem ao teclado numérico independente. Assim, na matriz internacional, os números e alguns caracteres são decodificados separadamente para o teclado numérico. Já no Expert somente alguns caracteres são decodificados separadamente e os números correspondem aos mesmos do teclado alfanumérico.

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) do BIOS. Por se tratar de periférico lento, o acesso pelo BIOS é preferível ao acesso direto. A rotina SNSMAT está descrita na página seguinte.

SNSMAT (0141H/Main)
Função: Lê uma linha da matriz de teclado.
Entrada: A - linha da matriz a ser lida (0 a 10)
Saída: A - status da linha especificada. Quando algum bit for 0, a tecla correspondente está sendo pressionada.

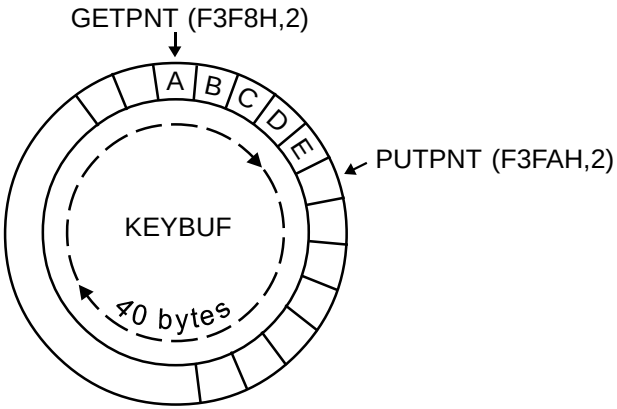
Outras rotinas relacionadas à interface de teclado são as seguintes:

CHSNS (009CH/Main) - checa o status 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 na seção "BIOS EM ROM" no capítulo 2.

3.2 - VARREDURA DE TECLADO

O MSX varre automaticamente toda a matriz de teclado 60 vezes por segundo, 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.

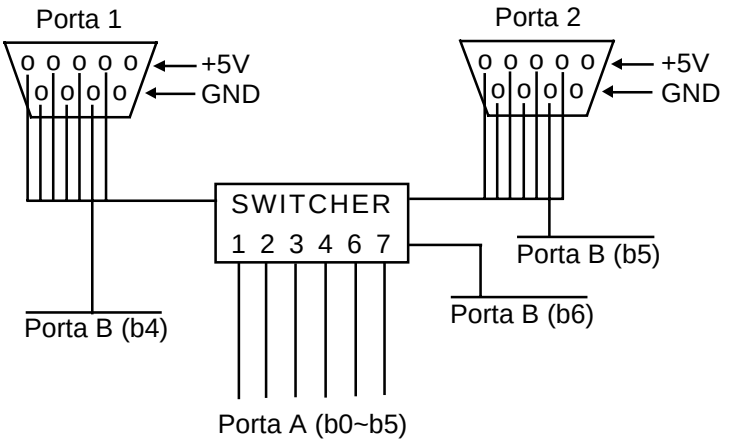


GETPNT aponta para o próximo caractere a ser obtido pela rotina CHGET e PUTPNT aponta para a próxima posição livre no buffer, a ser preenchida com o valor da próxima tecla pressionada.

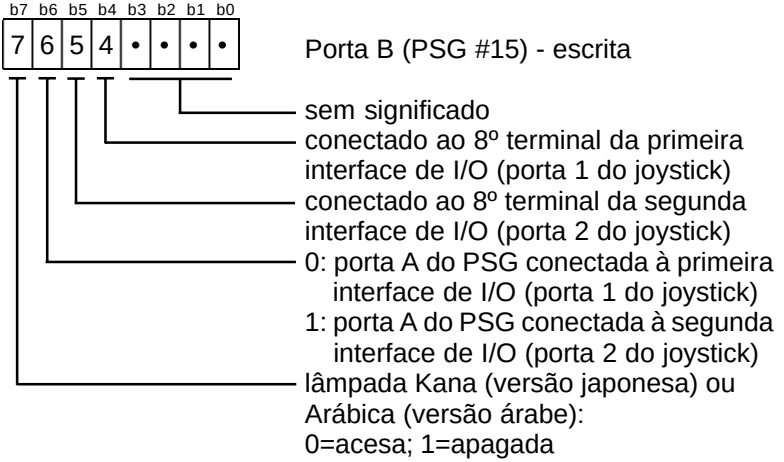
4 - INTERFACE UNIVERSAL DE I/O

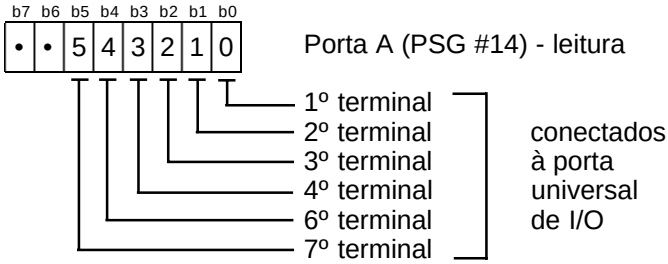
Como descrito no capítulo 5, o PSG tem duas portas de I/O para uso geral. Essas portas são conectadas à interface universla de I/O (portas do joystick). Vários dispositivos, além do joystick, podem ser conectados a essa porta, como mouse ou paddles. Para facilitar o acesso, existem algumas rotinas no BIOS que dão suporte a essas portas.

Essas interfaces são conectadas como ilustrado abaixo.



As duas portas do PSG são usadas como descrito abaixo:





O acesso à interface universal de I/O deve ser feito preferencialmente pelas rotinas do BIOS descritas abaixo:

GTSTCK (005DH/Main) - lê status do joystick
GTTRIG (00D8H/Main) - lê status 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 na seção “BIOS EM ROM” no capítulo 2.

Capítulo 9
O MSX TURBO R

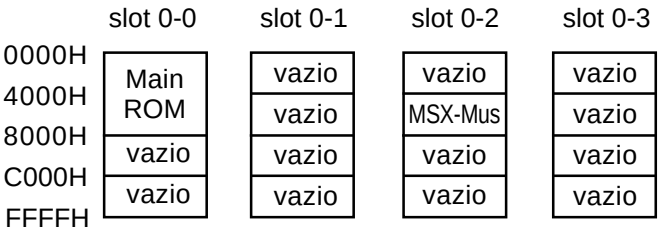
Nos modelos MSX turbo R, foi introduzida uma CPU de 16 bits, totalmente compatível com o Z80 a nível de instruções. A CPU R800 é construída em um chip LSI com encapsulamento QFP de 100 terminais. O clock interno do R800 é de 7,16 MHz. Ele também dispõe e 2 canais DMA e saída para multitarefa, mas essas opções não foram utilizadas.

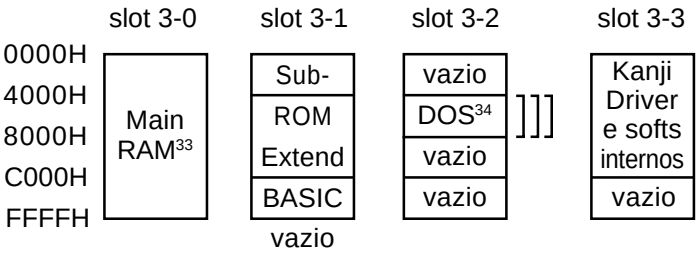
O set de instruções do R800 engloba todas as instruções do Z80 e acrescenta mais algumas, como multiplicação direta de 8 e 16 bits e tratamento dos registradores de índice IX e IY como dois registradores de 8 bits cada, denominados de .ixl, .ixh, .iyl e .iyh.

Como o R800 é totalmente compatível co o Z80 a nível de instruções, é possível fazer um programa que funcione do 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 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 do BIOS, BDOS ou BASIC não há nenhum problema, pois o BIOS compensa as diferenças de timing quando necessário.

1 - ORGANIZAÇÃO DE SLOTS E PÁGINAS

No MSX turbo R a organização de slots e páginas foi padronizada, por causa da RAM que é conectada diretamente ao R800. 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:





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.

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/DOS1 e R800/DOS2, mas nada impede que o DOS1 funcione sob o modo R800. 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, a menos que a tecla “1” seja pressionada durante o reset, o que força o sistema a entrar no modo Z80.

Uma observação importante é 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

---

Nota 33: o slot 3-0 deve conter, no mínimo, 256 Kbytes de RAM mapeada.
Nota 34: o DOS Kernel ocupa 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.

rotinas do 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 ao 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.

```
        RDSLT:  EQU 0000CH
        CALSLT: EQU 0001CH
        CHGCPU: EQU 00180H
        SLTROM: EQU 0FCC1H
        ;
        ;--- VERIFICA VERSAO ---
        ;
        LD   A,(SLTROM)
        LD   HL,0002DH
        CALL RDSLT
        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 do 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.

3.1 - COMPARAÇÃO DE VELOCIDADE

A tabela abaixo mostra o ganho de velocidade quando se usa o R800 no lugar do Z80.

Instruções	Z80(μs)	R800(μs)	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 microssegundos, foram usadas rotinas otimizadas para o Z80.

3.2 - INSTRUÇÕES ESPECÍFICAS DO R800

As instruções que foram acrescentadas para o R800 e que não existem no Z80 são as seguintes:

Memônico	Ilustração	Flags	Binário	Hex
		S Z H P N C	7 6 5 4 3 2 1 0	
ld u,u'	u ← u´	• • • • • •	1 1 0 1 1 1 0 1 0 1 u u'	DDH

Memônico	Ilustração	Flags	Binário	Hex
		S Z H P N C	7 6 5 4 3 2 1 0	
ld v,v'	v ← v´	• • • • • •	1 1 1 1 1 1 0 1 0 1 v v'	FDH
ld u,n	u ← n	• • • • • •	1 1 0 1 1 1 0 1 0 0 u 1 1 0 ------ n ------	DDH
ld v,n	v ← n	• • • • • •	1 1 0 1 1 1 0 1 0 0 v 1 1 0 ------ n ------	FDH
add .a,p	.a ← .a+p	↕ ↕ ↕ V 0 ↕	1 1 0 1 1 1 0 1 1 0 0 0 0 p	DDH
add .a,q	.a ← .a+q	↕ ↕ ↕ V 0 ↕	1 1 1 1 1 1 0 1 1 0 0 0 0 q	FDH
addc .a,p	.a ← .a+p+C	↕ ↕ ↕ V 0 ↕	1 1 0 1 1 1 0 1 1 0 0 0 1 p	DDH
addc .a,q	.a ← .a+q+C	↕ ↕ ↕ V 0 ↕	1 1 1 1 1 1 0 1 1 0 0 0 1 q	FDH
sub .a,p	.a ← .a-p	↕ ↕ ↕ V 1 ↕	1 1 0 1 1 1 0 1 1 0 0 1 0 p	DDH
sub .a,q	.a ← .a-q	↕ ↕ ↕ V 1 ↕	1 1 1 1 1 1 0 1 1 0 0 1 0 p	FDH
subc .a,p	.a ← .a+p-C	↕ ↕ ↕ V 1 ↕	1 1 0 1 1 1 0 1 1 0 0 1 1 p	DDH
subc .a,q	.a ← .a+q-C	↕ ↕ ↕ V 1 ↕	1 1 1 1 1 1 0 1 1 0 0 1 1 q	FDH
dec p	p ← p-1	↕ ↕ ↕ V 1 •	1 1 0 1 1 1 0 1 0 0 p 1 0 1	DDH
dec q	q ← q-1	↕ ↕ ↕ V 1 •	1 1 1 1 1 1 0 1 0 0 q 1 0 1	FDH
and .a,p	.a ← .a ∧ p	↕ ↕ 1 P 0 0	1 1 0 1 1 1 0 1 1 0 1 0 0 p	DDH
and .a,q	.a ← .a ∧ q	↕ ↕ 1 P 0 0	1 1 1 1 1 1 0 1 1 0 1 0 0 q	FDH
or .a,p	.a ← .a ∨ p	↕ ↕ 0 P 0 0	1 1 0 1 1 1 0 1 1 0 1 1 0 p	DDH
or .a,q	.a ← .a ∨ q	↕ ↕ 0 P 0 0	1 1 1 1 1 1 0 1 1 0 1 1 0 q	FDH
xor .a,p	.a ← .a ∀ p	↕ ↕ 0 P 0 0	1 1 0 1 1 1 0 1 1 0 1 0 1 p	DDH
xor .a,q	.a ← .a ∀ q	↕ ↕ 0 P 0 0	1 1 1 1 1 1 0 1 1 0 1 0 1 q	FDH

Memônico	Ilustração	Flags	Binário	Hex
		S Z H P N C	7 6 5 4 3 2 1 0	
cmp .a,p	.a - p	↕ ↕ ↕ V 1 ↕	1 1 0 1 1 1 0 1 1 0 1 1 1 p	DDH
cmp .a,q	.a - q	↕ ↕ ↕ V 1 ↕	1 1 1 1 1 1 0 1 1 0 1 1 1 q	FDH
mulub .a,r	.hl ← .a*r	0 ↕ • 0 • ↕	1 1 1 0 1 1 0 1 1 1 r 0 0 1	EDH
muluw .hl,ss	de:hl←.hl*ss	0 ↕ • 0 • ↕	1 1 1 0 1 1 0 1 1 1 ss 0 0 1 1	EDH
in .f,(c)	.f ← (.c)	↕ ↕ 0 P 0 •	1 1 1 0 1 1 0 1 0 1 1 1 0 0 0 0	EDH 70H

Convenção dos registradores:

	000	001	010	011	100	101	110	111	00	11
u	.b	.c	.d	.e	.ixh	.ixl	•	.a	•	•
v	.b	.c	.d	.e	.iyh	.iyl	•	.a	•	•
p	•	•	•	•	.ixh	.ixl	•	•	•	•
q	•	•	•	•	.iyh	.iyl	•	•	•	•
r	.b	.c	.d	.e	•	•	•	•	•	•
ss	•	•	•	•	•	•	•	•	.bc	.sp

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.

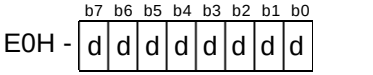
4.1 - ACESSO À MSX-MIDI

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 e na página seguinte:

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

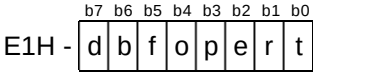
ECH - Contador 0
EDH - Contador 1
EEH - Contador 2
EFH - Controle dos contadores (escrita somente)

4.2 - DESCRIÇÃO DAS PORTAS DA MIDI EXTERNA



Porta de dados

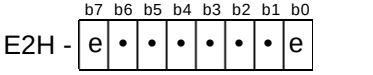
b0~b7 - dados
escrita → transmite (TXD)
leitura → recebe (RXD)



Porta de controle

Leitura:

b7 - DSR 8253 data set ready (1=pronto)
b6 - BRK 8251 parada detectada (1=detectada)
b5 - FE 8251 flag de erro de frame (1=erro)
b4 - OE 8251 flag de erro de overrun (1=erro)
b3 - PE 8251 flag de erro de paridade (1=erro)
b2 - EMPTY 8151 buffer de transmissão vazio (1=vazio)
b1 - RRDY 8251 status de recepção (1=dado presente)
b0 - TRDY 8251 status de transmissão (1=pronto)

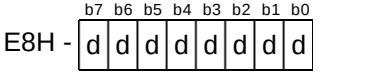


Porta de seleção

Escrita:

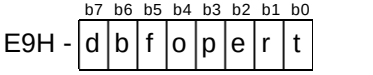
b7 - EN habilitação da MIDI externa (0=habilita)
[na inicialização, b7 é setado em 1]
b0 - E8 seleção de endereço da interface MIDI
[0=E8/E9H; 1=E0H,E1H]

4.3 - DESCRIÇÃO DAS PORTAS DA MIDI INTERNA



Porta de dados

***** Organização idêntica à da porta E0H.



Porta de controle

***** Organização idêntica à porta E1H para leitura. Para escrita, a organização é a seguinte:

Escrita:
Modo: b7=S2; b6=S1; b5=EP; b4=PEN;
b3=L2; b2=L1; b1=B2; b0=B1.
Comando:
b7 = EH - normalmente 0;
b6 = IR - normalmente 0;
b5 = RIE - habilita transmissão MIDI IN (1=habilita);
b4 = ER - reseta erro (1=reseta flags de erro;
0=sem operação);
b3 = SBRK - normalmente 0;
b2 = PE - habilita recepção MIDI IN (1=habilita);
b1 = TIE - timer 8253 (contador #2) - habilita transmissão (1=habilita);
b0 = TEN - habilita transmissão MIDI OUT (1=habilita).

Quando um dado for escrito no modo comando, é necessário uma espera de 16 ciclos T (3,58 MHz) para o resultado. Quando for escrita uma seqüência de comandos na porta de comando, é necessária a espera antes de escrever os dados.

EAH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

dados 8253

Escrita: 8253 OUT2 - latch dos sinais do terminal
Leitura: sem efeito

EBH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

Essa porta é uma imagem de EAH

ECH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

Leitura/escrita: contador 0

EDH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

Leitura/escrita: contador 1

EEH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

Leitura/escrita: contador 2

EFH -

b7	b6	b5	b4	b3	b2	b1	b0
d	d	d	d	d	d	d	d

Controle dos contadores

Escrita: b7~b6 - (SC1,SC0) seleciona contador
b5~b4 - (RW1,RW0) modo leitura/escrita do contador
b3~b1 - (M2,M1,M0) modo do contador
b0 - seleciona contador binário / contador BCD
Leitura: sem efeito

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).

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 8 μs 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.

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.

APÊNDICE

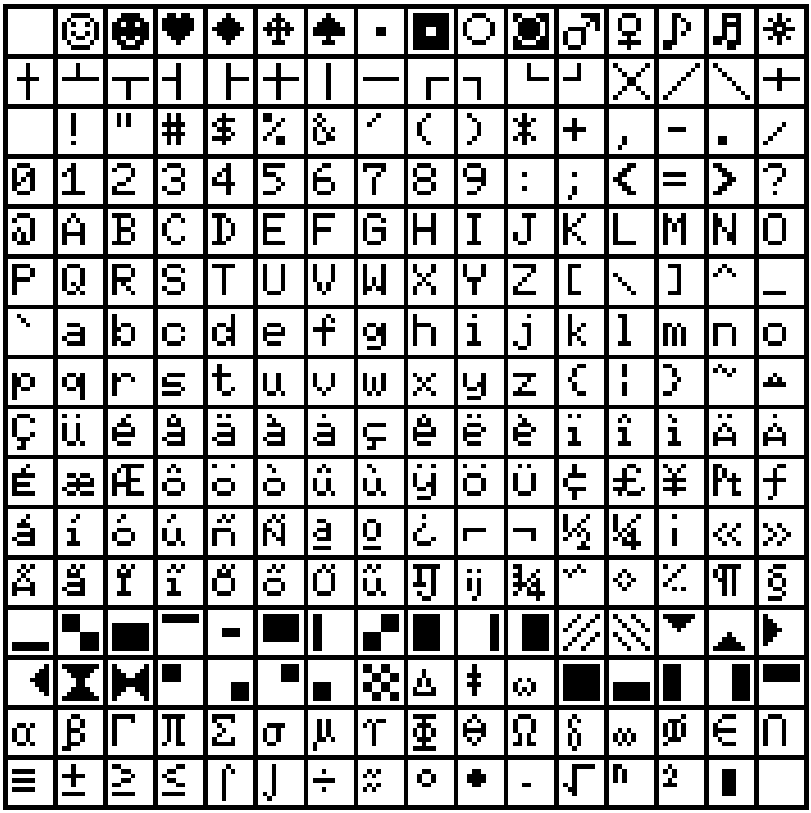
1 - TABELAS DE CARACTERES

1.1 - TABELA DE CARACTERES JAPONESA

A tabela abaixo é a que vem nos micros japoneses.

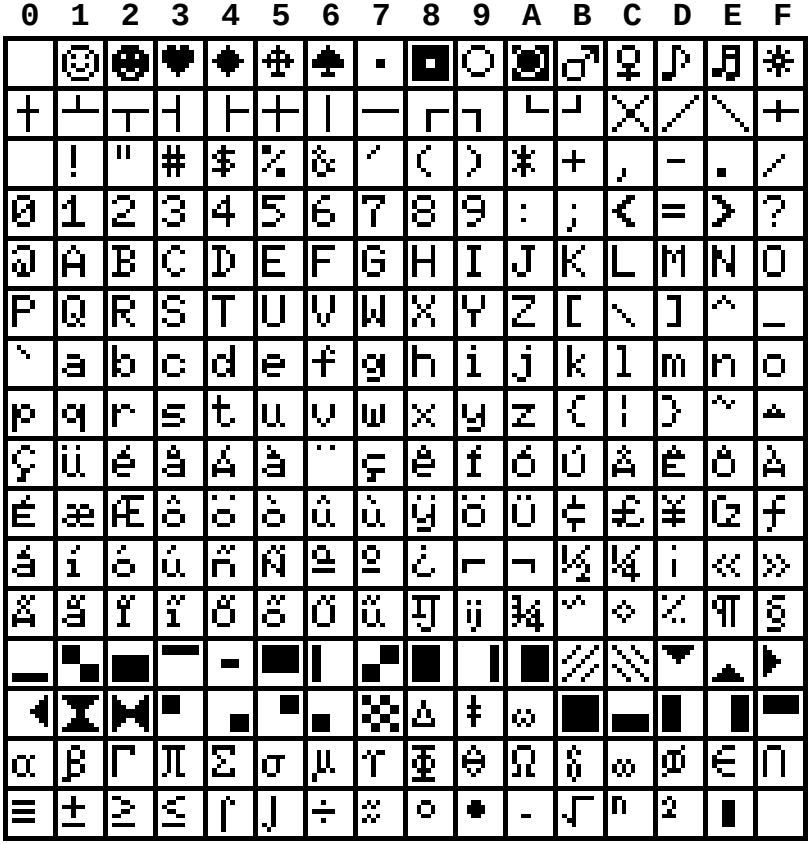
1.2 - TABELA DE CARACTERES INTERNACIONAL

A tabela de caracteres internacional é a adotada por todos os países da Europa (Reino Unido, França, Alemanha, etc.).



1.3 - TABELA DE CARACTERES BRASILEIRA

No Brasil, optou-se por uma tabela de caracteres ligeiramente diferente da internacional. Isso foi necessário para adaptar a tabela de caracteres à língua portuguesa.



2 - TABELA DE CORES PADRÃO

A tabela de cores ilustrada abaixo é a tabela de cores padrão para o MSX1. Para MSX2 em diante, é a tabela carregada quando do reset do micro.

nº paleta	Cor	nível vermelho	nível azul	nível 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

3 - CÓDIGOS DE CONTROLE

Teclado	Dec	Hex	Função
CTRL+A	001	01H	Determina caractere gráfico
CTRL+B	002	02H	Desvia o cursor para o início da palavra anterior
CTRL+C	003	03H	Encerra a condição de entrada
CTRL+D	004	04H	
CTRL+E	005	05H	Cancela caracteres do cursor até o fim da linha
CTRL+F	006	06H	Desvia o cursor para o início da palavra seguinte
CTRL+G	007	07H	Gera um beep
CTRL+H	008	08H	Apaga a letra anterior ao cursor (backspace)
CTRL+I	009	09H	Move o cursor p/ pos. de tabulação seguinte (TAB)
CTRL+J	010	0AH	Muda de linha (linefeed)
CTRL+K	011	0BH	Coloca o cursor na posição 1,1 (HOME)
CTRL+L	012	0CH	Limpa a tela e coloca o cursor na posição 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 o 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	
CTRL+[	027	1BH	(ESC)
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 o caractere que está sob o cursor (DEL)

4 - MAPA DAS PORTAS DE I/O DO Z80

00H~02H MIDI Saurus
00H~03H FAC MIDI Interface
00H~04H Music Module MIDI
00H~07H MD Telcom modem
08H~17H Sem uso conhecido
18H~19H Leitor de código de barras Philips NMS 1170/20
1AH~1FH Sem uso conhecido
20H~28H Ajuste secundário - Philips NMS1251 modem
Ajuste secundário - Miniware M4000 modem
29H~2FH Interface secundária Philips NMS 1210 (RS232C)
30H~38H Modem Philips NMS 1251
Modem Miniware M4000
Interface SCSI Green-Mak
Interface para CD-ROM
39H~5FH Sem uso conhecido
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 MIDI Saurus
71H~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 - Registrador 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

(continuação 80H~87H - interface serial RS232C padrão)
86H (R/W) 8253 - Contador 3
87H (W) Comando dos contadores
88H~8BH Portas do VDP para adaptação com MSX1
8CH~8DH Modem
8EH~8FH Megaram
8EH Seleção de páginas
8FH Megaram-Disk
90H~91H Impressora
90H (R) Status
91H (W) Dados
92H~97H Sem uso conhecido
98H~9BH VDP TMS9918/V9938/V9958
98H (R/W) Lê/escreve dados na VRAM
99H (R/W) Lê reg. de estado / escreve reg. 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 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 de teclado / click teclas)
ABH (W) Porta de comando da PPI
ACH~AFH MSX-Engine
B0H~B3H Expansão de memória (especificação SONY 8255)
B0H Linhas de endereço A0~A7
B1H Linhas de endereço A8~A10, A13~A15, controle, R/W
B2H Linhas de endereço A11~Ã12 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

C2H~C3H Sem uso conhecido

C4H~C7H Cartucho Moonsound (OPL4) - síntese FM
- C4H FM register array 0 (banco 1) e registrador de status
- C5H FM (dados)
- C6H FM register array 1 (banco 2)
- C7H Espelho de C5H (o acesso por C5H é preferido)

C8H~CFH MSX Interface

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~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~E5H Funções diversas para o MSX turbo R
- E4H Registradores
- E5H Dados

E6H~E7H Relógio do sistema para o MSX turbo R

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~F3H Sem uso conhecido

F4H Estado do RESET para o MSX turbo R

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 R (direito)
b1 - Audio L (esquerdo)
b2 - Seleciona entrada de vídeo
b3 - Detecta entrada de vídeo
b4 - Controle AV
b5 - Controle Ym
b6 - Inverso de b4 (VDP reg. #9 escrita)
b7 - Inverso de b5 (VDP reg. #9 leitura)

F8H~FBH Sem uso conhecido

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)

5 - CÓDIGOS DE ERRO DO MSX-BASIC

nº	Original inglês	Português
1	NEXT without FOR	NEXT sem FOR
2	Syntax error	Erro de sintaxe
3	RETURN without GOSUB	RETURN sem GOSUB
4	Out of DATA	Sem ´DATA´
5	Illegal function call	Chamada ilegal de função
6	Overflow	Overflow
7	Out of memory	Falta memória
8	Undefined line number	Número de linha não definido
9	Subsctipt out of range	Índice fora do limite
10	Redimensined array	Matriz redimensionada
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	Fórmula string muito complexa
17	Can´t CONTINE	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 inválido
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 inválido
57	Direct statement in file	Comando direto em arquivo
58	Sequential I/O only	Acesso seqüencial somente
59	File not OPEN	Arquivo não aberto
60	Bad FAT	Erro de FAT
61	Bad file mode	Modo errado de arquivo
62	Bad drive name	Nome errado de drive
63	Bad sector	Setor com erro
64	File still open	Arquivo já aberto
65	File already exists	Arquivo já existe

nº	Original inglês	Português
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	Arquivo 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
76~255	Unprintable error	Erro indefinido

6 - CÓDIGOS DE ERRO DO MSXDOS1

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	Input past end	Fim de arquivo
56	Bad file name	Nome de arquivo inválido
57	Direct statement in file	Comando direto em arquivo
58	Sequential I/O only	Acesso seqüencial somente
59	File not OPEN	Arquivo não aberto
60	Disk error	Erro de disco
61	Bad file mode	Modo errado de arquivo
62	Bad drive name	Nome errado de drive
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 de disco
70	Disk offline	Sem disco
71	RENAME across disk	RENAME em discos diferentes

7 - CÓDIGOS DE ERRO DO MSXDOS2

7.1 - 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	Unformatted 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 ruim
F1H	No message	Sem mensagem
F0H	Cannot format this drive	Este drive não pode ser formatado

7.2 - ERROS DAS FUNÇÕES DO MSXDOS

DFH	Internal error	Erro interno
DEH	Not enough memory	Memória insuficiente
DDH	-	
DCH	Invalid MSX-DOS call	Chamada ao MSXDOS inválida
DBH	Invalid drive	Especificação de drive inválida
DAH	Invalid filename	Nome de arquivo inválido
D9H	Invalid pathname	Nome do caminho inválido
D8H	Pathname too long	Nome do caminho 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 em duplicata
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	Invalide **.** or **..** operation	Operação com **.** ou **..** inválida
CDH	System file exists	Arquivo de sistema existe
CCH	Directory exists	Diretório existe
CBH	File exists	Arquivo existe

nº	Original inglês	Português
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	Erro de alocação de arquivo
C5H	Invalid process id	ID do processo inválida
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

7.3 - 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 standard
9BH	Error on standard input	Erro na entrada standard

7.4 - ERROS DE COMANDO

8FH	Wrong version of COMMAND	Versão errada do COMMAND.COM
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 para 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	Arq. 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	Arq. de destino não pode ser previamente escrito

8 - CÓDIGOS DE ERRO DO UZIX

nº	Original inglês	Português
1	Operation not permitted	Operação não permitida
2	No such file or directory	Não existe arquivo ou diretório
3	No such process	Não existe processo
4	Interrupted system call	Chamada de sistema interrompida
5	I/O error	Erro de I/O
6	No such device or address	Não existe dispositivo ou endereço
7	Arg list too long	Lista de argumentos muito longa
8	Exec format error	Erro de formato no Exec
9	Bad file number	Número de arquivo inválido
10	No child processes	Sem processos-filho
11	Try again	Tente novamente
12	Out of memory	Falta memória
13	Permission denied	Acesso negado
14	Bad address	Endereço inválido
15	Block device required	Dispositivo de bloco requerido
16	Device or resource busy	Dispositivo ou recurso ocupado
17	File exists	Arquivo existe
18	Cross-device link	Link para dispositivo cruzado
19	No such device	Não existe dispositivo
20	Not a directory	Não é um diretório
21	Is a directory	É um diretório
22	Invalid argument	Argumento inválido
23	File table overflow	Overflow na tabela de arquivo
24	Too many open files	Muitos arquivos abertos
25	Not a typewriter	Não é impressora
26	Text file busy	Arquivo de texto ocupado
27	File too large	Arquivo muito grande
28	No space left on device	Não há espaço no dispositivo
29	Illegal seek	Procura ilegal
30	Read-only file system	Sistema de arquivo somente leitura
31	Too many links	Muitos links
32	Broken pipe	Pipe quebrado
33	Math argument out of domain of func	Argumento matemático fora do domínio da função
34	Math result not representable	Resultado matemático não representável
35	Resource deadlock would occur	Travamento de recurso pode ocorrer
36	File name too long	Nome de arquivo muito longo
37	No record locks available	Não há proteção de gravação disponível
38	Function not implemented	Função não implementada
39	Directory not empty	Diretório não vazio
40	Too many symbolic links encountered	Muitos links simbólicos encontrados
41	It's a shell script	É um texto shell

GUIAS DE CONSULTA RÁPIDA

1 - MSX-BASIC

ABS (função, 1)
Formato: X = ABS (<exprN>)
Função: Retorna em X o valor absoluto (módulo) de <exprN>.

AND (operador lógico, 1)
Formato: <exprA1> AND <exprA2>
Função: Efetua operação lógica AND entre <exprA1> e <exprA2>.

ASC (função, 1)
Formato: X = ASC (<expr$>)
Função: Retorna em X o código ASCII do primeiro caractere de expr$.

ATN (função, 1)
Formato: X = ATN (<exprN>)
Função: Retorna em X o valor do arcotangente de exprN (exprN deve ser expresso em radianos).

AUTO (comando, 1)
Formato: AUTO [numlinha, [incremento]]
Função: Gera automaticamente números de linha, iniciando com [numlinha] e incrementado com o valor de [incremento].

BASE (variável de sistema, 1-2-3)
Formato: X = BASE (<n>) | BASE (<n>) = <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												
	SC0	SC1	SC2	SC3	SC4	SC5	SC6	SC7	SC8	SC10	SC11	SC12	TABELA DE
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 de 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 "<nomearq>"[,R[,<offset>]]
BLOAD "<nomearq>"[{,R | ,S}][,<offset>]] (D)
Função: Carrega um bloco binário na RAM ou na VRAM (,S). Se especificado [,R], executa programa em código de máquina.

BSAVE (comando, 1-D)
Formato: BSAVE "<nomearq>",<endini>,<endfim>[,<endexec>]
BSAVE "<nomearq>",<endini>,<endfim>[,<endexec>[,S]] (D)
Função: Salva em disco ou fita um bloco binário. Se especificado ,S salva um bloco da VRAM.

CALL (declaração, 1-2-3-4-D-M)
Formato: CALL <comando extendido> [(<argumento>[,argumento>...])]
Função: Executa comandos extendidos através de cartuchos de ROM.

CDBL (função, 1)
Formato: X# = CDBL(<exprN>)
Função: Converte o valor de <exprN> em um valor de dupla precisão e retorna o valor obtido em 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 em 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 arq"]
Função: Carrega um programa BASIC de fita cassete.

CLOAD? (comando, 1)
Formato: CLOAD? ["nome arq"]
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 verm.>,<nível verde>,<nível azul>)
Função: Especifica as cores da paleta.

COLOR = NEW (declaração, 2)
Formato: COLOR [= NEW]
Função: Inicializa a paleta de cores.

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 SPRITE$ (declaração, 2)
Formato: COLOR SPRITE$ (<nº do plano do sprite>)=<expr$>
onde <expr$> = CHR$(cor 1ª linha> + CHR$(cor 2ª linha) ...
Função: Especifica a cor de cada linha dos sprites.

CONT (comando, 1)
Formato: CONT
Função: Continua a execução de um programa que foi interrompido.

COPY (declaração, 1-2-D)
Formato: COPY "nomearq1" [TO "nomearq2"] (1-D)
Função: Copia o conteúdo de <nomearq1> para <nomearq2>.
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 | <nomearq>} (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> | <nomearq>} [,<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 <nomearq> TO <variável matriz> (2-D)
Função: Copia o conteúdo de um arquivo para uma variável matriz.
Formato: COPY <variável matriz> TO <nomearq> (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.

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 "<nomearq>" [,<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>
Função: 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 inteiras.

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.

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	X*série* - macro em série

DSKF (função, D)
Formato: X = DSKF(<nº drive>)
Função: Retorna o espaço livre no drive especificado em clusters.

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>.

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: Define a <var. string> para acesso aleatório ao disco.

FILES (comando, D)
Formato: FILES ["<nomearq>"]
Função: Apresenta os nomes de arquivos do disco de acordo com <nomearq>. Se <nomearq> for omitido, apresenta os nomes de todos os arquivos presentes no disco.

FIX (função, 1)
Formato: X = FIX(<exprN>)
Função: Retorna em X a parte inteira de <exprN>, 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 [#]<nº arq>[,<nº registro>]
Função: Lê um registro de um arquivo de acesso aleatório.

GET DATE (declaração, 2)
Formato: GET DATE <variável string> [,A]
Função: Retorna uma string com a data atual na <variável string>.

GET TIME (declaração, 2)
Formato: GET TIME <variável string> [,A]
Função: Retorna um string com a hora atual na <variável string>.

GOSUB (declaração, 1)
Formato: GOSUB <nº linha>
Função: Chama um subrotina que inicia na linha <nº linha>.

GOTO (declaração, 1)
Formato: GOTO <nº linha>
Função: Salta para a linha <nº linha>.

HEX$ (função, 1)
Formato: X$ = HEX$(<exprN>)
Função: Converte o valor de <exprN> em uma string hexadecimal e retorna o valor obtido em X$.

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>.

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.

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 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 contreúdo das teclas de função.

KILL (comando, D)
Formato: KILL "<nomearq>"
Função: Apaga arquivos no disco conforme especificado em <nomearq>.

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 ["<nomearq>"]
Função: Lista os nomes dos arquivos do disco na impressora de acordo com <nomearq>. Se <nomearq> 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).

LINE INPUT (declaração, 1)
Formato: LINE INPUT ["<prompt>";]<variável string>
Função: Lê uma seqüência 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 seqüência 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 "<nomearq>" [,R]
Função: Carrega um programa na memória e opcionalmente o executa.

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 [<cood. X>[,<coord. Y[,<tipo cursor>]]]
Função: Posiciona o cursor nas telas de texto.

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>.

LPRINT USING (declaração, 1)
Formato: LPRINT USING <"forma">;<exprA>[{; | ,}<exprA>...]
LPRINT USING <"forma expr$">
Função: Envia para a impressora os caracteres correspondentes às expressões <exprN> ou <expr$>, formatando. 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 "<nomearq>"
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 (declaração, 1)
Formato: MOTOR [{ON | OFF}]
Função: Liga ou desliga o motor do cassete.

NAME (comando, D)
Formato: "<nomearq1>" AS "<nomearq2>"
Função: Renomeia o arquivo <nomearq1> com <nomearq2>.

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>.

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 subrotina 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.

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 "<nomearq>" [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>.

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.

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 = PAD (<nº paddle>)
Função: Retorna em X o estado do paddle especificado.

PEEK (função, 1)
Formato: X = PEEK (<endereço>)
Função: Retorna em X o conteúdo do byte especificado por <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:

An~Gn	Toca nota cifrada com duração n (1~64, padrão é 4).
Rn	Pausa de duração n (1~64, o padrão é 4).
# ou +	Sustenido
-	Bemol
.	Aumento da duração em 50%.
On	Oitava (o padrão é 4)
Ln	Seta duração das notas (1~64, o padrão é 4)
Tn	Tempo e quartos de nota por minuto (32~255)
Vn	Volume (0~15, o padrão é 8)
Nn	Nota absoluta (1~96)
Mn	Período da envoltória (1~65535, o padrão é 255)
Sn	Forma de onda (0~15, o padrão é 0)
Xsérie	Executa o conteúdo da série.

PLAY (função, 1)
Formato: X = PLAY(<n>)
Função: Retorna em X o estado da voz <n> (tocando[-1] ou não[0]).

PLAY# (macro declaração, M-4)
Formato: PLAY #<n>,<expr$1>[,<expr$2>.....[,expr$12>]]]]]]]]]]]
Função: Toca as notas especificadas por <expr$> no PSG e/ou OPLL. Os comandos válidos para <expr$> são os mesmos que para a declaração PLAY, acrescidos dos descritos abaixo para o OPLL (**Obs.:** Mn e Sn são exclusivos do PSG):

Qn	Divisão de largura de som (1~8, o padrão é 8)
>	Aumenta uma oitava
<	Diminui uma oitava
=x;	Seta os parâmetros em x
&	Ligadura
{ }n	Define em n as notas entre { }. (n=1~8, padrão é Ln)
@n	Troca o instrumento (1~64)
@Vn	Seta mudança detalhada de volume (0~127)
@Nn	Mantém a duração definida por n (1~64, padrão é Ln)

Para as peças de bateria, os comandos são os seguintes:

B	Bass Drum
S	Snare Drum
W	Tom tom
C	Cymbals
H	Hi hat
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.

O valor <n> pode ser:

0	Toca somente o PSG (igual a PLAY)
1	Toca através da interface MIDI.
2 ou 3	Toca através do PSG e do OPLL (as 9 primeiras vozes são do OPLL e as três últimas do PSG).

POINT (função, 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 (delaração, 1-2)
Formato: PRESET {(X,Y) | STEP(X,Y)} [,<cor> [,<operação lógica>]]
Função: Apaga o ponto especificado por (X,Y) na tela gráfica.

PRINT (declaração, 1)
Formato: PRINT [<exprA>[{; | ,}<exprA>...]]
Função: Apresenta na tela os caracteres correspondentes às expressões <exprA>.

PRINT# (declaração, 1-D)
Formato: PRINT#<nº arq>,[<exprA>[{; | ,}<exprA>...]]
Função: Escreve o valor de <exprA> no arquivo especificado.

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. Os caracteres usados para formatar a saída estão descritos na página seguinte.

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 <"forma">;<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.

PUT (declaração, D)
Formato: PUT [#]<nº arq> [,<nº registro>]
Função: Grava um registro em um arquivo aleatório.

PUT KANJI (declaração, 1-2-K)
Formato: PUT KANJI [(X,Y)],<código JIS>[,<cor>[,<operação lógica> [,<modo>]]]
Função: Apresenta um caractere Kanji na tela.

PUT SPRITE (declaração, 1-2)
Formato: PUT SPRITE <plano do sprite>[,{(X,Y) | STEP(X,Y)} [,<cor> [,<nº do sprite>]]]
Função: Apresenta um sprite na tela.

READ (declaração, 1)
Formato: READ <nome variável>[,<nome variável>...]
Função: Lê os dados do comando DATA e os armazena nas variáveis.

REM (declaração, 1)
Formato: REM <comentários>
Função: Colocar comentários no programa.

RENUM (comando, 1)
Formato: RENUM [<novo nº linha>[,<nº linha antigo>[,<incremento>]]]
Função: Renumera as linhas de programa.

RESTORE (declaração, 1)
Formato: RESTORE [<nº de linha>]
Função: Especifica o número de linha DATA inicial a ser lido por READ.

RESUME (declaração, 1)
Formato: RESUME { [0] | NEXT | <nº de linha> }
Função: Finaliza rotina de tratamento de erros.

RETURN (declaração, 1)
Formato: RETURN [<nº de linha>]
Função: Retorna de uma subrotina.

RIGHT$ (função, 1)
Formato: X$ = RIGHT$ (<expr$>,<exprN>)
Função: Retorna em X$ os <exprN> caracteres direitos de <expr$>.

RND (função, 1)
Formato: X = RND [(<exprN>)]
Função: Retorna em X um número aleatório entre 0 e 1.

RSET (declaração, D)
Formato: RSET <variável string> = <expr$>
Função: Armazena o conteúdo de <expr$> à direita na variável string definida pela declaração FIELD.

RUN (comando, 1-D)
Formato: RUN [{<nº linha> | "nomearq" [,R]]
Função: Executa um programa na memória ou carrega um programa do disco e o executa.

SAVE (comando, 1-D)
Formato: SAVE "<nomearq>" [,A]
Função: Salva em disco ou fita o programa da memória.

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.

SGN (função, 1)
Formato: X = SGN (<exprN>)
Função: Retorna o resultado do sinal de <exprN> em X.

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>.

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: X$ = SPRITE$ (<nº sprite>) | SPRITE$ (<nº sprite>) = <expr$>
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.

STOP (declaração, 1)
Formato: STOP
Função: Paralisa a execução de um programa.
Formato: STOP {ON | OFF | STOP}
Função: Habilita, desabilita ou supende interrupção pelo pressionamento das teclas CTRL+STOP.

STRIG (função/declaração, 1)
Formato: X = STRIG (<nº porta joystick>)
Função: Examina a o estado dos botões de disparo e retorna o resultado em X.
Formato: STRIG (<nº porta joystick>) {ON | OFF | STOP}
Função: Habilita, desabilita ou supende 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 das duas variáveis.

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 | TIME = <exprN>
Função: Variável continuamente incrementada 60 vezes por segundo.

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.

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>)
Função: Retorna em X o endereço onde a variável está armazenada.
Formato: X = VARPTR (#<nº arq>)
Função: Retorna em X o endereço do FCB do arquivo especificado.

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: POKE <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: WIDHT <número>
Função: Especifica a número de caracteres por linha nos modos texto.

XOR (operador lógico, 1)
Formato: <exprA1> XOR <exprA2>
Função: Efetua operação lógica XOR entre <exprA1> e <exprA2>.

1.1 - SEQÜÊNCIA CALL

ANK (declaração, 1-2-K)
Formato: CALL ANK
Função: Sai do modo Kanji.

BGM (declaração, M)
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 a música (default).

CHDIR (declaração, D2)
Formato: CALL CHDIR (<expr$>)
Função: Troca subdiretório de acordo com o caminho <expr$>.

CHDRV (declaração, D2)
Formato: CALL CHDRV (<expr$>)
Função: Troca o drive de acordo com <expr$>.

CLS (declaração, K)
Formato: CALL CLS
Função: Limpa a tela no modo Kanji.

FORMAT (comando, D)
Formato: CALL FORMAT
Função: Formata um disquete.

KANJI (declaração, K)
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.

MDR (declaração, 4, opcional)
Formato: CALL MDR
Função: Ativa a saída do MSX-MUSIC para a interface MIDI.

MEMINI (declaração, 2)
Formato: CALL MEMINI [(tamanho da RAM disk)]
Função: Ativa a RAM disk nos 32K inferiores de memória.

MFILES (declaração, 2)
Formato: CALL MFILES
Função: Lista os arquivos da RAM disk dos 32K inferiores de memória.

MKDIR (declaração, D2)
Formato: CALL MKDIR (<expr$>)
Função: Cria um subdiretório com o nome especificado por <expr$>.

MKILL (declaração, 2)
Formato: CALL MKILL (“<nomearq>”)
Função: Apaga o arquivo <nomearq> da RAM disk dos 32K inferiores de memória.

MNAME (declaração, 2)
Formato: CALL MNAME (“<nomearq1>” AS “<nomearq2>”)
Função: Renomeia o arquivo <nomearq1> com <nomearq2> na RAM disk dos 32K inferiores de memória.

MUSIC (declaração, M)
Formato: CALL MUSIC [(<n1>[,0[,<n3>...[,n9]]]]]]]])]
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 - seleciona melodia
2 - seleciona bateria

PALETTE (declaração, 3)
Formato: CALL PALETTE (<nº paleta>,<R>,<G>,<B>)
Função: Especifica as cores para a paleta.

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 para a reprodução. [,S] especifica VRAM.

PCMREC (declaração, 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 e <salvamento> pode ser 0 ou 1 (1=salva na RAM, 0=não salva). [,S] grava na VRAM.

PITCH (declaração, M)
Formato: CALL PITCH (<n>)
Função: Ajuste fino do som. <n> pode variar de 410 a 459, sendo que o valor default é 440 (nota LÁ central).

PLAY (declaração, M)
Formato: CALL PLAY (<n>,<variável numérica>)
Função: Retorna na <variável numérica> o estado da voz <n> do OPLL (tocando[-1] ou não [0]). <n> pode variar de 0 a 9. Se for 0, todas as vozes são checadas. 1 a 9 checa a voz respectiva.

RAMDISK (declaração, D2)
Formato: CALL RAMDISK (<exprN1>,[<exprN2>])
Função: Cria uma RAMDISK com tamanho máximo <exprN1> e opcionalmente retorna o tamanho efetivamente criado em <exprN2>. A RAMDISK é acessada através do drive H:.

RMDIR (declaração, D2)
Formato: CALL RMDIR (<expr$>)
Função: Remove o subdiretório especificado por <expr$>.

STOPM (declaração, M)
Formato: CALL STOPM
Função: Interrompe a música tocada pelo MSX-MUSIC.

SYSTEM (comando, D)
Formato: CALL SYSTEM
Função: Chama o MSXDOS ou MSXDOS2.

TEMPER (declaração, M)
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 (default)	20 - Ritmo puro B- (G-)
10 - Ritmo puro C+ (A-)	21 - Ritmo puro H- (Gis-)

TRANSPOSE (declaração, M)
Formato: CALL TRANSPOSE (<n>)
Função: Muda de clave. <n> pode variar de -12799 a +12799, sendo que 100 unidades correspondem a meio tom. O valor default é 0.

VOICE (declaração, M)
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 default é 0.

VOICE COPY (declaração, M)
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.

1.2 - SEQÜÊNCIA 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.

BEEP (declaração, 2)
Formato: SET BEEP <timbre>,<volume>
Função: Seleciona o tipo e o volume do beep. Os valores válidos variam de 1 a 4.

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.

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.

PASSWORD (declaração, 2)
Formato: SET PASSWORD <expr$>
Função: Ativa a senha. <expr$> deve conter uma senha de no máximo 6 caracteres.

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.

SCREEN (declaração, 2)
Formato: SET SCREEN
Função: Grava na SRAM do relógio os dados da declaração SCREEN.

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.

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

VIDEO (declaração, 2, opcional)
Formato: SET VIDEO [<modo>[,<Ym>[,<CB>[,<sync>[,<voz>
[,<entrada de vídeo>[,<controle AV>]]]]]]]
Função: Define superimposição e outros modos.

1.3 - TABELAS E NOTAÇÕES

ABREVIAÇÕES DE INSTRUÇÕES

REM	´
PRINT	?
CALL	_

CÓDIGOS DE OPERAÇÃO LÓGICA

PSET	TPSET[35]	Usa a cor especificada (default)
PRESET	TPRESET[35]	Faz "NOT (cor especificada)
XOR	TXOR[35]	Faz "(cor destino) XOR (cor especificada)"
OR	TOR[35]	Faz "(cor destino) OR (cor especificada)"
AND	TAND[35]	Faz "(cor destino) AND (cor especificada)"

NOTAÇÕES

&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

1.4 - FORMATO

NOME DA INSTRUÇÃO (tipo da instrução, versão do BASIC)
Formato: Formatos válidos para a instrução.
Função: Forma de operação da instrução.

Nota 35: Quando a operação lógica for precedida por "T", nenhuma operação será feita quando a cor for transparente.

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

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.

2 - MSXDOS

ALIAS (interno, 2.41)
Formato: ALIAS [/P] [nome] [separador] [valor] | /R | {/L | /S} <nomearq>
Função: Apresenta ou define comando alias.
Detalhes: [/P] pausa a listagem ao completar uma tela.
[/R] remove todos os alias definidos.
[/L] carrega um alias definido em <nomearq>
[/S] salva o alias corrente no arquivo <nomearq>

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] <nomearq composto>
Função: Ativa/desativa arquivo oculto.
Detalhes: [/P] pausa as mensagens de erro ao completar uma tela.

ATTRIB (interno, 2-2.41)
Formato: ATTRIB {+|-H | +|-R | +|-S | +|-A} [/H] [/P] <nomearq composto>
Função: Altera atributos de arquivo oculto (H) somente leitura (R), arquivo de sistema (S, 24.1 somente) ou arquivado (A, 2.41 somente).
Detalhes: [/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.

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] <arqs fonte> <arq destino>
Função: Concatena todos os arquivos fonte em um único arquivo.
Detalhes: [/H] Arquivos ocultos também serão concatenados
[/S] Arquivos de sistema também serão concatenados (2.41)
[/P] Pausa as mensagens ao completar uma tela
[/B] Concatena sem interpretação (concatenação pura)
[/A] Reverte o efeito de [/B].
[/V] Verifica arquivo concatenado criado

COPY (interno, 1-2-2.41)
Formato: COPY [/H] [/S] [/P] [/A] [/B] [/V] [/T] <arqs fonte> <arqs dest>
Função: Copia arquivos.
Detalhes: [/H] Arquivos ocultos também serão copiados (2)
[/S] Arquivos de sistema também serão copiados (2.41)
[/P] Pausa as mensagens ao completar uma tela
[/A] Faz cópia ASCII (acrescenta Ctrl+Z no fim do arquivo)
[/B] Reverte o efeito de [/A]
[/V] Verifica arquivo copiado
[/T] Altera a data e hora do arquivo copiado para a atual

CPU (interno, 2.41)
Formato: CPU [número]
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.

DEL (interno, 1)
Formato: DEL [/S] [/H] [/P] <nomearq composto>
ERA [/S] [/H] [/P] <nomearq composto>
ERASE [/S] [/H] [/P] <nomearq composto>
Função: Deleta um ou mais arquivos.
Detalhes: [/S] Arquivos de sistema também serão deletados (2.41)
[/H] Arquivos ocultos também serão deletados
[/P] Pausa as mensagens ao completar uma tela

DIR (interno, 1-2-2.41)
Formato: DIR [/S] [/H] [/W] [/P] [/2] [<nomearq composto>]
Função: Apresenta os nomes dos arquivos do disco.
Detalhes: [/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:)
Detalhes: [/X] Suprime as mensagens durante a cópia

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 opcional [comando], alterna o Command Mode entre ON/OFF.

END (interno, 2.41)
Formato: END
Função: Termina um arquivo em lote (batch).

ENDIFF (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 valor default é 0).

FIXDISK (externo, 2)
Formato: FIXDISK [d:] [/S]
Função: Atualiza um disco para o formato MSXDOS2.
Detalhes: [/S] Atualização completa.

FORMAT (interno, 1-2.41)
Formato: FORMAT [d:] (1)
FORMAT [d: [opção [/X]]]
Função: Formata um disco. Se [opção] for especificada, formata com essa opção, sem apresentar lista de opções.
Detalhes: [/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 subrotina 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 [<nomearq>]
Função: Apresenta o arquivo de ajuda <nomearq).HLP ou lista todos.

HISTORY (interno, 2.41)
Formato: HISTORY [/P]
Função: Apresenta o histórico de comandos.
Detalhes: [/P] Pausa o histórico ao completar uma tela

IF (interno, 2.41)
Formato: IF [NOT] EXIST [d:][<caminho>] <nomearq> [THEN] <comando>
IF [NOT] <expr.1> == | EQ | LT | GT <expr.2> [AND | OR | XOR [NOT] <expr.3> == | EQ | LT | GT <expr.4> [AND | OR | XOR ...]] [THEN] <comando>
Função: Executa comando se a equação dada for verdadeira.
Detalhes: EQ Equivalência (igualdade)
LT Menor que
GT Maior que

IFF (interno, 2.41)
Formato: IFF [NOT] EXIST [d:][<caminho>] <nomearq> [THEN] <comando>
..... ENDIFF [<comando>]
IFF [NOT] <expr.1> == | EQ | LT | GT <expr.2> [AND | OR | XOR [NOT] <expr.3> == | EQ | LT | GT <expr.4> [AND | OR | XOR ...]] [THEN] <comando>
..... ENDIFF [<comando>]
Função: Liga o Command Mode se a equação dada for verdadeira e desliga caso contrário.
Detalhes: 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.
Detalhes: [/S] Atualiza o código de inicialização ou o drive [d:].

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.
Detalhes: [/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).

MOVE (interno, 2)
Formato: MOVE [/H] [/P] [/S] <nomearq composto> <caminho>
Função: Move arquivos para outra parte do disco.
Detalhes: [/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] <nomearq composto> <caminho>
Função: Move diretórios para outra parte do disco.
Detalhes: [/H] Diretórios ocultos também serão movidos
[/P] Pausa as mensagens ao completar uma tela

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.
Detalhes: + 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.
Detalhes: [/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 default, salvando os correntes.

RAMDISK (interno, 2)
Formato: RAMDISK [=] [<tamanho>[K]] [/D]
Função: Apresenta o tamanho ou cria uma RAMDISK.
Detalhes: [/D] Deleta a RAMDISK existente e cria outra.

RD (interno, 2)
Formato: RD [/H] [/P] <nomearq composto>
RMDIR [/H] [/P] <nomearq composto>
Função: Remove um ou mais subdiretórios.
Detalhes: [/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] <nomearq composto> <nomearq>
RENAME [/H] [/P] [/S] <nomearq composto> <nomearq>
Função: Renomeia o arquivo <nomearq composto> com <nomearq>.
Detalhes: [/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 subrotina 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] <nomearq composto> <nomearq>
Função: Renomeia o subdiretório <nomearq composto> com <nomearq>.
Detalhes: [/H] Arquivos ocultos também serão renomeados
[/P] Pausa as mensagens ao completar uma tela

SET (interno, 2-2.41)
Formato: SET [/P] [nome] [separador] [valor]
Função: Define ou apresenta itens de ambiente.
Detalhes: [/P] Pausa as mensagens ao completar uma tela
Os valores default 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

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.
Detalhes: [/N] Lista os diretórios contendo <parte_nome_subdirerório>.
[/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.
[/M] Cria novo diretório.
[/C] Cria novo diretório e entra nele.
[/H] Faz /M ou /C criarem diretório oculto.
[/D] Remove diretório.
[/R] Renomeira diretório.
[/V] Move subdiretório.
-n Nível dos subdiretórios.
\ Vai para o diretório raiz.

TREE (interno, 2.41)
Formato: TREE [d:] [<caminho>] [/P] [/?]
Função: Apresenta uma lista da árvore de diretórios no disco.
Detalhes: [/P] Pausa a listagem ao completar uma tela.
[/?] Apresenta uma tela de ajuda.

TYPE (interno, 1-2.41)
Formato: TYPE [/S] [/H] [/P] [/B] <nomearq composto> | “>”<dispositivo>
Função: Apresenta dados de um arquivo ou dispositivo.
Detalhes: [/S] Arquivos de sistema também serão apresentados (2.41).
[/H] Arquivos ocultos também serão apresentados.
[/P] Pausa a apresentação ao completar uma tela.
[/B] Desabilita a checagem de códigos de controle.

UNDEL (externo, 2)
Formato: UNDEL [<nomearq>]
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 [<nomearq> [<nomearq>]] [opções]
Função: Copia arquivos e diretórios. As opções são:
Detalhes: [/T] Altera a data do aquivo 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 [<nomearq>] [/H]
Função: Lista todos os arquivos do subdiretório corrente, em árvore.
Detalhes: [/H] Arquivos ocultos também serão listados.

2.1 - FORMATO

NOME DO COMANDO (tipo do comando, versão do Command)
Formato: Formatos válidos para o comando
Função: Forma de operação do comando
Detalhes: Descreve alguns detalhes sobre o formato

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. Na página seguinte 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)
K Necessário Kanji-ROM

NOTAÇÕES DE FORMATO

<nomearq>
Nome de arquivo na forma: A:\dir1\dir2\arquivo.ext

<nomearq 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.

{ } 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
PRN Impressora
NUL Nulo
AUX Auxiliar
COM Porta serial

Ou qualquer outro que esteja instalado.

3 - UZIX

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 standard.

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 econtradas.
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:*
[ugoa] [+ | -] [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=<número>] Copia <número> 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> 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 dhrystones.

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> [<nomearquzix>]
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 interpretanção 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 <nomearq>
Função: Executa um editor de texto padrão

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; 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 econtrada.
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
\nnn	valor numérico (estilo C)
: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
+	repere um ou mais
-	opcionalmente procura a expressão
[**..**]	qualquer destes (na faixa DE-PARA)
[^**..**]	qualquer exceto estes

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 rodando (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>
ln [-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 ou 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 movimentação 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-ↀ	:b - mesmo que Ç-ƒ
:r - mesmo que á-ↀÇ-ƒ	: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 standard.

3.1 - FORMATO

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 pa-

drão no UZIX 2.0, embora, em alguns casos, todos os detalhes acerca do formato não estejam descritos, como no editor ED e no comando TOP, por serem muito extensos.

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

Ou qualquer outro que esteja instalado.

4 - MEMÔNICOS DO Z80/R800

4.1 - GRUPO DE CARGA DE 8 BITS

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
LD r,r´	r←r´	• • • • • •	01 r r´	--	04	01	01
LD r,n	r←n	• • • • • •	00 r 110	--	07	02	02
LD u,u´	u←u´	• • • • • •	11 011 101 01 u u´	DD --	--	--	02
LD v,v´	v←v´	• • • • • •	11 111 101 01 v v´	FD --	--	--	02
LD u,n	u←n	• • • • • •	11 011 101 00 u 110 ← n →	DD -- --	--	--	03
LD v,n	v←n	• • • • • •	11 111 101 00 v 110 ← n →	FD -- --	--	--	03
LD r,(HL)	r←(HL)	• • • • • •	01 r 110	DD	07	02	02
LD r,(IX+d)	r←(IX+d)	• • • • • •	11 011 101 01 r 110 ← d →	-- -- --	19	05	05
LD r,(IY+d)	r←(IY+d)	• • • • • •	11 111 101 01 r 110 ← d →	FD -- --	19	05	05
LD (HL),r	(HL)←r	• • • • • •	01 110 r	--	07	02	02
LD (IX+d),r	(IX+d)←r	• • • • • •	11 011 101 01 110 r ← d →	DD -- --	19	05	05
LD (IY+d),r	(IY+d)←r	• • • • • •	11 111 101 01 110 r ← d →	FD -- --	19	05	05
LD A,(BC)	A←(BC)	• • • • • •	00 001 010	0A	07	02	02
LD A,(DE)	A←(DE)	• • • • • •	00 011 010	1A	07	02	02
LD A,(nn)	A←(nn)	• • • • • •	00 111 010 ← n → ← n →	3A -- --	13	04	04
LD (BC),A	(BC)←A	• • • • • •	00 000 010	02	07	02	02
LD (HL),A	(HL)←A	• • • • • •	00 000 010	12	07	02	02
LD (nn),A	(nn)←A	• • • • • •	00 000 010 ← n → ← n →	32 -- --	13	04	04
LD A,I	I←A	• ↕ I ↕ • •	11 101 101 01 010 111	ED 57	09	02	02
LD A,R	I←R	• ↕ I ↕ • •	11 101 101 01 011 111	ED 5F	09	02	02
LD I,A	I←A	• • • • • •	11 101 101 01 000 111	ED 47	09	02	02
LD R,A	R←A	• • • • • •	11 101 101 01 001 111	ED 4F	09	02	02

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
u	B	C	D	E	IXH	IXL	•	A
v	B	C	D	E	IYH	IYL	•	A

4.2 - GRUPO DE CARGA DE 16 BITS

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
LD dd,nn	dd←nn	• • • • • •	00 dd0 001 ← n → ← n →	-- -- --	10	03	03
LD IX,nn	IX←nn	• • • • • •	11 011 101 00 100 001 ← n → ← n →	DD 21 -- --	14	04	04
LD IY,nn	IY←nn	• • • • • •	11 111 101 00 100 001 ← n → ← n →	FD 21 -- --	14	04	04
LD HL,(nn)	H←(nn+1) L←(nn)	• • • • • •	00 101 010 ← n → ← n →	2A -- --	16	05	05
LD dd,(nn)	ddh←(nn+1) ddl←(nn)	• • • • • •	11 101 101 01 dd1 011 ← n → ← n →	ED -- -- --	20	06	06
LD IX,(nn)	IXh←(nn+1) IXl←(nn)	• • • • • •	11 011 101 00 101 010 ← n → ← n →	DD 2A -- --	20	06	06
LD IY,(nn)	IYh←(nn+1) IYl←(nn)	• • • • • •	11 111 101 00 101 010 ← n → ← n →	FD 2A -- --	20	06	06
LD (nn),HL	(nn+1)←H (nn)←L	• • • • • •	00 100 010 ← n → ← n →	22 -- --	16	05	05
LD (nn),dd	(nn+1)←ddh (nn)←ddl	• • • • • •	11 101 101 01 dd0 011 ← n → ← n →	ED -- -- --	20	06	06
LD (nn),IX	(nn+1)←IXh (nn)←IXl	• • • • • •	11 011 101 00 100 010 ← n → ← n →	DD 22 -- --	20	06	06
LD (nn),IY	(nn+1)←IYh (nn)←IYl	• • • • • •	11 111 101 00 100 010 ← n → ← n →	FD 22 -- --	20	06	06

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
LD SP,HL	SP←HL	• • • • • •	11 111 001	F9	06	01	01
LD SP,IX	SP←IX	• • • • • •	11 011 101 11 111 001	DD F9	10	02	02
LD SP,IY	SP←IY	• • • • • •	11 111 101 11 111 001	FD F9	10	02	02
PUSH qq	(SP-2)←qql (SP-1)←qqh	• • • • • •	11 qq0 101	--	11	03	03
PUSH IX	(SP-2)←IXl (SP-1)←IXh	• • • • • •	11 011 101 11 100 101	DD E5	15	04	04
PUSH IY	(SP-2)←IYl (SP-1)←IYh	• • • • • •	11 111 101 11 100 101	FD E5	11	04	04
POP qq	qql←(SP+1) qqh←(SP)	• • • • • •	11 qq0 001	--	10	03	03
POP IX	IXl←(SP-2) IXh←(SP-1)	• • • • • •	11 011 101 11 100 001	DD E1	14	04	04
POP IY	IYl←(SP-2) IYh←(SP-1)	• • • • • •	11 111 101 11 100 001	FD E1	14	04	04

	00	01	10	11
dd	BC	DE	HL	SP
qq	BC	DE	HL	AF

4.3 - GRUPO DE TROCA

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
EX DE,HL	DE↔HL	• • • • • •	11 101 011	EB	04	01	01
EX AF,AF´	AF↔AF´	• • • • • •	00 001 000	08	04	01	01
EXX	BC↔BC´ DE↔DE´ HL↔HL´	• • • • • •	11 011 001	D9	04	01	01
EX (SP),HL	H↔(SP+1) L↔(SP)	• • • • • •	11 100 011	E3	19	05	05
EX (SP),IX	IXh↔(SP+1) IXl↔(SP)	• • • • • •	11 011 101 11 100 011	DD E3	23	06	06
EX (SP),IY	IYh↔(SP+1) IYl↔(SP)	• • • • • •	11 011 101 11 100 011	FD E3	23	06	06

4.4 - GRUPO DE TRANFERÊNCIA DE BLOCO

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1	• • ↕ • 0 0	11 101 101 10 100 000	ED A0	16	04	04

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1 {Até BC=0}	• • 0 • 0 0	11 101 101 10 110 000	ED A8	21 16	05 04	05 04
LDD	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1	• • ↕ • 0 0	11 101 101 10 101 000	ED B0	16	04	04
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 {Até BC=0}	• • 0 • 0 0	11 101 101 10 111 000	ED B8	21 16	05 04	05 04

4.5 - GRUPO DE PESQUISAS

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
CPI	A–(HL) HL←HL+1 BC←BC-1	• ↕ ↕ ↕ 1 ↕	11 101 101 10 100 001	ED A1	16	04	04
CPIR	A–(HL) HL←HL+1 BC←BC-1 {Até BC=0 ou A=(HL)}	• ↕ ↕ ↕ 1 ↕	11 101 101 10 110 001	ED B1	21 16	05 04	05 04
CPD	A–(HL) HL←HL-1 BC←BC-1	• ↕ ↕ ↕ 1 ↕	11 101 101 10 101 001	ED A9	16	04	04
CPDR	A–(HL) HL←HL-1 BC←BC-1 {Até BC=0 ou A=(HL)}	• ↕ ↕ ↕ 1 ↕	11 101 101 10 111 001	ED B9	21 16	05 04	05 04

4.6 - GRUPO LÓGICO E DE COMPARAÇÃO

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
AND A,r	A←A∧r	0 ↕ P ↕ 0 1	10 100 r	--	04	01	01
AND A,p	A←A∧p	0 ↕ P ↕ 0 1	11 011 101 10 100 p	DD	--	--	01
AND A,q	A←A∧q	0 ↕ P ↕ 0 1	11 111 101 10 100 q	FD	--	--	01
AND A,(HL)	A←A∧(HL)	0 ↕ P ↕ 0 1	10 100 110	A6	07	02	02
AND A,(IX+d)	A←A∧(IX+d)	0 ↕ P ↕ 0 1	11 011 101 10 100 110 ← d →	DD A6 --	19	05	05

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
AND A,(IY+d)	A←A ∧(IY+d)	0 ↕ P ↕ 0 1	11 111 101 10 100 110 ← d →	FD A6 --	19	05	05
AND A,n	A←A ∧ n	0 ↕ P ↕ 0 1	11 100 110 ← n →	E6 --	07	02	02
OR A,r	A←A ∨ r	0 ↕ P ↕ 0 1	10 110 r	--	04	01	01
OR A,p	A←A ∨ p	0 ↕ P ↕ 0 1	11 011 101 10 110 p	DD --	--	--	01
OR A,q	A←A ∨ q	0 ↕ P ↕ 0 1	11 111 101 10 110 q	FD --	--	--	01
OR A,(HL)	A←A ∨ (HL)	0 ↕ P ↕ 0 1	10 110 110	B6	07	02	02
OR A,(IX+d)	A←A ∨ (IX+d)	0 ↕ P ↕ 0 1	11 011 101 10 110 110 ← d →	DD B6 --	19	05	05
OR A,(IY+d)	A←A ∨ (IY+d)	0 ↕ P ↕ 0 1	11 111 101 10 110 110 ← d →	FD B6 --	19	05	05
OR A,n	A←A ∨ n	0 ↕ P ↕ 0 1	11 110 110 ← n →	F6 --	07	02	02
XOR A,r	A←A ∀ r	0 ↕ P ↕ 0 1	10 101 r	--	04	01	01
XOR A,p	A←A ∀ p	0 ↕ P ↕ 0 1	11 011 101 10 101 p	DD --	--	--	01
XOR A,q	A←A ∀ q	0 ↕ P ↕ 0 1	11 111 101 10 101 q	FD --	--	--	01
XOR A,(HL)	A←A ∀ (HL)	0 ↕ P ↕ 0 1	10 101 110	AE	07	02	02
XOR A,(IX+d)	A←A ∀ (IX+d)	0 ↕ P ↕ 0 1	11 011 101 10 101 110 ← d →	DD AE --	19	05	05
XOR A,(IY+d)	A←A ∀ (IY+d)	0 ↕ P ↕ 0 1	11 111 101 10 101 110 ← d →	FD AE --	19	05	05
XOR A,n	A←A ∀ n	0 ↕ P ↕ 0 1	11 101 110 ← n →	EE --	07	02	02
CP A,r	A - r	↕ ↕ V ↕ 1 ↕	10 111 r	--	04	01	01
CP A,p	A - p	↕ ↕ V ↕ 1 ↕	11 011 101 10 111 p	DD --	--	--	01
CP A,q	A - q	↕ ↕ V ↕ 1 ↕	11 111 101 10 111 q	FD --	--	--	01
CP A,(HL)	A - (HL)	↕ ↕ V ↕ 1 ↕	10 111 110	BE	07	02	02
CP A,(IX+d)	A - (IX+d)	↕ ↕ V ↕ 1 ↕	11 011 101 10 111 110 ← d →	DD BE --	19	05	05
CP A,(IY+d)	A - (IY+d)	↕ ↕ V ↕ 1 ↕	11 111 101 10 111 110 ← d →	FD BE --	19	05	05
CP A,n	A - n	↕ ↕ V ↕ 1 ↕	11 111 110 ← n →	FE --	07	02	02

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
p	•	•	•	•	IXH	IXL	•	•
q	•	•	•	•	IYH	IYL	•	•

4.7 - GRUPO ARITMÉTICO DE 8 BITS

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
ADD A,r	A←A+r	↕ ↕ V ↕ 0 ↕	10 000 r	--	04	01	01
ADD A,p	A←A+p	↕ ↕ V ↕ 0 ↕	11 011 101 10 000 p	DD --	--	--	01
ADD A,q	A←A+q	↕ ↕ V ↕ 0 ↕	11 111 101 10 000 q	FD --	--	--	01
ADD A,(HL)	A←A+(HL)	↕ ↕ V ↕ 0 ↕	10 000 110	86	07	02	02
ADD A,(IX+d)	A←A+(IX+d)	0 ↕ P ↕ 0 1	11 011 101 10 000 110 ← d →	DD 86 --	19	05	05
ADD A,(IY+d)	A←A+(IY+d)	0 ↕ P ↕ 0 1	11 111 101 10 000 110 ← d →	FD 86 --	19	05	05
ADD A,n	A←A+n	0 ↕ P ↕ 0 1	11 000 110 ← n →	C6 --	07	02	02
ADC A,r	A←A+r+CY	0 ↕ P ↕ 0 1	10 001 r	--	04	01	01
ADC A,p	A←A+p+CY	0 ↕ P ↕ 0 1	11 011 101 10 101 p	DD --	--	--	02
ADC A,q	A←A+q+CY	0 ↕ P ↕ 0 1	11 111 101 10 101 q	FD --	--	--	02
ADC A,(HL)	A←A+(HL)+CY	0 ↕ P ↕ 0 1	10 001 110	8E	07	02	02
ADC A,(IX+d)	A←A+(IX+d)+CY	0 ↕ P ↕ 0 1	11 011 101 10 001 110 ← d →	DD 8E --	19	05	05
ADC A,(IY+d)	A←A+(IY+d)+CY	0 ↕ P ↕ 0 1	11 111 101 10 001 110 ← d →	FD 8E --	19	05	05
ADC A,n	A←A+n+CY	0 ↕ P ↕ 0 1	11 001 110 ← n →	CE --	07	02	02
SUB A,r	A←A-r	↕ ↕ V ↕ 1 ↕	10 010 r	--	04	01	01
SUB A,p	A←A-p	↕ ↕ V ↕ 1 ↕	11 011 101 10 010 p	DD --	--	--	02
SUB A,q	A←A-q	↕ ↕ V ↕ 1 ↕	11 111 101 10 010 q	FD --	--	--	02
SUB A,(HL)	A←A-(HL)	↕ ↕ V ↕ 1 ↕	10 010 110	96	07	02	02
SUB A,(IX+d)	A←A-(IX+d)	↕ ↕ V ↕ 1 ↕	11 011 101 10 010 110 ← d →	DD 96 --	19	05	05
SUB A,(IY+d)	A←A-(IY+d)	↕ ↕ V ↕ 1 ↕	11 111 101 10 010 110 ← d →	FD 96 --	19	05	05

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
SUB A,n	A←A-n	↕ ↕ V ↕ 1 ↕	11 010 110 ← n →	D6 --	07	02	02
SBC A,r	A←A-r-CY	↕ ↕ V ↕ 1 ↕	10 011 r	--	04	01	01
SBC A,p	A←A-p-CY	↕ ↕ V ↕ 1 ↕	11 011 101 10 011 p	DD --	--	--	02
SBC A,q	A←A-q-CY	↕ ↕ V ↕ 1 ↕	11 111 101 10 011 q	FD --	--	--	02
SBC A,(HL)	A←A-(HL)-CY	↕ ↕ V ↕ 1 ↕	10 011 110	9E	07	02	02
SBC A,(IX+d)	A←A-(IX+d)-CY	↕ ↕ V ↕ 1 ↕	11 011 101 10 011 110 ← d →	DD 9E --	19	05	05
SBC A,(IY+d)	A←A-(IY+d)-CY	↕ ↕ V ↕ 1 ↕	11 111 101 10 011 110 ← d →	FD 9E --	19	05	05
SBC n	A←A-n-CY	↕ ↕ V ↕ 1 ↕	11 011 110 ← n →	DE --	07	02	02
INC r	r←r+1	• ↕ V ↕ 0 ↕	00 r 100	--	04	01	01
INC p	p←p+1	• ↕ V ↕ 0 ↕	11 011 101 00 p 100	DD --	--	--	02
INC q	q←q+1	• ↕ V ↕ 0 ↕	11 111 101 00 q 100	FD --	--	--	02
INC (HL)	(HL)←(HL)+1	• ↕ V ↕ 0 ↕	00 110 100	34	11	03	04
INC (IX+d)	(IX+d)← ←(IX+d)+1	• ↕ V ↕ 0 ↕	11 011 101 00 110 100 ← d →	DD 34 --	23	06	07
INC (IY+d)	(IY+d)← ←(IY+d)+1	• ↕ V ↕ 0 ↕	11 111 101 00 110 100 ← d →	FD 34 --	23	06	07
DEC r	r←r-1	• ↕ V ↕ 1 ↕	00 r 101	--	04	01	01
DEC p	p←p-1	• ↕ V ↕ 1 ↕	11 011 101 00 p 101	DD --	--	--	02
DEC q	q←q-1	• ↕ V ↕ 1 ↕	11 111 101 00 q 101	FD --	--	--	02
DEC (HL)	(HL)←(HL)-1	• ↕ V ↕ 1 ↕	00 110 101	35	11	03	04
DEC (IX+d)	(IX+d)← ←(IX+d)-1	• ↕ V ↕ 1 ↕	11 011 101 00 110 101 ← d →	DD 35 --	23	06	07
DEC (IY+d)	(IY+d)← ←(IY+d)-1	• ↕ V ↕ 1 ↕	11 111 101 00 110 101 ← d →	FD 35 --	23	06	07
MULUB A,r	HL←A*r	↕ ↕ 0 0 • •	11 101 101 11 r 001	ED --	--	--	14

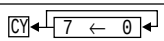
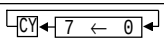
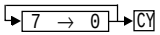
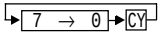
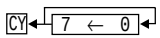
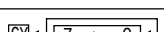
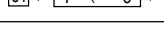
	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A
p	•	•	•	•	IXH	IXL	•	•
q	•	•	•	•	IYH	IYL	•	•

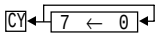
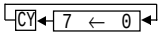
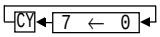
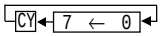
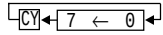
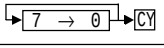
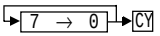
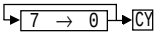
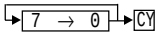
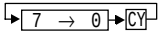
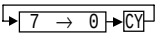
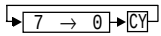
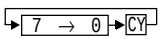
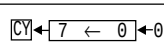
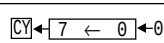
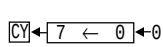
4.8 - GRUPO ARITMÉTICO DE 16 BITS

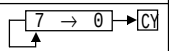
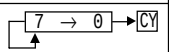
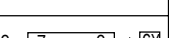
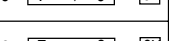
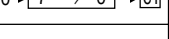
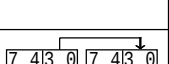
Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
ADD HL,ss	HL←HL+ss	↕ • • • 0 ?	00 ss1 001	--	11	03	01
ADD IX,pp	IX←IX+pp	↕ • • • 0 ?	11 011 101 00 ss1 001	DD	15	04	02
ADD IY,rr	IY←IY+rr	↕ • • • 0 ?	11 111 101 00 ss1 001	FD --	15	04	02
ADC HL,ss	HL← ←HL+ss+CY	↕ ↕ V ↕ 0 ?	11 101 101 01 ss1 010	ED --	15	04	02
SBC HL,ss	HL← ←HL-ss-CY	↕ ↕ V ↕ 0 ?	11 101 101 01 ss0 010	ED --	15	04	02
INC ss	ss←ss+1	• • • • • •	00 ss0 011	--	06	01	01
INC IX	IX←IX+1	• • • • • •	11 011 101 00 100 011	DD 23	10	02	02
INC IY	IY←IY+1	• • • • • •	11 111 101 00 100 011	FD 23	10	02	02
DEC ss	ss←ss-1	• • • • • •	00 ss1 011	--	06	01	01
DEC IX	IX←IX-1	• • • • • •	11 011 101 00 101 011	DD 2B	10	02	02
DEC IY	IY←IY-1	• • • • • •	11 111 101 00 101 011	FD 2B	10	02	02
MULUW HL,ss	DE:HL← ←HL*tt	↕ ↕ 0 0 • •	11 101 101 11 tt0 011	ED --	--	--	36

	00	01	10	11
ss	BC	DE	HL	SP
pp	BC	DE	IX	SP
rr	BC	DE	IY	SP
tt	BC	--	--	SP

4.9 - GRUPO DE DESLOCAMENTO E ROTAÇÃO

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
RLCA		↕ • • • 0 0	00 000 111	07	04	01	01
RLA		↕ • • • 0 0	00 010 111	0F	04	01	01
RRCA		↕ • • • 0 0	00 001 111	17	04	01	01
RRA		↕ • • • 0 0	00 011 111	1F	04	01	01
RLC r		↕ ↕ P ↕ 0 0	11 001 011 00 000 r	CB --	08	02	02
RLC (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 000 110	CB 06	15	04	05
RLC (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 000 110	DD CB -- 06	23	06	07

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
RLC (IY+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 000 110	FD CB -- 06	23	06	07
RL r		↕ ↕ P ↕ 0 0	11 001 011 00 010 r	CB --	08	02	02
RL (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 010 110	CB 16	15	04	05
RL (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 010 110	DD CB -- 16	23	06	07
RL (IY+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 010 110	FD CB -- 16	23	06	07
RRC r		↕ ↕ P ↕ 0 0	11 001 011 00 001 r	CB --	08	02	02
RRC (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 001 110	CB 0E	15	04	05
RRC (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 001 110	DD CB -- 0E	23	06	07
RRC (IY+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 001 110	FD CB -- 0E	23	06	07
RR r		↕ ↕ P ↕ 0 0	11 011 011 00 001 r	CB --	08	02	02
RR (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 011 110	CB 1E	15	04	05
RR (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 011 110	DD CB -- 1E	23	06	07
RR (IY+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 011 110	FD CB -- 1E	23	06	07
SLA r		↕ ↕ P ↕ 0 0	11 011 011 00 100 r	CB --	08	02	02
SLA (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 100 110	CB 26	15	04	05
SLA (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 100 110	DD CB -- 26	23	06	07

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
SLA (IX+d)		↕ ↕ P ↕ 0 0	11 100 011 11 001 011 ← d → 00 100 110	FD CB -- 1E	23	06	07
SRA r		↕ ↕ P ↕ 0 0	11 011 011 00 101 r	CB --	08	02	02
SRA (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 101 110	CB 2E	15	04	05
SRA (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 101 110	DD CB -- 2E	23	06	07
SRA (IX+d)		↕ ↕ P ↕ 0 0	11 100 011 11 001 011 ← d → 00 101 110	FD CB -- 2E	23	06	07
SRL r		↕ ↕ P ↕ 0 0	11 011 011 00 111 r	CB --	08	02	02
SRL (HL)		↕ ↕ P ↕ 0 0	11 001 011 00 111 110	CB 3E	15	04	05
SRL (IX+d)		↕ ↕ P ↕ 0 0	11 011 011 11 001 011 ← d → 00 111 110	DD CB -- 3E	23	06	07
SRL (IY+d)		↕ ↕ P ↕ 0 0	11 100 011 11 001 011 ← d → 00 111 110	FD CB -- 3E	23	06	07
RLD		• ↕ P ↕ 0 0	11 101 101 01 101 111	ED 6F	18	05	07
RRD		• ↕ P ↕ 0 0	11 101 101 01 100 111	ED 67	18	05	07

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	•	A

4.10 - GRUPO DE TESTE E MANIPULAÇÃO DE BITS

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
BIT b,r	$Z \leftarrow \overline{r}_b$	• ↕ ? ? 0 1	11 001 011 01 b r	CB --	08	02	02
BIT b,(HL)	$Z \leftarrow \overline{(HL)}_b$	• ↕ ? ? 0 1	11 001 011 01 b 110	CB --	12	03	03
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	• ↕ ? ? 0 1	11 011 101 11 001 011 ← d → 01 b 110	DD CB -- --	20	05	05

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
BIT b,(IY+d)	$Z \leftarrow (\overline{IY+d})_b$	• ↕ ? ? 0 1	11 111 101 11 001 011 ← d → 01 b 110	FD CB -- --	20	05	05
SET b,r	$r_b \leftarrow 1$	• • • • • •	11 001 011 11 b r	FD --	08	02	02
SET b,(HL)	$(HL)_b \leftarrow 1$	• • • • • •	11 001 011 11 b 110	FD --	15	04	05
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	• • • • • •	11 011 101 11 001 011 ← d → 11 b 110	FD CB -- --	23	06	07
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	• • • • • •	11 111 101 11 001 011 ← d → 11 b 110	FD CB -- --	23	06	07
RES b,r	$r_b \leftarrow 0$	• • • • • •	11 001 011 10 b r	FD --	08	02	02
RES b,(HL)	$(HL)_b \leftarrow 0$	• • • • • •	11 001 011 10 b 110	FD --	15	04	05
RES b,(IX+d)	$(IX+d)_b \leftarrow 0$	• • • • • •	11 011 101 11 001 011 ← d → 10 b 110	FD CB -- --	23	06	07
RES b,(IY+d)	$(IY+d)_b \leftarrow 0$	• • • • • •	11 111 101 11 001 011 ← d → 10 b 110	FD CB -- --	23	06	07

	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

4.11 - GRUPO DE SALTO

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
JP nn	PC←nn	• • • • • •	11 000 011 ← n → ← n →	C3 -- --	10	03	03
JP cc,nn	Se cc=Verd, PC←nn	• • • • • •	11 cc 010 ← n → ← n →	-- -- --	10	03	03
JR e	PC←PC+e	• • • • • •	00 011 000 ← e-2 →	18 --	12	03	03
JR C,e	Se C=1, PC←PC+e	• • • • • •	00 111 000 ← e-2 →	38 --	07 12	02 03	02 03
JR NC,e	Se C=0, PC←PC+e	• • • • • •	00 110 000 ← e-2 →	30 --	07 12	02 03	02 03

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
JR Z,e	Se Z=1, PC←PC+e	• • • • • •	00 101 000 ← e-2 →	28 --	07 12	02 03	02 03
JR NZ,e	Se Z=0, PC←PC+e	• • • • • •	00 100 000 ← e-2 →	20 --	07 12	02 03	02 03
JP (HL)	PC←HL	• • • • • •	11 101 001	E9	04	01	01
JP (IX)	PC←IX	• • • • • •	11 011 101 11 101 001	DD E9	08	02	02
JP (IY)	PC←IY	• • • • • •	11 111 101 11 101 001	FD E9	08	02	02
DJNZ e	B←B-1 Se B≠0, PC←PC+e	• • • • • •	00 010 000 ← e-2 →	10 --	08 13	02 03	02 02

	000	001	010	011	100	101	110	111
cc	NZ	Z	NC	C	PO	PE	P	M

4.12 - GRUPO DE CHAMADA E RETORNO

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
CALL nn	$(SP-1)\leftarrow PC_h$ $(SP-2)\leftarrow PC_l$ PC←nn	• • • • • •	11 001 101 ← n → ← n →	CD -- --	17	05	05
CALL cc,nn	Se cc=Verd, $(SP-1)\leftarrow PC_h$ $(SP-2)\leftarrow PC_l$ PC←nn	• • • • • •	11 cc 100 ← n → ← n →	-- -- --	10 17	03 05	03 05
RET	$PC_l\leftarrow(SP)$ $PC_h\leftarrow(SP+1)$	• • • • • •	11 001 001	C9	10	03	03
RET cc	Se cc=Verd, $PC_l\leftarrow(SP)$ $PC_h\leftarrow(SP+1)$	• • • • • •	11 cc 000	--	05 11	01 03	01 03
RETI	Retorna da interrupção	• • • • • •	11 101 101 01 001 101	ED 4D	14	04	05
RETN	Ret. interr. não mascar.	• • • • • •	11 101 101 01 000 101	ED 45	14	04	05
RST p	$(SP-1)\leftarrow PC_h$ $(SP-2)\leftarrow PC_l$ $PC_l\leftarrow p*8$ $PC_h\leftarrow 0$	• • • • • •	11 p 111	--	11	03	04

	000	001	010	011	100	101	110	111
cc	NZ	Z	NC	C	PO	PE	P	M

4.13 - GRUPO DE ENTRADA E SAÍDA

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
IN A,(n)	A←(n)	• • • • • •	11 011 011 ← n →	28 --	11	03	03

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
IN r,(C)	r←(C)	• ↕ P ↕ 0 ↕	11 101 101 01 r 000	ED --	11	03	03
INI	(HL)←(C) B←B-1 HL←HL+1	• ↕ ? ? 1 ?	11 101 101 10 100 010	ED A2	16	04	04
INIR	(HL)←(C) B←B-1 HL←HL+1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 110 010	ED B2	21 16	05 04	04 03
IND	(HL)←(C) B←B-1 HL←HL-1	• ↕ ? ? 1 ?	11 101 101 10 101 010	ED AA	16	04	04
INDR	(HL)←(C) B←B-1 HL←HL-1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 111 010	ED BA	21 16	05 04	04 03
OUT (n),A	(n)←A	• • • • • •	11 010 011 ← n →	D3 --	11	03	03
OUT (C),r	(C)←r	• • • • • •	11 101 101 01 r 001	ED --	12	03	03
OUTI	(C)←(HL) B←B-1 HL←HL+1	• ↕ ? ? 1 ?	11 101 101 10 100 011	ED A3	16	04	04
OTIR	(C)←(HL) B←B-1 HL←HL+1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 110 011	ED B3	21 16	05 04	04 03
OUTD	(C)←(HL) B←B-1 HL←HL-1	• ↕ ? ? 1 ?	11 101 101 10 110 011	ED AB	16	04	04
OTDR	(C)←(HL) B←B-1 HL←HL-1 {Até B=0}	• 1 ? ? 1 ?	11 101 101 10 111 011	ED BB	21 16	05 04	04 03

	000	001	010	011	100	101	110	111
r	B	C	D	E	H	L	F	A

4.14 - GRUPO DE CONTROLE E MISCELÂNEA

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
DAA	Conv. BCD	↕ ↕ P ↕ • ↕	00 100 111	27	04	01	01
CPL	A←NOT(A)	• • • • 1 1	00 101 111	2F	04	01	01
NEG	A←0-A	↕ ↕ V ↕ 1 ↕	00 101 101 01 000 100	ED 44	08	02	02
CCF	CY←NOT(CY)	↕ • • • 0 ?	00 111 111	3F	04	01	01
SCF	CY←1	1 • • • 0 0	00 110 111	37	04	01	01
NOP	Não-oper.	• • • • • •	00 000 000	00	04	01	01

Memônico	Ilustração	C Z P/V S N H	Binário	Hex	TZ	MZ	MR
HALT	CPU parada	• • • • • •	01 110 110	76	04	01	01
DI	IFF←0	• • • • • •	11 110 011	F3	04	01	01
EI	IFF←1	• • • • • •	11 111 011	FB	04	01	01
IM 0	Modo 0 de interrupção	• • • • • •	11 101 101 01 000 110	ED 46	08	02	02
IM 1	Modo 1 de interrupção	• • • • • •	11 101 101 01 010 110	ED 56	08	02	02
IM 2	Modo 2 de interrupção	• • • • • •	11 101 101 01 011 110	ED 5E	08	02	02

4.15 - FORMATO

Os memônicos do Z80 e do R800 estão separados em 14 grupos por semelhança de função. Todas as instruções, à exceção de MULUB e MULUW, que são exclusivas do R800, são comuns aos dois processadores. As instruções que manipulam os registradores IXH, IXL, IYH e IYL são as instruções "secretas" do Z80; elas foram oficializadas no R800. Abaixo há uma curta descrição de cada campo das tabelas.

Memônico: Código memônico na notação do Z80.
Ilustração: Curta descrição da operação realizada pela instrução. Uma descrição entre parênteses é uma observação.
C Z P/V S N H: Sinalizadores (flags) afetados. A notação é a seguinte:

- • sinalizador não afetado
- 0 sinalizador desligado
- 1 sinalizador ligado
- ? sinalizador desconhecido
- ↕ sinalizador afetado de acordo com o resultado da operação
- I o conteúdo do circuito biestável de ativação de interrupções (IFF) é copiado para o sinalizador

Binário: Código binário da instrução
Hex: Código hexadecimal da instrução
TZ: Número de ciclos T para o Z80
MZ: Número de ciclos de máquina para o Z80
MR: Número de ciclos de máquina para o R800

Nota: Quando houver duas descrições de ciclos, elas referem às duas condições que a instrução pode assumir. A indicação "--" está presente nas instruções que manipulam os registradores IXH, IXL, IYH e IYL (instruções "secretas" do Z80) e nas instruções MULUW e MULUB do R800.

REFERÊNCIAS BIBLIOGRÁFICAS

APROFUNDANDO-SE NO MSX
Piazzi - Maldonado - Oliveira (Editora Aleph, 1986)

LIVRO VERMELHO DO MSX, O (The Red Book)
McGraw Hill / Avalon Software (1988 / 1985)

MANUAL DO MICROPROCESSADOR Z-80
William Barden Jr. (Editora Campus, 1985)

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)

MSX2 TECHNICAL HANDBOOK
ASCII Corpotation (1985)

OPL4 YMF278B - APPLICATION MANUAL
Yamaha Corporation (1994)

PROGRAMAÇÃO AVANÇADA EM MSX
Figueredo - Maldonado - Rosseto (Editora Aleph, 1986)

V9938 MSX-VIDEO - APPLICATION MANUAL
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)