

# O LIVRO VERMELHO DO MSX

"The Red Book"



McGraw-Hill

AVALON SOFTWARE





Valorize sua formação profissional,  
seu futuro, sua consciência

# O LIVRO VERMELHO do MSX

THE RED BOOK

**AVALON SOFTWARE**

*Tradução:*

**Lars Gustav Erik Unonius**

Engenheiro Eletrônico

*Revisão Técnica:*

**José Maurício Bussab**

McGraw-Hill

São Paulo

Rua Tabapuã, 1.105, Itaim-Bibi

CEP 04533

(011) 881-8604 e (011) 881-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá •  
San Juan • Santiago

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris •  
Singapore • Sydney • Tokyo • Toronto

Do original  
The MSX Red Book  
Copyright © 1985 by Avalon Software  
Copyright © 1988 da Editora McGraw-Hill, Ltda.

Todos os direitos para a língua portuguesa reservados pela Editora McGraw-Hill, Ltda.

Nenhuma parte desta publicação poderá ser reproduzida, guardada pelo sistema "retrieval" ou transmitida de qualquer modo ou por qualquer outro meio, seja este eletrônico, mecânico, de fotocópia, de gravação, ou outros, sem prévia autorização, por escrito, da Editora.

*Editor:* MILTON MIRA DE ASSUMPÇÃO FILHO

*Coordenadora de Revisão:* Daisy Pereira Daniel

*Supervisor de Produção:* José Rodrigues

*Capa: Layout:* Cyro Giordano

*Arte final:* R. Iacobucci – Studio de Artes Gráficas

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

M911

O livro vermelho do MSX / Avalon Software ; tradução Lars Gustav Erik Unonius ; revisão técnica José Maurício Bussab. — São Paulo : McGraw-Hill, 1988.

1. MSX (Computadores) 2. MSX (Computadores) – Programação 3. MSX-BASIC (Linguagem de programação para computadores) I. Avalon Software.

CDD-001.64  
-001.642  
-001.6424

88-0466

**Índices para catálogo sistemático:**

1. MSX : Computadores : Processamento de dados 001.64
2. MSX : Computadores : Programação : Processamento de dados 001.642
3. MSX-BASIC : Linguagem de programação : Processamento de dados 001.6424



## NOTA DO REVISOR TÉCNICO

Durante a tradução e revisão do Red Book, encontramos diversas dificuldades, a maior parte delas devido à terminologia.

Não existe ainda um consenso quanto à terminologia MSX em português. Além disso, muitos termos de computação podem ter diversas traduções, e cada autor nacional usa a que mais lhe agrada.

Diante disso, e devido à natureza técnica do Red Book, decidimos apresentar aqui um pequeno índice das traduções utilizadas, para que o leitor possa consultar rapidamente em caso de dúvida.

Os termos são apresentados com seu nome original e a tradução utilizada. Segue-se um comentário entre parênteses sempre que apropriado.

Termos próprios do padrão MSX e da Microsoft:

**Primary Slot** – Conector Primário

**Name Table** – Tabela de Nomes

**Character Pattern Table** – Tabela de Imagens de Caracteres (preferiu-se “Imagem” a “Padrão”, pois, este último pode dar a idéia de “standard”, quando o sentido correto é o de “desenho” ou “estampa”)

**Sprite Attribute Table** – Tabela de Atributos dos Sprites

**Runloop** – Ronda de Execução (o Runloop não é um “laço” no sentido clássico, mas uma parte do interpretador que é executada entre a interpretação de uma linha e a próxima)

**Mainloop** – Ronda Principal

**Putback Flag** – Indicador de devolução

**Token** – átomo (outras traduções possíveis para “token”, que é a unidade de texto de um programa BASIC, seriam “símbolo”, “ficha” etc. Preferiu-se “átomo” para distinguir o “token” de outros “símbolos” utilizados no texto)

**tokenization** – atomização (é a operação de transformação de um texto BASIC em ASCII para um texto totalmente formado por átomos)

**detokenization** – desatomização (a operação inversa)

**Statement Handler** – Manipulador de instrução

**Link** – elo (“Link” é utilizado, no interpretador BASIC, para indicar a área de memória, numa linha, que aponta para outras linhas do programa)

**Function dispatcher** – despachante de funções

**Parser** – Analisador Sintático

**Template** – Máscara

Outros termos de computação:

**String** – String (“string” vem sendo traduzido por “cadeia”, “cordão”, “colar” ... Para evitar confusão e pela inexistência, em português, de um termo de aceitação geral para esta palavra, preferiu-se deixá-la no original. O mesmo acontece com outros termos indicados abaixo.)

**Power-up** – Partida

**Statement** – Instrução

**To set up** – Inicializar

**Blank** – Limpo(a)

**To set a bit** – Ligar um bit

**To reset a bit** – Desligar um bit

**Clipping** – Corte

**Buffer** – Buffer

**Overhead** – Tempo extra

**Overflow** – Extravasamento

**Underflow** – Subextravasamento

**Push** – Empilhar

**Pop** – Desempilhar

**Random number** – Número alcatório

**MSB** – MSB

**LSB** – LSB (MSB é o byte mais significativo e LSB é o menos. Deixou-se os termos no original especialmente por sua concisão.)

**Default** – default

**Dummy** – Sem significado

**Prompt** – Prompt

**Assignment** – Atribuição

**Subscript** – Índice (de matriz)

**Assembler** – Assembler

**To disassemble** – desassemblar (uma tradução possível seria “desmontar” mas, convenhamos, ninguém desejaria “desmontar a ROM do MSX”.)

Os programas do capítulo 7 foram testados e adaptados usando o editor-assembler GEN e foram incluídos dois pequenos programas em BASIC que mostram possíveis aplicações dos referidos programas.

Algumas constantes e tabelas específicas das máquinas inglesas foram adaptadas usando-se um HOT-BIT. Nesses casos, tanto o dado original do Red Book quanto o referente à máquina nacional foram incluídos.



## SUMÁRIO

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| Introdução .....                                                    | XI  |
| <b>Capítulo 1.</b> Interface do Periférico Programável (PPI) .....  | 1   |
| <b>Capítulo 2.</b> Processador de Apresentação de Vídeo (VDP) ..... | 7   |
| <b>Capítulo 3.</b> Gerador de Som Programável (PSG) .....           | 22  |
| <b>Capítulo 4.</b> BIOS em ROM .....                                | 29  |
| <b>Capítulo 5.</b> Interpretador BASIC em ROM .....                 | 107 |
| <b>Capítulo 6.</b> Mapeamento da Memória .....                      | 250 |
| <b>Capítulo 7.</b> Programas em Assemblér .....                     | 293 |



## INTRODUÇÃO

### Objetivos

Este livro diz respeito aos computadores MSX e como eles funcionam. Os fabricantes dos computadores MSX, por motivos técnicos e comerciais fornecem apenas uma pequena quantidade de informação (referente ao projeto de suas máquinas) disponível ao usuário final. Normalmente, esta informação compreende uma descrição bastante detalhada do BASIC MSX da Microsoft, juntamente com um delineamento amplo do hardware do sistema. Este nível de documentação é adequado ao usuário comum, porém insuficiente para um outro interessado em uma programação mais sofisticada.

O objetivo deste livro é de fornecer uma descrição bastante detalhada do hardware e software do MSX, em nível suficiente para satisfazer o usuário mais exigente: o programador de código de máquina. Não é um curso introdutório sobre programação; é necessariamente de natureza técnica. Presume-se que você já possua ou tencione adquirir, por outros meios, um conhecimento da linguagem de máquina do Microprocessador Z80. Como já existem vários livros especializados sobre o Z80, qualquer descrição de suas características, simplesmente, duplicaria uma informação amplamente disponível.

## Organização

O padrão MSX especifica os principais componentes funcionais de qualquer computador MSX:

1. Um Microprocessador Z80A da Zilog.
2. Uma Interface Periférica Programável 8255 da Intel.
3. Um Processador de Display de Vídeo 9929 da Texas
4. Um Gerador de Som Programável 8910 da G.I.
5. 32KB de ROM contendo BASIC
6. Um mínimo de 8KB de RAM

Apesar de, evidentemente, existir um grande número adicional de componentes envolvidos no projeto de um computador MSX, são eles todos de pequena escala, não-programáveis e, portanto, “invisíveis” ao usuário. Os fabricantes, geralmente, têm liberdade considerável na seleção destes componentes de pequena escala. Os componentes programáveis não podem variar e, portanto, todas as máquinas MSX são idênticas no que diz respeito ao programador.

Os Capítulos 1, 2 e 3 descrevem a operação da Interface Periférica Programável, Processador de Display de Vídeo e o Gerador de Som Programável, respectivamente. Estes três dispositivos constituem a interface entre o Z80 e o hardware periférico de uma máquina MSX padrão. Todos ocupam posições na Via de E/S (Entrada/Saída) do Z80.

O Capítulo 4 descreve o software contido na primeira parte da ROM do MSX. Esta seção da ROM serve para controlar o hardware da máquina no nível mais baixo de detalhe, e é conhecida como BIOS (Basic Input/Output System ou Sistema Básico de Entrada/Saída). Ele é estruturado de forma que a maioria das funções úteis ao programador em código de máquina, como interfaces de teclado e de vídeo, estão prontamente disponíveis.

O Capítulo 5 descreve o software contido no restante da ROM, o Interpretador BASIC MSX da Microsoft. Apesar de ser, principalmente, um programa cujo funcionamento é controlado por um texto e, portanto, de pouca utilidade para o programador, um exame detalhado revela diversos pontos não documentados pelos fabricantes.

O Capítulo 6 refere-se à organização da memória do sistema. Especial atenção é dada à Área de Trabalho, uma seção da RAM de F380H a FFFFH, utilizada como



rascunho pelo BIOS e pelo Interpretador BASIC, onde contém dados que são de grande valia para programas aplicativos.

O Capítulo 7 fornece alguns exemplos de programas em código de máquina que utilizam recursos da ROM, minimizando assim, o esforço de desenvolvimento.

Acredita-se que este livro esteja num nível satisfatório. Se você tiver outra opinião, o autor gostaria de manter contato com você. Este livro é dedicado aos interessados em resolver problemas de difícil solução.



## INTERFACE DE PERIFÉRICO PROGRAMÁVEL (PPI)

O PPI 8255 é um dispositivo de interface paralela de utilização geral, compreendendo três portas de dados de oito bits, denominados A, B, e C e uma porta de modo. Desta forma, aparece ao Z80 quatro portas de E/S, pelas quais o teclado, o hardware de comutação de memória, o motor do cassete, a saída do cassete, o LED Caps Lock e o “Click” de Tecla podem ser controlados. Desde que o PPI tenha sido inicializado, o acesso a uma determinada peça do hardware envolve apenas escrever ou ler a porta de E/S relevante.

### Porta A do PPI (Porta de E/S A8H)

|                                    |   |                                    |   |                                    |   |                                    |   |
|------------------------------------|---|------------------------------------|---|------------------------------------|---|------------------------------------|---|
| 7                                  | 6 | 5                                  | 4 | 3                                  | 2 | 1                                  | 0 |
| Página 3<br>PSLOT #<br>C000 - FFFF |   | Página 2<br>PSLOT #<br>8000 - BFFF |   | Página 1<br>PSLOT #<br>4000 - 7FFF |   | Página 0<br>PSLOT #<br>0000 - 3FFF |   |

**Figura 1** Registrador de Conector Primário (Primary Slot Register).

Esta porta de saída, conhecida como Registrador de Conector Primário na terminologia MSX, é utilizada para controlar o hardware de comutação de memória. O microprocessador Z80 pode acessar, diretamente, apenas 64KB de memória. Atualmente, esta

limitação é vista como sendo restritiva, e muitos computadores pessoais empregam métodos para sobrepujá-la.

As máquinas MSX podem ter diversos dispositivos de memória ocupando o mesmo endereço e o Z80 poderá selecionar qualquer um deles quando for necessário. A memória é vista como sendo duplicada “lateralmente” em quatro áreas separadas de 64KB cada uma, denominadas Conectores Primário (Primary Slots) 0 a 3. Cada um dos quatro conectores recebe o seu próprio “sinal de seleção de conector”, além dos sinais normais do Z80. O conteúdo do Registrador de Conector Primário determina qual “sinal de seleção de conector” está ativo e, portanto, qual o Conector Primário que está sendo selecionado.

Para aumentar a flexibilidade, cada página de 16KB de espaço de endereço do Z80 poderá ser selecionada de um Conector Primário diferente. Conforme mostrado na Figura 1, são necessários dois bits do Registrador de Conector Primário para definir o número do Conector Primário referente a cada página de 16KB.

A primeira operação executada pela ROM do MSX na partida (power-up) é procurar RAM nas páginas 2 e 3 (8000-FFFF) de cada conector. O Registrador de Conector Primário é inicializado de forma que os conectores contendo RAM sejam selecionados, fazendo com que esta seja “enxergada” pelo processador. A configuração de memória de qualquer máquina MSX pode ser determinada com a instrução BASIC, vista abaixo, que mostra o Registrador de Conector Primário:

```
PRINT RIGHT$("0000000" + BIN$(INP(&HA8)),8)
```

Como exemplo, “10100000” seria produzido em um Toshiba HX10 em que as páginas 3 e 2 (a RAM) viriam ambas do Conector Primário 2 e as páginas 1 e 0 (a ROM do MSX) do Conector Primário 0. A ROM do MSX deve ser sempre colocada no Conector Primário 0, uma vez que este é o conector selecionado pelo hardware na partida. Os fabricantes podem colocar outros dispositivos de memória RAM e qualquer ROM adicional em qualquer outro conector.

Uma máquina inglesa típica tem um Conector Primário contendo a ROM do MSX, um outro conector contendo 64KB de RAM e dois conectores ligados às saídas externas. A maioria das máquinas japonesas tem um conector do tipo cartucho em cada uma destas saídas externas. As máquinas inglesas, normalmente, têm um conector tipo cartucho e um conector IDC.

## Expansores

A memória do sistema poderá ser aumentada a um máximo teórico de dezesseis áreas de 64KB, utilizando interfaces de expansão. Um expansor é plugado, em qualquer Conector Primário, para fornecer quatro Conectores Secundários de 64KB, numerados de 0 a 3, em lugar de apenas um primário. Cada expansor tem o seu próprio hardware local, denominado Registrador de Conector Secundário, que deve selecionar qual dos Conectores Secundários deverá aparecer no Conector Primário. Da mesma forma que antes, as páginas podem ser selecionadas de Conectores Secundários diferentes.

|                   |   |                   |   |                   |   |                   |   |
|-------------------|---|-------------------|---|-------------------|---|-------------------|---|
| 7                 | 6 | 5                 | 4 | 3                 | 2 | 1                 | 0 |
| Página 3<br>SSLOT |   | Página 2<br>SSLOT |   | Página 1<br>SSLOT |   | Página 0<br>SSLOT |   |

**Figura 2** Registrador de Conector Secundário.

Cada Registrador de Conector Secundário, que apesar de ser realmente um latch do tipo ler/escrever de oito bits, aparece como a posição de memória FFFFH de seu Conector Primário pelo hardware do expansor. Para se conseguir acesso a esta posição, em um determinado expansor, é normalmente necessário comutar primeiro a página 3(COOOH a FFFFH) daquele Conector Primário para o espaço de endereço do processador. O Registrador de Conector Secundário poderá, em seguida, ser modificado e, se necessário, a página 3 será restaurada à sua condição original no Conector Primário. O acesso às memórias em expansores pode se tornar um processo bastante complicado.

Fica claro que deve haver um meio para se determinar se um Conector Primário contém RAM comum ou um expansor, para que possa ser corretamente acessado. Para possibilitar isto, os Registradores de Conectores Secundários são projetados para inverter seus conteúdos quando lidos. Durante a procura da RAM na partida, a posição de memória FFFFH de cada Conector Primário é examinada para sabermos se ela se comporta normalmente ou se o conector contém um expansor. Os resultados destes testes são armazenados no mapa de recursos do sistema (EXPTBL) da Área de Trabalho, para utilização posterior. Isto é feito na partida, devido à dificuldade na execução de testes, quando os Registradores de Conectores Secundários contiverem valores significativos.

Comutação de memória é, obviamente, uma tarefa que exige cuidados adicionais, particularmente, com os mecanismos hierárquicos necessários ao controle dos expansores. Deve-se ter cuidado para evitar o desligamento da página onde um programa

está sendo executado ou, quando a página contendo a pilha estiver sendo utilizada. Há um número de rotina padronizadas disponíveis ao programador em código de máquina, na seção BIOS da ROM do MSX para simplificar esse processo.

O próprio Interpretador BASIC tem quatro métodos para acessar ROMs de expansão. Os três primeiros são utilizados com programas em código de máquina em ROM, colocados na página 1(4000H a 7FFFH), e são eles:

1. Ganchos (Capítulo 6).
2. Instrução "CALL" (Capítulo 5).
3. Nomes de dispositivos adicionais (Capítulo 5).

O Interpretador BASIC poderá executar também um programa BASIC em ROM detectado na página 2(8000H a BFFFH), durante a procura de ROM na partida. O que o Interpretador BASIC não poderá fazer é utilizar qualquer RAM escondida atrás de um outro dispositivo de memória. Esta limitação é o reflexo da dificuldade que se tem em converter um programa já consagrado para que se possa tirar vantagens de máquinas novas e mais complexas. Existe uma situação parecida com a versão do BASIC da Microsoft disponível ao PC da IBM. De um total de 1MB de espaço de memória, apenas 64KB podem ser utilizados para armazenamento de programas.

### Porta B do PPI (Porta de E/S A9H)

7 6 5 4 3 2 1 0

Entradas da Coluna do Teclado

**Figura 3**

Esta porta de entrada é utilizada para ler os oito bits de dados de coluna, da linha normalmente selecionada do teclado. O teclado do MSX é uma matriz de onze linhas por oito colunas, varrida pelo software, e formada por teclas, freqüentemente abertas. As máquinas atuais normalmente têm teclas apenas nas linhas de zero a oito. A conversão das operações das teclas em códigos de caracteres é executada pelo manipulador de interrupções em ROM do MSX. Este processo será descrito no Capítulo 4.

**Porta C do PPI (Porta de E/S AAH)**

| 7              | 6       | 5         | 4         | 3                     | 2 | 1 | 0 |
|----------------|---------|-----------|-----------|-----------------------|---|---|---|
| Click de Tecla | LED Cap | Saída Cas | Motor Cas | Seleção Linha Teclado |   |   |   |

**Figura 4**

Esta porta de saída controla uma variedade de funções. Os quatro bits de Seleção de Linha do Teclado selecionam qual das onze linhas do teclado, numeradas de 0 a 10, deverá ser lida pela porta B do PPI.

O bit Motor do Cassete determina a situação do relê do motor do cassete: 0=ligado, 1=desligado.

O bit de Saída do Cassete é filtrado e atenuado, antes de ser levado ao soquete DIN do cassete, como sinal do Microfone. Toda geração de tom do cassete é executada por software.

O bit LED Cap determina o estado do LED Caps Lock: 0=ligado, 1=desligado.

A saída "Click" do teclado é atenuada e misturada à saída de áudio do Gerador de Som Programável. Para, realmente, gerar um som, este bit deve ser ativado e desativado.

Observe que há rotinas padrões na BIOS em ROM, para poder acessar todas as funções relacionadas com esta porta. Caso isto seja possível, deve-se utilizar estas funções e não manipular diretamente o hardware.

**Porta de Modo da PPI (Porta de E/S ABH)**

| 7 | 6        | 5 | 4     | 3     | 2        | 1     | 0     |
|---|----------|---|-------|-------|----------|-------|-------|
| 1 | Modo A&C |   | Dir A | Dir C | Modo B&C | Dir B | Dir C |

**Figura 5** Seleção de Modo do PPI.

Esta porta é utilizada para estabelecer o modo de operação da PPI. Uma vez que o hardware do MSX é projetado para funcionar, exclusivamente, em uma determinada configuração, esta porta não deverá ser modificada sob quaisquer circunstâncias. Os detalhes a seguir são fornecidos apenas como complementação.

O bit 7 deverá ser 1 para que se possa alterar o modo de PPI; quando é 0 a PPI executa a função set/reset de um único bit, mostrado na Figura 6.

Os bits Modo A&C determinam o modo de operação para a porta A e os quatro bits superiores, apenas, para a porta C: 00=Modo Normal(MSX), 01=Modo Strobe, 10=Modo Bidirecional.

O modo DIR A determina o sentido para a porta A: 0=Saída(MSX), 1=Entrada.

O bit DIR C determina, apenas para a porta C, o sentido dos quatro bits superiores: 0=Saída(MSX), 1=Entrada.

Os bits do Modo B&C determinam o modo de operação para a porta B e os quatro bits inferiores, apenas para a porta C: 0=Modo Normal(MSX), 1=Modo Strobe.

O bit Dir B determina o sentido para a porta B: 0=Saída, 1=Entrada(MSX).

O bit DIR C determina, apenas para a porta C, o sentido dos quatro bits inferiores: 0=Saída(MSX), 1=Entrada.

| 7 | 6             | 5 | 4 | 3             | 2 | 1 | 0            |
|---|---------------|---|---|---------------|---|---|--------------|
| 0 | Não-utilizado |   |   | Número de bit |   |   | Set<br>Reset |

**Figura 6** Set/Reset do Bit de PPI.

A porta de Modo da PPI poderá ser utilizada para ativar ou desativar, diretamente, qualquer bit da porta C, quando o bit 7 é 0. O Número-de-Bit, de 0 a 7, determina qual o bit que será afetado. Seu novo valor será determinado pelo bit Set/Reset: 0=Res, 1=Set. A vantagem deste modo é que uma única saída poderá facilmente ser modificada. Como exemplo, o LED Caps Lock poderá ser aceso com a declaração em BASIC, OUT &HAB,12, e apagado com a declaração em OUT &HAB, 13.



## O PROCESSADOR DE APRESENTAÇÃO DE VÍDEO (VDP)

O VDP 9929 contém todos os circuitos necessários à geração da tela de vídeo. Ele aparece ao Z80 como duas portas de E/S, denominadas de Porta de Dados e Porta de Comando. Apesar de VDP ter sua própria VRAM de 16KB (RAM de Vídeo), cujo conteúdo define a imagem da tela-mestra ele não poderá ser acessado diretamente pelo Z80. Ele precisará utilizar as duas portas de E/S para modificar a VRAM e estabelecer as diversas condições de operação do VDP.

### Porta de Dados (Porta de E/S 98H)

A Porta de Dados é utilizada para ler ou escrever bytes na VRAM. O VDP possui um registrador de endereçamento interno apontando para uma posição na VRAM. A leitura da Porta de Dados permitirá a entrada do byte desta posição da VRAM, ao passo que escrever na Porta de Dados armazenará um byte nesta posição da VRAM. Após uma leitura, ou escrita, o registrador de endereço é, automaticamente, incrementado para apontar à próxima posição da VRAM. Bytes sequenciais poderão ser acessados, simplesmente, por meios de leituras ou escritas contínuas na Porta de Dados.

## Porta de Comando (Porta de E/S 99H)

A Porta de Comando é utilizada para três finalidades:

1. Para inicializar o registrador de endereço da Porta de Dados.
2. Para ler o registrador de Estado do VDP (Status Register).
3. Para escrever em um dos Registradores de Modo do VDP.

## Registrador de Endereço

O Registrador de Endereço da Porta de Dados deverá ser inicializado de forma diferente, dependendo se o próximo acesso será de leitura ou de escrita. Qualquer valor, de 0000H até 3FFFH poderá ser colocado no registrador de endereço escrevendo-se primeiro o LSB (Byte Menos Significativo) e depois o MSB (Byte Mais Significativo) na Porta de Comando. Os bits 6 e 7 do MSB são utilizados pelo VDP para determinar se o registrador de endereço está sendo preparado para leituras ou escritas subseqüentes, como segue:

|          |          |            |
|----------|----------|------------|
| Ler      | XXXXXXXX | 00XXXXXXXX |
| Escrever | XXXXXXXX | 01XXXXXXXX |

**Figura 7** Estabelecimento do endereço do VDP.

É importante que nenhum outro acesso seja feito ao VDP entre a escrita do LSB e do MSB, uma vez que isto atrapalharia a sua sincronização. O manipulador de interrupção em ROM do MSX lê continuamente o Registrador de Estado do VDP como uma tarefa secundária, de modo que as interrupções deverão ser desativadas, se necessário.

## Registrador de Estado do VDP (VDP Status Register)

A leitura da Porta de Comando permite a entrada do conteúdo do Registrador de Estado do VDP. Ele contém diversos indicadores:

|            |             |            |                         |   |   |   |   |
|------------|-------------|------------|-------------------------|---|---|---|---|
| 7          | 6           | 5          | 4                       | 3 | 2 | 1 | 0 |
| Indic<br>F | Indic<br>SS | Indic<br>C | Número do Quinto Sprite |   |   |   |   |

**Figura 8** Registrador de Estado do VDP.

Os bits que formam o Número do Quinto Sprite contém o número (0 a 31) do sprite que acionou o Indicador do Quinto Sprite.

O Indicador de Coincidência normalmente é 0, mas será colocado em 1 se quaisquer sprites tiverem um ou mais pixels sobrepostos. A leitura do Registrador de Status repõe este indicador em 0. Observe que a coincidência é verificada apenas à medida em que cada pixel é gerado, durante um quadro de vídeo; em uma máquina inglesa isto ocorre a cada 20 mS. Caso sprites em movimentação rápida passem um sobre o outro entre verificações, nenhuma coincidência será indicada.

O Indicador do Quinto Sprite normalmente é 0, mas será colocado em 1 quando houver mais de quatro sprites em uma linha de pixels qualquer. A leitura do Registrador de Estado repõe este indicador em 0.

O Indicador de Quadro normalmente é 0, mas será colocado em 1 ao final da última linha ativa do quadro de vídeo. Para as máquinas inglesas com uma frequência de quadro de 50 Hz, isto ocorrerá cada 20 mS. A leitura do Registrador de Estado reporá este indicador para 0. Há um sinal de saída associado, do VDP, que gera interrupções para o Z80 nessa mesma periodicidade; isto aciona o manipulador de interrupções em ROM do MSX.

## Registradores de Modo do VDP

O VDP tem oito registradores apenas-para-escrita (write-only), numerados de 0 a 7, que controlam sua operação geral. Um determinado registrador é selecionado primeiro, escrevendo um byte de dados; depois um byte de seleção do registrador na Porta de Comando. O byte de seleção do registrador contém o número do registrador nos três bits menos significativos: 1000RRR. Como os Registradores de Modo são apenas para escrita, e não podem ser lidos, a ROM do MSX mantém uma cópia exata dos oito registradores na Área de Trabalho da RAM (Capítulo 6). A utilização das rotinas padrões da ROM do MSX para as funções VDP garante que a imagem deste registrador seja atualizada corretamente.

**Registrador Modo 0**

| 7 | 6 | 5 | 4 | 3 | 2 | 1  | 0  |
|---|---|---|---|---|---|----|----|
| 0 | 0 | 0 | 0 | 0 | 0 | M3 | VE |

**Figura 9**

O bit VDP Externo determina se a entrada externa do VDP, deverá ser ativada ou desativada: 0=Desativada, 1=Ativada.

O bit M3 é um dos três bits de seleção de modo do VDP. Veja o Registrador Modo 1.

**Registrador Modo 1**

| 7     | 6     | 5  | 4  | 3  | 2 | 1       | 0   |
|-------|-------|----|----|----|---|---------|-----|
| 4/16K | Limpa | IE | M1 | M2 | 0 | Tamanho | MAG |

**Figura 10**

O bit Magnitude determina se os sprites deverão ser de tamanho normal ou duplicado: 0=Normal, 1=Duplicado.

O bit de Tamanho determina se os sprites serão de 8x8 bits ou 16x16 bits: 0=8x8, 1=16x16.

Os bits M1 e M2 determinam o modo de operação do VDP, juntamente com o bit M3 do Registrador Modo 0:

| M1 | M2 | M3 |                    |
|----|----|----|--------------------|
| 0  | 0  | 0  | Modo Texto 32x24   |
| 0  | 0  | 1  | Modos Gráficos     |
| 0  | 1  | 0  | Modo Multicolorido |
| 1  | 0  | 0  | Modo Texto 40x24   |

O bit de Ativação de Interrupção ativa ou desativa o sinal de saída de interrupção do VDP: 0=Desativa, 1=Ativa.

O bit "Limpa" é utilizado para ativar ou desativar toda a tela de vídeo: 0=Desativa, 1=Ativa. A tela, ao ficar vazia, terá a mesma cor do contorno.

O bit 4/16K altera as características de endereçamento da VRAM do VDP, para que este utilize chips de 4KB ou 16KB: 0=4KB, 1=16KB.

### Registrador Modo 2

|   |   |   |   |                         |   |   |   |
|---|---|---|---|-------------------------|---|---|---|
| 7 | 6 | 5 | 4 | 3                       | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | Base da Tabela de Nomes |   |   |   |

**Figura 11**

Registrador Modo 2 define o endereço inicial da Tabela de Nomes na VRAM do VDP. Os quatro bits disponíveis especificam apenas as posições 00BB BB00 0000 0000 do endereço completo, de modo que um conteúdo 0FH, no registrador, resulte em um endereço base igual a 3C00H.

### Registrador Modo 3

|                         |   |   |   |   |   |   |   |
|-------------------------|---|---|---|---|---|---|---|
| 7                       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Base da Tabela de Cores |   |   |   |   |   |   |   |

**Figura 12**

O Registrador Modo 3 define o endereço inicial da Tabela de Cores na VRAM do VDP. Os oito bits disponíveis especificam apenas as posições 00BB BBBB BB00 0000 do endereço completo, de modo que um conteúdo FFH, no registrador, resulte em um endereço base igual a 3FC0H. No Modo Gráfico apenas o bit 7 tem efeito, oferecendo desta maneira uma base de 0000H ou 2000H. Os bits 0 à 6 devem ser 1.

**Registrador Modo 4**

|   |   |   |   |   |                                     |   |   |
|---|---|---|---|---|-------------------------------------|---|---|
| 7 | 6 | 5 | 4 | 3 | 2                                   | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | Bases das Imagens dos<br>Caracteres |   |   |

**Figura 13**

O Registrador Modo 4 define o endereço inicial da Tabela de Imagens dos Caracteres na VRAM do VDP. Os três bits disponíveis especificam apenas as posições 00BB B000 0000 0000 do endereço completo, de modo que um conteúdo 07H no registrador resulte em um endereço base igual a 3800H. No Modo Gráfico apenas o bit 2 é efetivo, oferecendo assim uma base de 0000H ou 2000H. Os bits 0 e 1 devem ser 1.

**Registrador Modo 5**

|   |                               |   |   |   |   |   |   |
|---|-------------------------------|---|---|---|---|---|---|
| 7 | 6                             | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Base de Atributos dos Sprites |   |   |   |   |   |   |

**Figura 14**

O Registrador Modo 5 define o endereço inicial da Tabela de Atributo do Sprite na VRAM do DVP. Os setes bits disponíveis especificam apenas as posições 00BB BBBB B000 0000 do endereço completo, de modo que um conteúdo 7FH no registrador resulte em um endereço base igual a 3F80H.

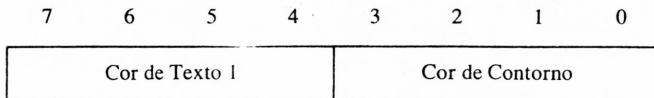
**Registrador Modo 6**

|   |   |   |   |   |                                  |   |   |
|---|---|---|---|---|----------------------------------|---|---|
| 7 | 6 | 5 | 4 | 3 | 2                                | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | Bases das Imagens<br>dos Sprites |   |   |

**Figura 15**

Registrador Modo 6 define o endereço inicial da Tabela de Imagens dos sprites na VRAM do VDP. Os três bits disponíveis especificam apenas as posições 00BB B000 0000 0000 do endereço completo, de modo que um conteúdo 07H no registrador resulte em um endereço base igual a 3800H.

### Registrador Modo 7



**Figura 16**

Os bits da Cor de Contorno determinam a cor da região que envolve a área ativa do vídeo, em todos os quatro modos do VDP. Determinam também a cor de todos os pixels 0 na tela, no Modo Texto 40x24. Observe que a região de contorno realmente se estende pela tela inteira, mas se tornará visível apenas na área ativa, se o pixel sobreposto for transparente.

O bits de Cor de Texto 1 determinam a cor de todos os pixels 1 no Modo Texto 40x24. Eles não têm nenhum efeito nos três outros modos, onde é fornecida uma flexibilidade maior, pela utilização da Tabela de Cores. Os códigos de cores do VDP são:

|                |                 |                  |                 |
|----------------|-----------------|------------------|-----------------|
| 0 Transparente | 4 Azul escuro   | 8 Vermelho       | 12 Verde escuro |
| 1 Preto        | 5 Azul claro    | 9 Vermelho claro | 13 Púrpura      |
| 2 Verde        | 6 Vermelho esc. | 10 Amarelo       | 14 Cinza        |
| 3 Verde claro  | 7 Azul celeste  | 11 Amarelo claro | 15 Branco       |

### Modo de Tela

O VDP tem quatro modos de operação, cada um oferecendo um conjunto de possibilidades ligeiramente diferentes. De forma geral, à medida que a resolução sobe, o preço a ser pago em tamanho de memória de vídeo e complexidade de atualização também aumenta. Em uma aplicação delicada, estes custos associados, de hardware e software, devem ser considerados. Para uma máquina MSX eles SÃO IRRELEVANTES, sendo,

portanto lamentável que uma tentativa maior para padronizar um determinado modo não tenha sido feita. O Modo Gráfico é capaz de executar, adequadamente, todas as funções dos demais modos, apenas com pequenas restrições.

Uma dificuldade adicional na utilização do VDP ocorre porque se deixou em seu projeto espaço insuficiente para a sobreposição de varredura, utilizada pela maioria dos televisores. A conseqüente perda de caracteres nas laterais da tela, forçou todo o software para o MSX relacionado-ao-vídeo a se basear em tamanhos peculiares de tela. As máquinas inglesas, normalmente, utilizam apenas os trinta e sete caracteres centrais disponíveis no Modo Texto 40x24. As máquinas japonesas, que utilizam o padrão NTSC (National Television Standards Committee), usam os trinta e nove caracteres centrais.

O elemento central do VDP, do ponto de vista do programador, é a Tabela de Nomes. Ela é apenas uma lista de códigos de caracteres de um byte, mantidos na VRAM. Tem um tamanho de 960 bytes no Modo Texto de 40x24, um tamanho de 768 bytes no Modo Texto 32x24, Modo Gráfico e Modo Multicolorido. Cada posição da Tabela de Nomes corresponde a uma determinada posição na tela.

Durante um quadro de vídeo, o VDP lerá seqüencialmente cada código de caractere da Tabela de Nomes iniciando pela base. À medida que cada código de caractere é lido, o padrão de pixel 8x8, correspondente, será procurado na Tabela de Imagens dos Caracteres e apresentado na tela. A apresentação na tela poderá assim ser modificada, pela alteração dos códigos de caracteres na Tabela de Nomes, ou dos padrões de pixels na Tabela de Imagens dos Caracteres.

Observe que o VDP não tem o cursor em hardware. Caso um cursor seja necessário, ele, terá de ser gerado por software.

## **Modo Texto 40x24**

A Tabela de Nomes ocupa 960 bytes de VRAM de 0000H a 03BFH:







## Modo Gráfico

A Tabela de Nomes ocupa 768 bytes de VRAM, desde 1800H até 1AFFH, o mesmo que no Modo Texto 32x24. A tabela é inicializada com a sequência de código de caracteres 0 a 255, repetido três vezes e depois permanece intacta; neste modo é a Tabela de Imagens de Caracteres que é modificada, durante a operação normal.

A Tabela de Imagens de Caracteres ocupa 6KB de VRAM, de 0000H a 17FFH. Apesar de sua estrutura ser a mesma que nos modos texto, ela não contém um conjunto de caracteres, mas é inicializada com todos os pixels iguais a 0. Os primeiros 2KB da Tabela de Imagens de Caracteres são endereçados pelos códigos de caractere da primeira terça parte da Tabela de Nomes, os segundos 2KB pela terça parte central da Tabela de Nomes e os últimos 2KB pela última terça parte da Tabela de Nomes. Em virtude do padrão sequencial na Tabela de Nomes, toda a Tabela de Imagens de Caracteres é lida linearmente, durante um quadro de vídeo. Colocar um ponto na tela significa descobrir aonde o bit correspondente se encontra na Tabela de Imagens de Caracteres e ligá-lo. Para um programa BASIC converter as coordenadas X,Y a um endereço, veja a rotina padrão MAPXYC no Capítulo 4.

|       |  |    |
|-------|--|----|
| 0000H |  | 0  |
| 0100H |  | 1  |
| 0200H |  | 2  |
| 0300H |  | 3  |
| 0400H |  | 4  |
| 0500H |  | 5  |
| 0600H |  | 6  |
| 0700H |  | 7  |
| 0800H |  | 8  |
| 0900H |  | 9  |
| 0A00H |  | 10 |
| 0B00H |  | 11 |
| 0C00H |  | 12 |
| 0D00H |  | 13 |
| 0E00H |  | 14 |
| 0F00H |  | 15 |
| 1000H |  | 16 |
| 1100H |  | 17 |
| 1200H |  | 18 |
| 1300H |  | 19 |
| 1400H |  | 20 |
| 1500H |  | 21 |
| 1600H |  | 22 |
| 1700H |  | 23 |

01234567890123456789012345678901

**Figura 20** Tabela de Imagens de Caracteres Gráficos.

A cor de contorno é definida pelo Registrador Modo 7 do VDP e, inicialmente, é azul. A Tabela de Cores ocupa 6KB de VRAM de 2000H até 37FFH. Existe um mapeamento exato byte-a-byte, da Tabela de Imagens de Caracteres à Tabela de Cores, mas,

pelo fato de ser necessário um byte inteiro para definir as cores do pixel 0 e do pixel 1, há uma resolução menor para cores do que há para pixels. Os quatros bits inferiores, de uma entrada de Tabela de Cores, definem a cor de todos os pixels 0, na linha de oito pixels correspondentes. Os quatros bits superiores definem a cor dos pixels 1. A Tabela de Cores é inicializada, de modo que a cor do pixel 0 e a cor do pixel 1 sejam azuis para toda a tabela. Pelo fato das duas cores serem a mesma, será necessário alterar uma das cores quando um bit for ligado na Tabela de Imagens de Caractere.

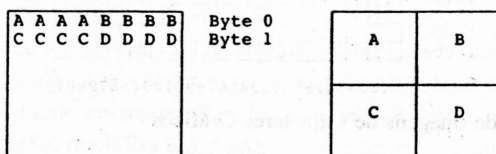
### Modo Multicolorido

A Tabela de Nomes ocupa 768 bytes de VRAM de 0800H até 0AFFH; o mapeamento da tela é o mesmo que no Modo Texto 32x24. A tabela é inicializada com o seguinte padrão de código de caractere.

00F a 1FH (Repetido quatro vezes)  
 20H a 3FH (Repetido quatro vezes)  
 40H a 5FH (Repetido quatro vezes)  
 60H a 7FH (Repetido quatro vezes)  
 80H a 9FH (Repetido quatro vezes)  
 A0H a BFH (Repetido quatro vezes)

Da mesma forma que no Modo Gráfico, esta tabela é usada apenas para mapear a Tabela de Imagens de Caracteres; esta última é a que será modificada durante a operação normal.

A Tabela de Imagens de Caracteres ocupa 1536 bytes da VRAM, de 0000H a 05FFH. Como nos outros modos, cada código de caractere corresponde a um bloco de oito bytes na Tabela de Imagens de Caracteres. Devido à resolução menor, neste modo, apenas dois bytes do bloco de imagens de caracteres são necessários para definir uma imagem de 8x8:



**Figura 21** Bloco de margem Multicolorido.

Como se vê na Figura 21 cada grupo de quatro bits do bloco contém um código de cor, definindo assim a cor de um dos quadrantes da imagem 8x8. Para que todos os oito bytes do bloco de imagem possam ser utilizados, um dado código de caractere utilizará um grupo de dois bytes diferentes, dependendo da posição na tela que corresponde ao código de caractere (isto é, a sua posição na Tabela de Nome):

- Linha de Vídeo 0,4,8,12,16,20 utiliza bytes 0 e 1
- Linha de Vídeo 1,5,9,13,17,21 utiliza bytes 2 e 3
- Linha de Vídeo 2,6,10,14,18,22 utiliza bytes 4 e 5
- Linha de Vídeo 3,7,11,15,19,23 utiliza bytes 6 e 7

Quando a Tabela de Nomes é preenchida, com a seqüência de códigos de caracteres, mostrada anteriormente, a Tabela de Imagens de Caracteres será lida linearmente, durante uma tela de vídeo:

|       |  |    |
|-------|--|----|
| 0000H |  | 0  |
| 0002H |  | 1  |
| 0004H |  | 2  |
| 0006H |  | 3  |
| 0100H |  | 4  |
| 0102H |  | 5  |
| 0104H |  | 6  |
| 0106H |  | 7  |
| 0200H |  | 8  |
| 0202H |  | 9  |
| 0204H |  | 10 |
| 0206H |  | 11 |
| 0300H |  | 12 |
| 0302H |  | 13 |
| 0304H |  | 14 |
| 0306H |  | 15 |
| 0400H |  | 16 |
| 0402H |  | 17 |
| 0404H |  | 18 |
| 0406H |  | 19 |
| 0500H |  | 20 |
| 0502H |  | 21 |
| 0504H |  | 22 |
| 0506H |  | 23 |

01234567890123456789012345678901

Figura 22 Tabela de Imagens de Caracteres Multicolorida.

A cor de contorno é definida pela Registrador Modo 7 do VDP e, inicialmente, é azul. Não há Tabela de Cor separada, uma vez que as cores são definidas diretamente pelo conteúdo da Tabela de Imagens de Caracteres; esta é inicialmente preenchida com azul.

## Sprites

O VDP pode controlar trinta e dois sprites em todos os modos, exceto no Modo Texto 40x24. Seu tratamento é idêntico em todos os modos e independente de qualquer atividade relacionada com os caracteres.

A Tabela dos Atributos dos Sprites ocupa 128 bytes de VRAM, desde 1B00H a 1B7FH. A tabela contém trinta e dois blocos de quatro bytes, um bloco para cada sprite. O primeiro bloco controla o sprite 0 (o sprite de “cima”), o segundo bloco controla o sprite 1, e assim por diante, até o sprite 31. O formato de cada bloco será:

|                    |   |   |   |               |   |   |   |        |
|--------------------|---|---|---|---------------|---|---|---|--------|
| 7                  | 6 | 5 | 4 | 3             | 2 | 1 | 0 |        |
| Posição Vertical   |   |   |   |               |   |   |   | Byte 0 |
| Posição Horizontal |   |   |   |               |   |   |   | Byte 1 |
| Número da Imagem   |   |   |   |               |   |   |   | Byte 2 |
| EC                 | 0 | 0 | 0 | Código de Cor |   |   |   | Byte 3 |

**Figura 23** Bloco de Atributo do Sprite.

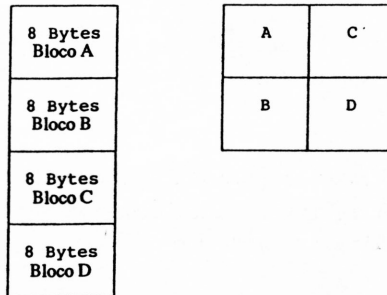
O byte 0 especifica a coordenada vertical (Y) do pixel superior esquerdo do sprite. O sistema de coordenadas varia de -1 (FFH) para a linha de pixel superior na tela, descendo até 190 (BEH) para a linha inferior. Valores menores que -1 poderão ser utilizados para “deslizar” o sprite para dentro da tela, vindo de cima. Os valores exatos necessários, obviamente, dependerão do tamanho do sprite. Curiosamente, não houve nenhuma tentativa no BASIC MSX para conciliar este sistema de coordenadas com a faixa de coordenadas gráficas normal, Y=0 até 191. Como consequência, um sprite estará sempre um pixel mais baixo na tela, do que o seu ponto gráfico equivalente. Observe que, o valor da coordenada vertical especial de 208 (DOH) colocado em um bloco de atributo de sprite fará com que o VDP ignore todos blocos subseqüentes da Tabela de Atributo de Sprites. Isto significa que qualquer sprite inferior desaparecerá da tela.

O byte 1 especifica a coordenada horizontal (X) do pixel superior esquerdo do sprite. O sistema de coordenadas vai de 0, para o pixel mais à esquerda, até 255(FFH), para o pixel mais à direita. Como este sistema de coordenadas não fornece nenhum mecanismo para “deslizar” um sprite para a tela a partir da esquerda, um bit especial no byte 3 é utilizado para esta finalidade, conforme veremos a seguir.

O byte 2 seleciona uma das 256 imagens 8x8, disponível na Tabela de Imagens de Sprites. Caso o bit Tamanho esteja ligado no Registrador Modo 1 do VDP, o que corresponde a imagens de 16x16 bits, ocupando 32 bytes cada, os dois bits menos significativos no número da imagem serão ignorados. Assim, os números de padrão 0, 1, 2 e 3 selecionariam, igualmente, a imagem número 0.

No byte 3 os quatro bits do Código de Cor definem a cor dos pixels de valor 1, no sprite; os pixels 0 são sempre transparentes. O bit “Adianta” (Early Clock), tem, normalmente, o valor 0. Quando seu valor for 1, o sprite será deslocado trinta e dois pixels, à esquerda. Assim, será possível fazer com que sprites “deslizem” para dentro da tela, a partir da esquerda mesmo não havendo coordenadas de reserva na direção horizontal.

A Tabela de Imagens de Sprites ocupa 2KB de VRAM, de 3800H até 3FFFH. Ela contém 256 imagens de pixel 8x8, numeradas de 0 a 255. Caso o bit Tamanho no Registrador Modo 1 do VDP seja 0, (sprite 8x8), cada bloco de imagem de sprite, de oito bytes, fica com a mesma estrutura que o bloco de imagem de caractere, mostrado na Figura 18. Caso o bit Tamanho seja 1, (sprite 16x16), quatro blocos de oito bytes serão necessários para definir a imagem, como indicado abaixo:



**Figura 24** Bloco de Imagens do Sprite 16x16.

## **GERADOR DE SOM PROGRAMÁVEL (PSG)**

Além de controlar três canais de som, o PSG 8910 contém duas portas de dados de oito bits, denominadas de A e B, pelas quais realiza a interface com o joystick e a entrada do cassete. O PSG aparece ao Z80 como três portas de E/S denominadas Porta de Endereço, Porta de Escrita de Dados e Porta de Leitura de Dados.

### **Porta de Endereço (Porta A0H de E/S)**

O PSG contém dezesseis registradores internos que definem completamente sua operação. Um determinado registrador é selecionado escrevendo seu número de 0 a 15, nesta porta. Uma vez selecionado o registrador, acessos repetidos a ela poderão ser feitos pelas duas portas de dados.

### **Porta de Escrita de Dados (Porta A1H de E/S)**

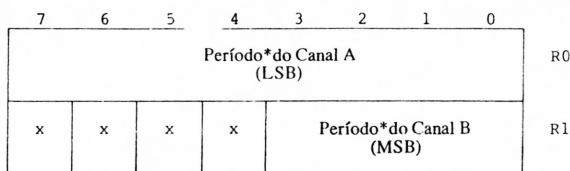
Esta porta é utilizada para escrever em qualquer registrador selecionado na Porta de Endereço.



## Porta de Leitura de Dados (Porta A2H de E/S)

Esta porta é utilizada para ler qualquer registrador selecionado na Porta de Endereço.

### Registradores 0 e 1



**Figura 25**

Estes dois registradores são utilizados para definir o período do Gerador de Som para o Canal A. Frequências variáveis são produzidas dividindo uma frequência mestre fixa pelo número mantido nos registradores 0 e 1. Este número poderá estar na faixa de 1 até 4095. O Registrador 0 contém os oito bits menos significativos e o Registrador 1 os quatro mais significativos. O PSG divide uma frequência externa de 1.7897725MHz\*\* por dezesseis para produzir uma frequência mestre do Gerador de Tom de 111861Hz. A Saída do Gerador de Tom poderá, portanto, variar de 111861Hz (dividida por 1) até 27.3Hz (dividida por 4095). Como exemplo, para produzir um “Lá” Central (440Hz) o valor do divisor nos Registradores 0 e 1 combinados seria 254.

\* No original foi usado o termo “frequência” em vez de “período”. Este último termo é o correto, conforme pode ser visto na página 39 do livro MSX-MÚSICA (Ed. McGraw-Hill). A frequência é o inverso do período.

\*\* Este valor de “Clock” é utilizado pelas máquinas inglesas. No Brasil: Expert: 1,788055KHz; Hot-bit: 1,789770KHz.

## Registradores 2 e 3

Estes dois registradores controlam o Gerador de Som do Canal B, de maneira análoga ao canal A.

## Registradores 4 e 5

Estes dois registradores controlam o Gerador de Som do Canal C, de maneira análoga ao canal A.

## Registrador 6

|   |   |   |                     |   |   |   |   |
|---|---|---|---------------------|---|---|---|---|
| 7 | 6 | 5 | 4                   | 3 | 2 | 1 | 0 |
| x | x | x | Frequência de Ruído |   |   |   |   |

**Figura 26**

Além de três Geradores de Som de onda quadrada, o PSG contém um único Gerador de Ruído. A frequência fundamental da fonte de ruído pode ser controlada de forma semelhante aos dos Geradores de Som. Os cinco bits menos significativos do Registrador 6 contém um divisor de 1 a 31. A frequência mestre do Gerador de Ruído é 111861Hz, como antes.

## Registrador 7

|                |                |            |            |            |          |          |          |
|----------------|----------------|------------|------------|------------|----------|----------|----------|
| 7              | 6              | 5          | 4          | 3          | 2        | 1        | 0        |
| Porta B<br>Dir | Porta A<br>Dir | Ruído<br>C | Ruído<br>B | Ruído<br>A | Som<br>C | Som<br>B | Som<br>A |

**Figura 27**

Este registrador ativa ou desativa o Gerador de Som e Gerador de Ruído para cada um dos três canais: 0=Ativa, 1=Desativa. Controla também, o sentido das portas de interface A e B, os quais o joystick e o cassette estão ligados: 0=Entrada, 1=Saída. O Registrador 7 deverá conter sempre 10xxxxxx ou poderá haver um possível dano ao PSG, pois há dispositivos ativos ligados aos seus pinos de E/S. A declaração "SOUND" do BASIC forçará estes bits a terem o valor correto para o Registrador 7, mas não há proteção a nível de código de máquina.

### Registrador 8

|   |   |   |      |                      |   |   |   |
|---|---|---|------|----------------------|---|---|---|
| 7 | 6 | 5 | 4    | 3                    | 2 | 1 | 0 |
| x | x | x | Modo | Amplitude do Canal A |   |   |   |

Figura 28

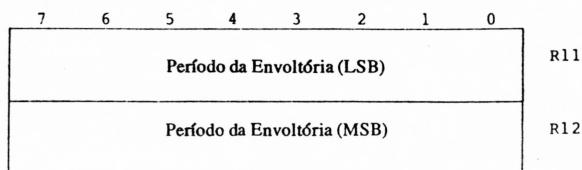
Os quatro bits de Amplitude determinam a amplitude do Canal A, de um mínimo de 0 a um máximo de 15.0 bit de Modo seleciona uma amplitude fixa ou modulada: 0=Fixo, 1=Modulado. Quando a amplitude modulada é selecionada, o valor fixo da amplitude é ignorado e o canal é modulado pelo Gerador de Envoltória.

### Registrador 9

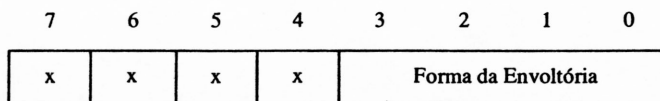
Este registrador controla a amplitude para o Canal B, da mesma forma que no Canal A.

### Registrador 10

Este registrador controla a amplitude para o Canal C, da mesma forma que o Canal A.

**Registrador 11 e 12****Figura 29**

Estes dois registradores controlam o período do Gerador de Envoltória utilizada para a modulação da amplitude. Da mesma forma que para os Geradores de Som, a frequência é determinada colocando um divisor de contagem nos registradores. O valor do divisor poderá variar de 1 a 65535: o Registrador 11 contém os oito bits menos significativos e o Registrador 12 os mais significativos. A frequência base do Gerador de Envoltória é 6991 Hz, de modo que a frequência da envoltória poderá variar de 6991 Hz (dividido por 1) até 0.11Hz (dividido por 65535).

**Registrador 13****Figura 30**

Os quatro bits de Forma da Envoltória determinam a forma da modulação da amplitude, produzida pelo Gerador de Envoltória:

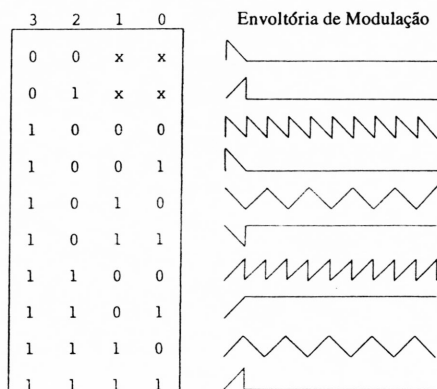


Figura 31

## Registrador 14

|      |      |        |        |         |     |      |        |
|------|------|--------|--------|---------|-----|------|--------|
| 7    | 6    | 5      | 4      | 3       | 2   | 1    | 0      |
| Entr | Modo | Gat. B | Gat. A | Direita | Esq | Retr | Frente |
| Cas  | Tec  | Joy    | Joy    | Joy     | Joy | Joy  | Joy    |

Figura 32

Este registrador é utilizado para ler a Porta A do PSG. Os seis bits do joystick refletem o estado das quatro chaves de direção e dos dois botões de acionamento em um joystick: 0=Ligado, 1=Não ligado. Ao invés disso, em um lugar de um joystick poderão ser ligados até seis Paddles. Apesar da maioria das máquinas MSX possuírem dois conectores de 9 pinos para joystick, apenas um, de cada vez poderá ser lido. O selecionado para ser lido é determinado pelo bit de Seleção de Joystick no Registrador 15 do PSG.

O bit Modo Teclado não é utilizado nas máquinas inglesas. Nas máquinas japonesas é ligado a um "jumper", que determina o conjunto de caracteres do teclado.

A Entrada Cassete é utilizada para ler o sinal da saída EAR do Cassete. O sinal de áudio passa por um computador que "limpa as bordas" e converte a níveis digitais, mas não é processado de outra forma.

**Registrador 15**

| 7           | 6          | 5          | 4          | 3 | 2 | 1 | 0 |
|-------------|------------|------------|------------|---|---|---|---|
| LED<br>Kana | Sel<br>Joy | Pulso<br>2 | Pulso<br>1 | 1 | 1 | 1 | 1 |

**Figura 33**

Este registrador é utilizado para fornecer uma saída à porta B. Os quatro bits menos significativos são ligados via buffers TTL de coletor aberto aos pinos 6 e 7 de cada conector de joystick. São normalmente colocados em 1, quando um paddle ou joystick é ligado, de modo que os pinos poderão funcionar como entradas. Quando um digitalizador é ligado, eles são utilizados como saídas para intercâmbio de informação (“hand-shaking”).

Os dois bits de Pulso são utilizados para produzir um curto pulso positivo, para quaisquer paddles ligados aos conectores 1 ou 2 do joystick. Cada paddle contém um temporizador monoestável com um resistor variável que contenha a duração de seu pulso. Quando o temporizador é acionado, a posição do resistor variável poderá ser determinada, contando até a queda do monoestável.

O bit de Seleção do Joystick determina qual conector de joystick é ligado à Porta A do PSG: 0=Conector 1, 1=Conector 2.

A saída LED Kana não é utilizada em máquinas inglesas, e nas máquinas japonesas é utilizada para acionar um indicador de modo teclado.

## BIOS EM ROM

O conteúdo da ROM do MSX tem importância se programas em códigos de máquina forem desenvolvidos eficientemente e operarem com confiabilidade. Quase todo programa, incluindo o próprio Interpretador BASIC, irá requerer um certo conjunto de funções primitivas para operar. Estas incluem funções para acionadores de tela e impressoras, para decodificador de teclado e, ainda, outras funções relacionadas ao hardware. Estas rotinas, estando separadas do Interpretador BASIC, poderão se tornar disponíveis a qualquer programa aplicativo. A seção da ROM de 0000H a 268BH, em sua maior parte é devotada a este tipo de rotina e é denominada de BIOS em ROM (Sistema Básico de Entrada e Saída).

Este capítulo fornece uma descrição funcional de cada rotina separada, na BIOS em ROM. Uma atenção especial será dada às rotinas “padrões”. Estas são documentadas pela Microsoft com a garantia de permanecerem consistentes em possíveis alterações de hardware e software. As primeiras centenas de bytes da ROM consistem em instruções “Jump” (JP) do Z80, que fornecem pontos de entrada em posições fixas a estas rotinas. Para compatibilidade máxima com um software futuro, um programa aplicativo deveria restringir a sua dependência na ROM, apenas a estas posições. A descrição da ROM se inicia com esta lista de pontos de entrada às rotinas padrões. Um breve comentário é colocado junto a cada ponto de entrada; a descrição completa é dada com a própria rotina.

## Áreas de Dados

É de se esperar que a maioria dos usuários deseje “desassemblar” certas porções da ROM (a listagem completa cobre aproximadamente quatrocentas páginas). Para facilitar este processo, as áreas de dados, que não contêm instruções Z80 são mostradas abaixo:

|             |             |             |              |
|-------------|-------------|-------------|--------------|
| 0004H-0007H | 185DH-1863H | 4B3AH-4B4CH | 73E4H-73E4H  |
| 002BH-002FH | 1B97H-1BAAH | 4C2FH-4C3FH | 752EH-7585H  |
| 0508H-050DH | 1BBFH-23BEH | 555AH-5569H | 7754H-7757H  |
| 092FH-097FH | 2439H-2459H | 5D83H-5DB0H | 7BA3H-7BCAH  |
| 0DA5H-0EC4H | 2CF1H-2E70H | 6F76H-6F8EH | 7ED8H-7F26H  |
| 1033H-105AH | 3030H-3039H | 70FFH-710CH | 7F41H-7FB8H  |
| 1061H-10C1H | 3710H-3719H | 7182H-7195H | 7FBEH-7FFFFH |
| 1233H-1252H | 392EH-3FE1H | 71A2H-71B5H |              |
| 13A9H-1448H | 43B5H-43C3H | 71C7H-71DAH |              |
| 160BH-1612H | 46E6H-46E7H | 72A6H-72B9H |              |

Observe que estas áreas de dados são para a ROM inglesa, havendo na ROM japonesa pequenas diferenças relacionadas ao decodificador de teclado e ao conjunto de caracteres de vídeo. As disparidades entre as ROMs estão restritas a estas regiões, porém o grosso do código é idêntico nos dois casos.

## Terminologia

Neste capítulo é feita, freqüentemente, uma referência às rotinas padrões e às variáveis da Área de Trabalho. Sempre que isto acontece, o nome recomendado pela Microsoft é utilizado em letras maiúsculas, por exemplo: “a rotina padrão FILVRM” e “SCRMOD é ativada”. Sub-rotinas que não têm denominação são referenciadas por um endereço entre parênteses, “a tela está liberada (0777H)”, por exemplo. Quando é feita uma referência aos indicadores de status do Z80, são utilizadas convenções em linguagem de montagem, por exemplo: “Flag C” significaria que o indicador de vai-um é ativado, ao passo que “Flag NZ” significaria que o indicador de zero está desligado. Os termos “EI” e “DI” significam interrupções ativadas e interrupções desativadas respectivamente.



---

**ENDEREÇO      NOME      DA      FUNÇÃO**


---

|       |        |       |                                                 |
|-------|--------|-------|-------------------------------------------------|
| 0000H | CHKRAM | 02D7H | Partida, verifica RAM                           |
| 0004H | .....  |       | Dois bytes, endereço do conj. caracteres na ROM |
| 0006H | .....  |       | Um byte, número da Porta de Dado VDP            |
| 0007H | .....  |       | Um byte, número da Porta de Dado VDP            |
| 0008H | SYNCHR | 2683H | Verifica caractere do programa BASIC            |
| 000BH | .....  |       | Nada (NOP)                                      |
| 000CH | RDSLT  | 01B6H | Lê RAM em qualquer conector                     |
| 000FH | .....  |       | Nada (NOP)                                      |
| 0010H | CHRGTR | 2686H | Pegue o próximo caractere do programa BASIC     |
| 0013H | .....  |       | Nada (NOP)                                      |
| 0014H | WRSLT  | 01D1H | Escreva na RAM em qualquer conector             |
| 0017H | .....  |       | Nada (NOP)                                      |
| 0018H | OUTDO  | 1B45H | Saída ao dispositivo atual                      |
| 001BH | .....  |       | Nada (NOP)                                      |
| 001CH | CALSLT | 0217H | Chame rotina em qualquer conector               |
| 001FH | .....  |       | Nada (NOP)                                      |
| 0020H | DCOMPR | 146AH | Compare os pares de registradores HL e DE       |
| 0023H | .....  |       | Nada (NOP)                                      |
| 0024H | ENALST | 025EH | Ativa qualquer conector permanentemente         |
| 0027H | .....  |       | Nada (NOP)                                      |
| 0028H | GETYPR | 2689H | Obter tipo de operando BASIC                    |
| 002BH | .....  |       | Número da Versão em cinco bytes                 |
| 0030H | CALLF  | 0205H | Chama rotina em qualquer conector               |
| 0033H | .....  |       | Cinco nadas (NOPS)                              |
| 0038H | KEYINT | 0C3H  | Manuseio de interrupção, varredura de teclado   |
| 003BH | INITIO | 049DH | Inicializa dispositivos de E/S                  |
| 003EH | INIFNK | 139DH | Inicializa strings de tecla de função           |
| 0041H | DISSCR | 0577H | Desativa tela                                   |
| 0044H | ENASCR | 0570H | Ativa tela                                      |
| 0047H | WRTVDP | 057FH | Escreve em qualquer registrador de VDP          |
| 004AH | RDVRM  | 07D7H | Lê byte da VRAM                                 |
| 004DH | WRTVRM | 07CDH | Escreve byte em VRAM                            |
| 0050H | SETRD  | 07ECH | Prepara VDP para leitura                        |
| 0053H | SETWRT | 07DFH | Prepara VDP para escrever                       |
| 0056H | FILVRM | 0815H | Preenche bloco de VRAM com byte de dado         |
| 0059H | LDIRMV | 070FH | Copia bloco da VRAM para memória                |
| 005CH | LDIRVM | 0744H | Copia bloco da memória para VRAM                |
| 005FH | CHGMOD | 084FH | Altera o modo VDP                               |
| 0062H | CHGCLR | 07F7H | Altera cores do VDP                             |
| 0065H | .....  |       | Nada (NOP)                                      |
| 0066H | NMI    | 1398H | Manipulador de Interrupção Não-Mascarável       |
| 0069H | CLRSPR | 06A8H | Limpar todos os sprites                         |
| 006CH | INITXT | 050EH | Inicializa VDP ao modo texto de 40x24           |

| ENDEREÇO | NOME    | DA    | FUNÇÃO                                             |
|----------|---------|-------|----------------------------------------------------|
| 006FH    | INIT32  | 0538H | Inicializa VDP ao modo texto de 32x24              |
| 0072H    | INIGRP  | 05D2H | Inicializa VDP ao modo gráfico                     |
| 0075H    | INIMLT  | 061FH | Inicializa VDP ao modo multicolorido               |
| 0078H    | SETTXT  | 0594H | Coloca VDP no Modo Texto de 40x24                  |
| 007BH    | SETT32  | 05B4H | Coloca VDP no Modo Texto de 32x24                  |
| 007EH    | SETGRP  | 0602H | Coloca VDP no Modo Gráfico                         |
| 0081H    | SETMLT  | 0659H | Coloca VDP no Modo Multicolorido                   |
| 0084H    | CALPAT  | 06E4H | Calcula endereço da imagem do sprite               |
| 0087H    | CALATR  | 06F9H | Calcula endereço do atributo do sprite             |
| 008AH    | GSPSIZ  | 0704H | Pega tamanho do sprite                             |
| 008DH    | GRPPRT  | 1510H | Imprime caractere na tela gráfica                  |
| 0090H    | GICINI  | 04BDH | Inicializa PSG (Pastilha GI)                       |
| 0093H    | WRTPSG  | 1102H | Escreve em qualquer registrador PSG                |
| 0096H    | RDPSG   | 110EH | Lê de qualquer registrador PSG                     |
| 0099H    | STRTMS  | 11C4H | Desempilha de fila musical                         |
| 009CH    | CHSNS   | 0D6AH | Verifica buffer do teclado por caractere           |
| 009FH    | CHGET   | 10CBH | Pegue caractere do buffer do teclado (espera)      |
| 00A2H    | CHPUT   | 08BCH | Saída de caractere na tela                         |
| 00A5H    | LPTOUT  | 085DH | Saída de caractere na impressora de linha          |
| 00A8H    | LPTSTT  | 0884H | Teste de status da impressora de linha             |
| 00ABH    | CNVCHR  | 089DH | Converte caractere com cabeçalho gráfico           |
| 00AEH    | PINLIN  | 23BFH | Pegue linha da console (editor)                    |
| 00B1H    | INLIN   | 23D5H | Pegue linha da console (editor)                    |
| 00B4H    | QINLIN  | 23CCH | Apresenta "?", pega linha da console (editor)      |
| 00B7H    | BREAKX  | 046FH | Verifique diretamente a tecla CTRL-STOP            |
| 00BAH    | ISCNTC  | 03FBH | Verifique a tecla CTRL-STOP                        |
| 00BDH    | CKCNTC  | 10F9H | Verifique a tecla CTRL-STOP                        |
| 00C0H    | BEEP    | 1113H | Faça um beep                                       |
| 00C3H    | CLS     | 0848H | Limpa tela                                         |
| 00C6H    | POSIT   | 088EH | Estabelecer posição do cursor                      |
| 00C9H    | FNKSB   | 0B26H | Verifica se display da tecla de função está ligado |
| 00CCH    | ERAFNK  | 0B15H | Apagar display da tecla de função                  |
| 00CFH    | DSPFNK  | 0B2BH | Apresenta teclas de função                         |
| 00D2H    | TOTEXT  | 083BH | Retorna VDP ao modo texto                          |
| 00D5H    | GTSECTK | 11EEH | Obter status do joystick                           |
| 00D8H    | GTTTRIG | 1253H | Obter status do gatilho                            |
| 00DBH    | GTPAD   | 12ACH | Obter status do digitalizador                      |
| 00DEH    | GTPDL   | 1273H | Obter status do paddle                             |
| 00E1H    | TAPION  | 1A63H | Entrada de fita ON                                 |
| 00E4H    | TAPIN   | 1ABCH | Entrada de fita                                    |
| 00E7H    | TAPIOF  | 19E9H | Entrada de fita OFF                                |
| 00EAH    | TAPOON  | 19F1H | Saída de fita On                                   |
| 00EDH    | TAPOUT  | 1A19H | Saída de fita                                      |
| 00F0H    | TAPOOF  | 19DDH | Saída de fita OFF                                  |

| ENDEREÇO | NOME   | DA    | FUNÇÃO                                            |
|----------|--------|-------|---------------------------------------------------|
| 00F3H    | STMOTR | 1384H | Coloca o motor em ON/OFF                          |
| 00F6H    | LFTQ   | 14EBH | Espaço na fila musical                            |
| 00F9H    | PUTQ   | 1492H | Coloca byte na fila musical                       |
| 00FCH    | RIGHTC | 16C5H | Mova à direita o endereço físico do pixel atual   |
| 00FFH    | LEFTC  | 16EEH | Mova à esquerda o endereço físico do pixel atual  |
| 0102H    | UPC    | 175DH | Mova para cima o endereço físico do pixel atual   |
| 0105H    | TUPC   | 173CH | Teste então UPC, se é válido                      |
| 0108H    | DOWNC  | 172AH | Mova para baixo o endereço físico do pixel atual  |
| 010BH    | TDOWNC | 170AH | Teste então DOWNC, se é válido                    |
| 010EH    | SCALXY | 1599H | Coloca as coordenadas gráficas em escala          |
| 0111H    | MAPXYC | 15DFH | Mapeia as coordenadas gráficas ao endereço físico |
| 0114H    | FETCHC | 1639H | Pega o endereço físico do pixel atual             |
| 0117H    | STOREC | 1640H | Armazena o endereço físico do pixel atual         |
| 011AH    | SETATR | 1676H | Coloca byte de atributo                           |
| 011DH    | READC  | 1647H | Lê atributo do pixel atual                        |
| 0120H    | SETC   | 167EH | Coloca atributo do pixel atual                    |
| 0123H    | NSETCX | 1809H | Coloca atributo de um número de pixels            |
| 0126H    | GTASPC | 18C7H | Pegue razão de aspecto                            |
| 0129H    | PNTINI | 18CFH | Inicialização de pintura (PAINT)                  |
| 012CH    | SCANR  | 18E4H | Varre pixels da direita                           |
| 012FH    | SCANL  | 197AH | Varre pixels da esquerda                          |
| 0132H    | CHGCAP | 0F3DH | Altera o LED de Caps Lock                         |
| 0135H    | CHGSND | 077AH | Altera saída de som do "Click" da Tecla           |
| 0138H    | RSLREG | 144CH | Lê Registrador do Conector Primário               |
| 013BH    | WSLREG | 144FH | Escreve no Registrador do Conector Primário       |
| 013EH    | RDVDP  | 1449H | Lê Registrador de Status do VDP                   |
| 0141H    | SNSMAT | 1452H | Lê linha da matriz de teclado                     |
| 0144H    | PHYDIO | 148AH | Disco, nenhuma ação                               |
| 0147H    | FORMAT | 148EH | Disco, nenhuma ação                               |
| 014AH    | ISFLIO | 145FH | Verifica E/S de arquivo                           |
| 014DH    | OUTDLP | 1B63H | Saída formatada para impressora de linha          |
| 0150H    | GETVCP | 1470H | Pega apontador de voz musical                     |
| 0153H    | GETVC2 | 1474H | Pega apontador de voz musical                     |
| 0156H    | KILBUF | 0468H | Limpa buffer de teclado                           |
| 0159H    | CALBAS | 01FFH | Chama BASIC de qualquer conector                  |
| 015CH    | .....  |       | Nada (NOP) para 01B5H para expansão               |

Endereço ..... 01B6H  
 Nome ..... RDSLT  
 Entrada ..... A=ID do Conector, HL=Endereço  
 Saída ..... A=Leitura de byte  
 Modifica ..... AF, BC, DE, DI

Rotina padrão para ler um único byte da memória em qualquer conector. O Identificador de Conector é formado por um número de Conector Primário, um número de Conector Secundário e um indicador:

|        |   |   |   |       |   |       |   |
|--------|---|---|---|-------|---|-------|---|
| 7      | 6 | 5 | 4 | 3     | 2 | 1     | 0 |
| Indic. | 0 | 0 | 0 | SSLOT |   | PSLOT |   |

**Figura 34** ID de Conector.

O indicador é, normalmente, 0, mas deverá ser 1 se um número de Conector Secundário for incluído na ID do Conector. O endereço de memória e a ID do conector são processados primeiro (027EH) para fornecer um conjunto de máscaras de bit, que serão aplicadas ao registrador do conector relevante. Se um número de Conector Secundário for especificado, então, o Registrador de Conector Secundário será modificado primeiro para selecionar a página relevante do Conector Secundário (02A3H). Em seguida, o Conector Primário será colocado no espaço de endereço do Z80, o byte será lido e o Conector Primário será restaurado à sua condição original pela rotina RDPRIM, na Área de Trabalho. Finalmente, se um número do Conector Secundário for incluído na ID do Conector, a condição original do Registrador de Conector Secundário será restaurado (01ECH).

Observe que, a não ser que seja o conector contendo a Área de Trabalho, qualquer tentativa em acessar a página 3 (C000H até FFFFH) fará o sistema “quebrar”, uma vez que RDPRIM se desliga. Observe, também, que as interrupções são mantidas desativadas por todas as rotinas de comutação de memória.

Endereço ..... 01D1H  
 Nome ..... WRSLT  
 Entrada ..... A=ID de Conector, HL=Endereço, E=Byte a ser escrito  
 Saída ..... Nenhuma  
 Modifica ..... AF, BC, D, DI

Rotina padrão para escrever um único byte na memória em qualquer conector. Sua operação é fundamentalmente a mesma que aquela da rotina padrão RDSLT, exceto que a rotina WRPRIM, da Área de Trabalho, é utilizada em lugar de RDPRIM.

|                    |                            |
|--------------------|----------------------------|
| Endereço . . . . . | 01FFH                      |
| Nome . . . . .     | CALBAS                     |
| Entrada . . . . .  | IX=Endereço                |
| Saída . . . . .    | Nenhuma                    |
| Modifica . . . . . | AF', BC', DE', HL', IY, DI |

Rotina padrão para chamar um endereço no Intérprete BASIC de qualquer conector. Normalmente, esta chamada será feita dentro de um programa em código de máquina, sendo executada em uma ROM de extensão na página 1 (4000H até 7FFFH). O byte mais significativo do par de registradores IY é carregado com a ID do Conector da ROM do MSX (00H) e o controle é transferido à rotina padrão CALSLT.

|                    |                                |
|--------------------|--------------------------------|
| Endereço . . . . . | 0205H                          |
| Nome . . . . .     | CALLF                          |
| Entrada . . . . .  | Nenhuma                        |
| Saída . . . . .    | Nenhuma                        |
| Modifica . . . . . | AF', BC', DE', HL', IX, IY, DI |

Rotina padrão para chamar um endereço em qualquer conector. A ID de Conector e o endereço são fornecidos como parâmetros em-linha, em vez de nos registradores, afim de caber dentro de um gancho (Capítulo 6), por exemplo:

```
RST 30H
DEFB ID do Conector
DEFW Endereço
RET
```

A ID do Conector é coletada primeiro e colocada no byte superior (mais significativo) do par de registradores IY. O endereço é, então, colocado no par de registradores IX e o controle vai para a rotina padrão CALSLT.

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| Endereço . . . . . | 0217H                                                   |
| Nome . . . . .     | CALSLT                                                  |
| Entrada . . . . .  | IY(Byte mais significativo)=ID do Conector, IX=Endereço |
| Saída . . . . .    | Nenhuma                                                 |
| Modifica . . . . . | AF', BC', DE', HL', DI                                  |

Rotina padrão para chamar um endereço em qualquer conector. Sua operação é fundamentalmente a mesma da rotina padrão RDSLT, exceto que a rotina CLPRIM da

Área de Trabalho seja utilizada, em vez de RDPRIM. Observe que CALBAS e CALLF são apenas pontos de entrada especializados à esta rotina padrão, que oferecem uma redução na quantidade de código requerido.

Endereço . . . . . 025EH  
Nome . . . . . ENASLT  
Entrada . . . . . A=Conector ID, HL=Endereço  
Saída . . . . . Nenhuma  
Modifica . . . . . AF, BC, DE, DI

Rotina padrão para dar entrada permanente a uma página, de qualquer conector. Ao contrário das rotinas padrões RDSLT, WRSLT e CALSLT, a comutação do Conector Primário é executada diretamente e não por meio de uma rotina de Área de Trabalho. Conseqüentemente, endereços na página 0(0000H até 3FFFH) provocarão uma quebra imediata do sistema.

Endereço . . . . . 027EH

Esta rotina é utilizada pelas rotinas padrões de comutação da memória, afim de transformar um endereço no par de registradores HL, e um ID de conector no registrador A, em um conjunto de máscaras de bit. Como exemplo, uma ID de Conector de FxxxSSPP e um endereço na Página 1(4000H a 7FFFH) devolveria o seguinte:

Registrador B=00 00 PP 00 (Máscara OR)  
Registrador C=11 11 00 11 (Máscara AND)  
Registrador D=PP PP PP PP (Reproduzida)  
Registrador E=00 00 11 00 (Máscara de página)

Os registradores B e C são derivados do número de Conector Primário e da máscara de página. São utilizados, posteriormente, para misturar o número novo do Conector Primário no conteúdo existente do Registrador de Conector Primário. O Registrador D contém o número do Conector Primário, reproduzido quatro vezes e o registrador B contém a máscara de página. Isto é produzido examinando os dois bits mais significativos do endereço para determinar o número da página, e depois deslocando a máscara para a posição relevante. Estes registradores são, posteriormente, utilizados durante a comutação de Conector Secundário.

No final da rotina, o bit 7 da ID de Conector é testado, para determinar se um Conector Secundário foi especificado, e neste caso o Flag M é devolvido.

Endereço . . . . . 02A3H

Esta rotina é utilizada pelas rotinas padrões de comutação de memória para modificar um Registrador de Conector Secundário. O ID do Conector é fornecido no registrador A, enquanto os registradores D e E contêm as máscaras de bit mostradas na rotina anterior.

Os bits 6 e 7 do registrador D são primeiro copiados no registrador do Conector Primário. Isto leva para dentro a página 3 do Conector Primário especificado pela ID do Conector e torna disponível o Registrador do Conector Primário requerido. Este é, em seguida, lido da posição de memória FFFFH e a máscara de página invertida é utilizada para liberar os dois bits requeridos. O número do Conector Secundário é transferido à posição relevante e misturado. Finalmente, a nova situação é colocada no Registrador de Conector Secundário e o Registrador de Conector Primário é restaurado à sua condição original.

Endereço . . . . . 02D7H  
 Nome . . . . . CHKRAM  
 Entrada . . . . . Nenhuma  
 Saída . . . . . Nenhuma  
 Modifica . . . . . AF, BC, DE, HL, SP

Uma rotina padrão para executar a inicialização da memória na partida. Ela testa de forma não-destrutiva a RAM nas páginas 2 e 3 em todos os dezesseis conectores possíveis, depois prepara os registradores do Conector Primário e Secundário para comutarem a maior área encontrada. Toda a Área de Trabalho (F380H até FFC9H) é zerada, e EXPTBL e SLTTBL são preenchidas para mapear qualquer interface de expansão existente. O Modo de Interrupção 1 é preparado e o controle é transferido ao restante da rotina de inicialização de partida (7C76H).

Endereço . . . . . 03FBH  
 Nome . . . . . ISCNTC  
 Entrada . . . . . Nenhuma  
 Saída . . . . . Nenhuma  
 Modifica . . . . . AF, EI

Uma rotina padrão para verificar se a tecla CTRL-STOP ou STOP foi pressionada. É utilizada pelo Interpretador BASIC, ao final de cada instrução, para verificar uma possível interrupção do programa. Primeiro BASROM é examinado para verificar se contém um valor não-zero; em caso afirmativo, a rotina termina imediatamente. Isto é feito

para impedir que os usuários interfiram em qualquer ROM de extensão, que contenha um

BASIC

Em seguida, INTFLG é verificado para determinar se o manipulador de interrupção colocou os códigos de CTRL-STOP, ou STOP, (03H ou 04H) naquela posição. Caso STOP tenha sido detectado o cursor é ligado (09DAH) e INTFLG é continuamente verificado, até que o código de uma das duas teclas reapareça. O cursor é desligado, em seguida (0A27H) e, se a tecla for STOP, a rotina chegará ao fim.

Caso CTRL-STOP tenha sido detectado, então, o buffer do teclado é primeiro liberado por meio da rotina padrão KILBUF, e TRPTBL é verificado para saber se uma declaração "ON STOP GOSUB" está ativa. Em caso afirmativo, a entrada relevante de TRPTBL é atualizada (0EF1H) e a rotina termina uma vez que o evento será tratado pela Ronda de Execução ("Runloop") do Interpretador. Caso contrário, a rotina padrão ENASLT é utilizada para colocar a página 1 da ROM do MSX (no caso de uma ROM de extensão estar utilizando a rotina) e o controle é transferido ao manipulador da instrução "STOP" (63E6H).

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 0468H   |
| Nome . . . . .     | KILBUF  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | HL      |

Uma rotina padrão para limpar KEYBUF, o buffer de quarenta caracteres do teclado. Há dois apontadores para este buffer, PUTPNT aonde o manipulador de interrupção coloca caractere, e GETPNT de onde os programas aplicativos os pegam. Sendo o número de caracteres no buffer indicado pela diferença entre estes dois apontadores, KEYBUF é, simplesmente esvaziado, tornando os dois iguais.

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| Endereço . . . . . | 046FH                                                 |
| Nome . . . . .     | BREAKX                                                |
| Entrada . . . . .  | Nenhuma                                               |
| Saída . . . . .    | Indicador C caso a tecla CTRL-STOP esteja pressionada |
| Modifica . . . . . | AF                                                    |

Uma rotina padrão que testa diretamente as linhas 6 e 7 do teclado, para determinar se as teclas CTRL e STOP estão ambas pressionadas. Caso estejam, KEYBUF é liberado e a linha 7 de OLDKEY é modificada para impedir que o manipulador de interrupção também pegue as teclas. Esta rotina poderá, freqüentemente, ser mais adequada



para ser utilizada, por um programa aplicativo, preferencialmente a ISCNTC, pois funciona mesmo com as interrupções desativadas, durante uma E/S de cassete por exemplo, e não causa uma saída para o Intérprete.

|                    |           |
|--------------------|-----------|
| Endereço . . . . . | 049DH     |
| Nome . . . . .     | INITIO    |
| Entrada . . . . .  | Nenhuma   |
| Saída . . . . .    | Nenhuma   |
| Modifica . . . . . | AF, E, EI |

Uma rotina padrão para inicializar o PSG e a Porta de Status Centronics. O Registrador 7 do PSG é, primeiro, colocado em 80H, tornando a Porta B=Saída e Porta A=Entrada no PSG. Registrador 15 do PSG é colocado em CFH para inicializar o hardware de controle do conector do Joystick. Em seguida, o Registrador 14 do PSG é lido e o bit de Modo Teclado colocado em KANAMD, o que não têm relevância para as máquinas inglesas.

Finalmente, um valor de FFH é enviado à Porta de Status Centronics (Porta de E/S 90H) para colocar o sinal STROBE em “alto”. Em seguida, o controle passa para a rotina padrão GICINI para completar a inicialização.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 04BDH   |
| Nome . . . . .     | GICINI  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | EI      |

Uma rotina padrão para inicializar o PSG e as variáveis da Área de Trabalho, associadas com o comando “PLAY”.QUETAB, VCBA, VCBB e VCBC são inicializadas com os valores mostrados no Capítulo 6. Os registradores 8, 9 e 10 do PSG são colocados em amplitude zero e o Registrador 7 é inicializado com B8H. Isto ativa o Gerador de Sons e desativa o Gerador de Ruído em cada canal.

|                    |       |
|--------------------|-------|
| Endereço . . . . . | 0508H |
|--------------------|-------|

Esta tabela de seis bytes contém os parâmetros da declaração “PLAY”, inicialmente colocadas em VCBA, VCBB e VCBC pela rotina padrão GICINI: Oitava=4, Tempo=120, Duração=4, Volume=88H, Envoltória=00FFH.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 050EH              |
| Nome . . . . .     | INITXT             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Uma rotina padrão para inicializar o VDP no Modo Texto 40x24. A tela é temporariamente desativada pela rotina padrão DISSCR, e o valor 00H é colocado em SCRMOD e OLDSCR. Os parâmetros requeridos pela rotina padrão CHPUT são estabelecidos copiando LINL40 em LINLEN, TXTNAM em NAMBAS e TXTCGP em CGPBAS. Em seguida, as cores do VDP são estabelecidas pela rotina padrão CHGCLR e a tela é limpa (077EH). O conjunto de caracteres atual é copiado na Tabela de Imagens de Caracteres da VRAM(071EH). Finalmente, o modo VDP e endereços de base são estabelecidos pela rotina padrão SETTXT e a tela é ativada.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 0538H              |
| Nome . . . . .     | INIT32             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Uma rotina padrão para inicializar o VDP no Modo de Texto 32x24. A tela é temporariamente desativada por meio da rotina padrão DISSCR, e o valor 01H é colocado em SCRMOD e OLDSCR. Os parâmetros requeridos pela rotina padrão CHPUT são estabelecidos copiando LINL32 em LIMLEN, T32NAM em NAMBAS, T32CGP em CGPBAS, T32PAT em PATBAS e T32ATR em ATRBAS. As cores do VDP são, depois, estabelecidas pela rotina padrão CHGCLR e a tela é limpa (077EH). O conjunto atual de caracteres é copiado na Tabela de Caracteres da VRAM (071EH) e todos os sprites limpos (06BBH). Finalmente, o modo VDP e endereços de base são estabelecidos via a rotina padrão SETT32 e a tela é ativada.

|                    |            |
|--------------------|------------|
| Endereço . . . . . | 0570H      |
| Nome . . . . .     | ENASCR     |
| Entrada . . . . .  | Nenhuma    |
| Saída . . . . .    | Nenhuma    |
| Modifica . . . . . | AF, BC, EI |

Rotina padrão para ativar a tela. Isto, simplesmente, significa ligar o bit 6 do Registrador Modo 1 do VDP.

|                    |            |
|--------------------|------------|
| Endereço . . . . . | 0577H      |
| Nome . . . . .     | DISSCR     |
| Entrada . . . . .  | Nenhuma    |
| Saída . . . . .    | Nenhuma    |
| Modifica . . . . . | AF, BC, EI |

Rotina padrão para desativar a tela. Isto, simplesmente, significa desligar o bit 6 do Registrador Modo 1 do VDP.

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| Endereço . . . . . | 057FH                                                  |
| Nome . . . . .     | WRTVDP                                                 |
| Entrada . . . . .  | B=Byte de dado, C=Número do Registrador de Modo do VDP |
| Saída . . . . .    | Nenhuma                                                |
| Modifica . . . . . | AF, B, EI                                              |

Rotina padrão para escrever um byte de dados em qualquer Registrador de Modo do VDP. O byte de seleção do registrador é escrito primeiro na Porta de Comando do VDP, seguido pelo byte de dado. Isto é, em seguida, copiado na imagem do registrador relevante, RGOSAV a RG7SAV, na Área de Trabalho.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 0594H              |
| Nome . . . . .     | SETTXT             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para estabelecer, parcialmente, o VDP no Modo Texto 40x24. Os bits de modo M1, M2 e M3 são estabelecidos primeiros nos Registradores de Modo 0 e 1 do VDP. Os cinco endereços de base da tabela da VRAM, iniciando com TXTNAM, são depois copiados da Área de Trabalho nos Registradores de Modo 2, 3, 4, 5 e 6(0677H) do VDP.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 05B4H              |
| Nome . . . . .     | SETT32             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para estabelecer, parcialmente, o VDP ao Modo Texto 32x24. Os bits de modo M1, M2 e M3 são estabelecidos primeiro nos Registradores de Modo 0 e 1 do VDP. Os cinco endereços de base da tabela da VRAM, iniciando com T32NAM, são, em seguida, copiados da Área de Trabalho nos Registradores de Modo 2, 3, 4, 5 e 6(0677H).

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 05D2H              |
| Nome . . . . .     | INIGRP             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para inicializar o VDP no Modo Gráfico. A tela é, temporariamente, desativada por meio da rotina padrão DISSCR, e SCRMOD colocado em 02H. Os parâmetros requeridos pela rotina padrão GRPPRF são estabelecidos copiando GRPPAT em PATBAS e GRPATR em ATRBAS. O mapeamento de códigos dos caracteres é, em seguida, copiado na Tabela de Nome VDP, a tela é limpa (07A1H) e todos os sprites são limpos (06BBH). Finalmente, o modo VDP e endereços básicos são estabelecidos por meio da rotina padrão SETGRP e a tela é ativada.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 0602H              |
| Nome . . . . .     | SETGRP             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para colocar o VDP, parcialmente, no Modo Gráfico. Os bits de modo M1, M2 e M3 são estabelecidos primeiro nos Registradores de Modo 0 e 1 do VDP. Os cinco endereços de base da tabela da VRAM, iniciando com GRPNAM, são, em seguida, copiados da Área de Trabalho nos Registradores de Modo 2, 3, 4, 5 e 6 do VDP(0677H).

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 061FH              |
| Nome . . . . .     | INIMLT             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para inicializar o VDP no Modo Multicolorido. A tela é, temporariamente, desativada pela rotina padrão DISSCR, e SCRMOD é colocado em 03H. Os parâmetros requeridos pela rotina padrão GRPPRT são estabelecidos copiando MLTPAT em PATBAS e MLTATR em ATRBAS. O mapeamento de códigos de caracteres é, então copiado na Tabela de Nomes do VDP, a tela é limpa (07B9H) e todos os sprites são limpos (06BBH). Finalmente, o modo VDP e os endereços de base são estabelecidos por meio da rotina padrão SETMLT e a tela é ativada.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 0659H              |
| Nome . . . . .     | SETMLT             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para colocar, parcialmente, o VDP no Modo Multicolorido. Os bits de modo M1, M2 e M3 são, primeiramente, estabelecidos nos Registradores de Modo 0 e 1 do VDP. Os cinco endereços de base da tabela da VRAM, iniciando com MLT-NAM, são, em seguida, copiados da Área de Trabalho nos Registradores de Modo 2, 3, 4, 5 e 6 do VDP.

|                    |       |
|--------------------|-------|
| Endereço . . . . . | 0677H |
|--------------------|-------|

Rotina utilizada pelas rotinas padrões SETTXT, SETT32, SETGRP e SETMLT para copiar um bloco de cinco endereços de base de tabela da Área de Trabalho nos Registradores de Modo 2, 3, 4, 5 e 6 do VDP. Na entrada, o par de registradores HL aponta para o grupo de endereços relevantes. Os endereços de base são lidos um por vez. De cada um é deslocado ("Shifted") o número necessário de posições e depois escrito no Registrador de Modo relevante, por meio da rotina padrão WRTVDP.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 06A8H              |
| Nome . . . . .     | CLRSPR             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para limpar todos os sprites. Toda a Tabela de Imagens de Sprites de 2KB é, inicialmente, preenchida com zeros por meio da rotina padrão FILVRM. A coordenada vertical de cada um dos trinta e dois blocos de atributo de sprite é colocada em - 47 (D1H) posicionando o sprite acima, na parte superior do topo da tela; a coordenada horizontal fica inalterada.

Os números de imagens na Tabela de Atributos de Sprite são inicializados com a série 0, 1, 2, 3, 4,...31 para sprite são inicializados com a série 0, 1, 2, 3, 4,...,31 para sprites de 8x8 ou com a série 0, 4, 8, 12, 16,...124 para sprites de 16x16. A série a ser gerada é determinada pelo bit de Tamanho, no Registrador de Modo 1. Finalmente, o byte de cor de cada bloco de atributo do sprite é preenchido com o código de cor contido em FORCLR, que inicialmente é branco.

Observe que os bits Tamanho e Magnitude no Registrador Modo 1 não são afetados por esta rotina. Observe também, que as rotinas padrões INIT32, INIGRP e INIMLT utilizam esta rotina com um ponto de entrada em 06BBH, deixando a Tabela de Imagens de Sprites inalterada.

|                    |                              |
|--------------------|------------------------------|
| Endereço . . . . . | 06E4H                        |
| Nome . . . . .     | CALPAT                       |
| Entrada . . . . .  | A=Número do padrão sprite    |
| Saída . . . . .    | HL=Endereço do padrão sprite |
| Modifica . . . . . | AF, DE, HL                   |

Rotina padrão para calcular o endereço de uma imagem de sprite. O número padrão é multiplicado por oito e depois, estão selecionados sprites 16x16, multiplicado por quatro. O resultado, em seguida, é somado ao endereço de base da Tabela de Imagens de Sprites, tomado de PATBAS, para obter o endereço final.

Este sistema de numeração está de acordo com a maneira pela qual o Interpretador BASIC numera as imagens em lugar da maneira do VDP, quando sprites 16x16 são selecionados. Como exemplo, enquanto o Interpretador chama a segunda imagem de "número um", na verdade ela é a número quatro para o VDP. Esta utilização significa que o número 16x16 máximo de imagem que esta rotina deveria permitir, quando sprites são

selecionados, seria sessenta e três. Não há, realmente, nenhum controle sobre este limite, de modo que números de imagens grandes fornecerão endereços maiores do que 3FFFH. Tais endereços, quando passados às demais rotinas VDP, darão a volta pelo zero e estragarão a Tabela de Imagens de Caracteres na VRAM.

|                    |                                |
|--------------------|--------------------------------|
| Endereço . . . . . | 06F9H                          |
| Nome . . . . .     | CALATR                         |
| Entrada . . . . .  | A=Número sprite                |
| Saída . . . . .    | HL=Endereço do atributo sprite |
| Modifica . . . . . | AF, DE, HL                     |

Rotina padrão para calcular o endereço de um bloco de atributos de sprite. O número do sprite, de zero a trinta e um, é multiplicado por quatro e somado ao endereço base da Tabela de Atributos de Sprites (ATRBAS).

|                    |                                       |
|--------------------|---------------------------------------|
| Endereço . . . . . | 0704H                                 |
| Nome . . . . .     | GSPSIZ                                |
| Entrada . . . . .  | Nenhuma                               |
| Saída . . . . .    | A=Bytes na imagem do sprite (8 ou 32) |
| Modifica . . . . . | AF                                    |

Rotina padrão para desenvolver o número de bytes ocupados por cada imagem de sprite na Tabela de Imagens de Sprites. O resultado é determinado, simplesmente, examinando o bit de Tamanho no Registrador de Modo 1 do VDP.

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| Endereço . . . . . | 070FH                                                   |
| Nome . . . . .     | LDIRMV                                                  |
| Entrada . . . . .  | BC=Comprimento, DE=Endereço da RAM, HL=endereço da VRAM |
| Saída . . . . .    | Nenhuma                                                 |
| Modifica . . . . . | AF, BC, DE, EI                                          |

Rotina padrão para copiar um bloco da VRAM do VDP para a memória principal. O endereço inicial da VRAM é estabelecido pela rotina padrão SETRD e os bytes subsequentes são lidos da Porta de Dados do VDP e colocados na memória principal.

Endereço . . . . . 071EH

Esta rotina é utilizada para copiar um conjunto de caracteres de 2KB na Tabela de Imagens de Caracteres do VDP, em qualquer modo. O endereço básico da Tabela de Imagens de Caracteres na VRAM é tomado de CGPBAS. O endereço inicial do conjunto de caracteres é tomado de CGPNT. A rotina padrão RDSLT é utilizada para ler os dados do caractere, de modo que o conjunto poderá estar situado em uma ROM de extensão.

Na partida, CGPNT é inicializado com o endereço contido na posição de ROM 0004H, que é 1BBFH. CGPNT é facilmente alterado para produzir resultados interessantes, POKE&HF920,&HC7:SCREEN 0 fornece um exemplo bastante confuso.

Endereço . . . . . 0744H

Nome . . . . . LDIRVM

Entrada . . . . . BC=Comprimento, DE=Endereço da VRAM, HR=endereço da RAM

Sáda . . . . . Nenhuma

Modifica . . . . . AF, BC, DE, HL, EI

Rotina padrão para copiar um bloco da memória principal, para a VRAM. O endereço inicial da VRAM é estabelecido por meio da rotina padrão SETWRT e os bytes subseqüentes são tomados da memória principal e escritos na Porta de Dados do VDP.

Endereço . . . . . 0777H

Esta rotina vai limpar a tela em qualquer modo VDP. No Modo Texto 40x24 e no Modo Texto 32x24, a Tabela de Nomes, cujo endereço básico é tomado de NAMBAS, é primeiro preenchida com espaços ASCII. Em seguida, o cursor é colocado na posição "home" (0A7FH) e LINTTB e a tabela de endereçamento de linha é reinicializada. Finalmente, o display das teclas de função é restaurado, se estiver ativado, pela rotina padrão FNKSB.

No Modo Gráfico, a cor de contorno é primeiro estabelecida por intermédio do Registrador Modo 7 do VDP (0832H). Em seguida, a Tabela de Cores é preenchida com o código da cor de segundo plano, tomado de BAKCLR, para os pixels 0 e 1. Finalmente, a Tabela de Imagens de Caractere é preenchida com zeros.

No Modo Multicolorido, a cor de contorno é primeiro estabelecida por meio do Registrador de Modo 7 (0832H) do VDP. Em seguida a Tabela de Imagens de Caracteres é preenchida com a cor de segundo plano, tomado de BAKCLR.



|                    |                                  |
|--------------------|----------------------------------|
| Endereço . . . . . | 07CDH                            |
| Nome . . . . .     | WRTVRM                           |
| Entrada . . . . .  | A=Byte de dado, HL=Endereço VRAM |
| Saída . . . . .    | Nenhuma                          |
| Modifica . . . . . | EI                               |

Rotina de padrão para escrever um único byte na VRAM do VDP. O endereço da VRAM é primeiro estabelecido por meio da rotina padrão SETWRT e, depois, o byte de dados é escrito na Porta de Dados do VDP. Observe que as duas instruções EX(SP), HL, aparentemente, espúrias desta rotina, e muitas outras, são necessárias para atender às exigências de temporização do VDP.

|                    |                     |
|--------------------|---------------------|
| Endereço . . . . . | 07D7H               |
| Nome . . . . .     | RDVRM               |
| Entrada . . . . .  | HL=Endereço da VRAM |
| Saída . . . . .    | A=Leitura de byte   |
| Modifica . . . . . | AF, EI              |

Rotina padrão para ler um único byte da VRAM do VDP. O endereço da VRAM é primeiro preparado por meio da rotina padrão SETRD e, depois, o byte lido da Porta de Dados da VDP.

|                    |                     |
|--------------------|---------------------|
| Endereço . . . . . | 07DHF               |
| Nome . . . . .     | SETWRT              |
| Entrada . . . . .  | HL=Endereço da VRAM |
| Saída . . . . .    | Nenhuma             |
| Modifica . . . . . | AF, EI              |

Rotina padrão para estabelecer o VDP para escritas subseqüentes da VRAM através da Porta de Dados. O endereço contido no par de registradores HL é escrito primeiro na Porta de Comando LSB, e MSB em segundo, conforme é mostrado na Figura 7. Endereços maiores do que 3FFFH darão a volta pelo zero, pois os dois bits mais significativos do endereço serão ignorados.

|                    |                  |
|--------------------|------------------|
| Endereço . . . . . | 07ECH            |
| Nome . . . . .     | SETRD            |
| Entrada . . . . .  | HL=Endereço VRAM |
| Saída . . . . .    | Nenhuma          |
| Modifica . . . . . | AF, EI           |

Rotina padrão para estabelecer o VDP para leituras subseqüentes da VRAM através da Porta de Dados. O endereço contido no par de registradores HL é escrito

primeiro na Porta de Comando LSB, e MSB em segundo, conforme é mostrado na Figura 7. Endereços maiores do que 3FFFH darão a volta pelo zero, pois os dois bits mais significativos do endereço serão ignorados.

|                    |                |
|--------------------|----------------|
| Endereço . . . . . | 07F=7H         |
| Nome . . . . .     | CHGCLR         |
| Entrada . . . . .  | Nenhuma        |
| Saída . . . . .    | Nenhuma        |
| Modifica . . . . . | AF, BC, HL, EI |

Rotina padrão para estabelecer as cores do VDP. Primeiro, SCRMOD é examinado para determinar a ação adequada. No Modo Texto 40x24, o conteúdo de BAKCLR e FORCLR é escrito no Registrador Modo 7 do VDP para estabelecer a cor dos pixels 0 e 1, que, inicialmente, são azul e branco. Observe que neste modo não há nenhuma maneira de se especificar a cor de contorno, que será a mesma que a cor do pixel 0. No Modo Texto 32x24, Modo Gráfico ou Modo Multicolorido, o conteúdo de BDRCLR é escrito no Registrador Modo 7 do VDP para estabelecer a cor do contorno, que, inicialmente, é azul. Ainda no Modo Texto 32x24, o conteúdo de BAKCLR e FORCLR são copiados em toda a Tabela de Cores para determinar as cores do Pixel 0 e 1.

|                    |                                                   |
|--------------------|---------------------------------------------------|
| Endereço . . . . . | 0815H                                             |
| Nome . . . . .     | FILVRM                                            |
| Entrada . . . . .  | A=Byte de dados, BC=Comprimento, HL=endereço VRAM |
| Saída . . . . .    | Nenhuma                                           |
| Modifica . . . . . | Af, BC, EI                                        |

Rotina padrão para preencher um bloco da VRAM do VDP com um único byte de dados. O endereço inicial da VRAM, contido no par de registradores HL, é primeiro, preparado pela rotina padrão SETWRT. O byte de dados é, em seguida, escrito repetidamente na Porta de Dados VDP, para preencher posições sucessivas da VRAM.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 083BH              |
| Nome . . . . .     | TOTEXT             |
| Entrada . . . . .  | Nenhuma            |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | AF, BC, DE, HL, EI |

Rotina padrão para devolver o VDP ao Modo Texto 40x24 ou Modo Texto 32x24, se estiver, atualmente, no Modo Gráfico ou no Modo Multicolorido. É utilizada pela Ronda Principal ("Mainloop") do Interpretador BASIC e pelo manipulador do

comando "INPUT". Sempre que as rotinas padrões INITXT ou INIT32 são utilizadas, o byte de modo, 00H ou 01H, é copiado em OLDSCR. Caso o modo seja posteriormente mudado para Modo Gráfico ou Modo Multicolorido, e depois tiver que ser devolvido para um dos modos texto para que seja efetuada entrada pelo teclado, esta rotina assegura que a volta seja para o mesmo modo.

SCRMOD é examinado primeiro, e, se a tela já estiver em um dos modos texto, a rotina, simplesmente, termina sem nenhuma ação. Caso contrário, o modo texto anterior será tomado de OLDSCR e passado à rotina padrão CHGMOD.

|                    |                |
|--------------------|----------------|
| Endereço . . . . . | 0848H          |
| Nome . . . . .     | CLS            |
| Entrada . . . . .  | Flag Z         |
| Saída . . . . .    | Nenhuma        |
| Modifica . . . . . | AF, BC, DE, EI |

Rotina padrão para limpar a tela em qualquer modo. Não faz nada mais do que chamar a rotina 0777H. Ela é, na verdade, o manipulador do comando "CLS" e, simplesmente, retorna, se entrada com o Flag NZ (Isto indica que há erro no comando).

|                    |                                      |
|--------------------|--------------------------------------|
| Endereço . . . . . | 084FH                                |
| Nome . . . . .     | CHGMOD                               |
| Entrada . . . . .  | A=modo de tela requisitado (0,1,2,3) |
| Saída . . . . .    | Nenhuma                              |
| Modifica . . . . . | AF, BC, DE, HL, EI                   |

Rotina padrão para estabelecer um novo modo de tela. O Registrador A, contendo o modo de tela requerido, é testado e o controle transferido a INITXT, INIT32, INIGRP ou INIMLT.

|                    |                                             |
|--------------------|---------------------------------------------|
| Endereço . . . . . | 085DH                                       |
| Nome . . . . .     | LPTOUT                                      |
| Entrada . . . . .  | A=Caractere a ser impresso                  |
| Saída . . . . .    | Indicador C, caso haja terminação CTRL-STOP |
| Modifica . . . . . | AF                                          |

Rotina padrão para enviar um caractere à impressora pela Porta Centronics. O status da impressora é testado continuamente, por meio da rotina padrão LPTSTT, até que a impressora fique livre. Em seguida, o caractere é escrito na Porta de Dados Centronics

(Porta de E/S 91H) e o sinal de STROBE da Porta de Status Centronics (Porta de E/S 90H) é abaixado, por um instante. Observe que a rotina padrão BREAKX é utilizada para testar a tecla CTRL-STOP, se a impressora estiver ocupada. Caso seja detectado um CTRL-STOP, um código CR é escrito na Porta de Dados Centronics, afim de fazer fluir o buffer de linha da impressora, e a rotina terminará com o indicador C.

|                    |                                                    |
|--------------------|----------------------------------------------------|
| Endereço . . . . . | 0884H                                              |
| Nome . . . . .     | LPTSTT                                             |
| Entrada . . . . .  | Nenhuma                                            |
| Saída . . . . .    | A=0 e Indicador Z, se a impressora estiver ocupada |
| Modifica . . . . . | AF                                                 |

Rotina padrão para testar, na Porta de Status Centronics, o sinal de BUSY. Isto apenas significa ler a porta de E/S 90H e examinar o estado do bit 1 :0=Pronto, 1=Ocupado.

|                    |                   |
|--------------------|-------------------|
| Endereço . . . . . | 088EH             |
| Nome . . . . .     | POSIT             |
| Entrada . . . . .  | H=Coluna, L=Linha |
| Saída . . . . .    | Nenhuma           |
| Modifica . . . . . | AF, EI            |

Rotina padrão para estabelecer as coordenadas do cursor. As coordenadas da linha e coluna são enviadas à rotina padrão OUTDO como parâmetros, em uma sequência ESC, "X", Linha+1FH, Coluna+1FH. Observe que a posição "home", para o BIOS tem coordenadas 1,1 em vez 0,0, utilizadas pelo Interpretador BASIC.

|                    |                                                                               |
|--------------------|-------------------------------------------------------------------------------|
| Endereço . . . . . | 089DH                                                                         |
| Nome . . . . .     | CNVCHR                                                                        |
| Entrada . . . . .  | A=Caractere                                                                   |
| Saída . . . . .    | Indicador Z, NC=Cabeçalho; Indicador NZ, C= Gráfico;<br>Indicador Z, C=Normal |
| Modifica . . . . . | AF                                                                            |

Rotina padrão para testar e converter, se necessário, caracteres com cabeçalhos gráficos. Caracteres menores do que 20H, normalmente, são interpretados por acionadores de dispositivos de saída como caracteres de controle. Um código de caracteres, nesta faixa, pode ser tratado como um caractere apresentável, precedendo-se com um código de controle de cabeçalho gráfico (01H) e somando 40H ao seu valor. Por exemplo, para

apresentar diretamente o caractere de código ODH, em vez de tê-lo interpretado como um retorno de carro, será necessário enviar os bytes 01H, 4DH. Esta rotina é utilizada pelos acionadores dos dispositivos de saída, como, por exemplo, a rotina padrão CHPUT, para verificar estas seqüências.

Se o caractere for um cabeçalho gráfico GRPHED, ele é colocado em 01H e a rotina termina; caso contrário GRPHED é zerado. Se o caractere estiver fora da faixa 40H a 5FH, ele permanece inalterado. Caso esteja no interior desta faixa, e GRPHED, contenha valor 01H, indicando um cabeçalho gráfico anterior ele, é convertido subtraindo-se 40H.

|                |             |
|----------------|-------------|
| Endereço ..... | 08BCH       |
| Nome .....     | CHPUT       |
| Entrada .....  | A=Caractere |
| Saída .....    | Nenhuma     |
| Modifica ..... | EI          |

Rotina padrão para dar saída a um caractere para a tela no Modo Texto 40x24 ou no Modo Texto 32x24. Primeiro, SCRMOD é verificado e se o VDP estiver no Modo Gráfico ou no Modo Multicolorido a rotina termina, sem ação. Caso contrário, o cursor é removido (0A2EH), o caractere decodificado (08DFH) e, em seguida, o cursor substituído (09E1H). Finalmente, a posição da coluna do cursor é colocada em TTYPOS, para ser utilizada pelo comando "PRINT", e a rotina termina.

|                |       |
|----------------|-------|
| Endereço ..... | 08DFH |
|----------------|-------|

Esta rotina é utilizada pela rotina padrão CHPUT para decodificar um caractere e tornar a ação adequada. A rotina padrão CNVCHR é, primeiramente, utilizada para verificar um caractere gráfico; se o caractere for um código de cabeçalho (01H) a rotina termina sem nenhuma ação. Caso o caractere seja um gráfico convertido, então, a seção de decodificação do código de controle é pulada. Caso contrário, ESCCNT é verificado para sabermos se um caractere ESC (1BH) anterior foi recebido. Em caso afirmativo o controle é transferido ao processador de seqüências ESC (098FH). Caso contrário, o caractere é verificado para sabermos se é menor do que 20H. Em caso afirmativo, o controle é transferido ao processador de códigos de controle (0914H). O caractere é, em seguida, verificado para sabermos se é DEL (7FH). Em caso afirmativo o controle é transferido à rotina de cancelamento (0AE3H).

Considerando que o caractere seja apresentável, as coordenadas do cursor são tomadas de CSRX e CSRY e colocadas no par de registradores HL, H=Coluna, L=Linha. Estas, em seguida, são convertidas em um endereço físico da Tabela de Nomes VDP e o caractere colocado ali (0BE6H). A posição da coluna do cursor, em seguida, é incrementada (0A44H) e, considerando a coluna mais à direita não tenha sido excedida, a rotina termina. Caso contrário, a entrada da linha em LINTTB, a tabela de terminação de linha, é zerada para indicar uma linha lógica estendida, o número da coluna é colocado em 01H e um LF é executado.

Endereço . . . . . 0908H

Esta rotina executa a operação LP para o processador de controle da rotina padrão do CHPUT. A linha do cursor é incrementada (0A61H) e, presumindo que a linha mais baixa não tenha sido excedida, a rotina termina. Caso contrário, a tela é rolada para cima e a linha inferior cancelada (0A88H).

Endereço . . . . . 0914H

Este é o processador dos códigos de controle para a rotina padrão CHPUT. O código é procurado na tabela localizada em 092FH e o controle transferido ao endereço associado ao código.

Endereço . . . . . 092FH

Esta tabela contém os códigos de controle, cada um com um endereço associado, reconhecido pela rotina padrão CHPUT:

| CÓDIGO | PARA  | FUNÇÃO                                         |
|--------|-------|------------------------------------------------|
| 07H    | 1113H | O som é acionado (BELL)                        |
| 08H    | 0A4CH | Cursor para esquerda (BS)                      |
| 09H    | 0A71H | Cursor para próxima posição de tabulação (TAB) |
| 0AH    | 0908H | Cursor desce uma linha (LF)                    |
| 0BH    | 0A7FH | Cursor à posição inicial (HOME)                |
| OCH    | 077EH | Limpa tela e à posição inicial (FORMFEED)      |
| ODH    | 0A81H | Cursor vai à coluna mais à esquerda (CR)       |
| 1BH    | 0989H | Entra na sequência escape (ESC)                |
| 1CH    | 0A5BH | Cursor para direita (RIGHT)                    |
| 1DH    | 0A4CH | Cursor para esquerda (LEFT)                    |
| 1EH    | 0A57H | Cursor para cima (UP)                          |
| 1FH    | 0A61H | Cursor para baixo (DOWN)                       |

---

| CÓDIGO | PARA | FUNÇÃO |
|--------|------|--------|
|--------|------|--------|

---

|     |       |                                                |
|-----|-------|------------------------------------------------|
| 6AH | 077EH | Limpa tela e à posição inicial (ESC, "j")      |
| 45H | 077EH | Limpa tela e à posição inicial (ESC, "E")      |
| 4BH | 0AEEH | Limpa até o final da linha (ESC, "K")          |
| 4AH | 0B05H | Lima até fim da tela (ESC, "J")                |
| 6CH | 0AECH | Limpa a linha (ESC, "l")                       |
| 4CH | 0AB4H | Insere linha (ESC, "L")                        |
| 4DH | 0A85H | Cancela linha (ESC, "M")                       |
| 59H | 0986H | Estabelece as coordenadas do cursor (ESC, "Y") |
| 41H | 0A57H | Cursor para cima (ESC, "A")                    |
| 42H | 0A61H | Cursor para baixo (ESC, "B")                   |
| 43H | 0A44H | Cursor para direita (ESC, "C")                 |
| 44H | 0A55H | Cursor para esquerda (ESC, "D")                |
| 48H | 0A7FH | Cursor para posição inicial (ESC, "H")         |
| 78H | 0980H | Muda cursor (ESC, "x")                         |
| 79H | 0983H | Muda cursor (ESC, "y")                         |

Endereço ..... 0953H

Esta tabela contém os códigos de controle ESC, cada um com um endereço associado, reconhecido pela rotina padrão CHPUT:

Endereço ..... 0980H

Esta rotina executa a operação ESC, "x" para o processador de códigos de controle da rotina padrão CHPUT.ESCCNT é colocado em 01H para indicar que o próximo caractere a ser recebido será um parâmetro.

Endereço ..... 0983H

Esta rotina executa a operação ESC, "y" para o processador dos códigos de controle da rotina padrão CHPUT.ESCCNT é colocado em 02H para indicar que o próximo caractere a ser recebido será um parâmetro.

Endereço ..... 0986H

Esta rotina executa a operação ESC, "Y" para o processador de códigos de controle da rotina padrão CHPUT.ESCCNT é colocado em 04H para indicar que o próximo caractere a ser recebido será um parâmetro.

Endereço . . . . . 0989H

Esta rotina executa a operação ESC para o processador de códigos de controle da rotina padrão CHPUT. ESCCNT é colocado em FFH para indicar que o próximo caractere a ser recebido será o segundo caractere de controle.

Endereço . . . . . 098FH

Este é o processador de seqüências ESC da rotina padrão CHPUT. Se ESCCNT contiver FFH, o caractere será o segundo caractere de controle e o controle será transferido ao processador de código de controle (0919H) para vasculhar a tabela de código ESC (0953H).

Se ESCCNT contiver 01H, o caractere será o parâmetro único da seqüência ESC, "x". Caso o parâmetro seja "4"(34H), CSTYLE será colocado em 00H, resultando em um cursor tipo "bloco". Caso o parâmetro seja "5"(35H), CSRSW será colocado em 00H, tornando o cursor normalmente desativado.

Se ESCCNT contiver 02H, o caractere será o único parâmetro na seqüência ESC, "y". Caso o parâmetro seja "4"(34H), então, CSTYLE será colocado em 01H, resultando um cursor tipo sublinhado. Se o parâmetro for "5"(35H), então, CSRSSW será colocado em 01H, tornando o cursor normalmente ativado.

Se ESCCNT contiver 04H, o caractere será o primeiro parâmetro da seqüência ESC, "Y" e será a coordenada de linha. Do parâmetro, será subtraído 1FH e ele será colocado em CSRY:ESCCNT será, então, decrementado para 03H.

Se ESCCNT contiver 03H, o caractere será o segundo parâmetro da seqüência ESC, "Y" e será a coordenada da coluna. Do parâmetro será subtraído 1FH e ele será colocado em CSRX.

Endereço . . . . . 09DAH

Esta rotina é utilizada pela rotina padrão CHGET, por exemplo, para apresentar o caractere do cursor quando ele é normalmente desativado. Se CSRSW for não-zero a rotina, simplesmente, terminará sem nenhuma ação. Caso contrário, o cursor será apresentado (09E6H).

Endereço . . . . . 09E1H

Esta rotina é utilizada pela rotina padrão CHPUT, por exemplo, para apresentar o caractere do cursor quando ele é normalmente ativado. Se CSRSW for zero a rotina,



simplesmente, terminará sem nenhuma ação. SCRMOD será verificado e, se a tela estiver no Modo Gráfico ou Modo Multicolorido, a rotina terminará sem nenhuma ação. Caso contrário, as coordenadas do cursor serão convertidas em um endereço físico da Tabela de Nomes do VDP e o caractere lido daquela posição (0BD8H) será salvo em CURSAV.

A imagem de pixels de oito bytes do caractere será lida da Tabela de Imagens de Caracteres VDP para o buffer os LINWRK (0BA5H). Em seguida, será invertido o padrão de pixels de todos os oito bytes, se CSTYLE indicar um cursor de bloco, ou, apenas os três inferiores, se CSTYLE indicar um cursor sublinhado. O padrão de pixels é copiado, de volta, na posição correspondente ao caractere 255 na Tabela de Imagens de Caracteres VDP (0BBEH). O código do caractere 255, em seguida, será colocado na posição do cursor atual na Tabela de Nomes VDP (0BE6H) e a rotina terminará.

Este método para gerar o caractere do cursor, utilizando o caractere 255, poderá produzir efeitos curiosos, sob determinadas condições. Estes, poderão ser demonstrados pela execução do comando BASIC FOR N=1 TO 100:PRINT CHR\$(255);:NEXT e, depois, pressionando a tecla "cursor para cima".

Endereço ..... 0A27H

Esta rotina é utilizada pela rotina padrão CHGET, por exemplo, para remover o caractere do cursor quando este é normalmente desativado. Se CSRSW for não-zero, a rotina simplesmente terminará sem nenhuma ação e em caso contrário, o cursor será removido (0A33H).

Endereço ..... 0A2EH

Esta rotina é utilizada pela rotina padrão CHPUT, por exemplo, para remover o caractere do cursor quando ele é normalmente ativado. Se CSRSW for zero, a rotina simplesmente terminará sem nenhuma ação. SCRMOD será verificado e, se a tela estiver no Modo Gráfico ou Modo Multicolorido, a rotina terminará sem nenhuma ação. Caso contrário as coordenadas do cursor serão convertidas em um endereço físico da Tabela de Nomes do VDP e o caractere mantido em CURSAV será escrito naquela posição (0BE6H).

Endereço ..... 0A44H

Esta rotina executa a operação ESC, "C" para o processador de códigos de controle da rotina padrão CHPUT. Se a coordenada da coluna do cursor já estiver na

coluna mais à direita, indicada em LINLEN, então, a rotina terminará sem nenhuma ação. Caso contrário, a coordenada da coluna será incrementada e CSRX atualizado.

Endereço ..... 0A4CH

Esta rotina executa a operação “cursor à esquerda” (BS/LEFT) para o processador de códigos de controle da rotina padrão CHPUT. A coordenada da coluna do cursor é decrementada e CSRX atualizado. Caso a coordenada da coluna tiver ficado com um valor, além da posição mais à esquerda, será colocada nela, o valor da posição mais à direita (LINLEN) e uma operação “cursor p/cima” (UP) será executada.

Endereço ..... 0A55H

Esta rotina executa a operação ESC, “D” para o processador de códigos de controle da rotina padrão CHPUT. Se a coordenada da coluna do cursor já estiver na posição mais à esquerda, a rotina terminará sem nenhuma ação. Caso contrário, a coordenada da coluna será decrementada e CSRX atualizado.

Endereço ..... 0A57H

Esta rotina executa a operação ESC, “A” (UP) para o processador de códigos de controle da rotina padrão CHPUT. Se a coordenada de linha do cursor já estiver na posição mais alta, a rotina terminará sem nenhuma ação. Caso contrário, a coordenada da linha será decrementada e CSRY atualizado.

Endereço ..... 0A5BH

Esta rotina executa a operação “cursor à direita” (RIGHT) para o processador de códigos de controle da rotina padrão CHPUT. A coordenada da coluna do cursor será incrementada e CSRX atualizado. Se a coordenada da coluna tiver se movido, além da posição mais à direita, determinada por LINLEN, será colocada na posição mais à esquerda (01H) e uma operação “cursor p/baixo” (DOWN) será executada.

Endereço ..... 0A61H

Esta rotina executa a operação ESC, “B” (DOWN) para o processador de códigos de controle da rotina padrão CHPUT. Se a coordenada de linha do cursor já estiver na posição mais baixa, determinada por CRTCNT e CNSDFG(0C32H), então a

rotina terminará sem nenhuma ação. Caso contrário, a coordenada de linha será incrementada e CSRY atualizado.

Endereço ..... 0A71H

Esta rotina executa a operação TAB para o processador de códigos de controle da rotina padrão CHPUT. Serão enviados espaços ASCII (08DFH) até que CSRX venha a ser um múltiplo de oito, mais um. (Colunas BIOS 1, 9, 17, 25, 33).

Endereço ..... 0A7FH

Esta rotina executa a operação ESC, "H" (HOME) para o processador de códigos de controle da rotina padrão CHPUT:CSRX e CSRY serão, simplesmente, colocados em 1,1. O sistema de coordenadas de cursor de BIOS em ROM, embora funcionalmente idêntico ao utilizado pelo Interpretador BASIC, numera as linhas da tela de 1 a 24 e as colunas de 1 a 32/40.

Endereço ..... 0A81H

Esta rotina executa a operação CR para o processador de códigos da rotina padrão CHPUT. CSRX é, simplesmente, colocado em 01H.

Endereço ..... 0A85H

Esta rotina executa à função ESC, "M" para o processador de códigos de controle da rotina padrão CHPUT. Uma operação CR é executada para colocar a coordenada da coluna do cursor na posição mais à esquerda. O número de linhas da linha atual, até a parte inferior da tela, é, em seguida, determinado, e se for zero a linha atual é, simplesmente, cancelada (0AECH). O número de linhas contado é, primeiro, utilizado para rolar a seção relevante de LINTTB, a tabela de terminação de linha, em um byte para cima. Em seguida, é utilizado para rolar a seção relevante da tela uma linha por vez. Iniciando na linha abaixo da linha atual, cada linha é copiada da Tabela de Nomes do VDP no buffer LINWRK (0BAAH), em seguida, copiada de volta à Tabela de Nomes uma linha acima (0BC3H). Finalmente, a linha inferior, na tela, é cancelada (0AECH).

Endereço ..... 0AB4H

Esta rotina executa a operação ESC, "L" para o processador de códigos de controle da rotina padrão CHPUT. Primeiro, uma operação CR é executada, para colocar a

coordenada da coluna do cursor na posição mais à esquerda. Em seguida, é determinado o número e linhas da linha atual, até a linha inferior da tela. Caso seja zero, a linha atual, simplesmente, é cancelada (0AECH). A contagem de linhas é, primeiramente, utilizada para rolar a seção relevante de LINTTB, a tabela de terminação de linha, em um byte para baixo. Em seguida, é utilizada para rolar a seção relevante da tela uma linha por vez. Iniciando na ante-penúltima linha da tela, cada linha é copiada da Tabela de Nomes do VDP no buffer LINWRK (0BAAK), depois copiada de volta à Tabela de Nomes uma linha mais abaixo (0BC3H). Finalmente, a linha atual é cancelada (0AECH).

Endereço ..... 0AE3H

Esta rotina é utilizada para executar a operação DEL para o processador de códigos de controle da rotina padrão CHPUT. Primeiro uma operação LEFT é executada. Caso isto não possa ser feito, pelo fato do cursor já estar na posição inicial, a rotina termina sem nenhuma ação. Caso contrário, um espaço é escrito na Tabela de nomes do VDP, na posição física do cursor (0BE6H).

Endereço ..... 0AECH

Esta rotina executa a operação ESC, "1" para o processador de códigos de controle da rotina padrão CHPUT. A coordenada da coluna do cursor é colocada em 01H e o controle cai na rotina ESC, "K".

Endereço ..... 0AEEH

Esta rotina executa a operação ESC, "K" para o processador de códigos de controle da rotina padrão CHPUT. A entrada da linha na LINTTB, a tabela de terminação de linha, é colocada em não-zero, para indicar que a linha lógica não é estendida (0C29H). As coordenadas do cursor são convertidas em um endereço físico (0BF2H) da Tabela de Nomes do VDP, e o VDP preparado para escrita por meio da rotina padrão SETWRT. Em seguida, espaços são escritos diretamente à Porta de Dados do VDP, até que a coluna mais à direita, determinada por LINLEN, seja atingida.

Endereço ..... 0B50H

Esta rotina executa a operação ESC, "J" para o processador de códigos de controle da rotina padrão CHPUT. Uma operação ESC, "K" é executada em linhas sucessivas, iniciando com a atual, até que a parte inferior da tela seja alcançada.

|                    |            |
|--------------------|------------|
| Endereço . . . . . | 0B15H      |
| Nome . . . . .     | ERAFNK     |
| Entrada . . . . .  | Nenhuma    |
| Saída . . . . .    | Nenhuma    |
| Modifica . . . . . | AF, DE, EI |

Rotina padrão para desligar a apresentação das teclas de função. Primeiro CNSDFG é zerado. Se o VDP estiver no Modo Gráfico ou Modo Multicolorido, a rotina terminará sem nenhuma ação adicional. Se o VDP estiver no Modo Texto 40x24 ou no Modo Texto 32x24, a última linha na tela será, então, cancelada (0AECH).

|                    |                |
|--------------------|----------------|
| Endereço . . . . . | 0B26H          |
| Nome . . . . .     | FNKSB          |
| Entrada . . . . .  | Nenhuma        |
| Saída . . . . .    | Nenhuma        |
| Modifica . . . . . | AF, BC, DE, EI |

Rotina padrão para mostrar as teclas de função, se isto estiver habilitado. Se CNSDFG for zero a rotina terminará sem nenhuma ação; caso contrário, o controle irá para a rotina padrão DSPFNK.

|                    |                |
|--------------------|----------------|
| Endereço . . . . . | 0B2BH          |
| Nome . . . . .     | DSPFNK         |
| Entrada . . . . .  | Nenhuma        |
| Saída . . . . .    | Nenhuma        |
| Modifica . . . . . | AF, BC, DE, EI |

Rotina padrão para habilitar o display da tecla de função. CNSDFG é colocado em FFH e, se o VDP estiver no Modo Gráfico ou no Modo Multicolorido, a rotina terminará sem outra ação. Caso contrário, a coordenada da linha do cursor será verificada e, se o cursor estiver na última linha da tela, um código LF (OAH) será enviado à rotina padrão OUTDO, que rolará a tela para cima. O par de registradores HL é, em seguida, colocado apontando para as strings de função, com ou sem SHIFT, na Área de Trabalho, dependendo se a tecla SHIFT estiver ou não pressionada. Será subtraído quatro de LINLEN, para permitir um mínimo de um espaço entre campos, e será dividido por cinco, para determinar o tamanho do campo de cada string. Caracteres sucessivos serão, em seguida, tomados de cada string de função, verificados quanto a cabeçalhos gráficos, por meio da rotina padrão CNVCHR, e colocados no buffer LINWRK, até que a string seja

esgotada ou a zona esteja preenchida. Quando todas as cinco strings estiverem completas, o buffer LINWRK será escrito na última linha na Tabela de Nomes do VDP (0BC3H).

Endereço ..... 0B9CH

Esta rotina é utilizada pela rotina padrão relacionada ao display das teclas de função. O conteúdo do registrador A é colocado em CNSDFG; em seguida SCRMOD é testado e o indicador NC devolvido, se a tela estiver no Modo Gráfico ou Modo Multicolorido.

Endereço ..... 0BA5H

Esta rotina copia oito bytes da VRAM do VDP no buffer do LINWRK, o endereço físico da VRAM será fornecido no par de registradores HL.

Endereço ..... 0BAAH

Esta rotina copia uma linha completa de caracteres, com o comprimento determinado por LINLEN, da VRAM da VDP no buffer do LINWRK. A coordenada da linha do cursor será fornecida no registrador L.

Endereço ..... 0BBEH

Esta rotina copia oito bytes do buffer LINWRK na VRAM do VDP. O endereço físico da VRAM será fornecido no par de registradores HL.

Endereço ..... 0BC3H

Esta rotina copia uma linha inteira de caracteres, com o comprimento determinado por LINLEN, do buffer LINWRK na VRAM do VDP. A coordenada da linha do cursor será fornecida no registrador L.

Endereço ..... 0BD8H

Esta rotina lê um único byte da VRAM do VDP para o registrador C. A coordenada da coluna será fornecida no registrador H e a coordenada da linha no registrador L.

Endereço ..... 0BE6H

Esta rotina converte um par de coordenadas de tela, a coluna no registrador H e a linha no registrador L em um endereço físico da Tabela de Nomes do VDP. Este endereço será devolvido no par de registradores HL.

Primeiro, a coordenada da linha será multiplicada por trinta e dois ou quarenta, dependendo do modo da tela, e somada à coordenada de coluna. Isto, em seguida, é

somado ao endereço base da Tabela de Nomes, tomado de NAMBAS, para fornecer o endereço inicial.

Em virtude da largura de tela variável, contida em LINLEN, um valor adicional será acrescentado ao endereço inicial, para manter a região ativa grosseiramente centralizada na tela. A diferença entre o número “verdadeiro” de caracteres por linha (trinta e dois ou quarenta) e a largura atual é dividida por dois e depois arredondada para fornecer o início à esquerda. Para uma máquina inglesa, com largura de trinta e sete caracteres no Modo Texto 40x24, isto terá como resultado dois caracteres não utilizados do lado esquerdo e um do direito. O comando PRINT(41-WID)\2, onde WID é qualquer largura de tela, apresentará o desvio da coluna esquerda no Modo Texto 40x24.

Abaixo é dado um programa BASIC completo que emula esta rotina:

```
10 CPL=40:NOMES=BASE(0):LARG=PEEK(&HF3AE)
20 MDTELA=PEEK(&HFCAF):IF MDTELA=0 THEN 40
30 CPL=32:NOMES=BASE(5):LARG=PEEK(&HF3AF)
40 ESQ=(CPL+1-LARG)\2
50 RESULT=NOMES+(LIN-1)*CPL+(COL-1)+ESQ
```

Este programa foi projetado para o sistema de coordenada ROW e COL utilizado pelo BIOS em ROM, cuja origem é 1,1. A linha 50 poderá ser simplificada, removendo os fatores “-1”, caso o sistema de coordenadas de interpretações BASIC seja utilizado.

Endereço ..... 0C1DH

Esta rotina calcula o endereço da entrada da linha em LINTTB, a tabela de terminação de linha. A coordenada da linha é fornecida no registrador L e o endereço é devolvido no par de registradores DE.

Endereço ..... 0C29H

Esta rotina coloca um valor não-zero na entrada de uma linha em LINTTB quando chamada em 0C29H e zero quando chamada em 0C2AH. A coordenada da linha é colocada no registrador L.

Endereço ..... 0C32H

Esta rotina devolve o número de linhas da tela no registrador A. Normalmente, devolverá vinte e quatro se o display da tecla de função estiver desativado, e vinte e três se estiver ativado. Observe que o tamanho da tela é determinado por CRTCNT e poderá ser modificado com o comando BASIC, POKE &HF3B1H, 14:SCREEN 0, por exemplo.

Endereço ..... 0C3CH  
 Nome ..... KEYINT  
 Entrada ..... Nenhuma  
 Saída ..... Nenhuma  
 Modifica ..... EI

Rotina padrão para processar interrupções do Z80, que são geradas pelo VDP, a cada 20 mS, em uma máquina Inglesa. O Registrador de Status do VDP é lido e o bit 7 verificado para garantir que é uma interrupção de tela. Caso contrário, a rotina terminará sem nenhuma ação. O conteúdo do Registrador de Status é salvo em STATFL e o bit 5 é verificado quanto à coincidência de sprites. Caso o Indicador de Coincidência estiver ativo, a entrada relevante em TRPTBL é atualizada (0EF1H).

INTCNT, o contador de “INTERVAL”, é decrementado. Se tiver atingido zero, a entrada relevante em TRPTBL será atualizada (0EF1H) e o contador reposto com o conteúdo de INTVAL.

JIFFY, o contador “TIME”, em seguida, será incrementado. Este contador retornará a zero, quando completo (“overflow”).

MUSICF será examinado para determinar se alguma das três filas musicais, geradas pelo comando “PLAY”, está ativa. Para cada fila ativa, será chamada a rotina que desempilha filas (113BH), para pegar o próximo pacote musical e escrevê-lo no PSG.

SCNCNT, em seguida, será decrementado para determinar se uma varredura de teclado e joystick serão necessários. Em caso negativo, o manipulador de interrupção terminará sem nenhuma ação adicional. Este contador é utilizado para aumentar a produtividade e minimizar os problemas de rebatimento do teclado (“Keybounce”), garantindo que uma varredura seja executada apenas a cada três interrupções. Considerando que uma varredura seja necessária, o conector do joystick 1 é selecionado e os dois bits de Gatilho são lidos (120CH), seguido pelos dois bits de Gatilho do conector do joystick 2 (120CH) e a barra de espaço da linha 8 do teclado (1226H). Estas cinco entradas, que são todas relacionadas ao comando “STRIG”, são combinadas em um único byte, onde: 0=pressionado, 1=não pressionado:

| 7              | 6              | 5              | 4              | 3 | 2 | 1 | 0      |
|----------------|----------------|----------------|----------------|---|---|---|--------|
| Joy 2<br>Gat.B | Joy 2<br>Gat.A | Joy 1<br>Gat.B | Joy 1<br>Gat.A | 0 | 0 | 0 | Espaço |

**Figura 35** Entradas “STRIG”



Esta leitura será comparada com a anterior, mantida em TRGFLG, para fornecer um byte de transição ativo. TRGFLG será atualizado com a nova leitura. O byte de transição ativa, normalmente, é zero, mas contém um 1 em cada posição em que uma transição de não pressionado para pressionado ocorreu. Este byte de transição ativa é deslocado bit a bit, e a entrada relevante de TRPTBL é atualizada (0EF1H) para cada dispositivo ativo.

Uma varredura completa da matriz do teclado é executada, em seguida, para novas operações de teclas, e qualquer uma identificada é traduzida em códigos de tecla e colocada em KEYBUF (0D12H). Se KEYBUF estiver vazio no final deste processo, REPCNT será decrementado para verificar se o atraso de auto-repetição expirou. Em caso negativo, a rotina terminará se o período de atraso tiver terminado, REPCNT será respondido com o valor de repetição rápida (60mS), o mapa do teclado OLDKEY reinicializado e o teclado varrido novamente (0D4EH). Qualquer tecla pressionada continuamente, será compreendida como uma nova transição, durante a varredura. Observe que as teclas somente serão auto-repetidas, se o programa aplicativo mantiver KEYBUF vazio, pela leitura de caracteres. Em seguida, o manipulador de interrupções terminará.

Endereço . . . . . 0D12H

Esta rotina executa uma varredura completa de todas as onze linhas da matriz do teclado para o manipulador de interrupção. Cada uma das onze linhas é lida via PPI e colocada em NEWKEY em ordem ascendente. Em seguida, ENSTOP é verificado para ver se partidas a quente (warm starts) estão habilitadas. Se seu conteúdo for não-zero e as teclas CODE, GRPH, CTRL e SHIFT estiverem sendo pressionadas o controle será transferido ao Interpretador BASIC (409BH) por meio da rotina CALBAS. Este recurso será útil, uma vez que, mesmo um programa em código de máquina poderá ser interrompido, desde que o manipulador de interrupções esteja funcionando.

O conteúdo de NEWKEY é comparado com a varredura anterior contida em OLDKEY. Se qualquer alteração tiver ocorrido, REPCNT será carregado com o atraso de auto-repetição inicial (780 mS). Cada leitura de linha de NEWKEY é, em seguida, comparada com a anterior, mantida em OLDKEY, a fim de produzir um byte de transição ativo e OLDKEY será atualizada com a nova leitura. O byte de transição ativa, normalmente, é zero, mas contém um 1 em cada posição em que ocorreu uma transição de não-pressionado para pressionado. Caso a linha contenha quaisquer transições, estas serão decodificadas e colocadas em KEYBUF como códigos de teclas (0D89H). Quando todas as onze linhas forem verificadas, a rotina checará se há algum caractere em KEYBUF, subtraindo GETPNT de PUTPNT, e terminará.

Endereço ..... 0D6AH  
 Nome ..... CHSNS  
 Entrada ..... Nenhuma  
 Saída ..... Indicador, NZ, se houver caractere em KEYBUF  
 Modifica ..... AF, EI

Rotina padrão para verificar se algum caractere do teclado está pronto. Se a tela estiver no Modo Gráfico ou Modo Multicolorido, GETPNT será subtraído de PUTPNT (0D62H) e a rotina terminará. Se a tela estiver no Modo Texto 40x24 ou no Modo Texto 32x24, o estado da tecla SHIFT também será examinado e o display das teclas de função atualizado, via rotina padrão DSPFNK, se este estado tiver mudado.

Endereço ..... 0D89H

Esta rotina converte cada bit ativo em um byte de transição de uma linha de teclado em um código de tecla. Primeiro um bit é convertido em um número de tecla determinado pela sua posição na matriz do teclado:

|                |               |             |               |              |                |               |                |         |
|----------------|---------------|-------------|---------------|--------------|----------------|---------------|----------------|---------|
| 7<br>(07H)     | 6<br>(06H)    | 5<br>(05H)  | 4<br>(04H)    | 3<br>(03H)   | 2<br>(02H)     | 1<br>(01H)    | 0<br>(00H)     | Linha 0 |
| ;<br>(0FH)     | ]<br>(0EH)    | [<br>(0DH)  | \<br>(0CH)    | =<br>(0BH)   | -<br>(0AH)     | 9<br>(09H)    | 8<br>(08H)     | Linha 1 |
| B<br>(17H)     | A<br>(16H)    | £<br>(15H)  | /<br>(14H)    | ·<br>(13H)   | ´<br>(12H)     | ˆ<br>(11H)    | ˘<br>(10H)     | Linha 2 |
| J<br>(1FH)     | I<br>(1EH)    | H<br>(1DH)  | G<br>(1CH)    | F<br>(1BH)   | E<br>(1AH)     | D<br>(19H)    | C<br>(18H)     | Linha 3 |
| R<br>(27H)     | Q<br>(26H)    | P<br>(25H)  | O<br>(24H)    | N<br>(23H)   | M<br>(22H)     | L<br>(21H)    | K<br>(20H)     | Linha 4 |
| Z<br>(2FH)     | Y<br>(2EH)    | X<br>(2DH)  | W<br>(2CH)    | V<br>(2BH)   | U<br>(2AH)     | T<br>(29H)    | S<br>(28H)     | Linha 5 |
| F3<br>(37H)    | F2<br>(36H)   | F1<br>(35H) | CODE<br>(34H) | CAP<br>(33H) | GRAPH<br>(32H) | CTRL<br>(31H) | SHIFT<br>(30H) | Linha 6 |
| CR<br>(3FH)    | SEL<br>(3EH)  | BS<br>(3DH) | STOP<br>(3CH) | TAB<br>(3BH) | ESC<br>(3AH)   | F5<br>(39H)   | F4<br>(38H)    | Linha 7 |
| RIGHT<br>(47H) | DOWN<br>(46H) | UP<br>(45H) | LEFT<br>(44H) | DEL<br>(43H) | INS<br>(42H)   | HOME<br>(41H) | SPACE<br>(40H) | Linha 8 |

(continua na página seguinte)

(continuação)

|            |            |            |            |            |       |       |       |          |
|------------|------------|------------|------------|------------|-------|-------|-------|----------|
| 4<br>(4FH) | 3<br>(4EH) | 2<br>(4DH) | 1<br>(4CH) | 0<br>(4BH) | (4AH) | (49H) | (48H) | Linha 9  |
| (57H)      | (56H)      | (55H)      | (54H)      | (53H)      | (52H) | (51H) | (50H) | Linha 10 |
| 7          | 6          | 5          | 4          | 3          | 2     | 1     | 0     | Coluna   |

Figura 36 Número das teclas

| MATRIZ DE TECLADO DO HOT-BIT |       |       |       |       |       |       |        |       |
|------------------------------|-------|-------|-------|-------|-------|-------|--------|-------|
| COLUNA                       |       |       |       |       |       |       |        | linha |
| 7                            | 6     | 5     | 4     | 3     | 2     | 1     | 0      |       |
| 7                            | 6     | 5     | 4     | 3     | 2     | 1     | 0      | 0     |
| (07H)                        | (06H) | (05H) | (04H) | (03H) | (02H) | (01H) | (00H)  |       |
| C                            | "     | '     | =     | =     | -     | 9     | 8      | 1     |
| (0FH)                        | (0EH) | (0DH) | (0CH) | (0BH) | (0AH) | (09H) | (08H)  |       |
| B                            | A     | \     | /     | .     | ,     | [     | ]      | 2     |
| (17H)                        | (16H) | (15H) | (14H) | (13H) | (12H) | (11H) | (10H)  |       |
| J                            | I     | H     | G     | F     | E     | D     | C      | 3     |
| (1FH)                        | (1EH) | (1DH) | (1CH) | (1BH) | (1AH) | (19H) | (18H)  |       |
| K                            | Q     | P     | O     | N     | M     | L     | K      | 4     |
| (27H)                        | (26H) | (25H) | (24H) | (23H) | (22H) | (21H) | (20H)  |       |
| Z                            | Y     | X     | W     | V     | U     | T     | S      | 5     |
| (2FH)                        | (2EH) | (2DH) | (2CH) | (2BH) | (2AH) | (29H) | (28H)  |       |
| F3                           | F2    | F1    | CODE  | CAPS  | GRAPH | CTRL  | SHIFT  | 6     |
| (37H)                        | (36H) | (35H) | (34H) | (33H) | (32H) | (31H) | (30H)  |       |
| RET                          | SLCT  | DS    | STOP  | TAB   | ESC   | F5    | F4     | 7     |
| (3FH)                        | (3EH) | (3DH) | (3CH) | (3BH) | (3AH) | (39H) | (38H)  |       |
| DIR                          | BAIXO | CIMA  | ESQ   | DEL   | INS   | HOME  | barra  | 8     |
| seta                         | seta  | seta  | seta  |       |       |       | espaco |       |
| (47H)                        | (46H) | (45H) | (44H) | (43H) | (42H) | (41H) | (40H)  |       |
| (4FH)                        | (4EH) | (4DH) | (4CH) | (4BH) | (4AH) | (49H) | (48H)  | 9     |
| (57H)                        | (56H) | (55H) | (54H) | (53H) | (52H) | (51H) | (50H)  | 10    |
| 7                            | 6     | 5     | 4     | 3     | 2     | 1     | 0      |       |
| COLUNA                       |       |       |       |       |       |       |        |       |

Figura 36-B Matriz de teclado do HOT-BIT

Em seguida, o número da tecla será convertido em um código de tecla e colocado em KEYBUF (1021H). Quando todos os oito bits possíveis tiverem sido processados a rotina terminará.

Endereço . . . . . 0DA5H

Esta tabela contém os códigos correspondentes às teclas numeradas de 00H a 2FH, para diversas combinações das teclas de controle. Um valor zero na tabela significa que nenhum código de tecla será produzido, quando ela for pressionada:

|                        |     |     |     |     |     |     |     |     |         |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|---------|
| <u>NORMAL</u>          | 37H | 36H | 35H | 34H | 33H | 32H | 31H | 30H | Linha 0 |
|                        | 87H | FFH | FFH | 5CH | 3DH | 2DH | 39H | 38H | Linha 1 |
|                        | 62H | 61H | 3CH | 2FH | 2EH | 2CH | 5BH | FFH | Linha 2 |
|                        | 6AH | 69H | 68H | 67H | 66H | 65H | 64H | 63H | Linha 3 |
|                        | 72H | 71H | 70H | 6FH | 6EH | 6DH | 6CH | 6BH | Linha 4 |
|                        | 7AH | 79H | 78H | 77H | 76H | 75H | 74H | 73H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
| <u>SHIFT</u>           | 26H | 22H | 25H | 24H | 23H | 40H | 21H | 29H | Linha 0 |
|                        | 80H | 27H | FFH | 5EH | 28H | 5FH | 20H | 2AH | Linha 1 |
|                        | 42H | 41H | 3EH | 3FH | 3AH | 3BH | 5DH | FFH | Linha 2 |
|                        | 4AH | 49H | 4BH | 47H | 46H | 45H | 44H | 43H | Linha 3 |
|                        | 52H | 51H | 50H | 4FH | 4EH | 4DH | 4CH | 4BH | Linha 4 |
|                        | 5AH | 59H | 58H | 57H | 56H | 55H | 54H | 53H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
| <u>GRAPH</u>           | FBH | F4H | BDH | EFH | BAH | ABH | ACH | 09H | Linha 0 |
|                        | 06H | 0DH | 01H | 1EH | F1H | 17H | 07H | ECH | Linha 1 |
|                        | 11H | C4H | 7CH | 1DH | F2H | F3H | BBH | 05H | Linha 2 |
|                        | C6H | DCH | 13H | 15H | 14H | CDH | C7H | BCH | Linha 3 |
|                        | 18H | CCH | D8H | C2H | 1BH | 0BH | C8H | DDH | Linha 4 |
|                        | 0FH | 19H | 1CH | CFH | 1AH | C0H | 12H | D2H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
| <u>SHIFT<br/>GRAPH</u> | 00H | F5H | 00H | 00H | FCH | FDH | 00H | 0AH | Linha 0 |
|                        | 04H | 0EH | 02H | 16H | F0H | 1FH | 0BH | 00H | Linha 1 |
|                        | 00H | FEH | FFH | F6H | AFH | AEH | F7H | 03H | Linha 2 |
|                        | CAH | DFH | D6H | 10H | D4H | CEH | C1H | FAH | Linha 3 |
|                        | A9H | CBH | D7H | C3H | D3H | 0CH | C9H | DEH | Linha 4 |
|                        | F8H | AAH | F9H | D0H | D5H | C5H | 00H | D1H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
| <u>CODE</u>            | E1H | E0H | 9BH | 9BH | BFH | D9H | 9FH | EBH | Linha 0 |
|                        | 00H | DAH | EDH | 00H | E9H | EEH | 00H | E7H | Linha 1 |
|                        | B9H | 91H | A7H | 00H | 00H | E4H | E5H | A6H | Linha 2 |
|                        | E3H | 00H | 00H | ABH | A5H | BEH | A4H | 7DH | Linha 3 |
|                        | 9CH | ADH | E6H | DBH | 00H | 00H | EAH | E8H | Linha 4 |
|                        | 00H | 00H | 7BH | 9EH | BBH | E2H | 9DH | 92H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
| <u>SHIFT<br/>CODE</u>  | 01H | E5H | C5H | 1BH | ACH | DAH | 7FH | FEH | Linha 0 |
|                        | 20H | B9H | EDH | 01H | 94H | 21H | 00H | 20H | Linha 1 |
|                        | C1H | E1H | 7EH | 09H | 20H | 0EH | 23H | 0AH | Linha 2 |
|                        | 1BH | B7H | C3H | C1H | E1H | 1BH | ACH | C3H | Linha 3 |
|                        | C3H | 01H | E6H | C9H | 5FH | F8H | EBH | 3AH | Linha 4 |
|                        | 00H | 00H | 00H | 00H | 00H | 00H | 0FH | A0H | Linha 5 |
|                        |     |     |     |     |     |     |     |     |         |
|                        |     |     |     |     |     |     |     |     |         |
|                        | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   | Coluna  |

Endereço . . . . . 0EC5H

O controle é transferido para esta rotina, de 0FC3H, afim de completar a decodificação das cinco teclas de função. Primeiro a entrada relevante em FNKFLG será verificada para determinar se a tecla está associada com um comando "ON KEY

GOSUB". Em caso afirmativo, e desde que CURLIN mostre que o Interpretador BASIC esteja no modo de programa, a entrada relevante em TRPTBL será atualizada (0EF1H) e a rotina terminará. Se a tecla não estiver ligada a um comando "ON KEY GOSUB", ou se o Intérprete estiver no modo direto, a string de caracteres, associada à tecla de função, será devolvida em seu lugar. O número da tecla será multiplicado por dezesseis, pois cada string tem dezesseis caracteres de comprimento e somado ao endereço inicial das strings das teclas de função, na Área de Trabalho. Em seguida, caracteres seqüenciais serão tomados da string e colocados em KEYBUF (0F55H), até que o terminador zero seja atingido.

Endereço ..... 0EF1H

Esta rotina é utilizada para atualizar a entrada de um dispositivo em TRPTBL quando este tiver produzido uma interrupção de programa BASIC. Na entrada, o par de registradores HL apontará para o byte de status do dispositivo na tabela. Primeiro será verificado o bit 0 do byte de status. Se o dispositivo não estiver na condição "ON", a rotina terminará sem nenhuma ação. Bit 2, o indicador de evento, então, será verificado. Se este já estiver ligado, a rotina terminará, caso contrário o bit será ligado para indicar que ocorreu um evento. Bit 1, o indicador "STOP", será, então, verificado. Caso o dispositivo esteja interrompido a rotina terminará sem nenhuma ação adicional. Caso contrário, ONGSBF será incrementado para sinalizar a Ronda de Execução do Interpretador que o evento será processado.

Endereço ..... 0F06H

Esta seção do decodificador de tecla processa, apenas, a tecla HOME. O estado da tecla SHIFT será determinado por meio da linha 6 de NEWKEY e o código de tecla para HOME (OBH) ou CLS (OCH) será colocado em KEYBUF (0F55H), conforme o caso.

Endereço ..... 0F10H

Esta seção do decodificador do teclado processa as teclas de número 30H a 57H, exceto as teclas CAP, F1 a F5, STOP e HOME. O número da tecla será, simplesmente, utilizado para procurar o código da tecla na tabela em 1033H e este será colocado em KEYBUF(0F55H).

Endereço . . . . . 0F1FH

Esta seção do decodificador do teclado processa a tecla DEAD, encontrada nas máquinas MSX européias. Nas máquinas inglesas a tecla da linha 2 coluna 5, sempre gera o código da tecla de libra(9CH), mostrado na tabela em 0DA5H. Nas máquinas européias esta tabela terá o código de tecla FFH nesta posição. Este código de tecla serve, apenas de indicador para mostrar que a próxima tecla pressionada, caso seja uma vogal, deverá ser modificada para produzir um caractere gráfico acentuado.

A situação das teclas SHIFT e CODE será determinada por meio da linha 6 de NEWKEY e um dos seguintes valores será colocado em KANAST: 01H=DEAD, 02H=DEAD+SHIFT, 03H=DEAD+CODE, 04H=DEAD+SHIFT+CODE.

Endereço . . . . . 0F36H

Esta seção do decodificador do teclado processa a tecla CAP. O estado atual de CAPST será invertido e o controle irá para a rotina padrão CHGCAP.

Endereço . . . . . 0F3DH  
Nome . . . . . CHGCAP  
Entrada . . . . . A=ON/OFF(chave)  
Saída . . . . . Nenhuma  
Modifica . . . . . AF

Rotina padrão para ligar e desligar o LED Caps Lock, conforme determinado pelo conteúdo do registrador A:00H=ligado, NZ=desligado. O LED será modificado utilizando-se o recurso de ligar/desligar bits na Porta de Modo PPI. Como CAPST não será alterada, esta rotina não afetará os caracteres produzidos pelo teclado.

Endereço . . . . . 0F46H

Esta seção do decodificador do teclado processa a tecla STOP. O estado da tecla CTRL é determinado por meio da linha 6 de NEWKEY e o código de tecla para STOP(04H) ou CTRL/STOP(03H) produzido conforme apropriado. Caso o código CTRL/STOP seja produzido, ele será copiado em INTFLG, afim de ser utilizado pela rotina padrão ISCNTC e depois colocado em KEYBUF(0F55H). Se o código STOP for produzido também será copiado em INTFLG, mas não será colocado em KEYBUF. Em vez disto, apenas será gerado um click(0F64H). Isto significa que uma aplicação não poderá ler o código da tecla STOP por meio das rotinas padrões BIOS em ROM.

Endereço ..... 0F55H

Esta seção do decodificador do teclado colocará um código de tecla em KEYBUF e irá gerar um click audível. O endereço correto no buffer do teclado será lido de PUTPNT e o código colocado ali. O endereço será, então, incrementado(105BH). Caso tenha dado a volta e alcançado GETPNT, a rotina terminará sem nenhuma ação adicional, pois o buffer do teclado estará lotado. Caso contrário, PUTPNT será atualizado com o novo endereço.

CLIKSW e CLIKFL serão verificados para determinar se um click é necessário. CLIKSW é uma tecla de ativação/desativação geral ao passo que CLIKFL será utilizada para impedir clicks múltiplos quando as teclas de função forem operadas. Presumindo que um click será requerido, a saída da Tecla Click será estabelecida via Porta de Modo PPI e, após um atraso de 50 uS, o controle irá para a rotina padrão CHGSND.

Endereço ..... 0F7AH  
 Nome ..... CHGSND  
 Entrada ..... A=Tecla ON/OFF  
 Saída ..... Nenhuma  
 Modifica ..... AF

Rotina padrão para ligar ou desligar a saída Key Click pela Porta de Modo PPI: 00H=desligar, NZ=ligar. Esta saída de áudio tem um acoplamento AC, de modo que polaridades absolutas não deverão ser levadas muito a sério.

Endereço ..... 0F83H

Esta seção do decodificador de teclado processa as teclas de número 00H até 2FH. O estado das teclas SHIFT, GRPH e CODE será determinado por meio da linha 6 de NEWKEY e combinado com o número da tecla, afim de formar um endereço de consulta na tabela em 0DA5H. O código da tecla será depois tomado da tabela. Se for zero, a rotina terminará sem nenhuma ação adicional; se for FFH o controle será transferido ao processador da tecla DEAD(0F1FH). Se o código estiver na faixa 40H a 5FH ou 60H a 7FH e a tecla CTRL pressionada, o código de controle correspondente será colocado em KEYBUF(0F55H). Se o código estiver na faixa 01H a 1FH então, um código de cabeçalho gráfico (01H) será, primeiramente, colocado em KEYBUF(0F55H), seguido pelo código com um acréscimo de 40H. Se o código estiver na faixa 61H a 7BH e CAPST estiver indicando "caps lock ativado" o código será convertido para maiúsculas pela subtração de 20H. Considerando que KANAST contém zero, como sempre terá em máquinas inglesas, o código será colocado em KEYBUF(0F55H) e a rotina terminará. Nas

máquinas MSX européias, com uma tecla DEAD, em lugar da libra, os códigos das teclas que correspondem às vogais a, e, i, o, u poderão ter modificações adicionais em códigos gráficos.

Endereço ..... 0FC3H

Esta seção do decodificador do teclado processa as cinco teclas de função. O estado da tecla SHIFT será examinado por meio da linha 6 de NEWKEY e será acrescentado cinco ao número da tecla, caso esta esteja pressionada. O controle será, então, transferido para 0EC5H afim de completar o processamento.

Endereço ..... 1021H

Esta rotina realiza uma busca na tabela 1B97H, afim de determinar a que grupo de teclas o número da tecla fornecido no registrador C pertence. O endereço associado em seguida, será tomado da tabela e o controle será transferido àquela seção do decodificador do teclado. Observe que a tabela em si estará colocada no meio da rotina padrão OUTDO, como resultado das modificações feitas na ROM japonesa.

Endereço ..... 1033H

Esta tabela contém os códigos correspondentes às teclas de número 30H a 57H, excluindo-se as teclas especiais CAP, F1 a F5, STOP e HOME. Uma entrada zero, na tabela, significa que nenhum código de tecla será produzido quando aquela tecla for pressionada:

|     |     |     |     |     |     |     |     |     |          |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| 00H | 00H | 00H | 00H | 00H | 00H | 00H | 00H | 00H | Linha 6  |
| 0DH | 1BH | 0BH | 00H | 07H | 1BH | 00H | 00H |     | Linha 7  |
| 1CH | 1FH | 1EH | 1DH | 7FH | 12H | 0CH | 20H |     | Linha 8  |
| 00H | 00H | 00H | 00H | 2FH | 2AH | 2DH | 2BH |     | Linha 9  |
| 00H | 00H | 00H | 00H | 00H | 00H | 00H | 00H |     | Linha 10 |
| 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |     | Coluna   |

Endereço ..... 105BH

Esta rotina, simplesmente, zera KANAST e depois transfere o controle a 10C2H.

Endereço ..... 1061H

Esta tabela contém os caracteres gráficos que substituem as vogais a, e, i, o, u nas máquinas européias.



Endereço . . . . . 10C2H

Esta rotina incrementa o apontador de buffer do teclado, PUTPNT ou GETPNT, fornecido no par de registradores HL. Se o apontador exceder o fim do buffer do teclado, ele voltará ao começo.

Endereço . . . . . 10CBH  
 Nome . . . . . CHGET  
 Entrada . . . . . Nenhuma  
 Saída . . . . . A=Caractere do teclado  
 Modifica . . . . . AF, EI

Rotina padrão para pegar um caractere do buffer do teclado. O buffer será primeiro verificado para ver se já contém um caractere(0D6AH). Em caso negativo, o cursor será ligado(09DAH), o buffer será verificado diversas vezes até que apareça um caractere(0D6AH) e, em seguida, o cursor será desligado(0A27H). O caractere será tomado do buffer utilizando GETPNT que depois será incrementado(10C2H).

Endereço . . . . . 10F9H  
 Nome . . . . . CKCNTC  
 Entrada . . . . . Nenhuma  
 Saída . . . . . Nenhuma  
 Modifica . . . . . AF, EI

Rotina padrão para verificar se as teclas CTRL-STOP ou STOP foram pressionadas. É utilizada pelo Interpretador BASIC dentro de comandos cujo processamento seja intenso, como "WAIT" e "CIRCLE", para verificar uma possível interrupção de programa. Primeiramente, o par de registradores HL é zerado e o controle transferido à rotina padrão ISCNTC. Quando o Interpretador está sendo executado, o par de registradores HL, normalmente, contém o endereço do caractere atual do texto do programa BASIC. Caso ISCNTC seja terminado com CTRL-STOP, este endereço será colocado em OLDTXT pelo manipulador do comando "STOP"(63E6H), a fim de ser utilizado por um comando "CONT" posterior. Zerar o par de registradores HL, antecipadamente, sinaliza ao manipulador de "CONT" que ocorreu um encerramento dentro de um comando e ele indicará um erro "Can't CONTINUE", caso a continuação seja tentada.

Endereço ..... 1102H  
Nome ..... WRTPSG  
Entrada ..... A=Número do registrador, E=Byte de dados  
Saída ..... Nenhuma  
Modifica ..... EI

Rotina padrão para escrever um byte de dados em qualquer um dos dezesseis registradores PSG. O número de seleção de registrador será escrito na Porta de Endereço PSG e o byte de dado escrito na Porta de Escrita de Dados do PSG.

Endereço ..... 110EH  
Nome ..... RDPSG  
Entrada ..... A=Número do registrador  
Saída ..... A=Byte de dados lido do PSG  
Modifica ..... A

Rotina padrão para ler um byte de dados de qualquer um dos dezesseis registradores PSG. O número de seleção do registrador será escrito na Porta de Endereço PSG e o byte de dados será lido da Porta de Leitura de Dados do PSG.

Endereço ..... 1113H  
Nome ..... BEEP  
Entrada ..... Nenhuma  
Saída ..... Nenhuma  
Modifica ..... AF, BC, E, EI

Rotina padrão para produzir um som via PSG. O canal A será preparado para enviar um Som de 1316Hz, depois ativado com uma amplitude sete. Depois de um atraso de 40 mS o controle será transferido à rotina padrão GICINI para reinicializar o PSG.

Endereço ..... 113BH

Esta rotina é utilizada pelo manipulador de interrupção para atender a uma fila musical. Como existem três delas, cada uma alimentando um canal PSG, a fila que deverá ser atendida é especificada fornecendo seu número no registrador A: 0=VOICAQ, 1=VOICBQ e 2=VOICCCQ.

Cada string em uma declaração "PLAY" será traduzida em uma série de pacotes de dados pelo Intérprete BASIC. Estes serão colocados em uma fila apropriada seguida

por um byte de final de dados(FFH). A tarefa de retirar os pacotes da fila, decodificá-los e ativar o PSG, ficará por conta do manipulador de interrupção. O Interpretador ficará livre para seguir imediatamente para o próximo comando, sem ter que esperar o término das notas.

Os dois primeiros bytes de qualquer pacote especificam o número e a duração de bytes. Os três bits mais significativos do primeiro byte especificam o número de byte, no pacote, que vem após o cabeçalho. O restante do cabeçalho especifica a duração do evento, em unidades de 20mS. Esta duração determina o tempo que transcorrerá até que o próximo pacote seja lido da fila.

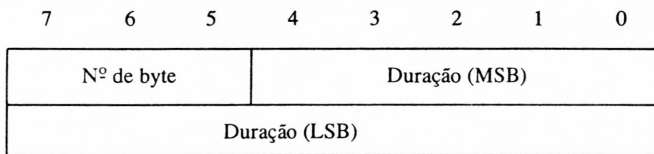
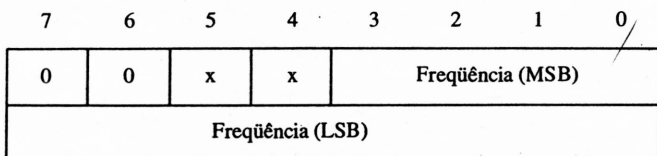
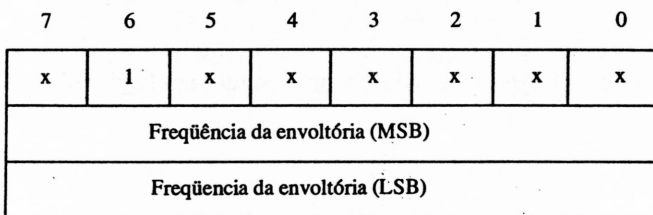


Figura 37 Cabeçalho do pacote

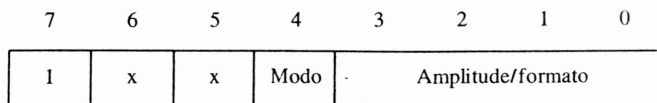
O cabeçalho do pacote poderá ser seguido por zero ou mais blocos, em qualquer ordem, contendo informação de frequência ou amplitude:



Bloco de frequência



Bloco de envoltória



Bloco de amplitude

**Figura 38** Tipos de blocos de pacotes.

A rotina localiza, primeiro, o contador de duração atual no buffer de voz relevante (VCBA, VCBB ou VCBC), por meio da rotina padrão GETVCP e o decrementa. Caso o contador tenha atingido zero, então, o próximo pacote terá que ser lido da fila; caso contrário a rotina terminará.

O número da fila será colocado em QUEUEN e um byte lido da fila (11E2H). O byte, em seguida, será verificado para sabermos se é o byte de fim de dados (FFH), e em caso afirmativo a fila terminará (11B0H). Caso contrário, o número de bytes será colocado no registrador C e MSB da duração no buffer de voz relevante. O segundo byte será lido (11E2H) e o LSB da duração será colocado no buffer de voz relevante. Em seguida, o número de bytes será examinado, e caso não haja nenhum byte, depois do cabeçalho do pacote, a rotina terminará. Caso contrário, bytes sucessivos serão lidos da fila, com a tomada da ação apropriada, até que a contagem de byte seja esgotada.

Se um bloco de frequência for encontrado, um segundo byte será lido e dois bytes serão escritos nos Registradores PSG 0 e 1, 2 e 3 ou 4 e 5, dependendo do número da fila.

Se um bloco de amplitude for encontrado, os bits de Modo e Amplitude serão escritos nos Registradores PSG 8, 9 ou 10, dependendo do número da fila. Se o bit de Modo for 1, selecionando amplitude modulada em vez fixa, então o byte também será escrito no Registrador 13 do PSG, para estabelecer a forma da envoltória.

Se um bloco de envoltória for encontrado, ou se o bit 6 de um bloco de amplitude for ligado, mais dois bytes serão lidos da fila e escritos nos Registradores 11 e 12 do PSG.

Endereço . . . . . 11B0H

Esta rotina será utilizada quando uma marca de término de dados (FFH) for encontrada em uma das três filas musicais. Um valor de amplitude zero será escrito no Registrador 8, 9 ou 10 do PSG, dependendo do número da fila, para silenciar o canal. O bit do canal em MUSICF será desligado e o controle irá para a rotina padrão STRTMS.

Endereço ..... 11C4H  
 Nome ..... STRTMS  
 Entrada ..... Nenhuma  
 Saída ..... Nenhuma  
 Modifica ..... AF, HL

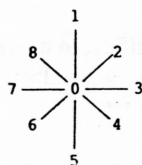
Rotina padrão utilizada pelo manipulador do comando "PLAY" para dar início ao desempilhamento das filas musicais pelo manipulador de interrupção. MUSICF será examinado primeiro. Se algum canal já estiver em processamento, a rotina terminará sem nenhuma ação. PLYCNT será, então, decrementado. Se não houver mais strings de "PLAY" enfileiradas a rotina terminará caso contrário, os três contadores de duração, em VCBA, VCB B e VCBC serão colocados em 0001H, de modo que o primeiro pacote do novo grupo será retirado da fila na próxima interrupção. e MUSICF será colocado em 07H para ativar todos os três canais.

Endereço ..... 11E2H

Esta rotina carrega o registrador A com o número da fila atual, de QUEUEN, e depois lê um byte da fila(14ADH).

Endereço ..... 11EEH  
 Nome ..... GTSTCK  
 Entrada ..... A=Identificação do joystick (0, 1 ou 2)  
 Saída ..... A=Código de posição do joystick  
 Modifica ..... AF, B, DE, HL, EI

Rotina padrão para ler a posição de um joystick ou as quatro teclas de controle do cursor. Se a identificação fornecida for zero, o estado das teclas de cursor será lido pela Porta B da PPI(1226H) e convertido para um código de posição, utilizando a tabela de consulta em 1243H. Caso contrário, o conector do joystick 1 ou 2 será lido (120CH), e os quatro bits de direcionamento convertidos em um código de posição, utilizando a tabela de consulta em 1233H. Os códigos de posição devolvidos são:



Endereço . . . . . 120CH

Esta rotina lê do joystick especificado pelo conteúdo do registrador A: 0=Conector 1, 1=Conector 2. O conteúdo atual do Registrador 15 do PSG é lido e depois escrito com o bit de Seleção de Joystick, adequadamente, preparado. O Registrador 14 do PSG será, então, copiado no registrador A(110CH) e a rotina terminará.

Endereço . . . . . 1226H

Esta rotina lê a linha 8 da matriz do teclado. O conteúdo atual da Porta C do PPI é lido e depois reescrito com os quatro bits de Seleção de Linha do Teclado estabelecidos para a linha 8. As entradas da coluna são, em seguida, lidas no registrador A da Porta B do PPI.

Endereço . . . . . 1253H

Nome . . . . . GTTRIG

Entrada . . . . . A=Identificação do gatilho(0,1,2,3 ou 4)

Saída . . . . . A=Código de status

Modifica . . . . . AF, BC, EI

Rotina padrão para verificar o gatilho do joystick ou status da tecla de espaço. Se o ID fornecido for zero; a linha 8 da matriz do teclado será lida(1226H) e convertida para um código do status. Caso contrário, o conector 1 ou 2 do joystick será lido (120CH) e convertido para um código de status. As Identificações dos gatilhos são:

0=Tecla de Espaço

1=Joystick 1, gatilho A

2=Joystick 2, gatilho A

3=Joystick 1, gatilho B

4=Joystick 2, gatilho B

O valor retornado é FFH, se o gatilho relevante estiver pressionado, e zero, em caso contrário.

Endereço . . . . . 1273H

Nome . . . . . GTPDL

Entrada . . . . . A=Identificação do paddle (1 a 12)

Saída . . . . . A=Valor do paddle (0 a 255)

Modifica . . . . . AF, BC, DE, EI

Rotina padrão para ler o valor de qualquer paddle ligado a um conector de joystick. Cada uma das seis linhas de entrada (quatro direções mais dois gatilhos) de um

conector pode corresponder a um paddle, de modo que seja possível um total de doze paddles. Os paddles ligados ao conector do joystick têm identificadores de entrada 1, 3, 5, 7, 9 e 11. Aqueles ligados ao conector 2 têm identificadores de entrada 2, 4, 6, 8, 10 e 12. Cada paddle é, basicamente, um gerador de pulso monoestável, sendo a largura do pulso controlado por um resistor variável. Um pulso inicial é enviado ao conector do joystick especificado pelo Registrador 15 do PSG. É, em seguida, mantida uma contagem de quantas vezes o Registrador 14 do PSG tem que ser lido até que a entrada relevante caia. Cada incremento de uma unidade representa um período aproximado de 12uS em uma máquina MSX, com um estado de espera (Wait state).

|                    |                            |
|--------------------|----------------------------|
| Endereço . . . . . | 12ACH                      |
| Nome . . . . .     | CTPAD                      |
| Entrada . . . . .  | A=Código de função (0 a 7) |
| Saída . . . . .    | A=Status ou valor          |
| Modifica . . . . . | AF, BC, DE, HL, EI         |

Rotina padrão para acessar um digitalizador ligado a um dos conectores de joystick. Códigos de função disponíveis para o conector 1 do joystick são:

- 0=Retorna status de atividade
- 1=Devolve a coordenada "X"
- 2=Devolve a coordenada "Y"
- 3=Devolve o status da tecla

Os códigos de função 4 a 7 têm o mesmo efeito com respeito ao conector 2 do joystick. A função Status de Atividade devolve FFH se o Digitalizador estiver sendo tocado, e zero em caso contrário. A função de Status de Tecla devolve FFH, se a tecla estiver sendo pressionada, e zero em caso contrário. As duas funções de solicitação de coordenadas devolvem as coordenadas da última posição tocada. Estas coordenadas são, realmente, armazenadas nas variáveis da Área de Espaço de Trabalho PADX e PADY, quando uma chamada com os códigos de função 0 ou 4 detectam atividade. Observe que estas variáveis são compartilhadas pelos dois conectores de joystick.

|                    |                          |
|--------------------|--------------------------|
| Endereço . . . . . | 1384H                    |
| Nome . . . . .     | STMOTR                   |
| Entrada . . . . .  | A=Código ON/OFF do motor |
| Saída . . . . .    | Nenhuma                  |
| Modifica . . . . . | AF                       |

Rotina padrão para ligar ou desligar o relê do motor do cassete pela porta C do PPI: 00H=desligado, 01H=ligado, FFH=estado de corrente reversa.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 1398H   |
| Nome . . . . .     | NMI     |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | Nenhuma |

Rotina padrão para processar uma Interrupção Não-mascarável do Z80 (NMI). Em uma máquina MSX padrão, retorna sem ação.

|                    |            |
|--------------------|------------|
| Endereço . . . . . | 139DH      |
| Nome . . . . .     | INIFNK     |
| Entrada . . . . .  | Nenhuma    |
| Saída . . . . .    | Nenhuma    |
| Modifica . . . . . | BC, DE, HL |

Rotina padrão para inicializar as dez strings das teclas de função com seus conteúdos padrões. Os cento e sessenta bytes de dados, iniciando em 13A9H, são copiados no buffer FNKSTR na Área do Espaço de Trabalho.

|                    |       |
|--------------------|-------|
| Endereço . . . . . | 13A9H |
|--------------------|-------|

Esta área contém as strings de inicialização para as dez teclas de função. Cada string tem dezesseis caracteres de comprimento. Posições não utilizadas contêm zeros:

| <u>F1 a F5</u> | <u>F6 a F10</u>  |
|----------------|------------------|
| color          | color 15, 4,4 CR |
| auto           | cload"           |
| goto           | contCR           |
| list           | list. CR UP UP   |
| run CR         | run CR           |

|                    |                                            |
|--------------------|--------------------------------------------|
| Endereço . . . . . | 1449H                                      |
| Nome . . . . .     | RDVDP                                      |
| Entrada . . . . .  | Nenhuma                                    |
| Saída . . . . .    | A=Conteúdo do Registrador de Status do VDP |
| Modifica . . . . . | A                                          |

Rotina padrão que permite a leitura do conteúdo do Registrador de Status do VDP pela Porta de Comando. Observe que a leitura do Registrador de Status do VDP limpará os flags associados. Isso poderá afetar o manipulador de interrupção.



|                    |                                                |
|--------------------|------------------------------------------------|
| Endereço . . . . . | 144CH                                          |
| Nome . . . . .     | RSLREG                                         |
| Entrada . . . . .  | Nenhuma                                        |
| Saída . . . . .    | A=Conteúdo do Registrador de Conector primário |
| Modifica . . . . . | A                                              |

Rotina padrão que permite a leitura do Registrador do Conector Primário pela Porta A do PPI.

|                    |                    |
|--------------------|--------------------|
| Endereço . . . . . | 144FH              |
| Nome . . . . .     | WSLREG             |
| Entrada . . . . .  | A=Valor a escrever |
| Saída . . . . .    | Nenhuma            |
| Modifica . . . . . | Nenhuma            |

Rotina padrão que permite a escrita do Registrador do Conector Primário pela Porta A do PPI.

|                    |                                 |
|--------------------|---------------------------------|
| Endereço . . . . . | 1452H                           |
| Nome . . . . .     | SNSMAT                          |
| Entrada . . . . .  | A=Um número de linha do teclado |
| Saída . . . . .    | A=Colunas da linha do teclado   |
| Modifica . . . . . | AF, C, EI                       |

Rotina padrão para ler uma linha completa da matriz do teclado. A Porta C do PPI é lida e reescrita com o número da linha, ocupando os quatro bits de Seleção de Linha do Teclado. Em seguida, a Porta B do PPI é lida no registrador A (as oito colunas da linha). As outras quatro saídas de controle da Porta C do PPI não são afetadas por esta rotina.

|                    |                                               |
|--------------------|-----------------------------------------------|
| Endereço . . . . . | 145FH                                         |
| Nome . . . . .     | ISFLIO                                        |
| Entrada . . . . .  | Nenhuma                                       |
| Saída . . . . .    | Indicador NZ se está ocorrendo E/S de arquivo |
| Modifica . . . . . | AF                                            |

Rotina padrão para verificar se o Interpretador BASIC está direcionando sua saída ou entrada por algum buffer de E/S. Isto é determinado examinando-se PTRFIL que normalmente é zero, mas conterá o endereço do FCB (Bloco de Controle de Arquivo) de

um buffer sempre que comandos como “PRINT#1”, “INPUT#1” etc., estiverem sendo executados pelo Interpretador.

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| Endereço . . . . . | 146AH                                                                    |
| Nome . . . . .     | DCOMPR                                                                   |
| Entrada . . . . .  | HL, DE                                                                   |
| Saída . . . . .    | Indicador NC se HL > DE, Indicador Z<br>se HL=DE, Indicador C se HL < DE |
| Modifica . . . . . | AF                                                                       |

Rotina padrão utilizada pelo Interpretador BASIC para verificar os valores relativos dos pares de registradores HL e DE.

|                    |                              |
|--------------------|------------------------------|
| Endereço . . . . . | 1470H                        |
| Nome . . . . .     | GETVCP                       |
| Entrada . . . . .  | A=Número da voz (0,1,2)      |
| Saída . . . . .    | HL=Endereço no buffer de voz |
| Modifica . . . . . | AF, HL                       |

Rotina padrão que devolve o endereço do byte 2 no buffer de voz especificada (VCBA, VCBB ou VCBC).

|                    |                              |
|--------------------|------------------------------|
| Endereço . . . . . | 1474H                        |
| Nome . . . . .     | GETVC2                       |
| Entrada . . . . .  | L=Número do byte (0 a 36)    |
| Saída . . . . .    | HL=Endereço no buffer de voz |
| Modifica . . . . . | AF, HL                       |

Rotina padrão que devolve o endereço de qualquer byte no buffer de voz (VCBA, VCBB e VCBC), especificada pelo número da voz em VOICEN.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 148AH   |
| Nome . . . . .     | PHYDIO  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | Nenhuma |

Rotina padrão utilizada pelo BASIC de Disco. Simplesmente volta, em máquinas MSX padrão.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 148EH   |
| Nome . . . . .     | FORMAT  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | Nenhuma |

Rotina padrão para ser utilizada pelo BASIC de Disco. Simplesmente volta, em máquinas MSX padrão.

|                    |                                      |
|--------------------|--------------------------------------|
| Endereço . . . . . | 1492H                                |
| Nome . . . . .     | PUTQ                                 |
| Entrada . . . . .  | A=Número da fila, E=Byte de dado     |
| Saída . . . . .    | Indicador Z, se a fila estiver cheia |
| Modifica . . . . . | AF, BC, HL                           |

Rotina padrão para colocar um byte de dados em uma das três filas musicais. Os endereços de leitura e escrita na fila são obtidos da tabela QUETAB(14FAH). O endereço de escrita é incrementado, temporariamente, e comparado ao endereço de leitura, e, se forem iguais, a rotina terminará, pois a fila estará cheia. Caso contrário, o endereço da fila é tomado de QUETAB e o endereço de escrita acrescentado a ela. O byte de dado é colocado na fila nesta posição, o endereço de escrita é incrementado e a rotina termina. Observe que as filas musicais são circulares, de modo que elas voltarão ao início se os apontadores de escrita ou leitura atingirem a última posição na fila.

|                    |       |
|--------------------|-------|
| Endereço . . . . . | 14ADH |
|--------------------|-------|

Esta rotina é utilizada pelo manipulador de interrupção para ler um byte de uma das três filas musicais. O número da fila será fornecido no Registrador A, o byte de dado devolvido no Registrador A e a rotina devolverá o Indicador Z, se a fila estiver vazia. Os endereços de leitura e escrita na fila são obtidos da tabela QUETAB(14FAH). Se o indicador de devolução estiver ativo, o byte de dado será tomado de QUEBAK e a rotina terminará(14D1H). Este recurso não é utilizado nas versões atuais da ROM do MSX. O endereço de escrita é, em seguida, comparado com o endereço de leitura, e, se forem iguais, a rotina terminará, pois a fila estará vazia. Caso contrário, o endereço da fila será tomado de QUETAB e o endereço de leitura acrescentado a ele. O byte de dados será lido desta posição da fila, o endereço de leitura será incrementado e a rotina terminará.

Endereço . . . . . 14DAH

Esta rotina é utilizada pela rotina padrão GICINI para inicializar um bloco de controle da fila em QUETAB. O bloco de controle é localizado em QUETAB(1504H) e os bytes de leitura, escrita e o indicador de devolução são zerados. O byte de tamanho é obtido do registrador B e o endereço da fila do par de registradores DE.

Endereço . . . . . 14EBH  
Nome . . . . . LFTQ  
Entrada . . . . . A=Número da fila  
Saída . . . . . HL=Espaço livre deixado na fila  
Modifica . . . . . AF, BC, HL

Rotina padrão que devolve o número de bytes livres em uma fila musical. Os endereços de leitura e escrita da fila são tomados de QUETAB(14FAH) e o espaço livre determinado, subtraindo-se "escrita" de "leitura".

Endereço . . . . . 14FAH

Esta rotina devolve os parâmetros de controle de uma fila de QUETAB. O número da fila é fornecido pelo registrador A. O bloco de controle primeiro é localizado em QUETAB(1504H), a posição de escrita é, em seguida, colocada no registrador B, a posição de leitura no registrador C e o indicador de devolução no registrador A.

Endereço . . . . . 1504H

Esta rotina localiza o bloco de controle de uma fila em QUETAB. O número da fila é fornecido pelo registrador A e o endereço do bloco de controle devolvido no par de registradores HL. O número da fila é, simplesmente, multiplicado por seis, uma vez que há seis bytes por bloco, e acrescentado ao endereço de QUETAB, mantido em QUEUES.

Endereço . . . . . 1510H  
Nome . . . . . GRPPRT  
Entrada . . . . . A=Caractere  
Saída . . . . . Nenhuma  
Modifica . . . . . EI

Rotina padrão para apresentar um caractere na tela no Modo Gráfico ou no Modo Multicolorido. É, funcionalmente, equivalente à rotina padrão CHPUT.

A rotina padrão CNVCHR é primeiro utilizada para verificar se o caractere é gráfico; se o caractere for um código de cabeçalho(01H), então, a rotina terminará sem nenhuma ação. Se o caractere for resultado de uma conversão gráfica, então, a seção de decodificação do código de controle será pulada. Caso contrário, o caractere será verificado para ver se é um código de controle. Apenas o código CR(ODH) será reconhecido (157EH), e todos os caracteres menores do que 20H serão ignorados.

Considerando que o caractere possa ser apresentado, sua imagem de pixels, de oito bytes, será copiada do conjunto de caracteres da ROM no buffer PATWRK(0752H) e FORCLR será copiado em ATRBYT para estabelecer sua cor. As coordenadas gráficas atuais serão, em seguida, tomadas de GRPACX e GRPACY e utilizadas para estabelecer o endereço físico do pixel atual, via rotinas padrões SCALXY e MAPXYC.

A imagem de oito bytes em PATWRK é processada um byte por vez. No início de cada byte o endereço físico do pixel atual é obtido e salvo, via rotina padrão FETCHC. Os oito bits são, em seguida, examinados, um por vez. Se o bit for 1, o pixel associado será ligado pela rotina padrão SETC, e se for 0, não será tomada nenhuma ação. Após cada bit, o endereço físico do pixel atual é movido para a direita(16ACH). Quando o byte estiver terminado, ou o lado direito da tela for atingido, o endereço físico inicial do pixel atual será restaurado e deslocado para baixo uma posição, pela rotina padrão TDOWNC.

Quando a imagem estiver completa, ou a parte inferior da tela for atingida, GRPACX será atualizado. No Modo Gráfico o seu valor será aumentado de oito, e no Modo Multicolorido, de trinta e dois. Se CRPACX exceder 255, o lado direito da tela, uma operação CR será executada(157EH).

Endereço . . . . . 157EH

Esta rotina executa a operação CR para a rotina padrão GRPPRT, sendo que este código funciona como um CR, LF combinado. GRPACX é zerado e, dependendo do modo da tela, o valor oito ou trinta e dois é acrescentado a GRPACY. Depois, se GRPACY exceder 191, o limite inferior da tela, ele será zerado.

GRPACX e GRPACY podem ser manuseados diretamente, por um programa aplicativo, para compensar o número limitado de funções de controle disponíveis.

Endereço ..... 1599H  
 Nome ..... SCALXY  
 Entrada ..... BC=coordenada x, DE=coordenada y  
 Saída ..... Indicador NC, em caso de corte ("clipping")  
 Modifica ..... AF

Rotina padrão para cortar um par de coordenadas gráficas, se houver necessidade. O interpretador poderá produzir coordenadas na faixa -32768 até + 32767, embora isto exceda em muito o tamanho real da tela. Esta rotina modifica valores excessivos de coordenadas para que caibam dentro da faixa realizável fisicamente. Se a coordenada x for maior do que 255, ela será colocada em 255, e, se a coordenada y for maior do que 191, ela será colocada em 191. Se uma das coordenadas for negativa (maior do que 7FFFH) ela será colocada em zero. Finalmente se a tela estiver no Modo Multicolorido, ambas as coordenadas serão divididas por quatro, conforme requerido pela rotina padrão MAPXYC.

Endereço ..... 15D9H

Esta rotina é utilizada para verificar o modo da tela atual, devolvendo o Indicador Z se a tela estiver no Modo Gráfico.

Endereço ..... 15DFH  
 Nome ..... MAPXYC  
 Entrada ..... BC=coordenada x, DE=coordenada y  
 Saída ..... Nenhuma  
 Modifica ..... AF, D, HL

Rotina padrão para converter um par de coordenadas gráficas no endereço físico atual de pixel. A posição da Tabela de Imagens de Caracteres do byte contendo o pixel será colocada em CLOC. A máscara de byte, identificando o pixel dentro deste byte, será colocada em CMASK. Métodos de conversão ligeiramente diferentes serão utilizados para Modo Gráfico e Modo Multicolorido. Os programas equivalentes em BASIC são:

```
5 ' MODO GRAFICO
7 '
10 INPUT "Coordenadas X,Y":X,Y
20 E=(Y\8)*256+(Y AND 7)+(X AND &HFB)
30 PRINT "ENDERECO=";HEX$(BASE(12)+E); "H ";
40 RESTORE 100
50 FOR N=0 TO (X AND 7):READ M$:NEXT N
60 PRINT "MASCARA=";M$:PRINT
70 GOTO 10
100 DATA 10000000
110 DATA 01000000
120 DATA 00100000
130 DATA 00010000
140 DATA 00001000
150 DATA 00000100
160 DATA 00000010
170 DATA 00000001
```

```

5  ? MODO MULTICOLORIDO
7  ?
10 INPUT "Coordenadas X,Y"; X,Y
20 X=X\4:Y=Y\4
30 E=(Y\8)*256+(Y AND 7)+(X*4 AND &HFB)
40 PRINT "ENDERECO=";HEX$(BASE(17)+E); "H ";
50 IF X MOD 2=0 THEN M$="11110000"
   ELSE M$="00001111"
60 PRINT "MASCARA=";M$
70 GOTO 10

```

A faixa de entrada permissível para os dois programas é  $x=0$  até 255 e  $y=0$  até 191. As instruções "DATA", no programa de Modo Gráfico, correspondem às oito máscaras de byte começando em 160BH na ROM do MSX. A linha 20 no Modo Multicolorido realmente corresponde à divisão por quatro na rotina padrão SCALXY. Ela é incluída tornando o sistema de coordenadas consistente para os dois programas.

|                |                 |
|----------------|-----------------|
| Endereço ..... | 1639H           |
| Nome .....     | FETCHC          |
| Entrada .....  | Nenhuma         |
| Saída .....    | A=CMASK,HL=CLOC |
| Modifica ..... | A, HL           |

Rotina padrão para devolver o endereço físico do pixel atual, com o par de registradores HL carregado de CLOC e o registrador A carregado de CMASK.

|                |                  |
|----------------|------------------|
| Endereço ..... | 1640H            |
| Nome .....     | STOREC           |
| Entrada .....  | A=CMASK, HL=CLOC |
| Saída .....    | Nenhuma          |
| Modifica ..... | Nenhuma          |

Rotina padrão para estabelecer o endereço físico do pixel atual. O par de registradores HL é copiado em CLOC e o registrador A é copiado em CMASK.

|                |                                |
|----------------|--------------------------------|
| Endereço ..... | 1647H                          |
| Nome .....     | READC                          |
| Entrada .....  | Nenhuma                        |
| Saída .....    | A=Código da cor do pixel atual |
| Modifica ..... | AF, EI                         |

Rotina padrão para devolver a cor do pixel atual. O endereço físico da VRAM é obtido, primeiro, por meio da rotina padrão FETCHC. Se a tela estiver no Modo Gráfico,

o byte apontado por CLOC será lido da Tabela de Imagens de Caracteres por meio da rotina padrão RDVRM. Em seguida, o bit requerido será isolado por CMASK e utilizado para selecionar os quatro bits superiores ou inferiores da entrada correspondente na Tabela de Cor.

Se a tela estiver no Modo Multicolorido, o byte apontado por CLOC será lido da Tabela de Imagens de Caracteres por meio da rotina padrão RDMRM. Em seguida, CMASK será utilizado para selecionar os quatro bits superiores ou inferiores deste byte. O valor devolvido, em qualquer um dos dois casos, será um código de cor normal do VDP de zero a quinze.

|                    |                                       |
|--------------------|---------------------------------------|
| Endereço . . . . . | 1676H                                 |
| Nome . . . . .     | SETATR                                |
| Entrada . . . . .  | A=Código de cor                       |
| Saída . . . . .    | Indicador C, em caso de código ilegal |
| Modifica . . . . . | Indicadores                           |

Rotina padrão para estabelecer a cor da tinta de gráficos utilizada pelas rotinas padrões SETC e NSETCX. O código da cor, de zero a quinze, é, simplesmente, colocado em ATRBYT.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 167EH   |
| Nome . . . . .     | SETC    |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | AF, EI  |

Rotina padrão para estabelecer qualquer cor ao pixel atual, sendo que código da cor é tomado de ATRBYT. O endereço físico do pixel na VRAM é primeiro obtido pela rotina padrão FETCHC. No Modo Gráfico, tanto a Tabela de Imagens de Caracteres quanto a Tabela de Cores são modificadas(186CH).

No Modo Multicolorido, o byte apontado por CLOC será lida da Tabela de Imagens de Caracteres pela rotina padrão RDVRM. Em seguida, o conteúdo de ATRBYT será colocado nos quatro bits superiores ou inferiores, conforme determinado por CMASK, e o byte será reescrito por meio da rotina padrão WRTVRM.



Endereço ..... 16ACH

Esta rotina move o endereço físico do pixel atual uma posição à direita. Se o lado direito da tela for excedido, ele voltará com o Indicador C e o endereço físico será inalterado. No Modo Gráfico CMASK será deslocado um bit para à direita e, se o pixel ainda continuar no byte, a rotina terminará. Se CLOC estiver na posição de caractere mais à direita (LSB=F8H a FFH), então, a rotina terminará com o Indicador C(175AH). Caso contrário, CMASK será colocado em 80H, o pixel mais à esquerda e 0008H somado a CLOC.

No Modo Multicolorido o controle será transferido para uma rotina separada (1779H).

Endereço ..... 16C5H  
Nome ..... RIGHTC  
Entrada ..... Nenhuma  
Saída ..... Nenhuma  
Modifica ..... AF

Rotina padrão para mover o endereço físico do pixel atual uma posição à direita. No Modo Gráfico, CMASK será primeiro deslocado um bit para direita, e se o pixel continuar dentro do byte a rotina terminará caso contrário, CMASK será colocado em 80H, o pixel mais à esquerda, e 0008H somando a CLOC. Observe que serão produzidos endereços incorretos, se o lado direito da tela for excedido.

No Modo Multicolorido, o controle será transferido para uma rotina separada(178BH).

Endereço ..... 16D8H

Esta rotina move o endereço físico do pixel atual uma posição à esquerda. Se o lado esquerdo da tela estiver excedido, ela devolverá o Indicador C e o endereço físico ficará inalterado. No Modo Gráfico, CMASK será deslocado primeiro um bit à esquerda, e se o pixel continuar dentro do byte a rotina terminará. Se CLOC estiver na célula de caractere mais à esquerda (LSB=00H a 07H), a rotina terminará com o Indicador C(175AH). Caso contrário, CMASK será colocado em 01H, o pixel mais à esquerda e 0008H subtraído de CLOC.

No Modo Multicolorido, o controle será transferido para uma rotina separada(179CH).

|                |         |
|----------------|---------|
| Endereço ..... | 16EEH   |
| Nome .....     | LEFTC   |
| Entrada .....  | Nenhuma |
| Saída .....    | Nenhuma |
| Modifica ..... | AF      |

Rotina padrão para mover o endereço físico do pixel atual uma posição para a esquerda. No Modo Gráfico, CMASK será primeiro deslocado um bit para esquerda, e, se o pixel continuar dentro do byte, a rotina terminará caso contrário, CMASK será colocado em 01H, o pixel mais à esquerda e 0008H subtraído de CLOC. Observe que serão produzidos endereços incorretos, se o lado esquerdo da tela for ultrapassado.

No Modo Multicolorido, o controle será transferido para uma rotina separada (17ACH).

|                |                                      |
|----------------|--------------------------------------|
| Endereço ..... | 170AH                                |
| Nome .....     | TDOWNC                               |
| Entrada .....  | Nenhuma                              |
| Saída .....    | Indicador C, se estiver fora da tela |
| Modifica ..... | AF                                   |

Rotina padrão para mover o endereço físico do pixel atual uma posição para baixo. Se a borda inferior da tela estiver excedida, ela devolverá o Indicador C e o endereço físico será inalterado. No Modo Gráfico, CLOC será incrementado, e, se continuar dentro de um limite de oito bytes, a rotina terminará se CLOC estiver na linha de caracteres mais baixa ( $CLOC \geq 1700H$ ), então a rotina terminará com o Indicador C (1759H). Caso contrário, 00F8H será somado a CLOC.

No Modo Multicolorido, o controle será transferido para uma rotina separada (17C6H).

|                |         |
|----------------|---------|
| Endereço ..... | 172AH   |
| Nome .....     | DOWNC   |
| Entrada .....  | Nenhuma |
| Saída .....    | Nenhuma |
| Modifica ..... | AF      |

Rotina padrão para mover o endereço físico do pixel atual uma posição para baixo. No Modo Gráfico, CLOC será primeiro incrementado, e se continuar dentro de um

limite de oito bytes a rotina terminará. Caso contrário, 00F8H será somado ao CLOC. Observe que endereços incorretos serão produzidos, se o limite inferior da tela for excedido.

No Modo Multicolorido, o controle será transferido para uma rotina separada (17DCH).

|                    |                                      |
|--------------------|--------------------------------------|
| Endereço . . . . . | 173CH                                |
| Nome . . . . .     | TUPC                                 |
| Entrada . . . . .  | Nenhuma                              |
| Saída . . . . .    | Indicador C, se estiver fora da tela |
| Modifica . . . . . | AF                                   |

Rotina padrão para mover o endereço físico do pixel atual uma posição para cima. Se a parte superior da tela for excedida, ela voltará com o Indicador C e o endereço físico será inalterado. No Modo Gráfico, primeiro CLOC será decrementado e se continuar dentro de um limite de oito bytes a rotina terminará. Se CLOC estiver na linha de caracteres superiores (CLOC > 0100H), a rotina terminará com Indicador C. Caso contrário, 00F8H será subtraído de CLOC.

No Modo Multicolorido, o controle será transferido para uma rotina separada (17E3H).

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 175DH   |
| Nome . . . . .     | UPC     |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | AF      |

Rotina padrão para mover o endereço físico do pixel atual uma posição para cima. No Modo Gráfico, CLOC será primeiro decrementado, e se continuar dentro de um limite de oito bytes a rotina terminará. Caso contrário 00F8H será subtraído de CLOC. Observe que serão produzidos endereços incorretos, se a parte superior da tela for ultrapassada.

No Modo Multicolorido, o controle será transferido para uma rotina separada (17F8H).

Endereço . . . . . 1779H

Esta é a versão para o Modo Multicolorido da rotina em 16ACH. É idêntica à versão para o Modo Gráfico, exceto que CMASK será deslocado quatro bits para à direita e passará a ser FOH, caso o limite de uma célula seja cruzado.

Endereço . . . . . 178BH

Esta é a versão para o Modo Multicolorido da rotina padrão RIGHTC. É idêntica à versão para o Modo Gráfico, exceto que CMASK será deslocado quatro bits para à direita e passará a ser FOH, caso o limite de uma célula seja cruzado.

Endereço . . . . . 179CH

Esta é a versão para o Modo Multicolorido da rotina em 16D8H. É idêntica à versão para o Modo Gráfico, exceto que CMASK será deslocado quatro bits para à esquerda e passará a ser OFH, caso o limite de uma célula seja cruzado.

Endereço . . . . . 17ACH

Esta é a versão para o Modo Multicolorido da rotina padrão LEFTC. É idêntica à versão do Modo Gráfico, exceto que CMASK será deslocado quatro posições de bits para à esquerda e passará a ser OFH, caso o limite de uma célula seja cruzado.

Endereço . . . . . 17C6H

Esta é a versão do Modo Multicolorido da rotina padrão TDOWNC. É idêntica à versão para o Modo Gráfico, exceto que o endereço do limite inferior é 0500H, em vez de 17000H. Há um erro nesta rotina que fará com que ela se comporte de forma imprevisível, se MLTCGP (endereço base da Tabela de Imagens de Caracteres) for alterado de seu valor normal de zero. Deveria haver uma instrução EX DE, HL inserida no endereço 17CEH.

Caso a Tabela de Imagens de Caracteres tenha sua base aumentada, a rotina pensará que atingiu a parte inferior da tela, quando na verdade não chegou lá. Esta rotina será utilizada pela declaração "PAINT", de modo que o seguinte programa demonstrará o defeito:

```

10 BASE (17) = &H1000
20 SCREEN 3
30 PSET (200, 0)
40 DRAW "D180L100U180R100"
50 PAINT (150, 90)
60 GOTO 60

```

Endereço ..... 17DCH

Este é a versão para o Modo Multicolorido da rotina padrão DOWNC, e é idêntica à versão para o Modo Gráfico.

Endereço ..... 17E3H

Esta é a versão para o Modo Multicolorido da rotina padrão TUPC. É idêntica à versão para o Modo Gráfico, exceto que possui um erro como o de acima, e desta vez deveria haver uma instrução EX DE, HL inserida no endereço (17EBH).

Se a Tabela de Imagens de Caracteres tiver seu endereço base aumentado, a rotina preverá que está dentro da tabela, quando na verdade atingiu o limite superior da tela. Isto poderá ser demonstrado removendo a parte "R100" da Linha 40, no programa anterior.

Endereço ..... 17F8H

Esta é a versão para o Modo Multicolorido da rotina padrão UPC, e é idêntica à versão para o Modo Gráfico.

```

Endereço ..... 1809H
Nome ..... NSETCX
Entrada ..... HL=Número de pixels a colorir
Saída ..... Nenhuma
Modifica ..... AF, BC, DE, HL, EI

```

Rotina padrão para estabelecer a cor de múltiplos pixels horizontalmente e para a direita, a partir do endereço físico do pixel atual. Apesar de sua função poder ser duplicada pelas rotinas padrões SETC e RIGHTC, o resultado será mais lento. O número de pixels fornecido deverá ser escolhido de tal forma que o limite direito da tela não seja ultrapassado, pois isto produzirá um comportamento anormal. O endereço físico do pixel atual será inalterado por esta rotina.

No Modo Gráfico, CMASK será examinado, primeiro, para determinar o número de pixels à direita dentro da célula do caractere atual. Considerando que o número de pixels fornecido seja suficientemente grande, eles serão, então, ativados(186CH). O número de pixels restante será dividido por oito, para determinar o número de células de caracteres completas. Bytes sucessivos na Tabela de Imagens de Caracteres serão zerados e os bytes correspondentes na Tabela de Cores serão ativados de ATRBYT, para preencher estas células completas. O restante da contagem de ocupação será, em seguida, convertido à uma máscara de bit, utilizando a tabela de sete bytes em 185DH, e estes pixels serão ligados(186CH).

No Modo Multicolorido, o controle será transferido para uma rotina separada (18BBH).

Endereço . . . . . 186CH

Esta rotina pinta até oito pixels de uma célula de uma determinada cor, no Modo Gráfico. ATRBYT contém o código de cor, o par de registradores HL, o endereço do byte relevante na Tabela de Imagens de Caracteres e o registrador A – uma máscara de bit 11100000, por exemplo, onde cada 1 especifica um bit a ser ligado.

Se ATRBYT combina com a cor do pixel 1, existente no byte da Tabela de Cor correspondente, então, cada bit especificado é colocado em 1, no byte da Tabela de Imagens de Caracteres. Caso ATRBYT combine com a cor do pixel 0, existente no byte da Tabela de Cor correspondente, então, cada bit especificado será colocado em 0, no byte da Tabela de Imagens de Caracteres.

Caso ATRBYT não combine com nenhuma das cores existentes, no byte da Tabela de Cores, então, normalmente cada bit especificado será colocado em 1, no byte da Tabela de Imagens de Caracteres, e a cor do pixel 1 será alterada na Tabela de Cores. Entretanto, se isto resultar todos os bits serão colocados em 1, no byte da Tabela de Imagens de Caracteres, então, cada bit especificado será colocado em 0 e a cor do pixel 0 será alterada, no byte da Tabela de Cores.

Endereço . . . . . 18BBH

Esta é a versão em Modo Multicolorido da rotina padrão NSETCX. As rotinas padrões SETC e RIGHTC serão chamadas até que o número de bytes a colorir cheguem a zero. A velocidade de operação não é tão importante no Modo Multicolorido, em virtude da baixa resolução da tela e a conseqüente redução no número de operações requeridas.

Endereço ..... 18C7H  
 Nome ..... GTASPC  
 Entrada ..... Nenhuma  
 Saída ..... DE=ASPCT1, HL=ASPCT2  
 Modifica ..... DE, HL

Rotina padrão que retorna as razões de aspecto default da instrução "CIRCLE".

Endereço ..... 18CFH  
 Nome ..... PNTINI  
 Entrada ..... A=Cor da fronteira (0 a 15)  
 Saída ..... Indicador C, em caso de cor ilegal  
 Modifica ..... AF

Rotina padrão que estabelece a cor de contorno para a declaração "PAINT". No Modo Multicolorido, o código da cor fornecida será colocado em BDRATR. No Modo Gráfico BDRATR será copiado de ATRBYT, pois não é possível termos cores separadas para pintura e contorno.

Endereço ..... 18E4H  
 Nome ..... SCANR  
 Entrada ..... B=Indicador pinta/não pinta, DE=número de pulos  
 Saída ..... DE=Pulos restantes, HL=número de pixels  
 Modifica ..... AF, BC, DE, HL, EI

Rotina padrão utilizada pelo manipulador da instrução "PAINT" para percorrer à direita, a partir do endereço físico do pixel atual até que um código de cor igual a BDRATR seja encontrado, ou até que a borda da tela seja atingida. A posição final torna-se o endereço físico do pixel atual e a posição inicial é devolvida em CSAVEA e CSAVEM. O tamanho da região atravessada será devolvido no par de registradores HL e FILNAM+1. A região atravessada, normalmente, será preenchida, mas isto poderá ser inibido no Modo Gráfico, apenas utilizando-se um parâmetro de entrada zero no registrador B. A contagem de pulos no par de registradores DE determina o número máximo de pixels da cor requerida, que poderão ser ignorados a partir da posição de partida inicial. Este recurso é utilizado pelo manipulador da instrução "PAINT", para procurar lacunas em uma fronteira horizontal que esteja bloqueando seu avanço para cima.

|                    |                      |
|--------------------|----------------------|
| Endereço . . . . . | 197AH                |
| Nome . . . . .     | SCANL                |
| Entrada . . . . .  | Nenhuma              |
| Saída . . . . .    | HL=Contagem de pixel |
| Modifica . . . . . | AF, BC, DE, BL, EI   |

Rotina padrão para procurar para à esquerda, a partir do endereço físico do pixel atual, até que um código de cor igual a BDRATR seja encontrado ou a borda da tela seja atingida. A posição fina torna-se o endereço físico do pixel atual e o tamanho da região percorrida será devolvido no par de registradores HL. A região percorrida será sempre preenchida.

|                    |       |
|--------------------|-------|
| Endereço . . . . . | 19C7H |
|--------------------|-------|

Esta rotina é utilizada pelas rotinas padrões SCANL e SCANR, para verificar a cor do pixel atual em comparação com a cor de contorno de BDRATR.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 19DDH   |
| Nome . . . . .     | TAPOOF  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | EI      |

Rotina padrão para interromper o motor do cassete depois que os dados tiverem sido escritos no cassete. Após um atraso de 550mS, nas máquinas MSX com um estado de espera, o controle vai para a rotina padrão TAPIOF.

|                    |         |
|--------------------|---------|
| Endereço . . . . . | 19E9H   |
| Nome . . . . .     | TAPIOF  |
| Entrada . . . . .  | Nenhuma |
| Saída . . . . .    | Nenhuma |
| Modifica . . . . . | EI      |

Rotina padrão para interromper o motor cassete, depois da leitura de dados do cassete. O relê do motor será aberto através da Porta de Modo do PPI. Observe que interrupções, que deverão ser desativadas durante a transferência de dados no cassete por motivos de temporização, serão ativadas quando esta rotina terminar.



Endereço ..... 19F1H  
 Nome ..... TAPOON  
 Entrada ..... A= Indicador de comprimento de cabeçalho  
 Saída ..... Indicador C, em caso de terminação com CTR-STOP  
 Modifica ..... AF, BC, HL, DI

Rotina padrão para ligar o motor do cassete, aguardar 550mS até que a fita atinja a velocidade de regime e depois escrever um cabeçalho para o cassete. Um cabeçalho é uma série de ciclos HI escritos na frente de quaisquer blocos de dados, de modo que o valor do baud possa ser determinado quando os dados forem lidos.

O tamanho do cabeçalho será determinado pelo conteúdo do registrador A: 00H=Cabeçalho curto, NZ=Cabeçalho longo. As instruções BASIC selecionadas com o cassete "SAVE", "CSAVE" e "BSAVE" geram todas um cabeçalho longo no início do arquivo na frente do bloco de identificação, e a partir daí utilizam cabeçalhos curtos entre blocos de dados. O número de ciclos no cabeçalho, também, é modificado pelo valor da frequência de baud atual, de modo a manter a sua duração constante:

|                       |                    |              |
|-----------------------|--------------------|--------------|
| 1200 Baud CURTO ..... | 3840 Ciclos .....  | 1.5 Segundos |
| 1200 Baud LONGO ..... | 15360 Ciclos ..... | 6.1 Segundos |
| 2400 Baud CURTO ..... | 7936 Ciclos .....  | 1.6 Segundos |
| 2400 Baud LONGO ..... | 31744 Ciclos ..... | 6.3 Segundos |

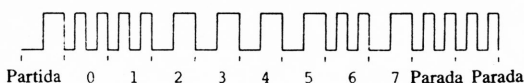
Depois que o motor tiver sido ligado e o atraso ocorrido, o conteúdo de HEADER será multiplicado por duzentos e cinquenta e seis, caso o registrador A seja não-zero, com um fator adicional de quatro para produzir a contagem de ciclos. Os ciclos HI serão gerados(1A4DH), até que a contagem seja esgotada, sendo o controle transferido à rotina padrão BREAKX. Pelo fato da tecla CTRL-STOP ser examinada apenas no final, é impossível sairmos no meio desta rotina.

Endereço ..... 1A19H  
 Nome ..... TAPOUT  
 Entrada ..... A=Byte de dado  
 Saída ..... Indicador C, no caso de terminação com CTRL-STOP  
 Modifica ..... AF, B, HL

Rotina padrão para escrever um único byte de dados no cassete. A ROM do MSX utiliza o método DSK(Frequency Shift Keyed), controlado por Software, para o armazenamento de informação no cassete. Na frequência de 1200 baud, ele é idêntico ao Padrão Kansas City, utilizado pela BBC para a distribuição dos programas BASICODE.

A 1200 baud cada bit 0 é escrito como um ciclo L0 de 1200Hz completo, e cada bit 1 como dois ciclos HI de 2400Hz completos. A frequência dos dados é, portanto, constante uma vez que os bits 0 e 1 têm a mesma duração. Quando a taxa de 2400 baud é selecionada, as frequências mudam para 2400Hz e 4800Hz, mas o formato não é alterado.

Um byte de dados é escrito com um bit de partida 0 (Start Bit) (1A50H), oito bits de dados com o bit menos significativo primeiro, e dois bits 1 de parada (Stop Bits) (1A40H). Na frequência de 1200 baud, um único byte terá uma duração nominal de  $11 \times 833 \mu\text{S} = 9.2\text{mS}$ . Depois que os bits de parada tiverem sido escritos, o controle é transferido à rotina padrão BREAKX, para verificar a tecla CTRL-STOP. Abaixo é mostrado como o byte 43H seria escrito no cassete:



**Figura 39** Byte de Dado em Cassete.

É importante não deixar um intervalo muito grande entre-bytes ao escrever os dados, uma vez que isto aumentará a taxa de erro. Um intervalo entre-bytes de  $80\mu\text{S}$ , por exemplo, produz uma taxa de falha de leitura de aproximadamente doze por cento. Se uma quantidade substancial de processamento for requerida entre cada byte, deverá ser utilizado um buffer para amontoar os dados em blocos. O formato "SAVE" do BASIC é deste tipo.

Endereço ..... 1A39H

Esta rotina escreve um único ciclo L0 com um comprimento de aproximadamente  $816\mu\text{S}$  no cassete. O comprimento de cada metade do ciclo é tomado de LOW e o controle é transferido ao gerador de ciclo geral(1A50H).

Endereço ..... 1A40H

Esta rotina escreve dois ciclos HI no cassete. O primeiro ciclo é gerado(1A4DH) seguido por um atraso de  $17\mu\text{S}$  e depois o segundo ciclo(1A4DH).

Endereço ..... 1A4DH

Esta rotina escreve um único ciclo HI com um comprimento de aproximadamente  $396\mu\text{S}$  no cassete. O comprimento de cada metade do ciclo é tomado de HIGH e o controle vai para o gerador de ciclo geral.

Endereço ..... 1A50H

Esta rotina escreve um único ciclo no cassete. O comprimento da primeira metade do ciclo é fornecido no registrador L e a sua segunda metade no registrador H. O primeiro comprimento é contado regressivamente e depois o bit Cas Out é ligado por meio da Porta de Modo da PPI. O segundo comprimento é contado regressivamente e o bit Cas Out é desligado.

Em todas as máquinas MSX o Z80 processa com uma frequência de Clock de 3.579.545MHz (280nS), com um estado de espera (Wait State) durante o ciclo M1. Uma vez que esta rotina conta a cada 16T estados, cada incremento de unidade na contagem do comprimento representa um período de 4.47 $\mu$ S. Há também um tempo extra (overhead), fixo de 20.7 $\mu$ S, associado à rotina, qualquer que seja a contagem do comprimento.

Endereço ..... 1A63H  
 Nome ..... TAPION  
 Entrada ..... Nenhuma  
 Saída ..... Indicador C, no caso de terminação com CTRL-STOP  
 Modifica ..... AF, BC, DE, HL, DI

Rotina padrão para ligar o motor do cassete, ler o cassete até encontrar um cabeçalho e depois determinar a frequência de baud. Ciclos sucessivos são lidos do cassete e o comprimento de cada um medido(1B34H). Quando 1,111 ciclos tiverem sido encontrados com menos de 35 $\mu$ S de variação em seus comprimentos, um cabeçalho foi localizado.

Os próximos 256 ciclos são, em seguida, lidos(1B34H) e sua média é calculada para determinar o comprimento do ciclo HI do cassete. Este valor é multiplicado por 1.5 e colocado em LOWLIN. Este valor define o menor comprimento aceitável de um bit de partida 0. O comprimento do ciclo HI é encontrado em WINWID e será utilizado para discriminar entre ciclos L0 e HI.

Endereço ..... 1ABCH  
 Nome ..... TAPIN  
 Entrada ..... Nenhuma  
 Saída ..... A=byte ciclo, Indicador C em caso de  
 CTRL-STOP ou erro de E/S  
 Modifica ..... AF, BC, DE, L

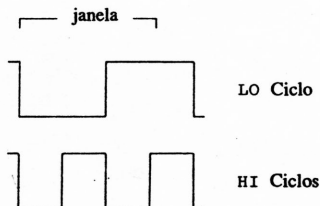
Rotina padrão para ler um byte de dado do cassete. O cassete é primeiro lido continuamente, até que um bit de partida seja encontrado. Isto é feito localizando uma

transição negativa, medindo o comprimento do ciclo seguinte(1B1FH) e comparando isto para ver se é maior do que LOWLIN.

Cada um dos oito bits de dados é, em seguida, lido contando o número de transições ocorridas dentro de um período fixo, de tempo(1B03H). Se zero ou uma transição for encontrada será um bit 0. Se duas ou três forem encontradas será um bit 1. Se forem encontradas mais do que três transições, a rotina terminará com o Indicador C, uma vez que isto significa um erro de hardware de algum tipo. Depois que o valor e cada bit tiver sido determinado serão lidas mais uma ou duas transições adicionais(1B23H) para manter a sincronização. Com uma montagem ímpar de transições será lida mais uma, com uma contagem par de transições, mais duas.

Endereço . . . . . 1B03H

Esta rotina é utilizada pela rotina padrão TAPIN para contar o número de transições no cassete, em um período fixo de tempo. A duração deste período (ou “janela”) está contida em WINWID e é, aproximadamente, 1.5 vezes o comprimento do ciclo HI, conforme indicado abaixo:



**Figura 40** Janela do Cassete

O bit Cas Input é, continuamente, mostrado por meio do Registrador 14 do PSG e comparado com a leitura anterior, contida no registrador E. Cada vez que uma mudança de estado é encontrada, o registrador C é incrementado. A frequência de amostragem é de uma vez a cada  $17.3\mu\text{S}$ , de modo que o valor em WINWID, que foi determinado pela rotina padrão TAPION, com uma frequência de contagem de  $11.45\mu\text{S}$ , é efetivamente multiplicado uma vez e meia.

Endereço . . . . . 1B1FH

Esta rotina mede o tempo até a próxima transição na entrada de cassete. O bit de Entrada de Cassete é mostrado continuamente por meio do Registrador 14 do PSG, até ficar diferente do estado fornecido no Registrador E. O indicador de estado, será, então

invertido e a contagem de duração retornada no registrador C, sendo que cada unidade representa um período de  $11.45\mu\text{S}$ .

Endereço ..... 1B34H

Esta rotina mede o comprimento de um ciclo completo do cassete, de uma transição negativa a uma transição negativa. O bit de entrada de Cassete é mostrado via Registrador 14 do PSG, até que vá a zero. O indicador de transição do registrador E será colocado em zero e a duração será medida até a transição positiva (1B23H). Em seguida, a duração será medida até a transição negativa (1B25H) e o total entregue ao registrador C.

Endereço ..... 1B45H  
 Nome ..... OUTDO  
 Entrada ..... A=Caractere a sair  
 Saída ..... Nenhuma  
 Modifica ..... EI

Rotina padrão utilizada pelo Interpretador BASIC para permitir a saída no dispositivo atual. A rotina padrão ISFLIO é, primeiro: utilizada para verificar se a saída está atualmente dirigida a um buffer de E/S. Em caso afirmativo, o controle é transferido à saída sequencial (6C48H), via a rotina padrão CALBAS. Se PRTFLG for zero, o controle será transferido à rotina padrão CHPUT para enviar do caractere à tela. Considerando que a impressora esteja ativa, RAWPRT será verificado. Se ele for não-zero, o caractere passa diretamente à impressora (1BABH), caso contrário, o controle irá para a rotina padrão OUTDLP.

Endereço ..... 1B63H  
 Nome ..... OUTDLP  
 Entrada ..... A=Caratere para saída  
 Saída ..... Nenhuma  
 Modifica ..... EI

Rotina padrão para permitir a saída de uma caractere na impressora. Se o caractere for um código TAB(09H) serão enviados espaços para a rotina padrão OUTDLP, até que LPTPOS seja um múltiplo de oito (0, 8, 16 etc.). Se o caractere for um código CR(0DH), LPTPOS será zerado; se for qualquer outro código de controle, LPTOS não será afetado, e se for um caractere para ser apresentado, LPTPOS será incrementado.

Se NTMSXP for zero, significando que uma impressora específica do MSX está conectada, o caractere é passado diretamente à impressora(1BABH). Considerando que uma impressora normal esteja conectada, a rotina padrão CNVCHR, que verifica caracteres gráficos, é utilizada. Se o caractere for um código de cabeçalho(01H) a rotina terminará sem nenhuma ação. Se for um caractere gráfico convertido será substituído por um espaço. Todos os outros caracteres serão passados para a impressora(1BACH).

Endereço ..... 1B97H

Esta tabela de vinte bytes é utilizada pelo decodificador de teclado, afim de encontrar a rotina correta para um determinado número de tecla:

| <u>NÚMERO DA TECLA</u> | <u>PARA</u> | <u>FUNÇÃO</u>    |
|------------------------|-------------|------------------|
| 00H to 2FH             | 0F83H       | Rows 0 to 5      |
| 30H to 32H             | 0F10H       | SHIFT,CTRL,GRAPH |
| 33H                    | 0F36H       | CAP              |
| 34H                    | 0F10H       | CODE             |
| 35H to 39H             | 0FC3H       | F1 to F5         |
| 3AH to 3BH             | 0F10H       | ESC,TAB          |
| 3CH                    | 0F46H       | STOP             |
| 3DH to 40H             | 0F10H       | BS,CR,SEL,SPACE  |
| 41H                    | 0F06H       | HOME             |
| 42H to 57H             | 0F10H       | INS,DEL,CURSOR   |

Endereço ..... 1BABH

Esta rotina é utilizada pela rotina padrão OUTDLP para passar um caractere à impressora. Ele será enviado através da rotina padrão LPTOUT, e, caso retorne o Indicador C, o controle será transferido ao gerador de "Erro de Dispositivo de E/S"(73B2H), através da rotina padrão CALBAS.

Endereço ..... 1BBFH

Os 2KB seguintes contêm o conjunto de caracteres de partida. Os primeiros oito bytes contêm a imagem do caractere Código 00H, os oito bytes seguintes contêm a imagem do caractere código 01H e, assim por diante, até o caractere código FFH.

|                    |                                                                         |
|--------------------|-------------------------------------------------------------------------|
| Endereço . . . . . | 23BFH                                                                   |
| Nome . . . . .     | PINLIN                                                                  |
| Entrada . . . . .  | Nenhuma                                                                 |
| Saída . . . . .    | HL=Início de texto, Indicador C, caso haja uma terminação com CTRL-STOP |
| Modifica . . . . . | AF, BC, DE, HL, EI                                                      |

Rotina padrão utilizada pela Ronda Principal do Interpretador BASIC para coletar uma linha lógica de texto do console. O controle é transferido à rotina padrão INLIN, logo depois do ponto em que a linha anterior foi cortada(23EOH).

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| Endereço . . . . . | 23CCH                                                            |
| Nome . . . . .     | QINLIN                                                           |
| Entrada . . . . .  | Nenhuma                                                          |
| Saída . . . . .    | HL=Início do texto; Indicador C, no caso de terminação CTRL-STOP |
| Modifica . . . . . | AF, BC, DE, HL, EI                                               |

Rotina padrão utilizada pelo manipulador de instrução "INPUT" para coletar uma linha lógica de texto do console. Os caracteres "?" são apresentados através da rotina padrão OUTDO e o controle vai para a rotina padrão INLIN.

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| Endereço . . . . . | 23D5H                                                               |
| Nome . . . . .     | INLIN                                                               |
| Entrada . . . . .  | Nenhuma                                                             |
| Saída . . . . .    | HL=Início do texto, Indicador C, caso haja uma terminação CTRL-STOP |
| Modifica . . . . . | AF, BC, DE, HL, EI                                                  |

Rotina padrão utilizada pelo manipulador da instrução "LINE INPUT" para coletar uma linha lógica de texto do console. Os caracteres são lidos do teclado até que as teclas CR ou CTRL-STOP sejam pressionadas. Em seguida, a linha lógica será lida da tela, caractere por caractere, e colocada na Área de Trabalho (no buffer de texto BUF).

As coordenadas da tela atuais são, primeiro, tomadas de CSRX e CSRY e colocadas em FSTPOS. A linha da tela, imediatamente acima da atual, terá sua entrada em LINTTB feita não-zero (0C29H), para impedir que se estenda logicamente na linha atual.

Cada leitura de caractere de teclado, através da rotina padrão CHGET, é verificada(0919H) com a tabela de teclas de edição em 4239H. O controle, em seguida, é

transferido para uma das rotinas de edição ou para o manipulador default de teclas (23FFH), conforme o caso. Este processo continua até que o Indicador C seja devolvido pela rotina CTRL-STOP ou pela rotina CR. Em seguida, o par de registradores HL é apontado para o começo de BUF e a rotina termina. Observe que o indicador de vai-um (carry flag) é zerado quando o Indicador NZ é devolvido, para distinguir entre um CR ou uma terminação CTRL-STOP protegida e uma terminação CTRL-STOP normal.

Endereço . . . . . 23FFH

Esta rotina processa todos os caracteres para a rotina padrão INLIN, exceto as teclas de edição especiais. Se o caractere for um código TAB(09H), espaços serão acionados(23FFH), até que CSRX venha a ser um múltiplo de oito mais um (colunas 1, 9, 17, 25, 33). Se o caractere for um código de cabeçalho gráfico(01H), simplesmente, será ecoado para a rotina padrão OUTDO. Todos os outros códigos de controle, menores do que 20H, serão ecoados para a rotina padrão OUTDO, após o que INSFLG e CSTYLE serão zerados. Para os caracteres apresentáveis INSFLG será primeiro verificado em um espaço inserido(24H2H); se for o caso, antes que o caractere seja ecoado para a rotina padrão OUTDO.

Endereço . . . . . 2439H

Esta tabela contém teclas de edição especiais reconhecidas pela rotina padrão INLIN, juntamente com os endereços relevantes:

| CÓDIGO | PARA  | FUNÇÃO                           |
|--------|-------|----------------------------------|
| 08H    | 2561H | BS, retrocesso                   |
| 12H    | 24E5H | INS, modo de inserção            |
| 1BH    | 23FEH | ESC, nenhuma ação                |
| 02H    | 260EH | CTRL-B, palavra anterior         |
| 06H    | 25F8H | CTRL-F, próxima palavra          |
| 0EH    | 25D7H | CTRL-N, fim da linha lógica      |
| 05H    | 25B9H | CTRL-E, limpa até o fim da linha |
| 03H    | 24C5H | CTRL-STOP, término               |
| ODH    | 245AH | CR, término                      |
| 15H    | 254EH | CTRL-U, limpa linha              |
| 7FH    | 2550H | DEL, cancela caractere           |



Endereço . . . . . 245AH

Esta rotina executa a operação CR para a rotina padrão INLIN. As coordenadas iniciais da linha lógica são encontradas(266CH), e o cursor é removido da tela(0A2EH). Até 254 caracteres serão em seguida, lidos da VRAM do VDP(0BD8H) e colocados no BUF. Quaisquer códigos nulos(00H) serão ignorados, e caracteres menores do que 20H serão substituídos por um código de cabeçalho gráfico(01H) e o caractere em si acrescido de 40H. À medida que o fim de cada linha física é alcançado, LINTTB é verificada(0C1DH) para checar se a linha lógica se estende na linha física seguinte. Espaços posteriores são, em seguida, retirados do BUF e um byte zero é acrescentado como um marcador de fim de texto. O cursor é restaurado à tela(09E1H) e as suas coordenadas ficam sendo as da última linha física da linha lógica. Isto é feito através da rotina padrão POSIT. Um código LF é enviado para a rotina padrão OUTDO, INSFLG é zerado e a rotina termina com um código CR(0DH) no registrador A e os Indicadores NZ, C. Este código CR será ecoado para a tela pela rotina padrão INLIN, logo antes de terminar.

Endereço . . . . . 24C5H

Esta rotina executa a operação CTRL-STOP para a rotina padrão INLIN. A última linha física da linha lógica é encontrada, examinando LINTTB(0C1DH); CSTYLE é zerado, um byte zero é colocado no início de BUF e todas as variáveis musicais são zerados pela rotina padrão GICINI. Em seguida, TRPTBL é examinada(0454H) para verificar se uma declaração "ON STOP" está ativa, e, em caso afirmativo, o cursor é repostado(24AFH) e a rotina termina com os Indicadores NC, C. Em seguida, BASBOM é verificado para ver se uma ROM protegida está em processamento, e em caso afirmativo, o cursor é repostado(24AFH) e a rotina termina com os Indicadores NZ, C. Caso contrário, o cursor é repostado(24B2H) e a rotina termina com os Indicadores Z, C.

Endereço . . . . . 24E5H

Esta rotina executa a operação INS para a rotina padrão INLIN. O estado atual de INSFLG é invertido e o controle termina através da rotina que atualiza CSTYLE (242CH).

Endereço . . . . . 24F2H

Esta rotina insere um caractere de espaço para a seção de teclas default da rotina padrão INLIN. O cursor é removido(0A2EH) e as coordenadas do cursor atual são

tomadas de CSRX e CSRY. O caractere desta posição é lido da VRAM do VDP (0BD8H) e substituído por um espaço(0BE6H). Em seguida, caracteres sucessivos são copiados uma posição de coluna à direita, até que o fim da linha física seja atingido.

Neste ponto LINTTB é examinada(0C1DH) para determinar se a linha lógica é estendida; em caso afirmativo o processo continua na próxima linha física. Caso contrário, o caractere tomado da posição da última coluna é examinado, e, se for um espaço, a rotina termina recolocando o cursor(09E1H). Caso contrário, a entrada da linha física em LINTTB é zerada para indicar uma linha lógica estendida. O número da próxima linha física é comparado com o número de linhas na tela (0C32H). Se a próxima linha for a última, a tela é rolada para cima(0A88H). Caso contrário, uma linha em branco é inserida(0AB7H) e o processo de cópia continua.

Endereço . . . . . 2550H

Esta rotina executa a operação DEL para a rotina padrão INLIN. Se a posição do cursor atual corresponde à da coluna mais à direita e a linha lógica não for estendida, nenhuma outra ação é tomada, a não ser a de escrever um espaço na VRAM do VDP (2595H). Caso contrário, um código RIGHT(1CH) é acionado para a rotina padrão OUTDO e o controle vai para a rotina BS.

Endereço . . . . . 2561H

Esta rotina executa a operação BS para a rotina padrão INLIN. Primeiro, o cursor é removido(0A2EH) e a coordenada da coluna do cursor decrementada, a não ser que este esteja na posição mais à esquerda e a linha anterior não esteja estendida. Em seguida, caracteres são lidos da VRAM do VDP(0BD8H) e escritos numa posição à esquerda(0BE6H), até que o final da linha lógica seja alcançada. Neste momento, um espaço é escrito na VRAM do VDP(0BE6H) e o caractere do cursor é restaurado(09E1H).

Endereço . . . . . 25AEH

Esta rotina executa a operação CTRL-U para a rotina padrão INLIN. O cursor é removido(0A2EH) e o início da linha lógica localizado(266CH) e colocado em CSRX e CSRY. Em seguida, toda a linha lógica é limpa(25BEH).

Endereço . . . . . 25B9H

Esta rotina executa a operação CTRL-E para a rotina padrão INLIN. O cursor é removido(0AEEH) e o restante da linha física limpo(0AEEH). Este processo é repetido para sucessivas linhas físicas, até que o final da linha lógica seja encontrado em LINTBB(0C1DH). Em seguida, o cursor é restaurado(09E1H), INSFLG é zerado e CSTYLE volta a ser um cursor tipo "bloco"(242DH).

Endereço . . . . . 25D7H

Esta rotina executa a operação CTRL-N para a rotina padrão INLIN. O cursor é removido(0A2EH) e a última linha física da linha lógica é encontrada pelo exame de LINTTB(0C1DH). A partir da coluna mais à direita desta linha física, caracteres são lidos da VRAM do VDP(0BD8H) até que um caractere não-espaco seja encontrado. As coordenadas do cursor são, em seguida, colocadas uma coluna à direita desta posição(0A5BH) e a rotina termina pela restauração do cursor(25CDH).

Endereço . . . . . 25F8H

Esta rotina executa a operação CTRL-F para a rotina padrão INLIN. O cursor é removido(0A2EH) e movido sucessivamente para à direita(2624H), até que um caractere não-alfanumérico seja encontrado. O cursor é, em seguida, movido sucessivamente para à direita(2624H) até que um caractere alfanumérico seja encontrado. A rotina termina pela restauração do cursor(25CDH).

Endereço . . . . . 260EH

Esta rotina executa a operação CTRL-B para a rotina padrão INLIN. O cursor é removido(0A2EH) e movido sucessivamente para à esquerda(2634H), até que um caractere alfanumérico seja encontrado. O cursor é, em seguida, movido sucessivamente para à esquerda(2634H), até que um caractere não-alfanumérico seja encontrado e depois movido uma posição para a direita(0A5BH). A rotina termina pela restauração do cursor(25CDH).

Endereço . . . . . 2624H

Esta rotina move o cursor uma posição para a direita(0A5BH), carrega o registrador D com o número da coluna mais à direita, o registrador E com o número da linha mais baixa e depois testa, se existir um caractere alfanumérico na posição do cursor (263DH).

Endereço . . . . . 2634H

Esta rotina move o cursor uma posição para à esquerda(0A4CH), carrega o registrador D com o número da coluna mais à esquerda e o registrador E com o número da linha mais alta. As coordenadas atuais do cursor são comparadas com estes valores e a rotina termina com o Indicador Z, se o cursor estiver nesta posição. Caso contrário, o caractere nesta posição é lido da VRAM do VDP(0BD8H) e verificado para saber se é alfanumérico. Em caso afirmativo, a rotina termina com os Indicadores NZ, C; caso contrário, termina com os Indicadores NZ, NC.

Os caracteres alfanuméricos são os dígitos "0" a "9" e as letras "A" a "Z" e "a" a "z". Estão incluídos, também, os caracteres gráficos 86H a 9FH e A6H a FFH, que eram originariamente letras japonesas e deveriam ter sido excluídos durante a conversão à ROM inglesa.

Endereço . . . . . 266CH

Esta rotina encontra o início de uma linha lógica e devolve as suas coordenadas de tela no par de registradores HL. Cada linha física, acima da atual, é verificada por meio da tabela LINTTB(0C1DH), até que uma linha não-estendida seja encontrada. A linha imediatamente abaixo desta, na tela, é o início da linha lógica e o seu número de linha é colocado no registrador L. Isto é, em seguida, comparado com FSTPOS, que contém o número da linha quando a rotina padrão INLIN foi executada pela primeira vez, para ver se o cursor continua na mesma linha. Em caso afirmativo, a coordenada da coluna (registrador H) fica valendo esta posição inicial de FSTPOS. Caso contrário, o registrador H fica com o valor da posição mais à esquerda, para devolver toda a linha.

Endereço . . . . . 2680H, JP até rotina de partida (power-up) (7C76H).

Endereço . . . . . 2683H, JP até a rotina padrão SYNCHR(558CH).

Endereço . . . . . 2686H, JP até a rotina padrão CHRGT(4666H).

Endereço . . . . . 2689H, JP até a rotina padrão GETYPR(5597H).

## INTERPRETADOR BASIC EM ROM

O BASIC da Microsoft evoluiu durante anos até a sua posição atual como padrão da indústria. Originalmente foi escrito para o Microprocessador 8080, e mesmo a versão MSX mantida na forma da Linguagem de Montagem do 8080. Este processo de desenvolvimento contínuo significa que há menos instruções específicas do Z80 do que seria de se esperar em um programa mais moderno. Significa, também, que muitas alterações foram feitas, o que resultou num programa bastante turbulento. A estrutura do Interpretador torna improvável que num programa aplicativo seja capaz de utilizar seus diversos recursos. Entretanto, a maioria dos programas terá que se adaptar a ele de alguma forma, de modo que este capítulo fornece uma descrição detalhada de suas operações.

Existem quatro áreas importantes, facilmente identificáveis dentro do Interpretador. A mais familiar a qualquer usuário é a Ronda Principal (Mainloop) (4134H). Ela pega linhas numeradas de texto do console e as coloca em ordem na Área de Texto de Programa da memória, até que um comando direto seja recebido.

A Ronda de execução (Runloop) (4601H) é responsável pela execução de um programa. Examina o primeiro átomo (token) de uma linha de programa e chama a rotina apropriada para processar o restante da instrução, até não haver mais nenhum texto de programa, quando o controle, então, passa a Ronda Principal.

Em uma instrução, a análise de operandos string ou numéricos é executada pelo Avaliador de Expressão (Expression Evaluator) (4C64H). Cada expressão é formada por fatores, que por sua vez são analisados pelo Avaliador de Fatores (4DC7H), que são ligados entre si por operadores binários. Como há diversos tipos de operandos, que não podem fazer parte de uma expressão (como os números de linha, no BASIC da Microsoft)

o termo “avaliados” está sendo usado aqui apenas para referir-se àqueles que podem fazer parte de uma expressão. Caso contrário, será utilizado um termo como “computado”.

Um ponto que deve ser notado ao se examinar o Interpretador em detalhes é que ele contém diversos truques de programação assembler. Aparentemente, os criadores estavam especialmente interessados em pular no meio das instruções na preparação dos pontos de entrada de uma rotina. Como exemplo, vejamos a instrução:

3E D1      Normal: LD      A,0D1H

Quando encontrada da forma normal ela, sem dúvida, carregará o acumulador com o valor D1H. Entretanto, se entrarmos em “Normal+1”, será executada uma instrução POP DE. O Interpretador tem muitas seções igualmente obscuras.

Endereço ..... 268CH

Esta rotina é utilizada pelo Avaliador de Expressões para a subtração de dois operandos de precisão dupla. O primeiro operando está contido em DAC e o segundo em ARG (veja o Capítulo 6, endereço F7F6H) e o resultado será colocado em DAC. O sinal da mantissa do segundo operando é invertido e o controle vai para a rotina de soma.

Endereço ..... 269AH

Esta rotina é utilizada pelo Avaliador de Expressões para a soma de dois operandos de dupla precisão. O primeiro operando está contido em DAC e o segundo em ARG e o resultado será colocado em DAC. Se o segundo operando foi zero, a rotina termina sem nenhuma ação, e se o primeiro operando for zero, o segundo operando é copiado em DAC(2F05H) e a rotina termina. Os dois expoentes são comparados e se diferirem em mais do que  $10^{15}$ , a rotina termina tendo como resultado o operando maior. Caso contrário, a diferença entre os dois expoentes será utilizada para alinhar a mantissa pelo deslocamento do menor para a direita(27A3H), por exemplo:

$$\begin{aligned} 19.2100 &= .1921 \cdot 10^2 = .192100 \\ + .7436 &= .7436 \cdot 10^0 = .007436 \end{aligned}$$

Se os sinais das duas mantissas forem iguais, elas serão somadas(2759H), se forem diferentes as mantissas serão subtraídas(276BH). O expoente do resultado é, simplesmente, o maior dos dois expoentes originais. Caso tenha sido produzido um extravasamento (overflow) pela adição, a mantissa resultante é deslocada um dígito para a direita(27DBH) e o expoente incrementado. Caso, pela subtração tenham sido produzidos

zeros nos locais mais significativos, a mantissa resultante será renormalizada por um deslocamento à esquerda(2797H). O byte de guarda é, em seguida, examinado e o resultado arredondado, se o décimo quinto dígito for igual ou maior do que cinco.

Endereço ..... 2759H

Esta rotina soma as duas mantissas de dupla precisão contidas em DAC e ARG e coloca o resultado em DAC. A soma se inicia nas posições menos significativas, DAC+7 e ARG+7, continuando dois dígitos por vez, nos sete bytes.

Endereço ..... 276BH

Esta rotina subtrai as duas mantissas de dupla precisão contidas em DAC e ARG e coloca o resultado em DAC. A subtração se inicia no byte de guarda, DAC+8 e ARG+8, e segue dois dígitos por vez nos oito bytes. Se o resultado for um subextravazamento, isto será corrigido pela subtração de zero e inversão do sinal da mantissa, por exemplo:

$$0.17 - 0.85 = 0.32 = -0.68$$

Endereço ..... 2797H

Esta rotina desloca a mantissa de dupla precisão contida em DAC, um dígito à esquerda.

Endereço ..... 27A3H

Esta rotina desloca à direita uma mantissa de dupla precisão. O número dígitos a serem deslocados é fornecido no registrador A, e o endereço do byte mais significativo da mantissa será fornecido pelo par de registradores HL. O número de dígitos primeiro é dividido por dois, para separar as contagens de byte e dígitos. Em seguida, o número requerido de bytes completos é deslocado à direita e os bytes mais significativos zerados. Se um número ímpar de dígitos tiverem sido especificados, a mantissa será deslocada mais um dígito à direita.

Endereço ..... 27E6H

Esta rotina é utilizada pelo Avaliador de Expressões para multiplicar dois operandos de dupla precisão. O primeiro operando está contido em DAC e o segundo em

ARG, e o resultado será colocado em DAC. Se qualquer um dos operandos for zero, a rotina terminará tendo como resultado um zero(2E7DH). Caso contrário, os dois expoentes serão somados para fornecer o expoente resultante. Se este for menor do que  $10^{-63}$ , a rotina terminará com um resultado zero, e se ele for maior do que  $10^{63}$  será gerado um "Erro de extravasamento"(4067H). Em seguida, os dois sinais das mantissas serão processados para fornecer o sinal do resultado, que será positivo, se os dois forem iguais; e negativo, se forem diferentes.

Mesmo estando as mantissas no formato BCD, elas são multiplicadas utilizando o método binário normal de soma e deslocamento ("add and shift"). Para isto, o primeiro operando é, sucessivamente, multiplicado por dois(288AH), afim de produzir as constantes  $X*80$ ,  $X*40$ ,  $X*20$ ,  $X*10$ ,  $X*8$ ,  $X*4$ ,  $X*2$  e  $X$  no buffer HOLD8. O segundo operando permanece em ARG e o DAC e será zerado, afim de funcionar como acumulador do produto. A multiplicação prossegue, tomando pares sucessivos de dígitos do segundo operando e iniciando com o par menos significativo. Para cada bit 1 no par de dígitos o múltiplo apropriado do primeiro operando será somado ao produto. Como um exemplo a multiplicação simples  $1823*96$  forneceria:

$$1832*10010110=(1823*80)+(1823*10)+(1823*4)+(1823*2)$$

À medida que cada par de dígitos for completado, o produto será deslocado dois dígitos à direita. Quando todos os sete pares de dígitos tiverem sido processados, a rotina terminará renormalizando e arredondando o produto(26FAH).

O tempo requerido para uma multiplicação depende, principalmente, do número de bits 1 no segundo operando. No pior caso, quando todos os dígitos forem sete, ela poderá demorar até 11mS. A duração média é de 7mS, aproximadamente.

Endereço ..... 288AH

Esta rotina dobra uma mantissa de dupla precisão três vezes, sucessivamente, para fornecer os produtos  $X*2$ ,  $X*4$ , e  $X*8$ . O endereço do byte menos significativo da mantissa é fornecido no par de registradores DE. Os produtos são armazenados em endereços sucessivamente mais baixos, iniciando imediatamente abaixo do operando.

Endereço ..... 289FH

Esta rotina é utilizada pelo Avaliador de Expressões para dividir dois operandos de dupla precisão. O primeiro operando está contido em DAC e o segundo em ARG, e o resultado colocado em DAC. Se o primeiro operando for zero, a rotina termina tendo como



resultado um zero, e se o segundo operando for zero será gerado um erro de “divisão por zero”(4058H). Caso contrário, os dois expoentes são subtraído para produzir o expoente resultante e as duas mantissas terão seus sinais processados para a obtenção do sinal resultante. Se forem os mesmos, o resultado será positivo, e se forem diferentes o resultado será negativo.

As mantissas são divididas utilizando o método normal da divisão longa. O segundo operando é subtraído repetidamente do primeiro até haver um subextravasamento para produzir um único dígito do resultado. Em seguida, o segundo operando será somando para restaurar o resto(2761H), o dígito será armazenado em HOLD e o primeiro operando deslocado um dígito à esquerda. Quando o primeiro operando tiver sido completamente deslocado o resultado será copiado de HOLD ao DAC e depois normalizado e arredondado(2883H). O tempo requerido para uma divisão atinge um máximo de, aproximadamente, 25mS se o primeiro operando for formado, principalmente, por noves e o segundo operando por uns. Isto vai requerer um maior número de subtrações.

Endereço . . . . . 2993H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “COS” a um operando de dupla precisão contido em DAC. O operando é primeiro multiplicado (2C3BH) por  $1/(2 \cdot \pi)$ , de modo que a unidade corresponda a um ciclo completo de 360 graus. O operando tem, então, 0.25(90 graus) subtraído(2C32H), o sinal de sua mantissa será invertido(2E8DH) e o controle será para a rotina “SIN”.

Endereço . . . . . 29ACH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “SIN” a um operando de dupla precisão contido em DAC. Primeiramente, o operando é multiplicado por(2C3BH)  $1/(2 \cdot \pi)$ , de modo que a unidade corresponda a um ciclo completo de 360 graus. Uma vez que a função é periódica, apenas a parte fracionária do operando será requerida. Isto é extraído empilhando o operando(2CCCH), obtendo sua parte inteira (30CFH), copiando-a em ARG(2C4DH), desempilhando de volta o operando completo em DAC(2CE1H) e depois subtraindo a parte inteira(268CH).

O primeiro dígito da mantissa será, então, examinado para determinar o quadrante do operando. Se estiver no primeiro quadrante, será inalterado. Se estiver no segundo quadrante, será subtraído de 0.5(180 graus) para refleti-lo no eixo dos Y. Caso esteja no terceiro quadrante será subtraído de 0.5(180 graus) para refleti-los no eixo dos X. Caso esteja no quarto quadrante 1.0(360 graus) será subtraído para refleti-lo sobre os dois eixos. Em seguida, a função é computada por aproximação polinomial(2C88H)

utilizando a lista de coeficientes em 2DEFH. Estes são os oito primeiros termos da série de Taylor  $X - (X^3/3!) + (X^5/5!) - (X^7/7!)...$  com os coeficientes multiplicados por fatores sucessivos de  $2*PI$  para compensar a mudança inicial de unidade.

Endereço ..... 29FBH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "TAN" em um operando de dupla precisão contido em DAC. A função é computada utilizando-se a identidade trigonométrica  $TAN(X) = SIN(X)/COS(X)$ .

Endereço ..... 2A14H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "ATN" em um operando de dupla precisão contido em DAC. A função é computada por aproximação polinomial(2C88H) utilizando-se a lista de coeficientes em 2E30H. Estes são os oito primeiros termos da série de Taylor  $X - (X^3/3) + (X^5/5) - (X^7/7)$  com os coeficientes ligeiramente modificados para melhorar a aproximação.

Endereço ..... 2A72H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LOG" em um operando de dupla precisão colocado em DAC. A função é computada por aproximação polinomial utilizando-se a lista de coeficientes em 2DA5H.

Endereço ..... 2AFFH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "SQR" em um operando de dupla precisão colocado em DAC. Esta função é computada utilizando-se o método Newton-Raphson. O programa em BASIC a seguir ilustra este método:

```
10 INPUT "NUMERO"; X
20 CHUTE=10
30 FOR N=1 TO 7
40 CHUTE=(CHUTE+X/CHUTE)/2
50 NEXT N
60 PRINT CHUTE
70 PRINT SQR(X)
```

O programa acima utiliza uma adivinhação inicial fixa. Apesar disto funcionar sobre uma faixa limitada, uma precisão máxima somente, será conseguida se a adivinhação inicial

estiver próxima da raiz. O método utilizado pela ROM é tomar a metade do expoente, arredondando para cima, e depois dividir os dois primeiros dígitos operando por quatro e incrementar o primeiro dígito.

Endereço ..... 2B4AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "EXP" a um operando de precisão dupla contido em DAC. O operando é primeiro multiplicado por 0.4342944819, que é  $\text{LOG}(e)$  na Base 10, de modo que o problema passa a ser computar  $10^X$  em vez de  $e^X$ . Isto tem como resultado uma simplificação considerável, uma vez que a parte inteira poderá ser facilmente trabalhada. Em seguida, a função é computada por aproximação polinomial utilizando-se a lista de coeficientes em 2D6BH.

Endereço ..... 2BDFH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "RND" em um operando de dupla precisão colocado em DAC. Se o operando for zero, o número aleatório atual é copiado em DAC de RNDX e a rotina terminará. Se o operando for negativo será copiado em RNDX, passando a ser o número aleatório atual. O novo número aleatório é produzido copiando RNDX para HOLD, a constante em 2CF9H para ARG, a constante em 2CF1H para DAC e depois multiplicando (282EH). Os quatorze dígitos menos significativos do produto de comprimento duplo são copiados para RNDX, para formar a mantissa do novo número randômico. O byte de expoente em DAC será colocado em  $10^0$  para devolver um valor na faixa 0 a 1.

Endereço ..... 2C24H

Esta rotina é utilizada pelos manipuladores dos comandos "/NEW", "CLEAR" e "RUN" para inicializar RNDX com a constante em 2D01H.

Endereço ..... 2C2CH

Esta rotina acrescenta a constante cujo endereço é fornecido pelo par de registradores HL ao operando de dupla precisão contido em DAC.

Endereço ..... 2C32H

Esta rotina subtrai a constante, cujo endereço é fornecido pelo par de registradores HL do operando de dupla precisão contido em DAC.

Endereço ..... 2C3BH

Esta rotina multiplica o operando de dupla precisão contido em DAC pela constante, cujo endereço é fornecido pelo par de registradores HL.

Endereço ..... 2C41H

Esta rotina divide o operando de dupla precisão contido em DAC pela constante, cujo endereço é fornecido pelo par de registradores HL.

Endereço ..... 2C47H

Esta rotina executa a operação relacional no operando de dupla precisão contido em DAC e a constante, cujo endereço é fornecido no par de registradores HL.

Endereço ..... 2C4DH

Esta rotina copia um operando de dupla precisão de oito bytes de DAC para ARG.

Endereço ..... 2C59H

Esta rotina copia um operando de dupla precisão de oito bytes de ARG para DAC.

Endereço ..... 2C6FH

Esta rotina realiza um intercâmbio entre os oito bytes de DAC e os oito bytes atualmente no fundo da pilha do Z80.

Endereço ..... 2C80H

Esta rotina inverte o sinal da mantissa do operando contido em DAC(2E8DH). O mesmo endereço é, em seguida, empilhado na pilha para restaurar o sinal quando o chamador retornar.

Endereço . . . . . 2C88H

Esta rotina gera uma série ímpar, baseada no operando de dupla precisão contido no DAC. A série será da seguinte forma:

$$X \wedge (K_n) + X \wedge 3 * (K_n - 1) + X \wedge 5 * (K_n - 2) + X \wedge 7 * (K_n - 3) \dots$$

O endereço da lista de coeficientes é fornecido pelo par de registradores HL. O primeiro byte da lista contém o número de coeficientes. Seguem-se os coeficientes de dupla precisão: primeiro K1 e por último Kn. A série par é gerada (2C9AH) e multiplicada (27E6H) pelo operando original.

Endereço . . . . . 2C9AH

Esta rotina gera uma série par, baseada no operando de dupla precisão contido em DAC. A série é da seguinte forma:

$$X \wedge 0 * (K_n) + X \wedge 2 * (K_n - 1) + X \wedge 4 * (K_n - 2) + X \wedge 6 * (K_n - 3) \dots$$

O endereço da lista de coeficientes é fornecido no par de registradores HL. O primeiro byte da lista contém o número de coeficientes. Seguem-se os coeficientes de dupla precisão: primeiro K1 e por último Kn. O método utilizado para computar o polinômio é conhecido como método de Horner. Requer apenas uma multiplicação e uma adição por termo, sendo o equivalente em BASIC:

```
10 X=X*X
20 PROD=0
30 RESTORE 100
40 READ NUM
50 FOR N=1 TO NUM
60 READ K
70 PROD=(PROD*X)+K
80 NEXT N
90 END
100 DATA 8
110 DATA Kn-7
120 DATA Kn-6
130 DATA Kn-5
140 DATA Kn-4
150 DATA Kn-3
160 DATA Kn-2
170 DATA Kn-1
180 DATA Kn
```

O processamento do polinômio é feito a partir do coeficiente final passando pelos outros até o primeiro coeficiente, de modo que o produto parcial possa ser utilizado para economizar operações desnecessárias.

Endereço ..... 2CC7H

Esta rotina empilha um operando de precisão dupla de oito bytes, de ARG para a pilha do Z80.

Endereço ..... 2CCCH

Esta rotina empilha um operando de dupla precisão de oito bytes, de DAC para a pilha do Z80.

Endereço ..... 2CDCH

Esta rotina desempilha um operando de dupla precisão de oito bytes da pilha do Z80 e coloca-o em ARG.

Endereço ..... 2CE1H

Esta rotina retira um operando de dupla precisão de oito byte da pilha do Z80 e coloca-o em DAC.

Endereço ..... 2CF1H

Esta tabela contém as constantes de dupla precisão utilizadas pelas rotinas matemáticas. As três primeiras constantes têm zero na posição do expoente, uma vez que estão em uma forma intermediária especial utilizada pelo gerador de números aleatórios.

#### ENDEREÇO    CONSTANTE

|       |                  |            |
|-------|------------------|------------|
| 2CF1H | .14389820420821  | RND        |
| 2CF9H | .21132486540519  | RND        |
| 2D01H | .40649651372358  | RNDX       |
| 2D09H | .43429448190324  | LOG(e)     |
| 2D11H | .50000000000000  |            |
| 2D13H | .00000000000000  |            |
| 2D1BH | 1.00000000000000 |            |
| 2D23H | .25000000000000  |            |
| 2D2BH | 3.1622776601684  | SQR(10)    |
| 2D33H | .86858896380650  | 2*LOG(e)   |
| 2D3BH | 2.3025850929940  | 1/LOG(e)   |
| 2D43H | 1.5707963267949  | PI/2       |
| 2D4BH | .26794919243112  | TAN(PI/12) |
| 2D53H | 1.7320508075689  | TAN(PI/3)  |

#### ENDEREÇO    CONSTANTE

|       |                  |     |
|-------|------------------|-----|
| 2DAEH | 6.2503651127908  |     |
| 2DB6H | -13.682370241503 |     |
| 2DBEH | 8.5167319872389  |     |
| 2DC6H | 5                | LOG |
| 2DC7H | 1.00000000000000 |     |
| 2DCFH | -13.210478350156 |     |
| 2DD7H | 47.925256043873  |     |
| 2DDFH | -64.906682740943 |     |
| 2DE7H | 29.415750172323  |     |
| 2DEFH | 8                | SIN |
| 2DF0H | -.69215692291809 |     |
| 2DF8H | 3.8172886385771  |     |
| 2E00H | -15.094499474801 |     |
| 2E08H | 42.058689667355  |     |

|       |                  |          |       |                  |     |
|-------|------------------|----------|-------|------------------|-----|
| 2D5BH | .52359877559830  | PI/6     | 2E10H | -76.705859683291 |     |
| 2D63H | .15915494309190  | 1/(2*PI) | 2E18H | 81.605249275513  |     |
| 2D6BH | 4                | EXP      | 2E20H | -41.341702240398 |     |
| 2D6CH | 1.00000000000000 |          | 2E28H | 6.2831853071796  |     |
| 2D74H | 159.37415236031  |          | 2E30H | 8                | ATN |
| 2D7CH | 2709.3169408516  |          | 2E31H | -.05208693904000 |     |
| 2D84H | 4497.6335574058  |          | 2E39H | -.07530714913480 |     |
| 2D8CH | 3                | EXP      | 2E41H | -.09081343224705 |     |
| 2D8DH | 18.312360159275  |          | 2E49H | .11110794184029  |     |
| 2D95H | 831.40672129371  |          | 2E51H | -.14285708554884 |     |
| 2D9DH | 5178.0919915162  |          | 2E59H | .19999999948967  |     |
| 2DA5H | 4                | LOG      | 2E61H | -.33333333333160 |     |
| 2DA6H | -.71433382153226 |          | 2E69H | 1.00000000000000 |     |

Endereço ..... 2E71H

Esta rotina devolve o sinal da mantissa de um operando no Ponto Flutuante contido em DAC. O byte do expoente é testado e o resultado indicado no registrador A e nos indicadores:

Zero ..... A=00H, Indicadores Z, NC  
 Positivo ..... A=01H, Indicadores NZ, NC  
 Negativo ..... A=FFH, Indicadores NZ, C

Endereço ..... 2E7DH

Esta rotina, simplesmente, zera o byte do expoente no DAC.

Endereço ..... 2E82H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "ASB" num operando contido em DAC. O sinal do operando é primeiro verificado(2EA1H), e se for positivo a rotina simplesmente termina. Em seguida, o tipo do operando é verificado por intermédio da rotina padrão GETYPR. Se for uma string será gerado um erro de "Tipo incorreto" (type mismatch) (406DH). Se for um inteiro ele será negado(322BH). Se for um operando de precisão simples ou dupla, o bit de sinal da mantissa em DAC será invertido.

Endereço ..... 2E97H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "SGN" num operando contido em DAC. O sinal do operando será verificado(2EA1H), estendido no par de registradores HL e depois colocado no DAC como um inteiro:

Zero . . . . . 0000H  
 Positivo . . . . . 0001H  
 Negativo . . . . . FFFFH

Endereço . . . . . 2EA1H

Esta rotina devolve o sinal de um operando contido no DAC. Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR. Se for uma string, um erro de "Tipo incorreto" (type mismatch) será gerado (406DH). Se for um operando de simples ou dupla precisão o sinal da mantissa será examinado(2E71H). Se for um inteiro, o seu valor será tomado de DAC+2 e traduzido para os indicadores mostrados em 2E71H.

Endereço . . . . . 2EB1H

Esta rotina empilha um operando de simples precisão de quatro bytes de DAC na pilha do Z80.

Endereço . . . . . 2EC1H

Esta rotina copia o conteúdo dos registradores C, B, E e D em DAC.

Endereço . . . . . 2EECH

Esta rotina copia o conteúdo de DAC nos registradores C, B, E e D.

Endereço . . . . . 2ED6H

Esta rotina carrega os registradores C, B, E e D de posições seqüenciais ascendentes, iniciando no endereço fornecido no par de registradores HL.

Endereço . . . . . 2EDFH

Esta rotina carrega os registradores E, D, C e B de posições seqüenciais ascendentes, iniciando no endereço fornecido pelo par de registradores HL.

Endereço . . . . . 2EE8H

Esta rotina copia um operando de precisão simples do DAC para um endereço fornecido pelo par de registradores HL.



Endereço ..... 2EEFH

Esta rotina copia qualquer operando do endereço fornecido no par de registradores HL em ARG. O comprimento do operando está contido em VALTYP: 2=Inteiro, 3=String, 4=Simple Precisão, 8=Dupla Precisão.

Endereço ..... 2F05H

Esta rotina copia qualquer operando de ARG para DAC. O comprimento do operando está contido em VALTYP: 2=Inteiro, 3=String, 4=Precisão Simples, 8=Precisão Dupla.

Endereço ..... 2F0DH

Esta rotina copia qualquer operando do DAC para ARG. O comprimento do operando está contido em VALTYP: 2=Inteiro, 3=String, 4=Precisão Simples, 8=Precisão Dupla.

Endereço ..... 2F21H

Esta rotina é utilizada pelo Avaliador de Expressões para encontrar a relação ( $< > =$ ) entre dois operandos de simples precisão. O primeiro operando está contido nos registradores C, B, E e D e, o segundo, em DAC. O resultado é colocado no registrador A e nos indicadores:

Operando 1=Operando 2 .... A=00H, Indicador Z, NC  
 Operando 1<Operando 2 .... A=01H, Indicador NZ, NC  
 Operando 1>Operando 2 .... A=FFH, Indicador NZ, C

Note que, para os operadores relacionais, o Avaliador de Expressões encara os maiores números negativos como pequenos e os maiores números positivos como grandes.

Endereço ..... 2F4DH

Esta rotina é utilizada pelo avaliador de Expressões para encontrar a relação ( $< > =$ ) entre dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL. O resultado é o mesmo que para a versão de simples precisão(2F21H).

Endereço . . . . . 2F83H

Esta rotina é utilizada pelo Avaliador de Expressões para encontrar a relação ( $< > =$ ) entre dois operandos de dupla precisão. O primeiro operando está contido em DAC e o segundo em ARG. Os resultados são iguais aos da precisão simples(2F21H).

Endereço . . . . . 2F8AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CINT" num operando contido em DAC. Primeiro, o tipo de operando é verificado por meio da rotina padrão GETYPR, e se já for um inteiro, a rotina simplesmente termina. Se for uma string, será gerado um erro de "Tipo incorreto" (type Mismatch) (406DH). Se for um operando de simples precisão ou dupla precisão, será convertido em um inteiro binário com sinal, no par de registradores DE(305DH), e depois colocado em DAC como um inteiro. Valores fora da faixa resultam em um erro de "Extravasamento"(4067H).

Endereço . . . . . 2FA2H

Esta rotina verifica se DAC contém o operando de simples precisão -32768 e, em caso afirmativo, substituí-lo com o inteiro equivalente 8000H. Este passo é necessário durante uma conversão de entrada numérica(3299H), pela assimetria que existe na faixa de valores para os números inteiros.

Endereço . . . . . 2FB2H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CSNG" em um operando contido em DAC. Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR. Se for simples precisão, a rotina simplesmente termina. Se for uma string será gerado um erro de "Tipo incorreto" (type mismatch) (406DH). Se for dupla precisão, VALTYP é alterado(3053H) e a mantissa arredondada para cima, a partir do sétimo dígito(2741H). Se o operando for um inteiro, ele será convertido de binário a um máximo de cinco dígitos BCD por divisões sucessivas, utilizando as constantes 10000, 1000, 100, 10, 1. Estes são colocados no DAC para formar a mantissa de simples precisão. O expoente é igual ao número de dígitos significativos na mantissa. Por exemplo, se há cinco, o expoente será  $10^5$ .

Endereço . . . . . 3030H

Esta tabela contém as cinco constantes utilizadas pela rotina "CSNG": -10000, -1000, -100, -10, -1.

Endereço . . . . . 303AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CDBL", num operando contido em DAC. Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR. Se já for precisão dupla, a rotina simplesmente termina. Se for uma string, será gerado um erro "Tipo incorreto" (type mismatch) (406DH). Se for um inteiro, será convertido primeiro para simples precisão (2FC8H). Em seguida, os oito dígitos menos significativos serão zerados e VALTYP ficará com o valor 8.

Endereço . . . . . 3058H

Esta rotina verifica se o operando atual é do tipo string. Em caso negativo, será gerado um erro de "Tipo incorreto" (type mismatch) (406DH).

Endereço . . . . . 305DH

Esta rotina é utilizada pela rotina "CINT"(2F8AH) para converter um operando BCD de dupla precisão ou simples precisão em um inteiro binário com sinal, no par de registradores DE, devolvendo o Indicador C, caso tenha ocorrido um extravasamento. Dígitos sucessivos são tomados da mantissa e somados ao produto, iniciando com o mais significativo. Após cada adição, o produto é multiplicado por dez. O número de dígitos a serem processados é determinado pelo expoente. Por exemplo, cinco dígitos seriam tomados com um expoente de  $10^5$ . Finalmente, o sinal da mantissa é verificado e o produto negado (3221H), se for o caso.

Endereço . . . . . 30BEH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "FIX", num operando contido em DAC. Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR. Se for um inteiro, a rotina simplesmente termina. Em seguida, o sinal da mantissa é verificado (2E71H), e caso seja positivo o controle será transferido à rotina "INT"(30CFH). Caso contrário, o sinal será invertido para ficar positivo, a função "INT" será executada (30FCH) e o sinal restaurado para negativo.

Endereço . . . . . 30CFH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "INT", num operando contido em DAC. Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR, e se for um inteiro, a rotina simplesmente termina. O número de dígitos fracionários é determinado subtraindo o expoente do número de dígitos do tipo: 6 para precisão simples, 14 para dupla precisão.

Se o sinal da mantissa for positivo, estes dígitos fracionários serão simplesmente zerados. Se o sinal da mantissa for negativo, cada dígito fracionário será examinado, antes de ser zerado. Se todos os dígitos forem previamente zero, a rotina simplesmente termina. Caso contrário, -1.0 é somado ao operando pela rotina de adição de precisão simples(324EH) ou pela rotina de adição de precisão dupla(269AH). Note que o tipo de um operando não é normalmente alterado pela função "CINT".

Endereço . . . . . 314AH

Esta rotina multiplica os inteiros binários sem sinal contido, nos pares de registradores BC e DE, e o resultado será colocado no par DE. O método padrão de "deslocamento e soma" é utilizado, o produto sucessivamente multiplicado por dois e o par de registradores BC somado a ele para cada bit 1, no par de registradores DE. A rotina é utilizada pela rotina de Busca de Variável(SEA4H), para calcular a posição de um elemento em uma Matriz e um erro "Subscript out of range" é gerado(601DH), caso ocorra um extravasamento.

Endereço . . . . . 3167H

Esta rotina é utilizada pelo Avaliador de Expressões para a subtração de dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, com o resultado colocado em DAC. O segundo operando é negado(3221H) e o controle vai para a rotina de adição.

Endereço . . . . . 3172H

Esta rotina é utilizada pelo Avaliador de Expressões para somar dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, com o resultado colocado em DAC. Os operandos binários com sinal, normalmente são apenas somados e colocados em DAC. Entretanto, caso tenha ocorrido um extravasamento, ambos os operandos serão convertidos para simples precisão(2FCBH)

e o controle transferido ao somador de precisão simples(324EH). Ocorre um extravasamento quando os dois operandos são de mesmo sinal e o resultado de sinal oposto, por exemplo:

$$30000 + 15000 = -20536$$

Endereço ..... 3193H

Esta rotina é utilizada pelo Avaliador de Expressões para multiplicar dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, com o resultado colocado em DAC. Os sinais dos dois operandos são temporariamente salvos e os dois operandos feitos positivos(3215H). A multiplicação continua, utilizando-se o método padrão de “deslocamento e soma” com o par de registradores HL, sendo o acumulador do produto o par de registradores BC que contém o primeiro operando e o par de registradores DE que contém o segundo. Caso o produto exceda 7FFFH, em qualquer instante durante a multiplicação, os dois operandos são convertidos para precisão simples(2FCBH) e o controle é transferido ao multiplicador de simples precisão(325CH). Caso contrário, os sinais originais serão restaurados, e, se forem diferentes, o produto será negado antes de ser colocado em DAC como um inteiro (321DH).

Endereço ..... 31E6H

Esta rotina é utilizada pelo Avaliador de Expressões para fazer uma divisão inteira(\) de dois operandos. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, com o resultado colocado em DAC. Se o segundo operando for zero, haverá um erro de “Divisão por zero” (4058H). Caso contrário os sinais dos dois operandos serão salvos e os dois operados serão tornados positivos(3215H). A divisão prossegue, utilizando-se o método padrão de “deslocamento e subtração” com o par de registradores HL contendo o resto, o par de registradores BC o segundo operando e o par de registradores DE o primeiro operando e o produto. Quando a divisão estiver completa os sinais iniciais serão restaurados e, se diferirem, o produto será negado antes de ser colocado em DAC como um inteiro(321FH).

Endereço ..... 3215H

Esta rotina é utilizada para tornar positivos dois inteiros binários com sinal, nos pares de registradores HL e DE. Os sinais dos dois operandos são devolvidos como um

indicador no bit 7 do registrador B: 0=iguais, 1=diferentes. Em seguida, cada operando será examinado e, se for negativo será tornado positivo, ao se subtrair de zero.

Endereço ..... 322BH

Esta rotina é utilizada pela função “ABS”, para tornar positivo um inteiro negativo contido em DAC. O operando é tomado de DAC, negado e depois devolvido a DAC(3221H). Se o valor do operando for 8000H, ele será convertido para simples precisão(2FCCH), pois não há inteiro de valor + 32768.

Endereço ..... 323AH

Esta rotina é utilizada pelo Avaliador de Expressões para realizar a operação “MOD” entre dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, com o resultado colocado em DAC. O sinal do primeiro operando é salvo e os dois operandos são divididos(31E6H). Uma vez que o resto é devolvido em dobro pelo processo de divisão, o par de registradores DE é deslocado uma posição para a direita, afim de restaurar o valor correto. O sinal do primeiro operando é, em seguida, restaurado, e se for negativo, o resto será negado antes de ser colocado em DAC como um inteiro(321DH).

Endereço ..... 324EH

Esta rotina é utilizada pelo Avaliador de Expressões para somar dois operandos de simples precisão. O primeiro operando está contido no registrador C, B, E, D e o segundo em DAC e o resultado é colocado em DAC. O primeiro operando é copiado em ARG(3280H), o segundo operando é convertido para precisão dupla(3042H) e o controle é transferido ao somador de dupla precisão(269AH).

Endereço ..... 3257H

Esta rotina é utilizada pelo Avaliador de Expressões para a subtração de dois operandos de simples precisão. O primeiro operando está contido nos registradores C, B, E, D e o segundo em DAC, com o resultado colocado em DAC. O segundo operando é negado(2E8DH) e o controle é transferido ao somador de simples precisão(324EH).

Endereço ..... 325CH

Esta rotina é utilizada pelo Avaliador de Expressões para multiplicar dois

operandos de precisão simples. O primeiro operando está contido nos registradores C, B, E, D e o segundo em DAC, com o resultado colocado em DAC. O primeiro operando é copiado no ARG(3280H), e o segundo operando é convertido para dupla precisão(3042H), com o controle transferido ao multiplicador de dupla precisão(27E6H).

Endereço . . . . . 3265H

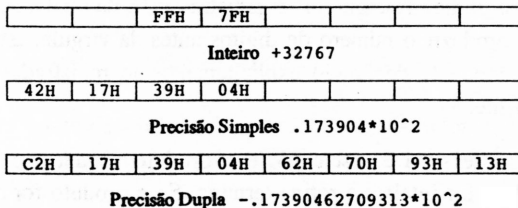
Esta rotina é utilizada pelo Avaliador de Expressões para dividir dois operandos de simples precisão. O primeiro operando está contido nos registradores C, B, E, D e o segundo em DAC, com o resultado colocado em DAC. O primeiro e o segundo operandos são trocados de modo que o primeiro fica em DAC e o segundo nos registradores. Em seguida, o segundo operando é copiado em ARG(3042H) e o controle transferido ao divisor de dupla precisão(289FH).

Endereço . . . . . 3280H

Esta rotina copia o operando de simples precisão contido nos registradores C, B, E e D em ARG, e depois zera os quatro bytes menos significativos.

Endereço . . . . . 3299H

Esta rotina converte um número na forma string em um dos tipos numéricos internos padrões e é utilizada durante a atomização e pelos manipuladores das instruções "VAL", "INPUT" e "READ". Na entrada, o par de registradores HL aponta para o primeiro caractere da string de texto a ser convertido. Na saída, o par de registradores HL aponta para o caractere que segue a string, o operando numérico está em DAC e o código de tipo em VALTYP. Exemplos dos três tipos:



**Figura 41** Tipos numéricos no DAC.

Um inteiro é um número binário de dezesseis bits armazenado em DAC+2 no formato de complemento de dois, LSB primeiro, MSB em segundo. Um inteiro pode variar de 8000H(-32786) até 7FFFH(+ 32767).

Um número em ponto flutuante consiste em um byte de expoente e uma mantissa de três ou sete bytes. O expoente é mantido no formato binário com sinal e pode variar de 01H(-63), passando pelo 40H(0) até 7FH(+63), com o valor especial de 00H utilizado para o número zero. Estes valores de expoente são para uma mantissa normalizada. O Interpretador apresenta ao usuário números na forma exponencial, com um dígito antes da vírgula. Por essa razão, o expoente pode variar de E-64 até E+62. O bit 7 do byte do expoente contém o sinal da mantissa, 0 para positivo e 1 para negativo, com a mantissa em si mantida no formato BCD condensado com dois dígitos por byte. Note que o Interpretador utiliza o conteúdo de VALTYP para determinar o tipo do número, e não o formato do número em si.

A conversão se inicia pelo exame do primeiro caractere do texto. Se este for um "&" o controle é transferido à rotina especial de conversão de base(4EB8H); se for um caractere de sinal, será salvo temporariamente. Em seguida, caracteres numéricos sucessivos são tomados e adicionados ao produto inteiro, que vai sendo multiplicado por dez cada vez que um novo dígito é encontrado. Se o valor do produto exceder 32767, ou for encontrado um ponto decimal, o produto será convertido para simples precisão e quaisquer caracteres adicionais serão colocados diretamente no DAC. Se for encontrado um sétimo dígito, o produto será mudado para precisão dupla, e se forem encontrados mais do que quatorze dígitos o excesso será lido, mas ignorado.

A conversão termina quando é encontrado um caractere não-numérico. Se este for um caractere de definição de tipo ("%", "#" ou "!"), será chamada a rotina de conversão apropriada e o controle será transferido ao ponto de saída(331EH). Se for um prefixo de expoente("E", "e", "D" ou "d"), uma das rotinas de conversão também será utilizada e, em seguida, os dígitos seguintes serão convertidos em um expoente binário no registrador E. No ponto de saída(331EH) o tipo do produto é verificado por meio da rotina padrão GETYPR. Se for precisão simples ou dupla, o expoente será calculado subtraindo primeiro o número de dígitos fracionário no registrador B, da contagem total de dígitos no registrador D, para produzir o número de dígitos antes da vírgula. Isto, em seguida, será somado a qualquer expoente declarado explicitamente no registrador E, e colocado em DAC+0 como expoente.

O caractere de sinal é restaurado e o produto negado, se houver necessidade (2E86H). Se o produto for inteiro, a rotina termina. Se o produto for de precisão simples, o controle termina pela verificação do valor especial de -32768(2FA2H). Se o produto for de dupla precisão o controle termina pelo arredondamento do décimo-quinto dígito(273CH).



Endereço . . . . . 340AH

Esta rotina é utilizada pelo manipulador de erro para apresentar a mensagem “in”(6678H) seguida pelo número da linha fornecido no par de registradores HL(3412H).

Endereço . . . . . 3412H

Esta rotina apresenta o inteiro binário sem sinal fornecido pelo par de registradores HL. O operando é colocado em DAC como um inteiro(2F99H), convertido em texto(3441H) e depois apresentado(6677H).

Endereço . . . . . 3425H

Esta rotina converte o operando numérico contido em DAC em uma string em FBUFFR. O endereço do primeiro caractere do texto resultante é devolvido no par de registradores HL, e o texto termina com um byte zero. Primeiro, o operando é convertido para dupla precisão(375FH). Os dígitos BCD da mantissa são, em seguida, desempacotados, convertidos em ASCII e colocados em FBUFFR(36B3H). A posição do ponto decimal é determinada pelo expoente, por exemplo:

$.999 * 10^+2 = 99.9$

$.999 * 10^+1 = 9.99$

$.999 * 10^+0 = .999$

$.999 * 10^{-1} = .0999$

Se o expoente estiver fora da faixa  $10^{-1}$  até  $10^{14}$ , o número será apresentado na forma exponencial. Neste caso, o ponto decimal será colocado depois do primeiro dígito, o expoente será convertido do binário e colocado após a mantissa.

Existe um ponto de entrada alternativo para esta rotina em 3426H, e é usado pelo manipulador da instrução “PRINT USING”. Com este ponto de entrada o número de caracteres antes do ponto decimal, será fornecido no registrador B, o número de caracteres depois do ponto no registrador C e um byte de formatação no registrador A:

| 7 | 6 | 5 | 4  | 3 | 2     | 1 | 0   |
|---|---|---|----|---|-------|---|-----|
| 1 | , | * | \$ | + | Sinal | 0 | ^^^ |

**Figura 42** Byte de formatação.

A operação neste modo é muito parecida com a do modo normal, mas apresenta recursos adicionais. Assim que o operador tiver sido convertido para dupla precisão, a forma exponencial será considerada se o bit 0 do byte de formatação estiver ativado. A mantissa será deslocada para a direita, em DAC, e arredondada para que os dígitos, depois do ponto decimal que estiverem sobrando, sejam eliminados(377BH). Assim que a mantissa for convertida em ASCII(36B3H), vírgulas serão colocadas nos pontos apropriados, se o bit 6 do byte de formatação estiver ativado. Durante a formatação pós-conversão (351CH), posições antes do ponto não-utilizadas serão preenchidas com asteriscos se o bit 5 estiver ativado, e um cifrão será colocado se o bit 4 estiver ativado. O bit 3 põe um sinal “+” nos números positivos quando é ativado, caso contrário será utilizado um espaço. Se o bit 2 estiver ligado, o sinal do número será colocado no final deste. Se estiver desligado, na frente.

O ponto de entrada desta rotina em 3441H é utilizado para converter inteiros sem sinal, como os números de linha, em uma string. Por exemplo: 9000H, quando tratado como um inteiro normal, seria convertido para -28672. Utilizando este ponto de entrada será produzido a string 36864. O operando será convertido por divisões sucessivas com os fatores 10000, 1000, 100, 10 e 1 e os dígitos resultantes serão colocados em FBUFFER (36DBH).

Endereço ..... 3710H

Esta tabela contém as cinco constantes utilizadas pela rotina de saída numérica: 10000, 1000, 100, 10, 1.

Endereço ..... 371AH

Esta rotina é utilizada pela função “BIN\$” para converter um operando numérico contido no DAC em string. O registrador B é carregado com o tamanho de grupo (1) e o controle é transferido à rotina de conversão geral(3724H).

Endereço ..... 371EH

Esta rotina é utilizada pela função “OCT\$” para converter um operando numérico contido no DAC em string. O registrador B é carregado com o tamanho de grupo (3) e o controle é transferido à rotina de conversão geral(3724H).

Endereço ..... 3722H

Esta rotina é utilizada pela função "HEX\$" para converter um operando numérico contido em DAC para string. O registrador B é carregado com tamanho de grupo (4) e o operando convertido a um inteiro binário no par de registradores HL(5439H). Grupos sucessivos de 1, 3 ou 4 bits são deslocados para a direita, para fora do operando, convertidos em dígitos ASCII e colocados no FBUFFR. Quando o operando ficar valendo zero, a rotina terminará com o endereço do primeiro caractere de texto, no par de registradores HL, e a string terminará com um byte zero.

Endereço ..... 3752H

Esta rotina é utilizada durante uma saída numérica para devolver o número de dígitos de um operando no registrador B, e o endereço de seu byte menos significativo no par de registradores HL. Para precisão simples B=6 e HL=DAC+3; para dupla precisão B=14 e HL=DAC+7.

Endereço ..... 375FH

Esta rotina é utilizada durante uma saída numérica para converter o operando numérico no DAC para dupla precisão(303AH).

Endereço ..... 377BH

Esta rotina é utilizada durante uma saída numérica para deslocar a mantissa no DAC para a direita(27DBH). O inverso do número de dígitos será fornecido no registrador A. O resultado será arredondado para cima, a partir do dígito quinze(2741H).

Endereço ..... 37A2H

Esta rotina é utilizada durante uma saída numérica. Ela retoma o inverso do número de dígitos fracionários em um operando em ponto flutuante. Isto é computado subtraindo-se o expoente do número de dígitos do operando(6 ou 14).

Endereço ..... 37B4H

Esta rotina é utilizada durante a saída numérica para localizar o último dígito não-zero na mantissa contida no DAC. Seu endereço é devolvido no par de registradores HL.

Endereço . . . . . 37C8H

Esta rotina é utilizada pelo Avaliador de Expressões para realizar uma exponenciação (^) entre dois operandos de precisão simples. O primeiro operando está contido nos registradores C, B, E, D e o segundo em DAC, com o resultado colocado em DAC. O primeiro operando será copiado em ARG(3280H), empilhado(2CC7H) e intercambiado com DAC(2C6FH). Em seguida, o segundo operando será desempilhado e colocado em ARG e o controle irá para a rotina da exponenciação de precisão dupla.

Endereço . . . . . 37D7H

Esta rotina é utilizada pelo Avaliador de Expressões para realizar uma exponenciação (^) entre dois operandos de dupla precisão. O primeiro operando está contido em DAC e o segundo em ARG, sendo o resultado colocado em DAC. O resultado é, normalmente, computado utilizando:

$$X^P = \text{EXP}(P * \text{LOG}(X))$$

Uma alternativa, muito mais rápida, é possível se o operando P for um inteiro. Isto é testado extraindo-se a parte inteira deste operando e verificado se ele ainda se mantém igual ao valor original(391AH). Um resultado positivo neste teste significa que um método mais rápido poderá ser utilizado. Este método é descrito abaixo.

Endereço . . . . . 383FH

Esta rotina é utilizada pelo Avaliador de Expressões para realizar uma exponenciação (^) entre dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, sendo o resultado colocado em DAC. A rotina funciona pelo desdobramento do problema em multiplicações simples:

$$6^{\wedge}13 = 6^{\wedge}1101 = (6^{\wedge}8) * (6^{\wedge}4) * (6^{\wedge}1)$$

Quando o operando correspondente ao expoente, está na forma binária é suficiente um deslocamento simples para a direita, afim de determinar se um determinado produto intermediário precisará ser incluído no resultado. Os produtos intermediários em si são obtidos por multiplicação cumulativa do operando, cada vez que se passa pelo laço de computação. Caso o produto extravase, em qualquer momento, será convertido para simples precisão. Finalmente, o operando correspondente ao expoente é verificado, e, se for negativo, o resultado ficará valendo a recíproca do produto pois:  $X^{\wedge}-P = 1/X^{\wedge}P$ .

Endereço ..... 390DH

Esta rotina é utilizada durante uma exponenciação para a multiplicação de dois inteiros(3193H), devolvendo Indicador NZ se o resultado tiver extravasado a precisão simples.

Endereço ..... 391AH

Esta rotina é utilizada durante a exponenciação para verificar se um operando expoente de precisão dupla consiste apenas em uma parte inteira, devolvendo Indicador NC em caso afirmativo.

Endereço ..... 392EH

Esta tabela de endereço é utilizada pela Ronda de Execução do Interpretador para encontrar o manipulador correspondente ao átomo de uma instrução. As instruções correspondentes aos endereços foram incluídas aqui, apesar de não estarem presentes na tabela em ROM:

| ENDEREÇO | INSTRUÇÃO | ENDEREÇO | INSTRUÇÃO | ENDEREÇO | INSTRUÇÃO |
|----------|-----------|----------|-----------|----------|-----------|
| 63EAH    | END       | 00C3H    | CLS       | 5B11H    | CIRCLE    |
| 4524H    | FOR       | 51C9H    | WIDTH     | 7980H    | COLOR     |
| 6527H    | NEXT      | 485DH    | ELSE      | 5D6EH    | DRAW      |
| 485BH    | DATA      | 6438H    | TRON      | 59C5H    | PAINT     |
| 4B6CH    | INPUT     | 6439H    | TROFF     | 00C0H    | BEEP      |
| 5E9FH    | DIM       | 643EH    | SWAP      | 73E5H    | PLAY      |
| 4B9FH    | READ      | 6477H    | ERASE     | 57EAH    | PSET      |
| 4880H    | LET       | 49AAH    | ERROR     | 57E5H    | PRESET    |
| 47E8H    | GOTO      | 495DH    | RESUME    | 73CAH    | SOUND     |
| 479EH    | RUN       | 53E2H    | DELETE    | 79CCH    | SCREEN    |
| 49E5H    | IF        | 49B5H    | AUTO      | 7BE2H    | VPOKE     |
| 63C9H    | RESTORE   | 5468H    | RENUM     | 7A48H    | SPRITE    |
| 47B2H    | GOSUB     | 4718H    | DEFSTR    | 7B37H    | VDP       |
| 4821H    | RETURN    | 471BH    | DEFINT    | 7B5AH    | BASE      |
| 485DH    | REM       | 471EH    | DEFSNG    | 55A8H    | CALL      |
| 63E3H    | STOP      | 4721H    | DEFDBL    | 7911H    | TIME      |
| 4A24H    | PRINT     | 4B0EH    | LINE      | 786CH    | KEY       |
| 64AFH    | CLEAR     | 6AB7H    | OPEN      | 7E4BH    | MAX       |
| 522EH    | LIST      | 7C52H    | FIELD     | 73B7H    | MOTOR     |
| 6286H    | NEW       | 775BH    | GET       | 6EC6H    | BLOAD     |
| 48E4H    | ON        | 7758H    | PUT       | 6E92H    | BSAVE     |
| 401CH    | WAIT      | 6C14H    | CLOSE     | 7C16H    | DSKOS     |
| 501DH    | DEF       | 6B5DH    | LOAD      | 7C1BH    | SET       |
| 5423H    | POKE      | 6B5EH    | MERGE     | 7C20H    | NAME      |
| 6424H    | CONT      | 6C2FH    | FILES     | 7C25H    | KILL      |
| 6FB7H    | CSAVE     | 7C48H    | LSET      | 7C2AH    | IPL       |
| 703FH    | CLOAD     | 7C4DH    | RSET      | 7C2FH    | COPY      |
| 4016H    | OUT       | 6BA3H    | SAVE      | 7C34H    | CMD       |
| 4A1DH    | LPRINT    | 6C2AH    | LFILES    | 7766H    | LOCATE    |
| 5229H    | LLIST     |          |           |          |           |

Endereço ..... 39DEH

Esta tabela de endereços é utilizado pelo Avaliador de Fatores para encontrar o manipulador correspondente ao átomo de uma função. As funções correspondentes foram incluídas aqui, apesar de não estarem presentes na tabela em ROM:

| ENDEREÇO | FUNÇÃO  | ENDEREÇO | FUNÇÃO   | ENDEREÇO | FUNÇÃO |
|----------|---------|----------|----------|----------|--------|
| 6861H    | LEFT\$  | 4FCCH    | POS      | 30BEH    | FIX    |
| 6891H    | RIGHT\$ | 67FFH    | LEN      | 7940H    | STICK  |
| 689AH    | MID\$   | 6604H    | STR\$    | 794CH    | STRIG  |
| 2E97H    | SGN     | 68BBH    | VAL      | 795AH    | PDL    |
| 30CFH    | INT     | 680BH    | ASC      | 7969H    | PAD    |
| 2E82H    | ABS     | 681BH    | CHR\$    | 7C39H    | DSKF   |
| 2AFFH    | SQR     | 541CH    | PEEK     | 6D39H    | FPOS   |
| 2BDFH    | RND     | 7BF5H    | VPEEK    | 7C66H    | CVI    |
| 29ACH    | SIN     | 6846H    | SPACES\$ | 7C6BH    | CVS    |
| 2A72H    | LOG     | 65F5H    | OCT\$    | 7C70H    | CVD    |
| 2B4AH    | EXP     | 65FAH    | HEX\$    | 6D25H    | EOF    |
| 2993H    | COS     | 4FC7H    | LPOS     | 6D03H    | LOC    |
| 29FBH    | TAN     | 65FFH    | BIN\$    | 6D14H    | LOF    |
| 2A14H    | ATN     | 2F8AH    | CINT     | 7C57H    | MKI\$  |
| 69F2H    | FRE     | 2FB2H    | CSNG     | 7C5CH    | MKS\$  |
| 4001H    | INF     | 303AH    | CDBL     | 7C61H    | MKD\$  |

Endereço ..... 3A3EH

Esta tabela de endereço é utilizada durante a atomização de um programa, como um índice para a tabela de palavras-chaves do BASIC(3A72H). Cada uma das vinte e seis entradas apontam para o endereço inicial de um dos sub-blocos de palavras-chaves. A primeira entrada aponta para as palavras-chaves começadas por "A", a segunda aponta para aqueles começados por "B" e assim por diante.

|             |             |             |
|-------------|-------------|-------------|
| 3A72H ... A | 3B9FH ... J | 3C8EH ... S |
| 3A88H ... B | 3BA0H ... K | 3CDBH ... T |
| 3A9FH ... C | 3BA8H ... L | 3CF6H ... U |
| 3AF3H ... D | 3BE8H ... M | 3CFFH ... V |
| 3B2EH ... E | 3C09H ... N | 3D16H ... W |
| 3B4FH ... F | 3C18H ... O | 3D20H ... X |
| 3B69H ... G | 3C2BH ... P | 3D24H ... Y |
| 3B7BH ... H | 3C5DH ... Q | 3D25H ... Z |
| 3B80H ... I | 3C5EH ... R |             |

Endereço ..... 3A72H

Esta tabela contém as palavras-chaves do BASIC com os átomos correspondentes. Cada um dos vinte e seis blocos da tabela contém todas as palavras-chaves que começam com uma determinada letra, terminando com um byte zero. Cada palavra-chave é armazenada como texto, com o bit 7 ligado no último caractere, seguido imediatamente pelo átomo correspondente. O primeiro caractere da palavra-chave não precisa ser arma-

zenado, pois ele é implícito pela posição da palavra na tabela. As palavras-chaves e os átomos estão relacionados a seguir, em sua totalidade. Observe que os blocos correspondentes às letras "J", "Q", "Y" e "Z" estão vazios.

|        |     |         |     |         |     |          |     |
|--------|-----|---------|-----|---------|-----|----------|-----|
| AUTO   | A9H | DSKF    | 26H | LIST    | 93H | REM      | 8FH |
| AND    | F6H | DRAW    | BEH | LFILS   | BBH | RESUME   | A7H |
| ABS    | 06H | ELSE    | A1H | LOG     | 0AH | RSET     | B9H |
| ATN    | 0EH | END     | 81H | LOC     | 2CH | RIGHT\$  | 02H |
| ASC    | 15H | ERASE   | A5H | LEN     | 12H | RND      | 08H |
| ATTR\$ | E9H | ERROR   | A6H | LEFT\$  | 01H | RENUM    | AAH |
| BASE   | C9H | ERL     | E1H | LOF     | 2DH | SCREEN   | C5H |
| BSAVE  | D0H | ERR     | E2H | MOTOR   | CEH | SPRITE   | C7H |
| BLOAD  | CFH | EXP     | 0BH | MERGE   | B6H | STOP     | 90H |
| BEEP   | C0H | EOF     | 2BH | MOD     | FBH | SWAP     | A4H |
| BIN\$  | 1DH | EQV     | F9H | MKI\$   | 2EH | SET      | D2H |
| CALL   | CAH | FOR     | 82H | MKS\$   | 2FH | SAVE     | BAH |
| CLOSE  | B4H | FIELD   | B1H | MKD\$   | 30H | SPC(     | DFH |
| COPY   | D6H | FILES   | B7H | MID\$   | 03H | STEP     | DCH |
| CONT   | 99H | FN      | DEH | MAX     | CDH | SGN      | 04H |
| CLEAR  | 92H | FRE     | 0FH | NEXT    | 83H | SQR      | 07H |
| CLOAD  | 9BH | FIX     | 21H | NAME    | D3H | SIN      | 09H |
| CSAVE  | 9AH | FPOS    | 27H | NEW     | 94H | STR\$    | 13H |
| CSRLIN | E8H | GOTO    | 89H | NOT     | E0H | STRING\$ | E3H |
| CINT   | 1EH | GO TO   | 89H | OPEN    | B0H | SPACE\$  | 19H |
| CSNG   | 1FH | GOSUB   | 8DH | OUT     | 9CH | SOUND    | C4H |
| CDBL   | 20H | GET     | B2H | ON      | 95H | STICK    | 22H |
| CVI    | 28H | HEX\$   | 1BH | OR      | F7H | STRIG    | 23H |
| CVS    | 29H | INPUT   | 85H | OCT\$   | 1AH | THEN     | DAH |
| CVD    | 2AH | IF      | 8BH | OFF     | EBH | TRON     | A2H |
| COS    | 0CH | INSTR   | E5H | PRINT   | 91H | TROFF    | A3H |
| CHR\$  | 16H | INT     | 05H | PUT     | B3H | TAB(     | DBH |
| CIRCLE | BCH | INP     | 10H | POKE    | 98H | TO       | D9H |
| COLOR  | BDH | IMP     | FAH | POS     | 11H | TIME     | CBH |
| CLS    | 9FH | INKEY\$ | ECH | PEEK    | 17H | TAN      | 0DH |
| CMD    | D7H | IPL     | D5H | PSET    | C2H | USING    | E4H |
| DELETE | A8H | KILL    | D4H | PRESET  | C3H | USR      | DDH |
| DATA   | 84H | KEY     | CCH | POINT   | EDH | VAL      | 14H |
| DIM    | 86H | LPRINT  | 9DH | PAINT   | BFH | VARPTR   | E7H |
| DEFSTR | ABH | LLIST   | 9EH | PDL     | 24H | VDP      | CBH |
| DEFINT | ACH | LPOS    | 1CH | PAD     | 25H | VPOKE    | C6H |
| DEFSNG | ADH | LET     | 88H | PLAY    | C1H | VPEEK    | 18H |
| DEFDBL | AEH | LOCATE  | D8H | RETURN  | 8EH | WIDTH    | A0H |
| DSK\$  | DIH | LINE    | AFH | READ    | 87H | WAIT     | 96H |
| DEF    | 97H | LOAD    | B5H | RUN     | 8AH | XOR      | F8H |
| DSKI\$ | EAH | LSET    | B8H | RESTORE | 8CH |          |     |

Endereço ..... 3D26H

Esta tabela de vinte e um bytes é utilizada pelo Interpretador durante a atomização de um programa. Ela contém as dez palavras-chaves compostas de um único caractere e seus átomos:

+ ... F1H \*... F3H ^... F5H '... E6H =... EFH  
 -... F2H /... F4H \... FCH >... EEH <... FOH

Endereço ..... 3D3BH

Esta tabela é utilizada pelo Avaliador de Expressão para determinar o nível de precedência para um determinado operador infixo. Quanto maior o valor na tabela, tanto maior a precedência do operador. Não estão incluídas as precedências para os operadores relacionais(64H), o operador "NOT"(5AH) e o operador "NOT"(7DH), pois são definidas diretamente pelos Avaliadores de Expressões e de Fatores.

|      |      |     |      |       |     |
|------|------|-----|------|-------|-----|
| 79 H | ...  | +   | 46 H | ....  | OR  |
| 79 H | .... | -   | 3 CH | ...   | XOR |
| 7 CH | .... | *   | 32 H | ...   | EQV |
| 7 CH | .... | /   | 28 H | ...   | IMP |
| 7 FH | ...  | \   | 7 AH | ..    | MOD |
| 50H  | .    | AND | 7 BH | ..... | \   |

Endereço ..... 3D47H

Esta tabela é utilizada para converter o resultado de uma função definida pelo usuário no mesmo tipo que a Variável utilizada na definição da função. Ela contém os endereços das rotinas de conversões de tipo:

|       |       |                           |
|-------|-------|---------------------------|
| 303AH | ..... | CEBL                      |
| 0000H | ..... | Não utilizado             |
| 2F8AH | ..... | CINT                      |
| 3058H | ..... | Verifica se é tipo string |
| 2FB2H | ..... | CSNG                      |

Endereço ..... 3D51H

Esta tabela de endereços é utilizada pelo Avaliador de Expressões para encontrar o manipulador de um determinado operador matemático infixo, quando os dois operandos são de precisão dupla:

|       |       |         |
|-------|-------|---------|
| 269AH | ..... | +       |
| 268CH | ..... | -       |
| 27E6H | ..... | *       |
| 289FH | ..... | /       |
| 37D7H | ..... | ^       |
| 2F83H | ..... | Relação |



Endereço ..... 3D5DH

Esta tabela de endereço é utilizada pelo Avaliador de Expressões para encontrar o manipulador de um determinado operador matemático infix, quando os dois operandos são de precisão simples:

|       |       |         |
|-------|-------|---------|
| 324EH | ..... | +       |
| 3257H | ..... | -       |
| 325CH | ..... | *       |
| 3267H | ..... | /       |
| 37C8H | ..... | ^       |
| 2F21H | ..... | Relação |

Endereço ..... 3D69H

Esta tabela de endereços é utilizada pelo Avaliador de Expressões para encontrar o manipulador de um determinado operador matemático infix, quando ambos os operandos são inteiros:

|       |       |         |
|-------|-------|---------|
| 3172H | ..... | +       |
| 3167H | ..... | -       |
| 3193H | ..... | *       |
| 4DB8H | ..... | /       |
| 383FH | ..... | ^       |
| 2F4DH | ..... | Relação |

Endereço ..... 3D75H

Esta tabela contém as mensagens de erro do Interpretador, sendo que cada uma é armazenada como texto, terminado com um byte zero. Os códigos de erro associados são mostrados abaixo apenas para referência, pois não fazem parte da tabela:

|                           |                         |
|---------------------------|-------------------------|
| 01 NEXT without FOR       | 19 Device I/O error     |
| 02 Syntax error           | 20 Verify error         |
| 03 RETURN without GOSUB   | 21 No RESUME            |
| 04 Out of DATA            | 22 RESUME without error |
| 05 Illegal function call  | 23 Unprintable error    |
| 06 Overflow               | 24 Missing operand      |
| 07 Out of memory          | 25 Line buffer overflow |
| 08 Undefined line number  | 50 FIELD overflow       |
| 09 Subscript out of range | 51 Internal error       |
| 10 Redimensioned array    | 52 Bad file number      |
| 11 Division by zero       | 53 File not found       |
| 12 Illegal direct         | 54 File already open    |

|                               |                             |
|-------------------------------|-----------------------------|
| 13 Type mismatch              | 55 Input past end           |
| 14 Out of string space        | 56 Bad file name            |
| 15 String too long            | 57 Direct statement in file |
| 16 String formula too complex | 58 Sequential I/O only      |
| 17 Can't CONTINUE             | 59 File not OPEN            |
| 18 Undefined user function    |                             |

Endereço ..... 3FD2H

Esta é a mensagem "in" terminada por um byte zero.

Endereço ..... 3FD7H

Esta é a mensagem "Ok", CR, LF, terminada por um byte zero.

Endereço ..... 3FDCH

Esta é a mensagem "Break" terminada por um byte zero.

Endereço ..... 3FE2H

Esta rotina procura um bloco de parâmetros do laço "FOR", na pilha do Z80. O endereço da Variável de laço é fornecido no par de registradores DE. A procura é iniciada quatro bytes acima do SP atual do Z80, pulando o endereço de retorno ao chamador e o endereço de retorno à Ronda de Execução. Se não houver nenhum átomo "FOR"(82H), a rotina encerra com Indicador NZ. Se um átomo "FOR" for encontrado, o endereço da Variável do laço é lido do bloco de parâmetros e verificado. A rotina termina com o Indicador Z, se este for o bloco correto, tendo o par de registradores HL apontando para o byte de tipo do bloco de parâmetros. Caso contrário, a procura continua vinte e dois bytes para cima, no próximo bloco de parâmetros.

Endereço ..... 4001H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "INP" a um operando contido em DAC. O número da porta é verificado(5439H), a porta lida e o resultado colocado em DAC como um inteiro(4FCFH).

Endereço ..... 400BH

Esta rotina avalia um operando na faixa -32768 a + 65535(542FH) e o coloca no par de registradores BC. Após verificar a existência de uma vírgula, por meio da rotina

padrão SYNCHR esta rotina avalia um segundo operando na faixa 0 a 255(521CH) e o coloca no registrador A.

Endereço . . . . . 4016H

Este é o manipulador da instrução "OUT". O número da porta e o byte de dado são avaliados(400BH) e o byte de dado escrito na porta Z80 correspondente.

Endereço . . . . . 401CH

Este é o manipulador da instrução "WAIT". O número da porta e os operadores "AND" são avaliados primeiro(400BH) seguidos pelo operando "XOR" opcional (521CH). Em seguida, a porta é repetidamente lida e os operandos aplicados, XOR e depois AND, até que um resultado não-nulo seja obtido. Ao contrário do que informam alguns manuais MSX, o laço pode ser interrompido pela tecla CTRL-STOP, assim que a rotina padrão CKCNTC for chamada pela presente rotina.

Endereço . . . . . 4039H

Esta rotina é utilizada pela Ronda de Exceção, se o final de programa for encontrado dentro do Modo de Programa. ONEFLG é verificado para ver se ainda contém um código de erro ativo. Em caso afirmativo será gerado um erro "No RESUME". Caso contrário, o programa é terminado normalmente(6401H). A idéia por trás desta rotina é pegar qualquer manipulador "ON ERROR", sem uma instrução "RESUME" no final.

Endereço . . . . . 404FH

Esta rotina é utilizada pelo manipulador da instrução "READ", ao ser encontrado um erro na instrução "DATA". O número da linha contido em DATLIN é copiado em CURLIN, de modo que o manipulador de erro indicará a linha de "DATA" como instrução ilegal, em lugar da linha de programa. O controle, em seguida, irá para o gerador de "Erro de sintaxe" (synter error).

Endereço . . . . . 4055H

Este é um grupo de nove geradores de erro. O registrador E tem o código de erro e o controle passa para o manipulador de erro:

**END            ERRO**

|       |                                                          |
|-------|----------------------------------------------------------|
| 4055H | Erro de sintaxe (syntax error)                           |
| 4058H | Divisão por zero (Division by zero)                      |
| 405BH | NEXT sem FOR (NEXT without FOR)                          |
| 405EH | Matriz redimensionada (Redimensioned array)              |
| 4061H | Função de usuário não definida (Undefined user function) |
| 4064H | RESUME sem erro (RESUME without error)                   |
| 4067H | Erro de extravazamento (Overflow error)                  |
| 406AH | Operando faltante (Missing Operand)                      |
| 406DH | Tipo incorreto (Type mismatch)                           |

Endereço ..... 406F

Este é o manipulador de erro do Interpretador. Todos os geradores de erro são transferidos para cá como um código de erro no registrador E. VLZADR é verificado primeiro para saber se o manipulador da instrução "VAL" alterou o texto do programa, e, em caso afirmativo, o caractere original é restaurado de VLZDAT. O número da linha atual, em seguida, é copiado de CURLIN em ERRLIN e DOT e a pilha do Z80 é restaurada de SAVSTK(62F0H). O código de erro é colocado em ERRFLG, para ser utilizado pela função "ERR", e a posição no texto do programa atual copiada de SAVTXT para ERRTXT para ser utilizada pela instrução "RESUME". O número da linha do erro e a posição do texto do programa também são copiados em OLDLIN e OLDTXT, para serem utilizados pelo manipulador da declaração "CONT". Em seguida, ONELIN é verificado para saber se uma instrução "ON ERROR" anterior foi executada. Em caso afirmativo, e desde que nenhum código de erro já esteja ativo, o controle é transferido à Ronda de Execução(4620H) para que seja executada a parte do Programa BASIC que tratará do erro.

Caso contrário, o código do erro é utilizado para contar as mensagens na tabela de mensagens de erro em 3D75H, até que a correta seja encontrada. Um CR, LF é acionado(7323H) com a tela forçada a voltar ao modo texto por meio da rotina padrão TOTEXT. Em seguida, um código BELL é acionado e a mensagem de erro apresentada (6678H). Considerando que o Interpretador esteja no modo programa, em vez de modo direto, ela será seguida pelo número da linha(340AH) e o controle vai para o "OK".

Endereço ..... 411FH

Este é o ponto de reentrada na Ronda Principal do Interpretador para um programa que terminou. A tela é forçada ao modo texto por meio da rotina padrão TOTEXT, a impressora é limpa(7304H) e o buffer 0 de E/S fechado(6D7BH). Em seguida, um CR, LF é enviado para a tecla(7323H), a mensagem "OK" é apresentada (6678H) e o controle passa para a Ronda Principal.

Endereço . . . . . 4134H

Ronda Principal do Interpretador .CURLIN é colocado em FFFH para indicar modo direto e AUTFLG é verificado para saber se o modo "AUTO" está ativado. Em caso afirmativo, o próximo número de linha é tomado de AUTLIN e apresentado(3412H). A Área de Texto do Programa é pesquisado, em seguida, para verificar se esta linha já existe(4295H) e um asterisco ou um espaço é apresentado, conforme o caso.

A rotina padrão ISFLIO é, em seguida, utilizada para determinar se uma instrução "LOAD" está ativa. Em caso afirmativo, a linha do programa é lida do cassete(7374H). Caso contrário, é lida do console por meio da rotina padrão PINLIN. Se a linha estiver vazia ou a tecla CTRL-STOP tiver sido pressionada, o controle é devolvido ao começo da Ronda Principal(4134H), sem nenhuma outra ação. Caso a linha se inicie com um número de linha, este será convertido em um inteiro, sem sinal no par de registradores DE(4769H). Em seguida, a linha é atomizada e colocada em KBUF(42B2H). Caso nenhum número de linha tenha sido encontrado no começo da linha, o controle é transferido para a Ronda de Execução(6D48H), para que a instrução seja executada.

Supondo que a linha se inicie com um número de linha, ela será testada para ver se está vazia e o resultado temporariamente salvo. O número da linha é copiado em DOT e AUTLIN incrementado do valor contido em AUTINC. Caso AUTLIN agora exceda 65530, o modo "AUTO" é desligado. A Área de Texto do Programa é pesquisada(4295H) para ver se existe uma linha de mesmo número ou, caso este não exista, para encontrar a linha seguinte, em ordem crescente. Se não se encontrar nenhum número de linha igual, se a linha estiver vazia e o modo "AUTO" desligado será gerado um erro "Undefined line number" (481CH). Se for encontrado um número de linha igual, e se a linha estiver vazia e o modo "AUTO" ativado, a Ronda Principal simplesmente pulará para a próxima instrução(4237H).

Caso contrário, quaisquer apontadores na Área de Texto do Programa são reconvertidos para números de linha(54EAH) e qualquer linha de programa existente cancelada(5405H). Supondo que a nova linha de programa não se encontre vazia, é aberto um espaço do tamanho necessário na Área de Texto do Programa(6250H) e a linha de programa atomizado é copiada de KBUF.

A Área de Texto do Programa tem seus elos recalculados(4257H), as Áreas de Armazenamento Variáveis são limpas(629AH) e o controle transferido de volta ao início da Ronda Principal.

Endereço . . . . . 4253H

Esta rotina recalcula os elos da Área de Texto do Programa, depois de uma modificação no programa. Os dois primeiros bytes de cada linha do programa contêm o endereço inicial da linha seguinte, o que é chamado de elo. Apesar do elo aumentar a quantidade de armazenamento requerido por linha de programa o tempo necessário para que o Interpretador localize uma determinada linha é reduzido enormemente.

Um exemplo de uma linha de programa típica é mostrado abaixo. Neste caso a linha "10 print 9", localizado no início da Área de Texto do Programa(8001H):

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 09H | 80H | 0AH | 00H | 91H | 20H | 1AH | 00H |
|-----|-----|-----|-----|-----|-----|-----|-----|

Figura 43 Linha de Programa.

No exemplo acima o elo é armazenado na ordem normal do Z80(LSB, MSB), e é seguido imediatamente pelo número da linha em binário, armazenado na mesma ordem. A instrução em si é formada por um átomo "PRINT"(91H), um único espaço, o número nove e o caractere de fim de linha(00H). Detalhes adicionais do formato de armazenamento podem ser encontrados na rotina de atomização(42B2H).

Cada elo é recalculado simplesmente varrendo a linha até que seja encontrado o caractere de fim de linha. O processo está completo quando o final da Área de Armazenamento do Programa, marcado pelo elo especial 0000H, é alcançado.

Endereço . . . . . 4279H

Esta rotina é utilizada pelo manipulador da instrução "LIST" para pegar os operandos dos dois números de linha do texto do programa. Se o número da primeira linha estiver presente, será convertido em um inteiro, sem sinal, no par de registradores DE(475FH), senão será devolvido o valor default 0000H. Se o número da segunda linha estiver presente, ele tem que ser precedido pelo símbolo "-"(F2H) e será devolvido na pilha do Z80. Caso contrário, o valor default 65530 será devolvido. Em seguida, o controle vai para a rotina de busca no texto do programa, para encontrar a primeira linha de programa referenciada.

Endereço . . . . . 4295H

Esta rotina varre a Área de Texto do Programa buscando a linha de programa, cujo número de linha está no par de registradores DE. A partir do endereço contido em TXTTAB, cada linha de programa é examinada. Se um número de linha igual é encon-

trado, a rotina termina com os Indicadores Z, C e o par de registradores BC apontando para o começo da linha de programa. Caso seja encontrado um número de linha maior, a rotina termina com os Indicadores NZ, NC e, se o elo final for alcançado, a rotina terminará com os Indicadores Z, NC.

Endereço . . . . . 42B2H

Esta rotina é utilizada pela Ronda Principal do Interpretador para atomizar uma linha de texto. Na entrada, o par de registradores HL aponta para o primeiro caractere do texto em BUF. Na saída, a linha atomizada está em KBUF, o par de registradores BC contém seu comprimento e o par de registradores HL aponta para seu início.

Exceto após as aspas iniciais ou após as palavras-chaves "REM", "CALL" e "DATA", qualquer string de caracteres concordando com uma palavra-chave é substituído pelo átomo daquela palavra-chave. As letras minúsculas são passadas para maiúsculas para a comparação de palavras-chaves. O caractere "?" é substituído pelo átomo "PRINT" (91H) e o caractere "\"" por ":" (3AH), pelo átomo "REM"(8FH) e pelo átomo "!" (E6H). O átomo "ELSE"(A1H) é precedido por um separador de instruções(3AH). Quaisquer outros caracteres do texto serão copiados sem nenhuma alteração, exceto as letras minúsculas que serão convertidas para maiúsculas. Os átomos menores que 80H, átomos de função, não podem ser armazenados diretamente em KBUF, pois seriam interpretados como texto. Em vez disto, a sequência FFH, átomo+80H é utilizada.

As constantes numéricas são primeiro convertidas em um dos tipos padrões em DAC(3299H). Em seguida, são armazenadas de uma entre diversas maneiras, dependendo de seu tipo e magnitude. A idéia geral é minimizar a utilização de memória:

|                                       |                      |
|---------------------------------------|----------------------|
| OBH LSB MSB . . . . .                 | Número octal         |
| OCH LSM MSB . . . . .                 | Número hexadecimal   |
| 11H até 1AH . . . . .                 | Inteiros 0 a 9       |
| OFH LSB . . . . .                     | Inteiros 10 a 255    |
| 1CH LSB MSB . . . . .                 | Inteiros 256 a 32767 |
| 1DH EE DD DD DD . . . . .             | Precisão simples     |
| 1FH EE DD DD DD DD DD DD DD . . . . . | Dupla precisão       |

Não há nenhum átomo especial para os números binários; eles são mantidos como strings de caracteres. Isto parece ser um legado de versões anteriores do BASIC da Microsoft. Qualquer sinal colocado como prefixo de um número é visto como um operador e é armazenado como um átomo separado. Números negativos não são produzidos durante a atomização. Uma vez que os números de dupla precisão ocupam um espaço tão grande, uma

linha que contenha vários deles, por exemplo PRINT 1#, 1#, 1# etc. poderá provocar o “entupimento” de KBUF. Caso isto ocorra, será gerado um erro “Extravasamento do buffer de linha” (line buffer overflow).

Qualquer número, após um dos átomos das palavras-chaves contidas na tabela em 43B5H, é considerado um operando de número de linha e é armazenado com um átomo diferente:

|                   |                 |
|-------------------|-----------------|
| ODH LSB MSB ..... | Apontador       |
| OEH LSB MSB ..... | Número da linha |

Durante a atomização, apenas o tipo normal(OEH) é gerado. Quando um programa é realmente executado, estes operandos de número de linha são convertidos no tipo apontador de endereço(ODH).

Endereço ..... 43B5H

Esta tabela de átomos é utilizada durante a atomização para verificar as palavras-chaves que têm operandos de números de linha. Estas palavras-chaves são:

|         |        |
|---------|--------|
| RESTORE | RUN    |
| AUTO    | LIST   |
| RENUM   | LLIST  |
| DELETE  | GOTO   |
| RESUME  | RETURN |
| ERL     | THEN   |
| ELSE    | GOSUB  |

Endereço ..... 4524H

Este é o manipulador da instrução “FOR”. Primeiro, a Variável do laço é localizada recebendo seu valor inicial pelo manipulador “LET”(4880H). O endereço desta Variável de laço é devolvido no par de registradores DE. O fim da instrução é encontrado (485BH) e seu endereço colocado em ENDFOR. Em seguida, a pilha do Z80 é percorrida(3FE6H) e são procurados blocos de parâmetros que utilizem a mesma Variável de laço. Para cada um encontrado, o endereço ENDFOR atual é comparado com aquele do bloco de parâmetros. Caso ele seja igual, aquela seção da pilha é descartada. Isto é feito se houver algum laço incompleto como resultado de um “GOTO” de dentro do laço, de volta à declaração “FOR”.

O operando de terminação e o operando “STEP” são, em seguida, avaliados e convertidos ao mesmo tipo que a Variável de laço. Depois de verificar se há disponibi-



lidade de espaço na pilha, (625EH) um bloco de parâmetros de vinte e cinco bytes será empilhado na pilha do Z80. O bloco tem a seguinte estrutura:

|         |       |                               |
|---------|-------|-------------------------------|
| 2 bytes | ..... | Endereço ENDFOR               |
| 2 bytes | ..... | Número da linha atual         |
| 8 bytes | ..... | Valor de encerramento do laço |
| 8 bytes | ..... | Valor do STEP                 |
| 1 byte  | ..... | Tipo de laço                  |
| 1 byte  | ..... | Direção do STEP               |
| 2 bytes | ..... | Endereço da Variável do laço  |
| 1 byte  | ..... | Átomo FOR(82H)                |

O bloco de parâmetro permanece na pilha para ser utilizado pelo manipulador da instrução "NEXT" até que se alcance o fim, quando é descartado. O tamanho do bloco permanece constante mesmo que, para Variáveis de laço de precisão simples ou inteiras, não sejam necessários oito bytes para o valor de encerramento e de STEP. Neste caso, os bytes menos significativos são ignorados.

Note que o tipo de operação aritmética executada pelo manipulador da instrução "NEXT", e, portanto, a velocidade da execução do laço, depende inteiramente do tipo da Variável do laço e não no tipo de operando. Para execução mais rápida do programa, Variáveis inteiras como N%, por exemplo, deveriam ser utilizadas.

Endereço ..... 4601H

Esta é a Ronda de execução, para onde volta cada manipulador de instrução ao chegar ao final, para que o Interpretador possa passar para a próxima declaração. O valor atual do SP do Z80 é copiado em SAVSTK, com a finalidade de recuperar erros, e a tecla CTRL-STOP é verificada por meio da rotina padrão ISCNTC. Qualquer interrupção pendente é processada(6389H) e a posição do texto de programa atual, mantido no par de registradores HL durante toda a interpretação, é copiado em SAVTXT.

Em seguida, o caractere atual do programa é examinado, e se for um separador de instruções(3AH), o controle será imediatamente transferido ao ponto de execução (4640H). Se for qualquer outra coisa, que não um caractere de fim de linha(00H), será gerado um "Erro de sintaxe"(4055H), uma vez que há um texto espúrio no final da instrução. O par de registradores HL é avançado até o primeiro caractere da nova linha do programa e o elo examinado. Se este for zero o programa terminará(4039H). Caso contrário, o número de linha será tomado da linha nova e colocado em CURLIN. Se

TRCFLG for não-zero, o número da linha é apresentado(3412H) ladeado por colchetes, e o controle passará para o ponto de execução.

Endereço ..... 4640H

Este é o ponto de execução da Ronda de Execução. Um retorno para o início da Ronda de Execução(4601H) é colocado na pilha do Z80 e o primeiro caractere da nova instrução é lido por meio da rotina padrão CHRGT. Caso seja o caractere "sublinhado", (5FH) o controle é transferido ao manipulador da instrução "CALL"(55A7H). Se for menor que 81H, o menor átomo de instrução, o controle será transferido ao manipulador "LET"(4880H). Se for maior do que D8H, maior átomo de instrução, será verificado para vermos se é um dos átomos de função que podem funcionar como instrução(51ADH). Caso contrário, o endereço do manipulador é tomado na tabela em 392EH e colocado na pilha. Em seguida, o controle vai para a rotina padrão CHRGT, afim de pegar o próximo caractere do programa, antes que o controle seja transferido ao manipulador da instrução.

Endereço ..... 4666H

Nome ..... CHRGT

Entrada ..... HL aponta ao caractere atual do programa

Saída ..... A=Próximo caractere do programa

Modifica ..... AF, HL

Rotina padrão para pegar o próximo caractere do texto de programa. O par de registradores HL é incrementado e o caractere C colocado no registrador A. Se for um espaço, código TAB(09H) ou código LF(0AH), será pulado. Se for um separador de instruções(3AH) ou um caractere fim de linha(00H), a rotina terminará com os Indicadores Z, NC, caso seja um dígito de 0 a 9, a rotina termina com NZ, C. Caso seja qualquer caractere diferente dos átomos de prefixo numérico, a rotina termina com os Indicadores NZ, NC. Caso o caractere seja um dos átomos de prefixo numérico, ele é colocado em CONSAV e o operando copiado em CONLO. O código de tipo é colocado em CONTYP e o endereço do caractere seguinte em CONXT.

Endereço ..... 46E8H

Esta rotina é utilizada pelo Avaliador de Fatores, durante a desatomização, para recuperar um operando numérico quando um dos átomos de prefixo é devolvido pela rotina padrão CHRGT. O átomo do prefixo, primeiro é tomado de CONSAV, e se for alguma coisa diferente de um átomo de número de linha ou apontador, o operando será

copiado de CONLO para DAC e o código de tipo copiado de CNTYP para VALTYP. Se for um número de linha, será convertido para precisão simples e colocado em DAC (3236H). Se for um apontador, o número de linha original é recuperado da linha de programa apontada, convertido para precisão simples e colocado no DAC(3236H).

Endereço ..... 4718H

Este é o manipulador da instrução "DEFSTR". O registrador E é carregado com o código do tipo string(03H) e o controle é passado para a rotina geral de definição do tipo.

Endereço ..... 471BH

Este é o manipulador de instrução "DEFINT". O registrador E é carregado com o código do tipo inteiro(02H) e o controle é passado para a rotina geral de definição de tipo.

Endereço ..... 471EH

Este é o manipulador da instrução "DEFSNG". O registrador E é carregado com o código do tipo precisão simples(04H) e o controle é passado para a rotina geral de definição de tipo.

Endereço ..... 4721H

Este é o manipulador da instrução "DEFDBL". O registrador E é carregado com o código do tipo dupla precisão(08H) e o primeiro caractere definição de limite verificado(64A7H). Se não for alfabético e maiúsculo, será gerado um "erro de sintaxe"(4055H). Se o token "-"(F2H) for o caractere seguinte, o segundo caractere de definição de limites será lido e verificado(64A7H). A diferença entre os dois determinará o número de entradas em DEFTBL que serão preenchidas com o código de tipo.

Endereço ..... 4755H

Esta rotina avalia um operando e o converte em um inteiro, no par de registradores DE(520FH). Se o operando for negativo será gerado um erro de "Chamada ilegal à uma função".

Endereço ..... 475FH

Esta rotina é utilizada pelos manipuladores das instruções mostradas na tabela em 43B5H para pegar um único operando de número de linha do texto do programa e convertê-lo em um inteiro, sem sinal no par de registradores DE. Se o primeiro caractere no texto for um “.” (2EH), a rotina terminará com o conteúdo de DOT. Se for um dos átomos de número de linha(0DH) ou (0EH), a rotina terminará com o conteúdo de CONLO. Caso contrário, dígitos sucessivos serão tomados e somados ao produto, com multiplicações apropriadas por dez, até que seja encontrado um caractere não-numérico.

Endereço ..... 479EH

Este é o manipulador da instrução “RUN”. Se nenhum operando de número de linha estiver presente no texto do programa, o sistema será limpo(629AH) e o controle será entregue à Ronda de Execução, com o par de registradores HL apontando para o início da Área de Armazenamento de Programa. Se um operando de número de linha estiver presente, o sistema será limpo(62A1H) e o controle transferido ao manipulador da instrução “GOTO” (47E7H). Caso contrário, será considerado um nome de arquivo, por exemplo RUN “CAS:FILE”, e o controle será transferido ao manipulador da declaração “LOAD”.

Endereço ..... 47B2H

Este é o manipulador da instrução “GOSUB”. Depois de verificar se há espaço de pilha disponível(625EH), o operando do número de linha é coletado e colocado no par de registradores DE (4769H). O bloco de parâmetros de sete bytes, em seguida, é colocado na pilha e o controle transferido ao manipulador de “GOTO” (47EBH). O bloco de parâmetro é feito do seguinte modo:

|         |       |                              |
|---------|-------|------------------------------|
| 2 bytes | ..... | Endereço do fim da instrução |
| 2 bytes | ..... | Número da linha atual        |
| 2 bytes | ..... | 0000H                        |
| 1 byte  | ..... | Átomo de GOSUB(8DH)          |

O bloco de parâmetros permanece na pilha, até que seja executada uma instrução “RETURN”. Aí, ele será utilizado para determinar a posição original no texto do programa e descartado.

Endereço ..... 47CFH

Esta rotina é utilizada pelo processador de interrupção da Ronda de Execução (6389H), para a criação de um bloco de parâmetros do tipo "GOSUB" na pilha do Z80. Um bloco de interrupção é idêntico a um bloco normal, exceto que os dois bytes zero mostrados acima são substituídos pelo endereço da entrada do dispositivo em TRPTBL. Este endereço será utilizado pelo manipulador da instrução "RETURN", para atualizar o status de interrupção do dispositivo, assim que uma sub-rotina tenha terminado. Depois de empilhar o bloco de parâmetro, o controle é transferido à Ronda de Execução para executar a linha de programa, cujo endereço é fornecido no par de registradores DE.

Endereço ..... 47E8H

Este é o manipulador da instrução "GOTO". O operando do número de linha é coletado(4769H) e colocado no par de registradores HL. Se for um apontador, o controle será transferido imediatamente à Ronda de Execução, para iniciar a execução na nova posição de texto do programa. Caso contrário, o número da linha será comparado com o número da linha atual, para determinar a posição inicial de busca no texto do programa. Se for maior a busca iniciar-se-á no final desta linha(4298H), e se for menor iniciar-se-á no começo da Área de Texto do programa(4295H). Se a linha referenciada não for encontrada, um erro "Linha de número não definido" será gerado(481CH). Caso contrário, o operando do número da linha será substituído pelo endereço da linha de programa referenciado e seu átomo alterado para o tipo apontador(5583H). Em seguida, o controle será transferido à Ronda de Execução para executar a linha de programa referenciado.

Endereço ..... 481CH

Este é o gerador do erro "número de linha não definida".

Endereço ..... 4821H

Este é o manipulador da instrução "RETURN". Um endereço de Variável de laço sem significado (dummy) será colocado no par de registradores DE e a pilha do Z80 pesquisada(3FE2H) para encontrar o primeiro bloco de parâmetros que não pertença a um laço "FOR". Em seguida esta seção da pilha é descartada. Se nenhum átomo "GOSUB" for encontrado(8DH), será gerado um erro de "RETURN sem "GOSUB".

Em seguida, os dois próximos bytes são tomados do bloco, e se forem não-zero, o bloco foi gerado por uma interrupção e a condição "STOP" temporária será removida

(633EH). Em seguida, o texto do programa é examinado, e caso alguma coisa venha depois do átomo "RETURN" em si, presume-se que seja um operando de número de linha e o controle é transferido ao manipulador de "GOTO" (47E8H). Caso contrário, o número da linha de retorno e o endereço do texto do programa são tomados do bloco e o controle volta ao Laço-de-processamento.

Endereço . . . . . 485BH

Este é o manipulador da declaração "DATA". O texto do programa é pulado até que um separador de instruções(3AH) ou um caractere de fim de linha (00H) seja encontrado. Esta rotina também manipula as instruções "REM" e "ELSE", através do ponto de entrada em 485DH, onde apenas o caractere de fim de linha age como terminador.

Endereço . . . . . 4880H

Este é o manipulador de instrução "LET". A Variável é localizada(5EA4H), seu endereço salvo em TEMP e o operando avaliado(4C64H). Se for necessário, o tipo do operando é convertido para combinar com o da Variável(517AH). Se o operando for um dos três tipos numéricos, simplesmente será copiado de DAC à Variável, na Área de Armazenamento de Variáveis(2EF3H). Se o operando for do tipo string, o endereço do corpo da string será tomado do descritor e verificado. Se estiver em KBUF, como seria o caso de uma string explícita em uma instrução direta, o corpo será primeiro copiado na Área de Armazenamento de Strings e criado um novo descritor(6611H). Em seguida, o descritor será liberado de TEMPST(67EEH) e copiado na Variável na Área de Armazenamento de Variáveis(2EF3H).

Endereço . . . . . 48E4H

Este é o manipulador das instruções "ON ERROR", "ON DEVICE GOSUB" e "ON EXPRESSÃO". Se o próximo caractere de texto no programa não for o átomo "ERROR" (A6H), o controle será transferido para o manipulador de "ON DISPOSITIVO GOSUB" e "ON EXPRESSÃO" (490DH). O texto do programa é verificado para garantir que o átomo "GOTO"(89H) é o próximo e o operando do número de linha é coletado(4769H). O texto do programa é procurado para a obtenção do endereço da linha referencial(4293H) e esta é colocada em ONELIN. Se o operando do número de linha for não-zero, a rotina terminará. Se o número de linha do operando for zero, ONEFLG será verificado para verificar se já existe uma situação de erro (nesse caso, a instrução estaria

no interior de uma rotina de tratamento de erro em BASIC). Em caso afirmativo, o controle é transferido ao manipulador de erro(4096H) para forçar um erro imediato; caso contrário a rotina terminará normalmente.

Endereço ..... 490DH

Este é o manipulador das instruções “ON DISPOSITIVO GOSUB” e “ON EXPRESSÃO”. Se o próximo caractere de texto de programa não for um átomo de dispositivo(7810H), o controle será transferido ao manipulador de “ON EXPRESSÃO” (4943H). Depois de verificar se há um átomo GOSUB (8DH) no texto do Programa, os operandos de número de linha requeridos por um determinado dispositivo serão coletados um por vez(4769H). Sendo um dado operando de número de linha não-zero, o texto do programa será procurado para encontrar o endereço da linha referenciada(4293H), e este será colocado na entrada do dispositivo em TRPTBL(785CH). A rotina terminará quando não for encontrado mais nenhum operando de número de linha.

Endereço ..... 4943H

Este é o manipulador da instrução “ON EXPRESSÃO”. O operador é avaliado (521CH) e o átomo “GOSUB” seguinte(8DH) ou o átomo “GOTO” seguinte(89H) colocado no registrador A. Em seguida, o operando é utilizado para contagem do texto do programa, até que o par de registradores HL aponta para o operando de número de linha indicado. Em seguida, o controle é devolvido ao ponto de execução da Ronda de Execução(4646H) para decodificar o átomo “GOSUB” ou “GOTO”.

Endereço ..... 495DH

Este é o manipulador da instrução “RESUME”. Primeiro ONEFLG é verificado para assegurar que já existe uma condição de erro, caso contrário um “RESUME sem erro” será gerado(4064H). Se vier em seguida um operando não-zero de número de linha, o controle será transferido ao manipulador “GOTO” (47EBH). Se vier, em seguida, um átomo “NEXT”(83H), a posição do erro será restaurada de ERRTXT para ERRLIN, o início da próxima instrução será encontrado(485BH) e a rotina terminará. Se não houver nenhum operando de número de linha ou se este for zero, a posição do erro será encontrada de ERRTXT e ERRLIN e a rotina terminará.

Endereço ..... 49AAH

Este é o manipulador da instrução “ERROR”. O operando é avaliado e colocado

no registrador E(521CH). Se for zero, será gerado um erro “Chamada ilegal de função”(475AH), caso contrário o controle será transferido ao manipulador de erro(406FH).

Endereço ..... 49B5H

Este é o manipulador da instrução “AUTO”. Os operandos opcionais (início e incremento de número de linha), ambos com um valor default de dez, são coletados (475FH) e colocado em AUTLIN e AUTINC. Depois de fazer AUTFLG não-zero, o retorno à Ronda de Execução será destruído e o controle transferido diretamente à Ronda Principal(4134H).

Endereço ..... 49E5H

Este é o manipulador da instrução “IF”. O operando é avaliado(4C64H) e, depois de procurar por um símbolo de “GOTO”(89H) ou um símbolo de “THEN”(DAH), o seu sinal será testado. Se o operando for não-zero (“verdadeiro”), o texto seguinte será executado ou por meio de uma transferência imediata à Ronda Principal(4646H) ou, no caso de um operando de número de linha, ao manipulador de “GOTO”(47E8H). Se o operando for zero (falso) o texto da declaração é varrido(485BH) até que um símbolo de “ELSE” (A1H) seja encontrado, sem ser balanceado por um símbolo “IF”(8BH) e a execução é reiniciada.

Endereço ..... 4A1DH

Este é o manipulador da instrução “LPRINT”. PRTFLG” é colocado em 01H, para direcionar a saída à impressora, e o controle é transferido ao manipulador de “PRINT”(4A29H).

Endereço ..... 4A24H

Este é o manipulador da declaração “PRINT”. O texto do programa é primeiro verificado quanto ao número do buffer final e, se necessário, PTRFIL será ligado para direcionar a saída ao buffer de E/S requerido(6D57H). Caso não exista mais texto de programa será acionado um CR, LF(7328H) e a rotina termina(4AFFH). Caso contrário, caracteres sucessivos são tomados do texto de programa e analisados. Se for encontrado um átomo “USING”(E4H), o controle será transferido ao manipulador de “PRINT USING” (60B1H). Se for encontrado um caractere “;”, o controle simplesmente volta ao início para pegar o próximo item(4A2EH). Se for encontrada uma vírgula serão enviados



espaços suficientes para fazer com que a posição atual de impressão, TTYPOS, LPTPOS ou um FCB de um buffer de E/S, fique com um valor múltiplo inteiro de quatorze. Se a saída estiver direcionada à tela e a posição de impressão for igual ou maior que o conteúdo de CLMLST, ou se a saída estiver direcionada à impressora e a posição for igual ou maior que 238 então, em vez disto, será acionado um CR, LF(7328H). Se um átomo "SPC(" (DFH) for encontrado, o operando será avaliado(521BH) e o número requerido de espaços serão enviados para saída. Se for encontrado um átomo "TAB(" (DBH) o operando será avaliado(521BH) e serão enviados espaços suficientes para fazer com que a posição atual da impressora, TTYPOS, LPTPOS ou um FCB de um buffer de E/S fique com o valor requerido.

Se nenhum destes caracteres forem encontrados, o texto do programa conterá um item de dado, que será em seguida avaliado(4C64H). Se o operando for uma string, ela será simplesmente apresentada(667BH). Se for numérico, será primeiro convertido em texto em FBUFR(3425H) e criado um descritor de string(6635H). Se a saída tiver direcionada a um buffer de E/S, a string resultante será, então, apresentada(667BH). Se a saída estiver direcionada à tela ou à impressora, a posição atual da impressão, de TTYPOS ou LPTPOS será comparada com o tamanho da linha e CR, LF acionado(7328H), se a saída não contiver na linha. O tamanho máximo da linha para a impressora é 255 e é tomado de LINLEN para a tela. Assim que a string tiver sido apresentada, o controle será devolvido ao início do manipulador.

Endereço ..... 4AFFH

Esta rotina zera PRTFLG e PRTFIL para devolver a saída do Interpretador à tela.

Endereço ..... 4B0EH

Este é o manipulador das instruções "LINE INPUT", "LINE INPUT#" e "LINE". Se o caractere seguinte de texto do programa seja qualquer coisa diferente do átomo "INPUT" (85H), o controle será transferido ao manipulador da instrução "LINE"(58A7H). Se o caractere seguinte do texto de programa for "#" (23H), o controle será transferido ao manipulador da declaração "LINE, INPUT#" (6D8FH).

Qualquer string de prompt seguinte será avaliada e apresentada(4B7BH) e a Variável localizada(5EA4H) e verificada para garantir que seja string(3058H). A linha de texto é pega do console por meio da rotina padrão INLIN, e se o Indicador C(CTRL-STOP) for devolvido, o controle será transferido ao manipulador da instrução

“STOP”(63FEH). Caso contrário, o string de entrada é analisado e criado um descritor(6638H). Em seguida, o controle será transferido do manipulador da instrução “LET” para a atribuição(4892H). Note que a tela não será forçada ao modo texto antes que a entrada seja coletada.

Endereço ..... 4B3AH

Esta é a mensagem “?Redo from start”, CR, LF terminada por um byte zero.

Endereço ..... 4B4DH

Esta rotina é utilizada pelo manipulador das instruções “READ/INPUT”, caso tenha falhado em converter um item de dado para a forma numérica. Caso esteja no modo “READ”(FLGINP é não-zero) será gerado um “Erro de sintaxe”(404FH). Caso contrário, a mensagem “?Redo from start” será apresentada(6678H) e o controle voltará ao manipulador da instrução.

Endereço ..... 4B62H

Este é o manipulador da declaração “INPUT\$”. O número do buffer é avaliado e PTRFIL ligado para direcionar a entrada do buffer de E/S requerido(6D55H). O controle é depois transferido ao manipulador das instruções “READ/INPUT” (4B9BH).

Endereço ..... 4B6CH

Este é o manipulador da declaração “INPUT”. Se o próximo caractere de texto do programa for um “\$”, o controle será transferido ao manipulador da instrução “INPUT\$”(4B62H). Caso contrário, a tela será forçada ao modo texto, por meio da rotina padrão TOTXT, e qualquer string de prompt será analisada(6636H) e apresentada(667BH). Em seguida, será apresentado um ponto de interrogação e uma linha de texto tomada do console por meio da rotina padrão QINLIN. Caso seja retomado o Indicador C(CTRL-STOP), o controle é transferido ao manipulador de “STOP”(63FEH). Se o primeiro caractere de BUF for zero (entrada nula), o manipulador terminará pulando até o fim da declaração(485AH); caso contrário, o controle irá para o manipulador de “READ/INPUT” combinados.

Endereço ..... 4B9FH

Este é o manipulador da instrução “READ”, com uma grande seção também

utilizada pelas instruções “INPUT” e “INPUT\$”, de modo que a estrutura da rotina pode parecer bastante estranha. As variáveis encontradas no texto do programa são localizadas uma por vez(5EA4H), e para cada uma o item de dado correspondente é obtido e atribuído pelo manipulador “LET”(4893H). Quando no modo “READ”, os itens de dados são tomados do texto do programa utilizando-se conteúdo inicial de DATPRT(4C40H). Quando no modo “INPUT” ou “INPUT\$”, os itens de dado são tomados do buffer de texto BUF.

Se os itens de dados forem esgotados no modo “READ” será gerado um erro “Falta DATA”. Se forem esgotados no modo “INPUT”, dois pontos de interrogação são apresentados e outra linha tomada do console via a rotina padrão QUINLIN. Se forem esgotados em “INPUT\$”, outra linha de texto será copiada no BUF, do buffer de E/S relevante(6D83H). Se a lista de Variáveis for esgotada, enquanto no modo “INPUT” a mensagem “Extra ignored” será apresentada (6678H) e o manipulador terminará(4AFFH). No modo “INPUT\$” nenhuma mensagem será apresentada, enquanto no modo “READ” o controle termina atualizando DATPTR(63DEH). Caso um item de dado não possa ser convertido para a forma numérica(3299H), afim de combinar com uma variável numérica, o controle será transferido à rotina “?Redo from start”(4B4DH).

Endereço ..... 4C2FH

Esta é a mensagem “?Extra ignored”, CR, LF, terminada em um byte zero.

Endereço ..... 4C40H

Esta rotina é utilizada pelo manipulador “READ” para localizar o próximo “DATA” no texto do programa, com o endereço inicial fornecido no par de registradores HL. Cada instrução no programa é examinada, até que um átomo “DATA”(84H) seja encontrado e a rotina termine(4BD1H). Se o elo final for atingido será gerado um erro “Falta DATA”. À medida que a procura continua, o número da linha de cada linha de programa será colocado em DATLIN para ser utilizado pelo manipulador de erro.

Endereço ..... 4C5FH

Esta rotina checka se o próximo caractere no texto do programa é o átomo “=”(EFH) e depois vai para o Avaliador de Expressões. Quando entrada em 4C62H ela checka “(”.

Endereço . . . . . 4C64H

Este é o Avaliador de Expressões. Na entrada, o par de registradores HL aponta para o primeiro caractere da expressão a ser avaliado. Ao sair, o par de registradores HL aponta para o caractere que segue a expressão, com o resultado colocado em DAC e o código do tipo em VALTYP. Para um resultado string, o endereço do descriptor da string é devolvido em DAC+2. O descriptor em si, formado por um único byte para o comprimento da string e dois bytes para seu endereço estará em TEMPST ou dentro de uma Variável string.

Uma expressão é uma lista de fatores(4DC7H) ligados entre si por operadores com níveis de precedência diferentes. Para processar uma expressão deste tipo corretamente, o Avaliador de Expressões empilha temporariamente um resultado intermediário, se o operador seguinte tiver uma precedência mais alta que o operador atual, e um novo cálculo seja recommçado. O avaliador, portanto, tem duas operações básicas, empilha e calcula por exemplo:

|                        |                         |
|------------------------|-------------------------|
| 3 + 250\2^2 * 3^3 + 1, |                         |
| EMPILHA: 3+            | (Porque se segue um \)  |
| EMPILHA: 250\          | (Porque se segue um ^)  |
| CALCULA: 2^2 = 4       | (Porque se segue um *)  |
| EMPILHA: 4*            | (Porque se segue um ^)  |
| CALCULA: 3^3 = 27      | (Porque se segue um +)  |
| CALCULA: 4 * 27 = 108  | (Porque se segue um +)  |
| CALCULA: 250\108 = 2   | (Porque se segue um +)  |
| CALCULA: 3 + 2 = 5     | (Porque se segue um +)  |
| CALCULA: 5 + 1 = 6     | (Porque se segue uma ,) |

A avaliação terminará quando o próximo operador tiver uma precedência igual ou menor que a precedência inicial e a pilha estiver vazia. O delimitador de expressão, mostrado como uma vírgula no exemplo, é visto como tendo uma precedência de zero e desta forma interrompe sempre a avaliação. Normalmente, o Avaliador de Expressão inicia com uma precedência inicial de zero, mas o porto de entrada em 4C67H poderá ser utilizado para fornecer um valor alternativo no registrador D. Esta facilidade é utilizada pelo Avaliador de Fatores para restringir a faixa de avaliação ao aplicar os operadores de negação monádica e "NOT".

Endereço . . . . . 4D22H

Esta rotina é utilizada pelo Avaliador de Expressões para aplicar um operador

matemático infix (+ - \* / ^) a um par de operandos numéricos. Há rotinas separadas para os operadores relacionais(4F57H) e operadores lógicos(4F78H). O primeiro operando, seu código de tipo e o símbolo do operador são fornecidos na pilha do Z80; o segundo operando e o seu código de tipo são fornecidos em DAC e VALTYP. Primeiro os tipos dos dois operandos são comparados, e se diferirem, o operando de menor precisão será convertido de modo a combinar com o de maior precisão. Em seguida, os operandos são movidos às posições requeridas pelas rotinas matemáticas. Para os inteiros, o primeiro operando será colocado no par de registradores DE e o segundo no par de registradores HL. Para precisão simples, o primeiro operando será colocado nos registradores C, B, E, D e o segundo no DAC. Para dupla precisão, o primeiro operando será colocado no DAC e o segundo no ARG. Em seguida, o símbolo do operador será utilizado para a obtenção do endereço requerido da tabela em 3D51H, 3D5DH ou 3D69H, dependendo do tipo de operando, e o controle será transferido para a rotina matemática relevante.

Endereço . . . . . 4DB8H

Esta rotina é utilizada pelo Avaliador de Expressões para dividir dois operandos inteiros. O primeiro operando está contido no par de registradores DE e o segundo no par de registradores HL, e o resultado será colocado em DAC. Ambos os operandos são convertidos para precisão simples(2FCBH) e o controle será transferido para a rotina de divisão de precisão simples(3265H).

Endereço . . . . . 4DC7H

Este é o Avaliador de Fatores. No início da rotina, o par de registradores HL aponta para o caractere antes do fator a ser avaliado. Ao terminar, o par de registradores HL aponta para o caractere que segue o fator. O resultado é colocado no DAC e o código do tipo em VALTYP. Um fator poderá ser um dos seguintes exemplos:

- (1) Uma constante numérica ou string
- (2) Uma variável numérica ou string
- (3) Uma função
- (4) Um operador monádico (+ - NOT)
- (5) Uma expressão entre parênteses

O primeiro caractere é tomado do texto de programa por meio da rotina padrão CHRGT e é examinado. Se for um caractere de fim de instrução será gerado um erro "Falta operando"(406AH). Se for um dígito ASCII será convertido de forma textual para uma dos tipos numéricos padrão em DAC(3299H).

Se for uma letra maiúscula(64A8H) será uma Variável e seu valor atual é devolvido(4E9BH). Se for um átomo numérico o número será copiado de CONLO ao DAC(46B8H). Se for um dos átomos de função prefixados por FFH, mostrados na tabela em 39DEH, será decodificado para que o controle seja transferido ao manipulador da função correspondente(4EFCH). Se for um operador monádico “+” será simplesmente pulado. Apenas o operador monádico “-” (4E8DH) e o operador monádico “NOT” (4F63H) requerem alguma ação.

Se forem umas aspas de abertura, a string explícita que segue será analisada e será criado um descritor(6636H). Se for um “&”, será uma constante numérica não-decimal que será convertida para um dos tipos padrão em DAC(4EB8H). Se não for uma das funções indicadas a seguir, será uma expressão entre parênteses(4E87H). Caso contrário, será gerado um “Erro de sintaxe”. Os seguintes símbolos de função serão testados diretamente e o controle será transferido ao endereço indicado:

|            |       |             |       |
|------------|-------|-------------|-------|
| ERR .....  | 4DFDH | ATTR\$ .... | 7C43H |
| ERL .....  | 4E0BH | VARPTR ...  | 4E41H |
| POINT .... | 5803H | USR .....   | 4FD5H |
| TIME ..... | 7900H | INSTR ....  | 68EBH |
| SPRITE ... | 7A84H | INKEY\$ ... | 7347H |
| VDP .....  | 7B47H | STRING\$ .. | 6829H |
| BASE ..... | 7BCBH | INPUT\$ ... | 6C87H |
| PLAY ..... | 791BH | CSRLIN ...  | 790AH |
| DSKIS .... | 7C3EH | FN .....    | 5040H |

Endereço ..... 4DFDH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “ERR”. O conteúdo de ERRFLG é colocado em DAG como um inteiro(4FCFH).

Endereço ..... 4E0BH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “ERL”. O conteúdo de ERRLIN é copiado em DAC como um número de precisão simples (3236H).

Endereço ..... 4E41H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “VARPTR”. Se o símbolo da função for seguido por um “#”, o número do buffer será avaliado(521BH), o FCB do buffer de E/S localizado(6A6DH) e seu endereço colocado em DAC como um inteiro(2F99H). Caso contrário, a Variável será localizada(5F5DH) e seu endereço colocado no DAC como um inteiro(2F99H).

Endereço ..... 4E8DH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar o operador monádico “-”. Um valor de precedência 7DH é colocado no registrador D, o fator é avaliado(4C67H) e depois negado(2E86H).

Endereço ..... 4E9BH

Esta rotina é utilizada pelo Avaliador de Fatores para devolver o valor atual de uma Variável. Primeiro a Variável é localizada(5EA4H). Se for uma Variável string, seu endereço será colocado no DAC para apontar para o descritor. Caso contrário, o conteúdo da Variável é copiado no DAC(2F08H).

Endereço ..... 4E9BH

Esta rotina devolve no registrador A, o caractere apontado pelo par de registrador HL. Se for uma letra minúscula ela é, convertida em maiúscula.

Endereço ..... 4EB8H

Esta rotina é utilizada pelo Avaliador de Fatores e pela rotina de entrada numérica(3299H) para converter um número prefixado com “&”, da forma textual para um inteiro no DAC. À medida que cada caractere válido for encontrado, o produto será multiplicado por 2, 8 ou 16, dependendo do caractere que inicialmente seguiu o “&”, e o novo dígito somando a ele. Caso o produto extravase, um erro “Extravasamento” (Overflow) será gerado(4067H). A rotina terminará quando é encontrado um caractere inválido.

Endereço ..... 4EFCH

Esta rotina é utilizada pelo Avaliador de Fatores para processar os átomos de função prefixados por FFH. Caso o símbolo for “LEFT\$”, “RIGHT” ou “MID\$”, o operando string será avaliado(4C62H), o endereço de seu descritor colocado na pilha do Z80, o operando numérico seguinte também será avaliado(521CH) e empilhado. Caso contrário, o operando entre parênteses da função será avaliado(4E87H) e, para “SQR”, “RND”, “SIN”, “LOG”, “EXP”, “COS”, “TAN” ou “ATN” apenas, convertido em dupla precisão(303AH). O átomo da função, em seguida, é utilizado para a obtenção do endereço requerido da tabela em 39DEH e o controle é transferido ao manipulador da função.

Endereço ..... 4F47H

Esta rotina é utilizada pela rotina de conversão de entrada numérica(3299H) para testar a existência de um caractere ou átomo “+” ou “-”. Devolve o registrador D=0 para positivo e o registrador D=FFH para negativo.

Endereço ..... 4F57H

Esta rotina é utilizada pelo Avaliador de Expressões para aplicar um operador relacional (< > = ou suas combinações) a um par de operandos. Se os operandos forem numéricos, o Avaliador de Expressões utiliza primeiro a rotina de operador matemático (4D22H) para aplicar a operação de relação geral aos operandos. Se os operandos forem strings, a rotina de comparação de strings(65C8H) será utilizada primeiro. Quando o controle chegar aqui, o resultado da relação estará no registrador A e nos Indicadores do Z80:

|                       |       |                           |
|-----------------------|-------|---------------------------|
| Operando 1=Operando 2 | ..... | A=00H, Indicadores Z, NC  |
| Operando 1<Operando 2 | ..... | A=01H, Indicadores NZ, NC |
| Operando 1>Operando 2 | ..... | A=FFH, Indicadores NZ, C  |

O Avaliador de Expressões fornece também na pilha do Z80, uma máscara de bits definindo os operadores originais. Esta máscara contém 1 em cada posição, onde a operação associada é requerida: 00000<=>. A máscara é aplicada ao resultado da relação resultado em zero se nenhuma das condições for satisfeita. O resultado é, em seguida, colocado no DAC como um inteiro “verdadeiro”(-1) ou “falso”(0) (2E9AH).

Endereço ..... 4F63H

Esta rotina é utilizada pelo Avaliador de Fatores, para aplicar o operando monádico “NOT”. Um nível de precedência de 5AH é colocado no registrador D, a expressão avaliada(4C67H) e convertida em um inteiro(2F8AH). Em seguida, é invertida e recolocada em DAC.

Endereço ..... 4F78H

Esta rotina é utilizada pelo Avaliador de Expressões para aplicar um operador lógico(“OR”, “AND”, “XOR”, “EQV” e “IMP”) ou os operadores “MOD” e “\” em um par de operandos numéricos. O primeiro operando, que já foi convertido em um inteiro, é fornecido na pilha do Z80 e o segundo é fornecido no DAC. O átomo do operador (na



verdade o seu nível de precedência) é fornecido no registrador B. Depois de converter o segundo operando em um inteiro(2F8AH) o operador é examinado. Há rotinas separadas para "MOD"(323AH) e "\"(31E6H), mas os operadores lógicos são processados localmente, utilizando-se as instruções lógicas correspondentes do Z80 nos pares de registradores DE e HL. O resultado é armazenado em DAC como um inteiro (2F99H).

Endereço ..... 4FC7H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LPOE" em um operando contido em DAC. O conteúdo de LPTPOS é colocado no DAC como um inteiro(4FCFH).

Endereço ..... 4FCCH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "POS" em um operando contido em DAC. O conteúdo de TTYPOS é colocado em DAC como um inteiro(2F99H).

Endereço ..... 4FD5H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "USR". O número do usuário é pego diretamente do texto do programa (não podendo ser uma expressão) e o endereço associado tomado de USRTAB(4FF4H). O operando que se segue entre parênteses é avaliado(4E87H) e deixado em DAC como o parâmetro passado. Se for um tipo string seu espaço será liberado(67D3H). A posição do texto do programa atual é colocada na pilha do Z80 seguida por um retorno para 3297H, e a rotina deste endereço restaurará a posição de texto do programa depois que a função do usuário termina. O controle, em seguida, será transferido ao endereço do usuário com o par de registradores HL apontando para o primeiro byte de DAC e o código de tipo de VALTYP, no registrador A. Adicionalmente, para um parâmetro string, o endereço do descritor é tomado de DAC e colocado no par de registradores DE.

A rotina de usuário pode modificar qualquer registrador, exceto o SP do Z80 e deveria terminar com uma instrução RET. Interrupções poderão ficar desativadas, se for necessário, uma vez que a Ronda de Execução irá reativá-las. Qualquer parâmetro numérico, a ser devolvido ao Interpretete, deveria ser colocado em DAC. Normalmente, este deveria ser do mesmo tipo numérico que o parâmetro passado, porém, se VALTYP for modificado o Interpretador irá também aceitá-lo.

Devolver uma string é mais difícil. Utilizar o mesmo método que as funções string de Avaliador de Fatores (que envolve copiar a string para a Área de Armazenamento de Strings e colocar um novo descritor em TEMPST) é complicado e vulnerável às mudanças no sistema MSX. Um método mais simples e mais confiável é utilizar o próprio parâmetro passado para criar espaço para o string resultante. Este não deveria ser uma string declarada explicitamente, uma vez que o texto do programa seria modificado. Em vez disto, um parâmetro implícito deveria ser utilizado. Entretanto, isto tem que ser feito com cuidado, porque é muito fácil termos a impressão de que o Interpretador aceitou a string quando na verdade não o fez. Observe o seguinte exemplo que nada faz, a não ser devolver o parâmetro passado:

```
10 POKE &H9000,&HC9
20 DEFUSR=&H9000
30 A$=USR(STRING$(12,"!"))
40 PRINT A$
50 B$=STRING$(9,"X")
60 PRINT A$
```

Inicialmente, parece que a string passada foi atribuída corretamente a A\$. Entretanto, quando a linha 60 é alcançada torna-se aparente que A\$ foi estragado pela atribuição subsequente de uma string para B\$. O que aconteceu é que o armazenamento temporário alocado ao parâmetro passado foi liberado da Área de Armazenamento de strings, antes do controle ser transferido à rotina do usuário. Em seguida, esta região foi utilizada para armazenar a string pertencente a B\$, modificando assim A\$.

Esta situação pode ser evitada atribuindo o parâmetro a uma Variável e depois passando a Variável como parâmetro, por exemplo:

```
10 A$=STRING$(12,"!")
20 A$=USR(A$)
```

A linha 10 faz com que doze bytes da Área de Armazenamento de string sejam permanentemente alocados a A\$. Quando entramos na função do usuário, o descritor, que é apontado pelo par de registradores DE, conterà o endereço inicial da região de doze bytes em que o resultado deve ser colocado. Caso a string devolvida seja menor que o passado, o byte de comprimento do descritor poderá ser alterado sem nenhum efeito secundário. Para detalhes adicionais sobre o armazenamento de strings, veja o coletor de lixo(66B6H).

Um ponto que vale a pena mencionar é que uma operação "CLEAR" não é estritamente necessária, antes que um programa em linguagem de máquina seja carregado. A região entre o topo da Área de Armazenamento de Matriz e a base da pilha do Z80 nunca é utilizada pelo Interpretador. Um programa poderá existir nesta região, desde que as duas áreas circundantes não a sobreponham.

Endereço . . . . . 500EH

Este é o manipulador da instrução “DEFUSR”. O número do usuário é coletado diretamente do texto do programa (não podendo ser uma expressão) e a entrada associada em USRTAB localizada(4FF4H). Em seguida, o operando de endereço é avaliado(542FH) e colocado em USRTAB.

Endereço . . . . . 501DH

Este é o manipulador das instruções “DEF FN” e “DEFUSR”. Se o caractere seguinte for um átomo “USR” o controle será transferido ao manipulador da instrução “DEFUSR”(500EH), caso contrário, é verificado se o próximo átomo é um “FN”(DEH). A Variável com o nome da função é criada(51A1H) e, depois de verificar se o Interpretador está no modo de programa(5193H), a posição atual no texto do programa é colocada ali. Cada uma das Variáveis na lista de parâmetros formais é criada em sequência(5EA4H), simplesmente para garantir a criação. A rotina termina pulando o restante da instrução(485BH), uma vez que o corpo da função não é necessário neste momento.

Endereço . . . . . 5040H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “FN”. Primeiro, a Variável do nome da função é localizada(51A1H) para se obter o endereço da definição da função no texto do programa. Cada Variável formal da definição da função é localizada(5EA4H) e seu endereço colocado na pilha do Z80. A medida que cada uma é encontrada o parâmetro real correspondente é avaliado(4C64H) e colocado na pilha com ela. Se for necessário, o tipo do parâmetro real será convertido para combinar com aquele do parâmetro formal(517AH).

Quando as duas listas estão esgotadas, cada endereço formal de Variável e parâmetro atual são retirados da pilha. Em seguida, cada variável é copiada, da Área de Armazenamento de Variáveis, em PARM2, com seu valor substituído pelo parâmetro verdadeiro. Observe que, pelo fato de PARM2 ter apenas cem bytes de tamanho, será permitido um máximo de nove parâmetros de dupla precisão. Quando todos os parâmetros atuais forem copiados em PARM2, o conteúdo inteiro de PARM1 (a área atual dos parâmetros) será colocada na pilha do Z80 e PARM2 será copiado em PARM1(518EH). O par de registradores HL, em seguida, é atribuído com o início do corpo de função no texto do programa e a expressão é avaliada(4C5FH). Os conteúdos antigos de PARM1 serão

retirados da pilha e restaurados. Finalmente, o resultado da avaliação sofre uma conversão de tipo, se for necessário, para combinar com o tipo do nome da função(517AH).

Uma função definida pelo usuário difere de uma expressão normal em apenas um aspecto: tem seu próprio conjunto de Variáveis locais. Estas variáveis são criadas em PARM1, quando a função é evocada, e desaparecem quando ela termina. Quando uma procura normal de Variável é iniciada pelo Avaliador de Expressões, a região examinada é a Área de Armazenamento de Variáveis. Entretanto, se NOFUNS for não-zero, indicando ao menos uma função de usuário ativa, PARM1 será vasculhado em seu lugar, e se isto falhar, a busca será efetuada nas Variáveis globais da Área de Armazenamento de Variáveis. Utilizar uma área de Variável local específica para cada chamada de função significa que os mesmos nomes de Variáveis poderão ser utilizados durante o tempo todo, sem que as Variáveis se sobreponham ou sobreponham às variáveis globais.

Vale a pena observar que uma função definida pelo usuário é mais lenta que uma expressão explícita ou mesmo uma sub-rotina. A procura levada a efeito para encontrar a Variável do nome da função, mais a grande quantidade de empilhamento e desempilhamento, tomam um tempo significativo.

Endereço ..... 5189H

Esta rotina move um bloco de memória do endereço apontado pelo par de registradores DE ao apontado pelo par de registradores HL, com o par de registradores BC definindo o tamanho.

Endereço ..... 5193H

Esta rotina gera um erro "Direto ilegal" se CURLIN mostrar que o Interpretador está no modo direto.

Endereço ..... 51A1H

Esta rotina verifica se o texto do programa tem um átomo "FN"(DEH) e depois cria a Variável do nome da função(5EA9H). Estas são distinguidas das Variáveis normais por ter o bit 7 ligado no primeiro caractere do nome da Variável.

Endereço ..... 51ADH

O controle é transferido à rotina do ponto de execução da Ronda de Execução

(460H), quando um átomo maior do que D8H for encontrado no início de uma instrução. Se não for um átomo de função prefixada por FFH, será gerado um “Erro de sintaxe” (4055H). Se o átomo da função for um daqueles que também podem funcionar como instrução, o controle será transferido ao manipulador relevante; caso contrário um “Erro de sintaxe” é gerado. As instruções deste grupo são “MID\$”(696EH), “STRIG”(77BFH) e “INTERVAL”(77B1H). Não há, realmente nenhum átomo específico para “INTERVAL”. O átomo “INT”(85H) é suficiente, sendo o restante dos caracteres verificado pelo manipulador da instrução.

Endereço ..... 51C9H

Este é o manipulador da instrução “WIDTH”. O operando é avaliado(521CH) e sua magnitude verificada. Se for zero ou maior que trinta e dois ou quarenta, dependendo do modo da tecla mantido em OLDSCR, será gerado um erro de “função ilegal”(475AH). Se for o mesmo que o conteúdo atual de LINLEN, a rotina terminará sem nenhuma ação adicional. Caso contrário, a tela atual é limpa com um código de controle FORMFEED (OCH), por meio da rotina padrão OUTDO, caso a tela deva ser encurtada. Em seguida, o operando será colocado em LINLEN e LINL32 ou LINL40, dependendo do modo de tela mantido em OLDSCR, com a tela limpa novamente, se tiver sido aumentada. Pelo fato da variável do tamanho de linha a ser alterada ser selecionada por OLDSCR, em vez de SCRMOD, a largura poderá ser mudada, mesmo que a tecla esteja, atualmente, no Modo Gráfico ou Modo Multicolorido. Neste caso, a alteração será efetuada ao ser feito um retorno à Ronda Principal do Interpretador, ou quando uma instrução “INPUT” for executada.

Endereço ..... 520EH

Esta rotina avalia a próxima expressão no texto do programa(4C64H), converte o resultado a um inteiro(2F8AH) e o coloca no par de registradores DE. A grandeza e o sinal do MSB serão testados depois, e a rotina terminará.

Endereço ..... 521BH

Esta rotina avalia o próximo operando no texto do programa(4C64H) e o converte em um inteiro(5212H). Se o operando for maior do que 255, um erro de “Chamada ilegal de função” será gerado(475AH).

Endereço . . . . . 5229H

Este é o manipulador da instrução "LLIST". PRFTLG será colocado em 01H, para direcionar a saída à impressora, e o controle irá para o manipulador da instrução "LIST".

Endereço . . . . . 522EH

Este é o manipulador da instrução "LIST". Os operandos dos números de linha inicial e final opcionais são coletados e a posição inicial é encontrada no texto do programa(4279H). Linhas de programa sucessivas são listadas até que o elo final seja encontrado, até que a tecla CTRL-STOP seja pressionada ou que o número de linha final seja alcançado. Em seguida, o controle será transferido, diretamente, ao ponto "OK" da Ronda Principal(411FH). A linha é listada da seguinte maneira: seu número é apresentado (3412H), a linha é desatomizada(5284H) e apresentada(527BH) e um CR, LF é enviado(7328H).

Endereço . . . . . 5284H

Esta rotina é utilizada pelo manipulador da instrução "LIST" para converter uma linha de programa atomizada para a forma textual. No início, o par de registradores HL aponta para o primeiro caractere da linha atomizada. Ao terminar, a linha de texto está no BUF, terminada com um byte zero.

Qualquer átomo normal ou prefixado por FFH é convertido na palavra-chave correspondente por uma simples procura linear dos símbolos na tabela em 3A72H. Exceções serão feitas se aspas de abertura inicial, um símbolo "REM", ou um símbolo "DATA", for encontrado anteriormente. Normalmente, estes símbolos são seguidos por um texto normal. De qualquer forma, a verificação é feita para impedir que caracteres gráficos sejam interpretados como átomos. A sequência de três bytes ":"(3AH), o átomo "REM"(8FH) e o átomo "!"(E6H) é convertida no caractere "!"(27H), e o separador de instruções(3AH) precedendo um símbolo "ELSE"(A1H) é eliminado.

Se um dos símbolos numéricos for encontrado, seu valor e tipo são primeiro copiados de CONLO e CONTYP em DAC e VALTYP(46E8H). em seguida ele é convertido à forma textual em FBUFFR, pelas rotinas de conversão decimal(3425H), octal (371EH) ou hexa(3722H). Para os tipos octal e hexadecimal, o número é prefixado por um "&" e uma letra "O" ou "H". Um sufixo de tipo, "!" ou "#", será acrescentado aos números de simples precisão ou dupla precisão, apenas se não houver parte decimal e nenhuma parte exponencial("E" ou "D").

Endereço ..... 53E2H

Este é o manipulador da instrução "DELETE". Os operandos dos números de linha inicial e final opcionais são coletados e a posição inicial encontrada no texto de programa(4279H). Se existir algum apontador no texto do programa, ele será reconvertido em número de linha(54EAH). A linha de programa final é encontrada por uma procura no texto do programa(4295H). Se este endereço for menor que o inicial, será gerado um erro de "Chamada ilegal de função"(475AH): caso contrário a mensagem "OK" é apresentada (6678H). O bloco de memória do final da linha terminal ao início da Área de Armazenamento de Variáveis é copiado no início da linha inicial e VARTAB, ARYTAB e STREND serão atualizados com o valor do novo final de programa (agora menor). Em seguida, o controle é transferido diretamente ao final da Ronda de Execução(4237H) para atualizar os apontadores restantes e refazer os elos da Área de Texto de Programa. Observe que, pelo fato do controle não voltar ao ponto "OK" normal, a tela não voltará ao modo texto. Se a tela estiver no Modo Gráfico ou Modo Multicolorido quando um "DELETE" é executado (o que é muito pouco provável), o sistema entra em colapso.

Endereço ..... 541CH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "PEEK", em um operando contido em DAC. O operando de endereço é verificado(5439H), o byte é lido da memória e colocado em DAC como um inteiro(4FCFH).

Endereço ..... 5423H

Este é o manipulador da instrução "POKE". O operando de endereço é avaliado (542FH). Em seguida, o operando do dado é avaliado(521CH) e escrito na memória.

Endereço ..... 542FH

Esta rotina avalia o próximo operando no texto do programa(4C64H) e o coloca no par de registradores DE como um inteiro(5439H).

Endereço ..... 5439H

Esta rotina converte o operando numérico contido no DAC, em um inteiro no par de registradores HL. O operando tem que estar na faixa -32768 a +65535 e é normalmente um endereço conforme requerido por "POKE", "PEEK", "BLOAD" etc.

Primeiro, o tipo do operando é verificado por meio da rotina padrão GETYPR, e, se já foi um inteiro, simplesmente será colocado no par de registradores HL(2F8AH). Considerando que o operando seja de simples ou dupla precisão, seu sinal será verificado, e se for negativo será convertido em um inteiro(2F8AH). Caso contrário, será convertido para precisão simples(2FB2H) e sua grandeza verificada(2F21H). Se for maior do que 32767 e menor do que 65536, então -65536 será acrescentado(324EH) antes de ser convertido em um inteiro(2F8AH).

Endereço ..... 5468H

Este é o manipulador da instrução “RENUM”. Se existir um novo operando de número de linha inicial, ela será coletada(475FH); caso contrário, um valor default de dez será considerado. Se existir um operando de antigo número de linha, ele será coletado (475FH); caso contrário, um default de zero será considerado. Se existir um operando de incremento de número de linha, ele será coletado(4769H). Caso contrário, um valor default de dez é considerado.

Em seguida, o texto de programa será procurado por números de linhas existentes iguais ou maiores que o novo número de linha inicial(4295H) e o antigo número de linha inicial(4295H). Será gerado um erro de “Chamada ilegal de função”(475AH), se o novo endereço for menor que o endereço antigo. Isto é para evitar que linhas mais elevadas do programa fiquem com números de linhas mais baixas existentes.

Uma remuneração hipotética do texto do programa será executada a fim de certificar que nenhum número de linha novo será gerado com um valor maior do que 65529. Isto deverá ser feito, pois um erro no meio do caminho da conversão deixará o texto do programa em uma situação confusa. Estando tudo em ordem, quaisquer operandos de número de linha no texto do programa são convertidos em apontadores(54F6H). Isto resolve de forma satisfatória o problema de referências a números de linha, por exemplo, GOTO 50, pois o texto do programa não é tocado durante a renumeração. A partir da antiga posição inicial de texto do programa, cada número de linha de programa existente é substituído pelo seu novo valor. Quando o elo final é alcançado, qualquer apontador no texto do programa é reconvertido em um operando de número de linha(54F1H) e o controle transferido diretamente ao ponto “OK” da Ronda Principal(411EH).

Endereço ..... 54F6H

Quando entrado em 54F6H, esta rotina converte qualquer operando de número de linha no texto do programa em um apontador. Se digitada em 54F7H, executa a



operação inversa e converte qualquer apontador no texto do programa de volta para um operando de número de linha. A partir do início da Área de Texto do Programa, cada linha é examinada quanto a um átomo de apontador(ODH) ou a um átomo de operando do número de linha(OEH), dependendo do modo. No modo apontador para de número de linha, o apontador é substituído pelo número da linha de programa referenciada e o átomo mudado para OEH. No modo operando de número de linha para apontador, o texto do programa é vasculhado(4295H) para encontrar as linhas referidas, seu endereço substitui o operando de número de linha e o átomo é substituído por ODH. Se a procura não for bem sucedida, uma mensagem da forma "Undefined line NNNN in NNNN" será apresentada (6678H) e o processo de conversão continuará. Uma verificação especial será feita para a instrução "ON ERROR GOTO 0", a fim de impedir a geração de uma mensagem de erro espúria, mas nenhuma verificação será feita para a declaração "RESUME 0". Neste caso, uma mensagem de erro será apresentada, e isto deverá ser ignorado.

Endereço . . . . . 555AH

Esta é a mensagem de texto "Undefined line" terminada por um byte zero.

Endereço . . . . . 558CH

Nome . . . . . SYNCHR

Entrada . . . . . HL aponta o caractere a ser verificado

Saída . . . . . A=Próximo caractere do programa

Modifica . . . . . AF, HL

Rotina padrão para comparar o caractere corrente no texto do programa, cujo endereço é fornecido no par de registradores HL, com um caractere de referência. O caractere de referência é fornecido como um byte simples imediatamente após a instrução CALL ou REST, por exemplo:

```
RST B
DEFB ' , '
```

Caso os caracteres não coincidam, será gerado um "Erro de sintaxe" (4055H). Caso contrário, o controle é transferido à rotina padrão CHRGT, para pegar o próximo caractere do programa(4666H).

Endereço ..... 5597H  
 Nome ..... GTYPR  
 Entrada ..... Nenhuma  
 Saída ..... AF=Tipo  
 Modifica ..... AF

Rotina padrão que devolve o tipo do operando atual, determinado por VALTYP, como segue:

Inteiro ..... A=FFH, Indicador M, NZ, C  
 String ..... A=00H, Indicador P, Z, C  
 Precisão simples ..... A=01H, Indicador P, NZ, C  
 Precisão dupla ..... A=05H, Indicador P, NZ, NC

Endereço ..... 55A8H

Este é o manipulador da instrução "CALL". O nome da instrução estendida, que é uma string sem aspas com até quinze caracteres de tamanho terminando em "(", ":", ou um caractere de fim de linha(00H), é primeiro copiado do texto do programa para PROCNM, onde quaisquer bytes não utilizados serão preenchidos com zero. Em seguida, o bit 5 de cada entrada em SLTATR é examinado quanto a uma ROM de extensão com um manipulador de declaração. Se uma ROM adequada for encontrada, a sua posição no SLTATR será convertida em uma Identificação de conector no registrador A e um endereço de base da ROM no registrador H(7E2AH). O endereço do manipulador da instrução é lido das posições quatro e cinco da ROM(7E1AH) e colocado no par de registradores IX. O Identificador do conector é colocado no byte mais alto do par de registradores IY e o manipulador da instrução em ROM é chamado pela rotina padrão CALSLT.

A ROM examina o nome da declaração e devolve o Indicador C, caso não o reconheça. Caso contrário, executa a operação requerida. Se a chamada à ROM falhar, a procura de SLTATR continuará até que a tabela seja esgotada, quando será gerado um "Erro de sintaxe"(4055H). Se a chamada à ROM for bem sucedida, o manipulador terminará.

Endereço ..... 55F8H

Esta rotina é utilizada pelo analisador sintático de nomes de dispositivos(6F15H) quando não consegue reconhecer um nome de dispositivo encontrado no texto do

programa. No início o par de registradores HL aponta para o primeiro caractere do nome e o registrador B tem seu comprimento. Primeiro, o nome é copiado em PROCNM e terminado por um byte zero. O byte 6 de cada entrada em SLTATR é, em seguida, examinado quanto a uma ROM de extensão com um manipulador de dispositivo. Se uma ROM adequada for encontrada, a sua posição em SLTATR será convertida em uma Identificação de conector no registrador A e um endereço de base da ROM no registrador H(7E2AH). O endereço do manipulador de dispositivo é lido das posições seis e sete da ROM(7E1AH) e colocado no par de registradores IX. O Identificador do conector será colocado no byte mais alto do par de registradores IY, o código de dispositivo desconhecido (FFH) no registrador A e o manipulador do dispositivo ROM é chamado via rotina padrão CALSLT.

A ROM examina o nome do dispositivo e devolve o Indicador C, caso não o reconheça. Caso contrário, entrega seu próprio código interno de zero a três. Se a chamada a ROM falhar, a procura de SLTATR continuará, até que a tabela esteja esgotada, quando um erro de "Nome de arquivo errado" será gerado(6E6BH). Se a chamada a ROM for bem sucedida, o código interno da ROM é somado com a sua posição em SLTATR e multiplicado por quatro, para produzir um código de dispositivo global. O código de base para cada entrada em SLTATR será mostrado a seguir na forma hexadecimal. Os marcadores "SS" e "PS" mostram os números dos Conectores Primário e Secundário correspondentes, sendo que cada conector é formado de quatro páginas:

| SS0         | SS1         | SS2         | SS3         |     |
|-------------|-------------|-------------|-------------|-----|
| 00 04 08 0C | 10 14 18 1C | 20 24 28 2C | 30 34 38 3C | PS0 |
| 40 44 48 4C | 50 54 58 5C | 60 64 68 6C | 70 74 78 7C | PS1 |
| 80 84 88 8C | 90 94 98 9C | A0 A4 A8 AC | B0 B4 B8 BC | PS2 |
| C0 C4 C8 CC | D0 D4 D8 DC | E0 E4 E8 EC | F0 F4 F8 FC | PS3 |

**Figura 44** Códigos de dispositivos.

O código de dispositivo global é utilizado pelo Interpretador até chegar o momento da ROM executar uma operação real de dispositivo. Af é convertido de volta em uma Identificação do conector da ROM em um endereço de base e em um código interno de dispositivo para que o acesso à ROM seja efetuado. Observe que os códigos de 0 a 8 são reservados para identificadores de acionadores de disco e aqueles de FCH até FFH para os dispositivos padrão GRP, CRT, LPT e CAS. Com a estrutura de hardware atual do MSX, estes códigos correspondem a configurações de ROM fisicamente improváveis sendo, portanto, seguros para serem utilizados para operações específicas do Interpretador.

Endereço ..... 564AH

Esta rotina é utilizada pelo despachante de funções(6F8FH), quando ele encontra um código de dispositivo que não pertence a um dos dispositivos padrões. Primeiro, o código de dispositivo será convertido em uma posição na tabela SLTATR, em uma Identificação de Conector no registrador A e em um endereço base de ROM no registrador H(7E2DH). O endereço do manipulador de dispositivo em ROM será lido das posições seis e sete da ROM(7E1AH) e colocado no par de registradores IX. A Identificação do conector será colocado no byte mais alto do par de registradores IY, o código de dispositivo interno da Rom em DEVICE e o manipulador do dispositivo em ROM será chamado por meio da rotina padrão CALSLT.

Endereço ..... 566CH

Este ponto de entrada no analisador sintático de linguagem macro é utilizado pelo manipulador da instrução "DRAW". Um ponto de entrada posterior(65A2H) é utilizado pelo manipulador da instrução "PLAY". A string de comandos é avaliada (4C64H) e seu espaço liberado(67D0H). Depois de colocar um bloco com terminação zero na pilha do Z80, o comprimento e endereço do corpo da string serão colocados em MCLLEN e MCLPTR e o controle irá para a Ronda Principal do Analisador Sintático (RPAS).

Endereço ..... 56A2H

Esta é a Ronda Principal do Analisador Sintático (RPAS) da linguagem macro, e é utilizada para processar a string de comandos associado com as instruções "DRAW" e "PLAY". No início, o comprimento da string está em MCLLEN, o endereço da string em MCLPTR e o endereço da tabela de comandos relevantes está em MCLTAB. A tabela de comando de cada instrução contém as letras de comando válidos, juntamente com os endereços do manipulador de comando associado. A tabela "DRAW" está em 5D83H e a tabela "PLAY" em 752EH.

Primeiro, a RPAS pega o próximo caractere da string de comandos(56EEH). Se não houver mais nenhum caractere sobrando, o próximo descritor de string será retirado da pilha(568CH). Se este for zero, o analisador sintático terminará(5709H) se MCLFLG mostrar uma instrução "DRAW" ativada; caso contrário o controle será transferido de volta ao manipulador da declaração "PLAY"(7494H).

Existindo um caractere de comando, a tabela de comandos atual será pesquisada para verificar sua validade, e se nenhuma concordância for encontrada, será gerado um

erro de "Chamada ilegal de função"(475AH). Em seguida, a entrada da tabela de comandos será examinada, e caso o bit 7 esteja ativado o comando toma um parâmetro numérico opcional. Se este estiver presente, será coletado e colocado no par de registradores DE(571CH); caso contrário será tomado um valor default de um. Depois de colocar um retorno ao início da RPAS na pilha do Z80, o controle será transferido ao manipulador de comandos no endereço retirado da tabela de comandos.

Endereço ..... 56EEH

Esta rotina é utilizada pelo analisador sintático de linguagem macro para pegar o próximo caractere da string de comando. Se MCLLEN for zero, a rotina terminará com o Indicador Z, se não houver nenhum caractere sobrando. Caso contrário, o próximo caractere será tomado do endereço contido em MCLPTR e devolvido no registrador A, e se o caractere for minúsculo, será convertido em maiúsculo. Em seguida, MCLPTR será incrementado e MCLLEN decrementado.

Endereço ..... 570BH

Esta rotina é utilizada pelo analisador sintático de linguagem macro para devolver um caractere indesejável à string de comando. MCLLEN será incrementado e MCLPTR será decrementado.

Endereço ..... 5719H

Esta rotina é utilizada pelo analisador sintático de linguagem macro para coletar um parâmetro numérico da string de comando. O resultado, um inteiro com sinal, e não podendo ser uma expressão, será devolvido no par de registradores DE. O primeiro caractere é pego e examinado, e se for um "+", será ignorado e tomado o próximo caractere(5719H). Se for um "-", um retorno será preparado para a rotina de negação(5719H). Se for um "=", o valor da variável seguinte será devolvido(577AH). Caso contrário, caracteres sucessivos serão tomados e um produto binário será acumulado, até que um caractere não-numérico seja encontrado.

Endereço ..... 575AH

Esta rotina é utilizada pelos manipuladores de "=" e "X" do analisador sintático de linguagem macro. O nome da Variável é copiado em BUF, até que o delimitador ";" seja encontrado, e caso leve mais do que trinta e nove caracteres para ser encontrado, será gerado um erro de "Chamada ilegal de função(475AH). Caso contrário, o

controle será transferido ao manipulador de Variáveis do Avaliador de Fatores(4E9BH) e o conteúdo da Variável será devolvido em DAC.

Endereço . . . . . 577AH

Esta rotina é utilizada pelo analisador sintático de linguagem macro para processar o comando “=” em um parâmetro de comando. O valor da Variável será obtido(575AH), convertido para um inteiro(2F8AH) e colocado no par de registradores DE.

Endereço . . . . . 5782H

Esta rotina é utilizada pelo analisador sintático de linguagem macro para processar o comando “X”. A Variável será processada(575AH) e, depois de verificar que há espaço de pilha disponível(625EH), os conteúdos atuais de MCLLEN e MCLPTR são empilhados. Em seguida, o controle será transferido ao ponto de entrada do analisador sintático(5679H), para obtenção do descritor da Variável e para o processador a nova string de comandos.

Endereço . . . . . 579CH

Esta rotina é utilizada por diversas instruções gráficas para avaliar um par de coordenadas no texto do programa. As coordenadas têm que estar entre parênteses, com uma vírgula separando os operandos componentes. Se o par de coordenadas for precedido por um átomo “STEP”(DCH), cada valor de componente será somado ao componente correspondente das coordenadas gráficas atuais em GRPACX e GRPACY; caso contrário os valores absolutos serão devolvidos. A coordenada X será devolvida em GRPACX, GXPOS e no par de registradores BC. A coordenada Y será devolvida em GRPACY GYPOS e no par de registradores DE.

Há dois pontos de entrada para esta rotina, sendo que aquele que será utilizado, dependerá do chamador esperar o retorno de mais do que um par de coordenadas. A declaração “LINE”, por exemplo, aguarda dois pares de coordenadas, sendo o primeiro mais flexível. O ponto de entrada em 579CH é utilizado para coletar o primeiro par de coordenadas e aceitará os caracteres “-” ou “@-”, como representando as coordenadas gráficas atuais. O ponto de entrada em 57ABH é utilizado, para o segundo par de coordenadas e requer um operando explícito.

Endereço ..... 57E5H

Este é o manipulador da instrução "PRESET". A cor de fundo atual será tomada de BAKCLR e o controle irá para o manipulador de "PSET".

Endereço ..... 57EAH

Este é o manipulador de "PSET". Depois que o par de coordenadas tiver sido avaliado(57ABH) a cor de frente atual será tomada de FORCLR e utilizada como default, ao se colocar a cor da tinta(5850H). As coordenadas gráficas atuais serão convertidas em um endereço físico, via rotinas padrões SCALXY e MAPXYC, e a cor do pixel atual será estabelecida por meio da rotina padrão SETC.

Endereço ..... 5803H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "POINT". Os conteúdos atuais de CLOC, CMASK, GYPOS, CXPOS, GRPACY, e GRPACX são empilhados e o operando do par de coordenadas avaliado(57ABH). A cor do novo pixel é lida por meio das rotinas padrões SCALXY, MAPXYC e READC e colocada em DAC como um inteiro(2F99H). Os valores das coordenadas antigas serão retirados da pilha e restaurados. Observe que o valor de -1 será devolvido se as coordenadas do ponto estiverem fora da tela.

Endereço ..... 5850H

Esta rotina gráfica é utilizada para avaliar um operando de cor opcional no texto do programa e torná-la a cor atual da tinta. Depois de verificar o modo da tela(59BCH), o operando de cor será avaliado(521CH) e colocado no ATRBYT. Se não existir nenhum operando, o código da cor fornecido no registrador A será colocado no ATRBYT.

Endereço ..... 5871H

Esta rotina gráfica devolve, no par de registradores HL, a diferença entre os conteúdos de GXPOS e o par de registradores BC. Se o resultado for negativo (CXPOS<BC), será negado para produzir a amplitude absoluta e o Indicador C será devolvido.

Endereço ..... 5883H

Esta rotina gráfica devolve no par de registradores HL, a diferença entre o conteúdo de GYPOS e o par de registradores DE. Se o resultado for negativo ( $CPOS < DE$ ), será negado para produzir a amplitude absoluta e o Indicador C será devolvido.

Endereço ..... 588EH

Esta rotina gráfica permuta os conteúdos de GYPOS e o par de registradores DE.

Endereço ..... 5898H

Primeiro esta rotina gráfica permuta os conteúdos de GYPOS e o par de registradores DE(588EH), depois permuta os conteúdos de GXPOS e o par de registradores BC. Se for entrado em 589BH, apenas a segunda operação será executada.

Endereço ..... 58A7H

Este é o manipulador da instrução "LINE". O primeiro par de coordenadas (X1,Y1) será avaliado(579CH) e colocado nos pares de registradores BC, DE. Depois de verificar a existência do átomo "-" (F2H), o segundo par de coordenadas (X2,Y2) será avaliados(57ABH) e deixado em GRPACX, GRPACY e GXPOS, GYPOS. Depois de estabelecer a cor da tinta(584DH), o texto do programa será verificado pela existência de uma opção "B" ou "BF" e, conforme o caso, será executada a operação de retângulo(5912H), retângulo cheio(58BFH) ou traçado-de-linha(58FCH). Nenhuma destas operações afeta as coordenadas gráficas atuais em GRPACX e GRPACY, que são deixadas em X2, Yz.

Endereço ..... 58BFH

Esta rotina executa a operação de preenchimento de um retângulo. Dado que as coordenadas fornecidas definem pontos diagonalmente opostos do retângulo, duas quantidades precisarão ser derivadas delas. O tamanho horizontal do retângulo será obtido da diferença entre X1 e X2, fornecendo o número de pixels que deverão ser ligados por linha. O tamanho vertical será obtido pela diferença entre Y1 e Y2, fornecendo o número de linhas requeridas. Iniciando no endereço físico X1, Y1 e movendo para baixo, por meio da rotina padrão DOWNC, o número requerido de linhas de pixels serão preenchidas por meio da utilização repetida da rotina padrão NSETCX.



Endereço ..... 58FCH

Esta rotina executa a operação de traçado-de-linha. Depois de traçar a linha(593CH) CXPOS e GYPOS serão colocados em X2, Y2 de GRPACX e GRPACY.

Endereço ..... 5912H

Esta rotina executa a operação de um retângulo. O retângulo será produzido traçando uma linha(58FCH) entre cada um dos quatro cantos. As coordenadas de cada canto serão derivadas a partir dos operandos iniciais, intercambiando as coordenadas dos pares. A seqüência do desenho é:

- (1) X1, Y2 a X2, Y2
- (2) X1, Y1 a X2, Y1
- (3) X2, Y1 a X2, Y2
- (4) X1, Y1 a X1, Y2

Endereço ..... 593CH

Esta rotina traça uma linha entre os pontos (X1, Y1), fornecidos no par de registradores BC e DE e (X2, Y2), fornecidos em GXPOS e GYPOS. A operação do laço-principal de desenho(5993H) será melhor ilustrada por um exemplo, digamos LINE (0,0) - (10,4). Para alcançar o ponto final da linha a partir do começo deverão ser tomados em conjunto dez passos horizontais( $X2-X1$ ) e quatro passos para baixo ( $Y2-Y1$ ). A melhor aproximação de uma reta requer portanto, dois e meio passos horizontais para cada passo descendente ( $X2-X1/Y2-Y1$ ). Sendo isto impossível na prática, uma vez que só poderão ser tomados apenas passos inteiros, a taxa correta poderá ser conseguida na média.

O método utilizado é somar a diferença Y a um contador, cada vez que for tomado um passo para à direita. Quando o contador exceder, o valor da diferença X, ele será zerado e um passo para baixo será tomado, o que é realmente uma divisão inteira entre as duas diferenças. Algumas vezes, os passos para baixo serão produzidos a cada dois passos para à direita, outras vezes a cada três passos para à direita. A média, porém, será um passo para baixo a cada dois passos e meio para à direita. Um programa BASIC equivalente é mostrado abaixo com um LINE ligeiramente deslocado, para comparação:

```

10 SCREEN 0
20 INPUT "X,Y INICIAIS";X1,Y1
30 INPUT "X,Y FINAIS";X2,Y2
40 SCREEN 2
50 X=X1:Y=Y1:L=X2-X1:S=Y2-Y1:CT=L/2
60 PSET(X,Y)
70 CT=CT+S:IF CT>L THEN 90
80 CT=CT-L:Y=Y+1
90 X=X+1:IF X<=X2 THEN 60
100 LINE(X1,Y1+5)-(X2,Y2+5)
110 GOTO 110

```

O exemplo acima sofre de três limitações. A linha tem de deslizar para baixo, tem que ser para à direita e a inclinação não pode exceder quarenta e cinco graus da horizontal (um passo para baixo a cada passo à direita).

A rotina vencerá a primeira limitação se examinar as coordenadas Y1 e Y2 antes de iniciar o traçado. Se Y2 for maior ou igual a Y1, indicando uma linha com inclinação para cima ou horizontal, os dois pares de coordenadas serão trocados. Agora a linha está inclinada para baixo e será traçada do ponto final ao inicial.

A segunda limitação será vencida ao examinar X1 e X2, antecipadamente, para determinar o sentido da inclinação da linha. Se X2 for maior ou igual a X1, a inclinação da linha será para à direita e um JP do Z80 para a rotina RIGHTC será colocado em MINUPD/MAXUPD (veja em seguida), a fim de ser utilizado pelo laço principal de desenho; caso contrário será colocado ali um JP para a rotina padrão LEFTC.

A terceira limitação será vencida pela comparação da diferença entre as coordenadas X com a diferença entre as coordenadas Y, antes de traçar a linha, para determinar a inclinação. Se  $X2 - X1$  for menor do que  $Y2 - Y1$ , a inclinação da linha será menor do que quarenta e cinco graus com a horizontal. O método simples mostrado anteriormente para LINE (0,0) – (10,4) não funciona para inclinações maiores do que quarenta e cinco graus, uma vez que a taxa máxima de descida é obtida quando um passo para baixo é tomado para cada passo horizontal. Funcionará, no entanto, se as direções de passo forem trocadas. Assim LINE (0,0) – (4,10) requer um passo para à direita a cada dois passos e meio descendentes. MINUPD contém um JP do Z80 para a rotina padrão de direção do passo “normal” e MAXUPD contém um JP para a rotina padrão de direção do passo da “inclinação”. Estes JP serão utilizados pelo laço principal de desenho. Para ângulos rasos, MINUPD aponta para DOWNC e MAXUPD para LEFTC ou RIGHTC. Para ângulos maiores, MINUPD aponta para LEFTC ou RIGHTC e MAXUPD para DOWNC. Para ângulos maiores, os valores dos contadores também serão trocados, a diferença X será somada ao contador e a diferença Y utilizada como o limitador do contador. As variáveis MINDEL e MAXDEL serão utilizadas pelo laço principal do traçado para conter estes valores de contagem, com MINDEL contendo a diferença do ponto extremo menor e MAXDEL o maior.

Um ponto interessante é que o contador de referência, contido em CTR no programa acima e no par de registradores DE na ROM, é pré-carregado com a metade da maior diferença do ponto extremo, em vez de ser colocado em zero. Isto tem efeito de rachar a primeira “escada” da linha em duas seções, uma no começo da linha e outra no fim, melhorando a aparência da linha.

Endereço . . . . . 59B4H

Esta rotina gráfica desloca o conteúdo do par de registradores DE um bit para à direita.

Endereço . . . . . 59BCH

Esta rotina gera um erro de “Chamada ilegal de função”(475AH), caso a tela não esteja no Modo Gráfico ou Modo Multicolorido.

Endereço . . . . . 59C5H

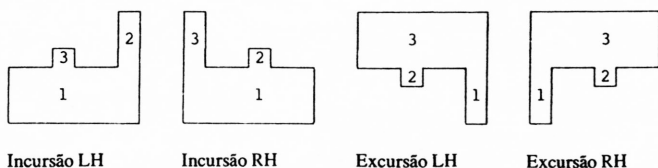
Este é o manipulador da instrução “PAINT”, O par de coordenadas iniciais é avaliado(579CH), a cor de tinta preparada(584DH) e o operando da cor de contorno opcional é avaliado(521CH) e colocado em BDRATR. O par de coordenadas iniciais é verificado para garantir que está dentro da tecla(5E91H) e ficará sendo o endereço físico do pixel atual, por uma chamada à rotina padrão MAPXYC. A distância à fronteira da direita é medida(5ADCH) e, se for zero, o manipulador terminará. Caso contrário, a distância ao limite esquerdo é medida(5AEDH) e a soma das duas será colocada no par de registradores DE como largura da região. Em seguida, a posição atual é empilhada duas vezes(5AECB), primeiro com um indicador de final(00H) e depois com um indicador de “para baixo”(40H). em seguida, o controle será transferido ao laço-principal da pintura(5A26H) com um indicador de “para cima”(C0H) no registrador B.

Endereço . . . . . 5A26H

Este é o laço principal da pintura. A largura da região é mantida no par de registradores DE, a direção da pintura para cima ou baixo, no registrador B e o endereço físico do pixel atual é aquele do pixel adjacente ao limite esquerdo da tela. É tomado um passo vertical para a linha seguinte, por meio das rotinas padrões TUPC ou TDOWNC, e a distância ao limite da tela do lado direito é medida(5ADCH). Em seguida, a distância ao limite da tela do lado esquerdo é medida e a linha entre os limites preenchidas(5AEDH). Se nenhuma alteração for encontrada na posição dos dois limites, o controle será transferido ao início do laço principal para continuar a pintar na mesma direção. Se uma alteração for encontrada, uma inflexão ocorreu e a ação apropriada será tomada.

Há quatro tipos de inflexões, LH ou incursivos, onde o limite relevante se move para dentro, e LH ou RH excursivos, em que se move para fora. Um exemplo de cada tipo será mostrado abaixo com zonas numeradas indicando a ordem da pintura, durante um

movimento para cima. Uma região secundária será mostrada dentro de cada região com inflexão para complementação:



**Figura 45** Inflexões de limite.

Ocorre uma excursão LH quando a distância ao limite esquerdo é não-zero, ocorre uma excursão RH quando a largura da região atual é maior do que aquela da linha anterior. A excursão será ignorada quando for menor do que dois pixels. A posição atual (o canto inferior esquerdo da região 3 na figura 45) será empilhada(5AC2H), a direção da pintura invertida, tendo a pintura reiniciando no canto superior esquerdo da região excursiva.

Uma inclusão RH ocorre quando a largura da região atual é menor do que aquela da linha anterior. Caso a incurção seja total, isto é, a largura da região atual for zero, atingimos uma zona morta e a última posição e direção serão retirados da pilha(5A1FH) e a pintura reiniciará naquele ponto. Caso contrário, a posição atual e a direção serão empilhadas(5AC2H) e a pintura reiniciará na parte inferior esquerda da região incursiva.

Uma incurção LH é cuidada, automaticamente, durante a procura do limite da direita e não requer nenhuma ação explícita do laço principal de pintura.

Endereço . . . . . 5AC2H

Esta rotina é utilizada pelo manipulador da instrução "PAINT" para salvar a posição de pintura atual e sua direção na pilha do Z80. O bloco de parâmetros de seis bytes é feito da seguinte forma:

|         |       |                         |
|---------|-------|-------------------------|
| 2 bytes | ..... | Conteúdo atual de CLOC  |
| 1 byte  | ..... | Direção atual           |
| 1 byte  | ..... | Conteúdo atual de CMASK |
| 2 bytes | ..... | Largura atual da região |

Depois que os parâmetros forem empilhados, uma verificação será feita para saber se ainda existe espaço suficiente de pilha(625EH).

Endereço ..... 5ADCH

Esta rotina é utilizada pelo manipulador da instrução "PAINT" para localizar o limite da direita. A largura da região da linha anterior é passada para a rotina padrão SCANR no par de registradores DE, indicando o número máximo de pixels da cor de contorno, que poderão inicialmente ser pulados. O resto da contagem-de-pulo será devolvido e colocado em SKPCNT e o número de pixels examinados, mas que não forem de contorno serão colocados em MOVCNT.

Endereço ..... 5AEDH

Esta rotina é utilizada pelo manipulador da instrução "PAINT" para localizar o limite esquerdo. O ponto final da procura do limite do lado direito será salvo e temporariamente e o ponto inicial será tomado de CSAVEA e CSAVEM e transformado no endereço físico atual do pixel. Em seguida, o limite da esquerda será localizado por meio da rotina padrão SCANL, que também preenche toda região, e o ponto final do lado direito recuperado e colocado em CSAVEA e CSAVEM.

Endereço ..... 5B0BH

Esta rotina é usada pelo manipulador de instrução "CIRCLE" para negar o conteúdo do par de registradores DE.

Endereço ..... 5B11H

Este é o manipulador da instrução "CIRCLE". Após avaliar as coordenadas do centro(579CH), o raio será avaliado(520FH), multiplicado(325CH) por  $\sin(\pi/4)$  e colocado em CNPNTS. A cor da tinta será estabelecida (584DH), o ângulo inicial avaliado(5D17H) e colocado em CSTGNT e o ângulo final avaliado(5D17H) e colocado em CENCNT. Se o ângulo final for menor do que o ângulo inicial, os dois valores serão permutados e CPLOTf será feito não-zero. A razão de aspecto será avaliado(4C64H) e, se for maior do que um, será tomado sua recíproca(3267H) e CSCLXY será feito não-zero, para indicar a compressão do eixo X. A razão de aspecto será multiplicada(325CH) por 256, convertida em um inteiro(2F8AH) e colocada em ASPECT como uma fração binária de um byte. Os pares de registradores DE e HL serão colocados na posição inicial, no perímetro do círculo ( $X = \text{RADIUS}$ ,  $Y = 0$ ) e o controle irá ao laço principal do círculo.

Endereço . . . . . 5BBDH

Este é o laço principal do círculo. Devido ao alto grau de simetria em um círculo será apenas necessário computar as coordenadas do arco de zero a quarenta e cinco graus. Os outros sete segmentos serão produzidos por rotação e reflexão destes pontos. A equação paramétrica para um círculo unitária, em que T é o ângulo de zero a  $\pi/4$  é:

$$X = \cos(T)$$

$$Y = \sin(T)$$

Uma computação direta utilizando esta equação, ou a forma funcional correspondente  $X = \text{SQRT}(1 - Y^2)$ , é muito lenta. Em seu lugar, é utilizada a primeira derivada:

$$dx/dy = -Y/X$$

Sendo o ponto de partida dado ( $X = \text{RADIUS}$ ,  $X = 0$ ), a alteração da coordenada X para cada alteração da coordenada Y poderá ser computada utilizando-se a derivada. Além do mais, pelo fato da resolução gráfica estar limitada a um pixel, será apenas necessário saber quando a soma das alterações da coordenada X alcança uma unidade e depois decrementar X. Portanto,

Decremente X quando  $(Y1/X) + (Y2/X) + (Y3/X) + \dots \Rightarrow 1$

Portanto, decremente quando  $(Y1 + Y2 + Y3 + \dots)/X \Rightarrow 1$

Portanto, decremente quando  $Y1 + Y2 + Y3 + \dots \Rightarrow X$

Tudo o que é preciso para identificar uma alteração de coordenada X é totalizar os valores da coordenada Y de cada passo, até que o valor da coordenada X seja excedido. O laço principal do laço contém a coordenada X no par de registradores HL, a coordenada Y no par de registradores DE e o total geral em CRCSUM.

Um programa BASIC equivalente para um círculo de raio arbitrário de 160 pixels será:

```
10 SCREEN 2
20 X=160:Y=0:SOMA=0
30 PSET (X,191-Y)
40 SOMA=SOMA+Y:Y=Y+1
50 IF SOMA<X THEN 30
60 SOMA=SOMA-X:X=X-1
70 IF X>Y THEN 30
80 CIRCLE (0,191),155
90 GOTO 90
```

Os pares de coordenadas gerados pelo laço principal são aqueles de um círculo “virtual” tais como tarefas como reflexão axial, compressão elíptica e translação do centro que será tratadas em um nível inferior(5C06H).

Endereço . . . . . 5C06H

Esta rotina é utilizada pelo laço principal do círculo, para converter um par de coordenadas, nos pares de registradores HL e DE, em oito pontos simétricos na tela. A coordenada Y será, inicialmente, negada(5BOBH), refletindo-a no eixo dos X, e os primeiros quatro pontos serão produzidos por rotações sucessivas, em sentido horário de noventa graus(5C48H). Em seguida, a coordenada Y será negada novamente(5BOBH) e mais quatro pontos serão produzidos(5C48H).

Uma rotação em sentido horário será executada pela substituição das coordenadas X e Y e negando a nova coordenada Y, assim um ponto (40,10) ficaria (10, -40). Presumindo uma razão de aspecto de 0.5, por exemplo, a seqüência completa de oito pontos seria:

- (1) X, -Y\*0.5
- (2) -Y, -X\*0.5
- (3) -X, Y\*0.5
- (4) Y, X\*0.5
- (5) Y, -X\*0.5
- (6) -X, -Y\*0.5
- (7) -Y, X\*0.5
- (8) X, Y\*0.5

Vemos acima que, ignorando o sinal das Coordenadas, há apenas quatro termos envolvidos. Portanto, em vez de executar a multiplicação da razão de aspecto relativamente lenta(5CEBH) para cada ponto, o termo  $X*0.5$  e  $Y*0.5$ , poderão ser preparados antecipadamente e a seqüência completa será gerada por intercâmbio e negação dos quatro termos. Com a razão de aspectos mostrada acima, as condições iniciais serão estabelecidas de modo que o par de registradores HL = X, o par de registradores DE = -Y\*0.5, CXOFF = Y e CYOFF = X\*0.5 e os pontos sucessivos serão produzidos pelas seguintes operações:

- (1) Troque HL e CXOFF, negue HL.
- (2) Troque DE e CYOFF, negue DE.

Em paralelo com a computação de cada coordenada do círculo, o número de pontos requeridos para se atingir o começo do segmento contendo o ponto é mantido em CPCNT8. Ele será, inicialmente, zero e aumentará de  $2 * \text{RADIUS} * \sin(\pi/4)$  à medida que cada rotação de noventa graus for feita. À medida que cada um dos oito pontos for produzidos, o valor de sua coordenada Y será somado ao conteúdo de CPCNT8 e comparado com os ângulos inicial e final, para determinar a ação correta. Se o ponto estiver entre os dois ângulos e CPLOTf for zero, ou se estiver fora dos ângulos e CPLOTf for não-zero, as coordenadas serão adicionadas às coordenadas do centro do círculo(5CDCH) e o ponto será ligado pelas rotinas padrões SCALXY, MAPXYC e SETC. Se o ponto for igual a um dos dois ângulos e o bit associado ativado em CLINEF, as coordenadas serão acrescentadas às coordenadas do centro do círculo(5CDCH) e uma linha será ligada ao centro(593CH). Se nenhuma destas condições for aplicável, nenhuma outra ação será tomada, a não ser prosseguir para o próximo ponto.

Endereço . . . . . 5CEBH

Esta rotina multiplica o valor das coordenadas fornecidas no par de registradores DE pela razão de aspecto contido em ASPECT, com o resultado devolvido no par de registradores DE. O método padrão de deslocamento e soma será utilizado, mas a operação será executada como duas multiplicações de um único byte para evitar problemas de extravasamento.

Endereço . . . . . 5D17H

Esta rotina é utilizada pelo manipulador da instrução "CIRCLE" para converter um operando de ângulo à forma requerida pelo laço principal do círculo, com o resultado colocado no par de registradores DE. O método utilizado será fundamentalmente correto, e eliminará uma computação trigonométrica, porém os resultados produzidos serão imprecisos. Isto é demonstrado pelo exemplo abaixo, que traça uma linha até o ponto a trinta graus no perímetro de um círculo:

```
10 SCREEN 2
20 PI=4*ATN(1)
30 CIRCLE(100,100),80,PI/6
40 LINE(100,100)-(100+80*COS(PI/6),
    100-80*SIN(PI/6))
50 GOTO 50
```



A rotina deveria devolver o número de pontos que deveriam ser produzidos pelo laço principal do círculo, antes que o círculo requerido seja alcançado. Isto pode ser computado notando-se primeiro que haverá  $\text{INT}(\text{ÂNGULO}/(\text{PI}/4))$ , segmentos de quarenta e cinco graus antes do segmento contendo o ângulo requerido. Além do mais, cada segmento de quarenta e cinco conterá  $\text{RAIO} * \text{SIN}(\text{PI}/4)$  pontos, uma vez que este é o valor da coordenada Y final. Portanto, o número de pontos requeridos para alcançar o início do segmento contendo o ângulo será o produto destes dois números. A contagem total será produzida somando este valor do número de pontos requeridos para cobrir qualquer ângulo que faltar no segmento final, isto é,  $\text{RAIO} * \text{SIN}(\text{ÂNGULO-QUE-FALTA})$  pontos.

Infelizmente, a rotina computa o número de pontos em um segmento por aproximação linear a partir do tamanho total do segmento na suposição errônea que pontos sucessivos subentendem ângulos iguais. Assim, no exemplo acima, a contagem de pontos computada para o ângulo é  $30/45 * (80 * 0.707107) \approx 37$ , em lugar do valor correto de quarenta. O erro produzido pela rotina está, portanto, no máximo, no centro de cada segmento de quarenta e cinco graus e reduzido a zero nos pontos terminais.

Endereço ..... 5D6EH

Este é o manipulador da instrução "DRAW". O par de registradores DE é estabelecido para apontar à tabela de comando em 5D83H e o controle será transferido ao analisador sintático de linguagem macro(566CH).

Endereço ..... 5D83H

Esta tabela contém as letras de comando válidas e endereços associados aos comandos de instrução "DRAW". Aqueles comandos da tabela que tomam um parâmetro, e conseqüentemente têm bit 7 ligado, estão indicados com um asterisco:

| COMANDO | ENDEREÇO |
|---------|----------|
| U*      | 5DB1H    |
| D*      | 5DB4H    |
| L*      | 5DB9H    |
| R*      | 5DBCH    |
| M       | 5DD8H    |
| E*      | 5DCAH    |
| F*      | 5DC6H    |
| G*      | 5DD1H    |
| H*      | 5DC3H    |
| A*      | 5E4EH    |
| B       | 5E46H    |
| N       | 5E42H    |
| X       | 5782H    |
| C*      | 5E87H    |
| S*      | 5E59H    |

Endereço . . . . . 5DB1H

Este é o manipulador do comando "U" da instrução "DRAW". O funcionamento dos comandos "D", "L", "R", "E", "F", "G" e "H" é bem semelhante, de modo que nenhuma descrição, em separado, de seus manipuladores será dada. O parâmetro numérico opcional é fornecido pelo analisador sintático de linguagem macro, no par de registradores DE. Um manipulador transforma este parâmetro inicial em um deslocamento horizontal no par de registradores BC e um desvio vertical no par DE. Por exemplo, caso um movimento para esquerda ou para cima seja requerido, o parâmetro será negado(5BOBH). Caso seja requerido um movimento diagonal, o parâmetro será duplicado, produzindo um deslocamento horizontal e um vertical. Assim que os deslocamentos tenham sido preparados, o controle será transferido para a rotina de desenho de linha(5DFFH).

Endereço . . . . . 5DD8H

Este é o manipulador do comando "M" da instrução "DRAW". O caractere que segue a letra do comando é examinado e, em seguida, os dois parâmetros são tomados da string de comando(5719H). Caso o caractere inicial seja "+" ou "-", os parâmetros serão vistos como deslocamentos e sua escala será calculada(5E66H), serão girados por passos sucessivos de noventa graus conforme determinado por DRWANG e depois adicionados às coordenadas gráficas atuais(5CDCH), para determinar os pontos terminais. Caso DRWFLG mostre o modo "B" como estando inativo, uma linha será traçada(5CCDH) das coordenadas gráficas atuais ao ponto terminal. Caso DRWFLG mostre o modo "N" como estando inativo, as coordenadas terminais serão colocadas em GRPACX e GRPACY para se tornarem as novas coordenadas gráficas atuais. Finalmente, DRWFLG será zerado, desligando os modos "B" e "N", e o manipulador terminará.

Endereço . . . . . 5E42H

Este é o manipulador de comando "N" da instrução "DRAW". DRWFLG fica valendo 40H, simplesmente.

Endereço . . . . . 5E46H

Este é o manipulador do comando "B" da instrução "DRAW". DRWFLG fica valendo 80H, simplesmente.

Endereço . . . . . 5E4EH

Este é o manipulador do comando "A" da instrução "DRAW". O parâmetro é verificado quanto ao tamanho e colocado em DRWANG.

Endereço . . . . . 5E59H

Este é o manipulador do comando "S" da instrução "DRAW". O parâmetro é verificado quanto ao tamanho e colocado em DRWSCL.

Endereço . . . . . 5E66H

Esta rotina é utilizada pelos manipuladores dos comandos "U", "D", "L", "R", "E", "F", "G", "H", e "M" (no modo deslocamento) da instrução "DRAW", para colocar em escala o deslocamento fornecido no par de registradores DE. A escala é indicada pelo conteúdo de DRWSCL. A não ser que DRWSCL seja zero, o que termina a rotina, o desvio será multiplicado com somas repetitivas e depois dividido por quatro(59B4H). Para eliminar a escala um comando "S0" ou "S4" deveria ser utilizado.

Endereço . . . . . 5E87H

Este é o manipulador do comando "C" da instrução "DRAW". O parâmetro será colocado em ATRBYT, por meio da rotina padrão SETATR. Não há nenhuma verificação no MSB do parâmetro, de modo que valores ilegais como "C265" serão aceitos sem uma mensagem de erro.

Endereço . . . . . 5E91H

Esta rotina é utilizada pelo manipulador da instrução "PAINT" para verificar, por meio da rotina padrão SCALXY, se as coordenadas nos pares de registradores RC e DE estão dentro da tela. Em caso negativo, um erro de "Chamada ilegal de função" será gerado(475AH).

Endereço . . . . . 5E9FH

Este é o manipulador de instrução "DIM". Um retorno para 5E9AH é preparado, de modo que várias matrizes poderão ser processadas, DIMFLG será colocado em não-zero e o controle irá para a rotina e busca de Variável.

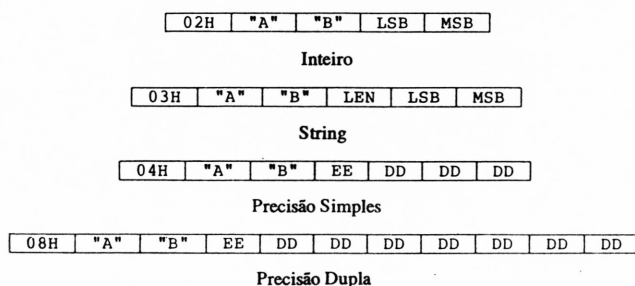
Endereço . . . . . 5EA4H

Esta é a rotina de busca da Variável. No início, o par de registradores HL aponta para o primeiro caractere do nome da Variável no texto do programa. No final, o par de registradores HL aponta para o caractere após o nome, e o par de registradores DE aponta para o primeiro byte do conteúdo da Variável da Área de Armazenamento de Variáveis. O primeiro caractere do nome será tomado do texto do programa, verificado para garantir que é um alfabético maiúsculo(67A7H) e colocado no registrador C. O segundo caractere opcional, com um valor default de zero será colocado no registrador B, podendo ser alfabético ou numérico. Qualquer caractere alfanumérico adicional é simplesmente pulado. Caso siga um caractere de sufixo de tipo ("% ", "\$ ", "!" ou " ") depois do nome, este será convertido no código de tipo correspondente (2, 3, 4 ou 8) e colocado em VALTYP. Caso contrário, o tipo default da Variável será tomado de DEFTBL utilizando-se a primeira letra do nome para localizar a entrada apropriada.

SUBFLG é verificado para determinar como qualquer índice entre parênteses seguindo o nome deverá ser tratado. Este indicador é normalmente zero, mas é modificado pelos manipuladores das instruções "ERASE"(01H), "FOR"(64H), "FN"(80H) ou "DEF FN"(80H) para indicar a ação apropriada. No caso de "ERASE", o controle será transferido diretamente para a rotina de pesquisa de Matriz(5FE8H), sem a necessidade da presença de um índice entre parênteses. Nos casos de "FOR", "FN" e "DEF FN" o controle será transferido diretamente para a rotina de pesquisa de variável simples (5FO8H), sem nenhuma verificação feita por um índice entre parênteses. Numa situação normal, o texto do programa é verificado quanto aos caracteres "(" ou "[". Se um deles estiver presente, o controle será transferido para a rotina de pesquisa de Matriz(5FBAH); caso contrário o controle irá para a rotina de pesquisa de Variáveis simples.

Endereço . . . . . 5FO8H

Esta é a rotina de pesquisa de Variável simples. Existem quatro tipos de Variáveis simples, cada um composto de um cabeçalho seguido pelo conteúdo de Variável. O primeiro byte do cabeçalho contém o código do tipo e os dois bytes seguintes contêm nome da Variável. O conteúdo da Variável será uma das três formas numéricas padrão ou, para o tipo string, o comprimento e o endereço da string. Cada um dos quatro tipos é apresentado a seguir:



**Figura 46** Variáveis simples.

Primeiro NOFUNS é verificado para determinar se uma função definida pelo usuário está atualmente sendo avaliada. Em caso afirmativo a procura é feita primeiro em PARM1, e apenas se esta falhar ela passa a ser feita na Área de Armazenamento de Variáveis. Um método de busca linear será utilizado, onde os dois caracteres do nome e byte de tipo de cada Variável da Área de Armazenamento serão comparados com os caracteres e tipo até que se encontre uma concordância ou que o fim da Área de Armazenamento seja atingido. Caso a busca seja bem-sucedida, a rotina terminará com o endereço do primeiro byte do conteúdo da Variável no par de registradores DE. Caso a procura não seja bem-sucedida, a Área de Armazenamento da Matriz será movida para cima e a nova Variável será acrescentada ao final das existentes e inicializadas com zero.

Há duas exceções à criação automática de uma nova Variável. Caso a procura esteja sendo levada a efeito pela função "VARPTR", e isto é determinado examinando-se o endereço de retorno, nenhuma Variável será criada. Em vez disto a rotina termina com o par de registradores DE colocados em zero(5F61H), provocando uma subsequente "Chamada ilegal de função". A segunda exceção ocorre quando a procura estiver sendo feita pelo Avaliador de Fatores, isto é, quando a Variável é recém-criada dentro de uma expressão. Neste caso DAC é zerado para tipo numérico, e carregado com o endereço de um descrito de string sem significado de comprimento zero, envolvendo, desta forma, um resultado zero(5FA7H). Estas ações são projetadas para impedir que o Avaliador de Expressões crie uma nova Variável ("VARPTR" é a única função a tomar diretamente uma Variável como argumento e não uma expressão requerendo assim proteção separada). Se isto não fosse assim, a atribuição à uma Matriz, por meio do manipulador da instrução "LET", falharia, pois qualquer Variável simples criada durante a avaliação de uma expressão alteraria o endereço da Matriz.

Endereço . . . . . 5FBAH

Esta é a rotina de procura de Matriz. Há quatro tipos de Matrizes, cada uma composta por um cabeçalho mais um número de elementos. O primeiro byte do cabeçalho contém o código de tipo, os dois bytes seguintes os nomes da Matriz e os dois seguintes, um desvio ao início da Matriz seguinte. Isto é seguido por um único byte, que contém o número de dimensões da Matriz e uma lista de números de elementos. Cada número de elementos de dois bytes contém o número máximo de elementos por dimensão. Estes são armazenados na ordem inversa, com o primeiro correspondendo ao último índice. O conteúdo de cada elemento da Matriz é idêntico ao conteúdo da Variável simples correspondente. A Matriz inteira AB%(3,4) será mostrada a seguir, com cada elemento identificado por seus índices, com a memória mais alta em direção ao topo da página:

|                                                                                                                                                                                                                                                                                                                                                      |       |       |                   |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|-------|-------------------|------------|------------------|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <table><tr><td>(0,4)</td><td>(1,4)</td><td>(2,4)</td><td>(3,4)</td></tr><tr><td>(0,3)</td><td>(1,3)</td><td>(2,3)</td><td>(3,3)</td></tr><tr><td>(0,2)</td><td>(1,2)</td><td>(2,2)</td><td>(3,2)</td></tr><tr><td>(0,1)</td><td>(1,1)</td><td>(2,1)</td><td>(3,1)</td></tr><tr><td>(0,0)</td><td>(1,0)</td><td>(2,0)</td><td>(3,0)</td></tr></table> |       |       |                   |            |                  |                  | (0,4) | (1,4) | (2,4) | (3,4) | (0,3) | (1,3) | (2,3) | (3,3) | (0,2) | (1,2) | (2,2) | (3,2) | (0,1) | (1,1) | (2,1) | (3,1) | (0,0) | (1,0) | (2,0) | (3,0) |
| (0,4)                                                                                                                                                                                                                                                                                                                                                | (1,4) | (2,4) | (3,4)             |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| (0,3)                                                                                                                                                                                                                                                                                                                                                | (1,3) | (2,3) | (3,3)             |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| (0,2)                                                                                                                                                                                                                                                                                                                                                | (1,2) | (2,2) | (3,2)             |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| (0,1)                                                                                                                                                                                                                                                                                                                                                | (1,1) | (2,1) | (3,1)             |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| (0,0)                                                                                                                                                                                                                                                                                                                                                | (1,0) | (2,0) | (3,0)             |            |                  |                  |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| 02H                                                                                                                                                                                                                                                                                                                                                  | "A"   | "B"   | Offset<br>2DH 00H | Dim<br>02H | Count<br>05H 00H | Count<br>04H 00H |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |

**Figura 47** Matriz Inteira.

Cada índice é avaliado, convertido em um inteiro(4755H) e colocado na pilha do Z80, até que um parênteses de fechamento seja encontrado, não precisando combinar com o inicial. Uma procura linear, em seguida, é feita na Área de Armazenamento de Matrizes pelos dois caracteres do nome e do tipo. Caso a procura seja bem sucedida, DIMFLS será verificado e será gerado um erro de “Matriz redimensionada”(405EH), caso mostre uma declaração “DIM” ativa. A não ser que uma declaração “ERASE” esteja ativa, a rotina termina com o par de registradores BC apontando ao início da Matriz(3297H); em seguida, o número de dimensões da Matriz será comparado com o número encontrado e um erro de “Índice fora de faixa” será gerado, caso não combinem. Passando-se por estes testes, o controle será transferido ao ponto de computação do endereço do elemento(607DH).

Se a procura não for bem sucedida e uma declaração “ERASE” estiver ativa, será gerado um erro de “Chamada ilegal de uma função”(475AH), caso contrário, a nova Matriz será acrescentada ao final da Área de Armazenamento de Matriz existentes. A inicialização da nova Matriz prossegue pelo armazenamento das duas características do nome, o código de tipo e a dimensionalidade (a contagem dos índices) seguido da contagem de elementos para cada dimensão. Caso DIMFLG mostre uma declaração “DIM” ativa, a contagem dos elementos será determinada pelos índices. Caso a matriz esteja sendo criada por default por exemplo, com uma declaração como “A(1, 2, 3) = 5”,

será utilizado um valor default de onze. À medida que cada contagem de elemento é armazenada, o tamanho total da Matriz será acumulado no par de registradores DE, por multiplicações sucessivas(314AH) das contagens de elementos e tamanho dos elementos (o tipo da Matriz). Depois de verificar se esta quantidade de memória está disponível(6267H) STREND será aumentado, a nova área zerada e o tamanho da Matriz armazenamento, de forma ligeiramente modificada, imediatamente após os dois caracteres do nome. A não ser que a Matriz esteja sendo criada por default, quando o endereço do elemento tem que ser computado, a rotina termina.

Este é o ponto de computação do endereço do elemento da rotina de procura da Matriz. A posição de um determinado elemento dentro de uma Matriz envolve a multiplicação(314AH) dos índices, contagem de elemento e tamanho do elemento. Como há uma variedade de maneiras em que isto poderia ser feito, o método realmente utilizado é melhor ilustrado com um exemplo. A localização do elemento (1, 2, 3) na Matriz 4\*5\*6 seria inicialmente computado como  $((3 * 5) + 2) * 4 + 1$ . Isto é, em seguida, multiplicado pelo tamanho do elemento (tipo) e acrescentado ao endereço base da Matriz, para a obtenção do endereço do elemento requerido. O método de computação é uma forma otimizada que minimiza o número de passos necessários, e é equivalente à avaliação  $(3 * (4 * 5)) + (2 * 4) + (1)$ . O endereço do elemento é colocado no par de registradores DE.

Endereço ..... 60B1H

Este é o manipulador da instrução "PRINT USING". O controle é transferido para cá, do manipulador da instrução "PRINT" geral, depois que o dispositivo de saída aplicável estiver preparado. Ao terminar, o controle volta para o ponto de saída da declaração "PRINT" geral(4AFFH) para restaurar a saída de vídeo normal. A string da formatação é avaliada(4C65H) e o endereço e comprimento do corpo da string serão obtidos do descritor. O apontador do texto do programa é, em seguida, salvo temporariamente. Cada caractere da string de formatação é examinado, até que um dos possíveis caracteres de máscara seja encontrado. Se o caractere não for uma máscara será simplesmente enviado para a saída, por meio da rotina padrão OUTDO. Uma vez que o início de uma máscara seja encontrado, esta será varrida, até que seja encontrado um caractere que não pertença a ela. Em seguida, o controle passa para a rotina de saída numérica(6192H) ou a rotina de saída da string(6211H).

Em qualquer dos casos o apontador de texto do programa é restaurado ao par de registradores HL e o próximo operando avaliado(4C64H). Para uma saída numérica, a informação recebida da varredura da máscara é passada para a rotina de conversão numérica(3426H) nos registradores A, B e C e a string resultante apresentado(6678H).

Para cada tipo de saída, o texto do programa e a string de formatação serão examinados, a fim de determinar se há mais algum caractere. Caso não exista nenhum operando, o manipulador termina. Se a string de formatação tiver sido exaurida, então será reinicializado(60BFH). Caso contrário, a varredura continua da posição atual para o operando seguinte(60F6H).

Endereço . . . . . 6250H

Esta rotina é utilizada pela Ronda Principal do interpretador e pela rotina de procura de Variáveis, para mover para cima um bloco de memória. Primeiro, uma verificação será feita para garantir se existe memória suficiente(6267H), e depois o bloco de memória será movido. O endereço fonte superior é fornecido no par de registradores BC e o endereço de destino superior no par de registradores HL. A cópia pára quando o conteúdo do par de registradores BC for igual ao do par de registradores DE.

Endereço . . . . . 625EH

Esta rotina é utilizada para verificar se existe memória suficiente disponível entre o topo da Área de Armazenamento de Matrizes e a base da pilha do Z80. No início, o registrador C contém o número de palavras que o chamador requer. Se isto estreitar o espaço para menos do que duzentos bytes, será gerado um erro de “Falta de memória”.

Endereço . . . . . 6286H

Este é o manipulador da instrução “NEW”. TRCFLG, AUTFLG e PTRFLG são zerados e o elo de terminação zero será colocado no início da Área de Texto do Programa. VARTAB será apontado para o byte que segue o elo final e o controle irá para a rotina RUN-CLEAR.

Endereço . . . . . 629AH

Esta rotina é utilizada pelos manipuladores das instruções “NEW”, “RUN” e “CLEAR”, para inicializar as variáveis do interpretador. Todas as interrupções são limpas(636EH) e os tipos de Variáveis default em DEFTBL serão colocados em dupla precisão. RNDX é repostado(2C24H) e ONEFLG, ONELIN e OLDTXT são zerados. MEMSIZ é copiado em FRETOP para limpar a Área de Armazenamento de String e DATPTR é colocado no início da Área de Texto do Programa(63C9H). O conteúdo de VARTAB é copiado em ARYTAB e STREND, para limpar qualquer Variável. Todos os buffers de E/S serão fechados(6C1CH) e NLOONLY será repostado. SAVSTK e o SP do Z80



serão repostos de STKTOP e TEMPPT será reposto no início de TEMPST para liberar qualquer descritor de string. A impressora é fechada(7304H) e a saída restaurada para a tela(4AFFH). Finalmente, PRMLN, NOFUNS, PRMLN2, FUNACT, PRMSTK e SUBFLG serão zerados e a rotina terminará.

Endereço ..... 631BH

Esta rotina é utilizada pelos manipuladores das instruções “DISPOSITIVO ON” para ativar uma fonte de interrupção, com o endereço do byte de status TRPTBL correspondente ao dispositivo no par de registradores HL. As interrupções são ativadas ligando o bit 0 do byte de status. Em seguida, os bits 1 e 2 serão examinados e, se o dispositivo tiver sido interrompido e uma interrupção ocorrida, ONGSBF é incrementado(634FH) de modo que a Ronda de Execução possa processá-la no fim da instrução. Finalmente, o bit 1 do byte de status é desligado para liberar qualquer condição de parada que possa existir.

Endereço ..... 632BH

Esta rotina é utilizada pelos manipuladores das instruções “DISPOSITIVO OFF” para desativar uma fonte de interrupção, com o endereço do byte de status TRPTBL correspondente ao dispositivo no par de registradores HL. Os bits 0 e 2 são examinados para determinar se uma interrupção ocorreu desde o fim da última instrução, e, em caso afirmativo, ONGSBF é decrementado(6362H) para impedir que a Ronda de Execução possa pegá-lo. Em seguida, o byte de status é zerado.

Endereço ..... 6331H

Esta rotina é utilizada pelos manipuladores das instruções “DISPOSITIVO STOP”, para suspender o processamento de interrupções de uma fonte de interrupção, com o endereço do byte de status TRPTBL correspondente ao dispositivo no par de registradores HL. Bits 0 e 2 são examinados para determinar se uma interrupção ocorreu desde o final da última instrução e, em caso afirmativo, ONGSBF é decrementado(6362H) para impedir que a Ronda Principal possa pegá-la. Em seguida, o bit 1 do byte de status será ativado.

Endereço ..... 633EH

Esta rotina é utilizada pelo manipulador da instrução “RETURN” para liberar a condição de parada temporária imposta durante as sub-rotinas em BASIC, acionadas por

meio de interrupção, com o endereço do byte de status TRPTBL correspondente ao dispositivo no par de registradores HL. Os bits 0, 1 e 2 são examinados para determinar se uma interrupção parada ocorreu desde que a sub-rotina foi ativada pela primeira vez. Em caso afirmativo, ONGSBF é incrementado(634FH) de modo que a Ronda de Execução a pegue no fim da instrução. Bit 1 do byte de status será então desligado. Observe que qualquer instrução "DISPOSITIVO STOP" dentro de uma sub-rotina acionada por uma interrupção não terá efeito.

Endereço . . . . . 6358H

Esta rotina é utilizada pelo processador de interrupção da Ronda de Execução(6389H) para limpar uma interrupção, antes de ativar a sub-rotina BASIC, com o endereço do byte de status TRFTBL correspondente ao dispositivo no par de registradores HL. ONGSBF é decrementado e o bit 2 do byte de status é desligado.

Endereço . . . . . 636EH

Esta rotina é utilizada pela rotina RUN-CLEAR(629AH) para limpar todas as interrupções. Os setenta e oito bytes de TRPTBL e os dez bytes de FNKFLG são zerados.

Endereço . . . . . 6389H

Este é o processador de interrupção da Ronda de Execução. Primeiro ONEFLG é examinado para determinar se existe uma condição de erro. Em caso afirmativo, a rotina termina, e nenhuma interrupção será processada até a liberação do erro. Em seguida, CURLIN é examinado e, caso o intérprete esteja no modo direto, a rotina terminará. Estando tudo em ordem, será feita uma procura nos vinte e seis bytes de status da TRPTBL para encontrar a primeira interrupção ativa. Observe que dispositivos perto do início da tabela terão prioridade em relação aos de mais adiante. Quando o primeiro byte de status ativo for encontrado, isto é, com os bits 0 e 2 ativados, o endereço associado será tomado de TRTPBL e colocado no par de registradores DE. Em seguida, a interrupção será limpa(6358H) e o dispositivo interrompido(6331H), antes que o controle seja transferido ao manipulador "GOSUB"(47CFH).

Endereço . . . . . 63C9H

Este é o manipulador da instrução "RESTORE". Caso não exista nenhum operando de linha, DATPTR será colocado no início da Área de Armazenamento do Programa. Caso contrário, o operando será coletado(4769H), o texto do programa

procurado para se encontrar a linha relevante(34295H) e seu endereço colocado em DATPTR.

Endereço ..... 63E3H

Este é o manipulador da instrução "STOP". Caso exista texto adicional na instrução, o controle será transferido ao manipulador da instrução "STOP ON/OFF/STOP"(77A5H). Caso contrário, o registrador A será colocado em 01H e o controle irá para o manipulador da instrução "END".

Endereço ..... 63EAH

Este é o manipulador da instrução "END". É utilizado também, com pontos de entrada diferentes, pela instrução "STOP", pelo CTRL-STOP e no encerramento do programa por fim-de-texto. Primeiro, ONEFLG será zerado e depois, apenas para a declaração "END", todos buffers de E/S serão fechados(6C1CH). A posição do texto do programa atual será colocada em SAVTXT e OLDTXT e o número de linha atual em OLDLIN para ser utilizado por qualquer instrução "CONT" subsequente. A impressora será fechada(7304H), será acionado um CR, LF para a tela(7323H) e o par de registradores HL serão colocados para apontar para a mensagem "Break", em 3FDCH. Para a declaração "END" e fim de texto o controle será transferido para o ponto "OK" da Ronda Principal(411EH). No caso de CTRL-STOP, o controle será transferido para o final do manipulador de erro(40FDH), a fim de apresentar a mensagem "Break".

Endereço ..... 6424H

Este é o manipulador da instrução "CONT". A não ser que seja zero, quando um erro "Não posso continuar" é gerado, o conteúdo de OLDTXT é colocado no par de registradores HL e o de OLDLIN em CURLIN. Em seguida, o controle volta à Ronda de Execução para executar a partir da posição antiga do texto do programa. Um programa não poderá continuar depois que CTRL-STOP for utilizada para interromper um programa DE DENTRO de uma instrução, pela rotina CKCNTC, ao invés de o ser entre instruções.

Endereço ..... 6438H

Este é o manipulador da instrução "TRON", com TRCFLG simplesmente feito não-zero.

Endereço . . . . . 6439H

Este é o manipulador da instrução “TROFF”, com TRCFLG simplesmente colocado em zero.

Endereço . . . . . 643EH

Este é o manipulador da instrução “SWAP”. A primeira Variável é localizada(5EA4H) e seu conteúdo é copiado em SWPTMP. As posições desta Variável e do final da Área de Armazenamento de Variáveis são salvas temporariamente. Em seguida, a segunda Variável é localizada(5EA4H) e o seu tipo comparado com o da primeira. Caso os tipos não concordem, será gerado um erro de “Tipo Incorreto”(406DH). O final atual da Área de Armazenamento de Variáveis é, em seguida, comparada com o final antigo e um erro de “Chamada ilegal de função” será gerado(475AH) caso difiram. Finalmente, o conteúdo da segunda Variável será copiado na posição da primeira Variável(2EF3H) e o conteúdo de SWPTMP na posição da segunda Variável(2EF3H).

As verificações feitas pelo manipulador significam que a segunda Variável, caso seja simples e não uma Matriz, terá que existir sempre antes que uma declaração “SWAP” for encontrada, ou será gerado um erro. A razão disto é que, supondo que a primeira Variável seja uma Matriz, então a criação de uma segunda Variável (simples) moveria para cima a Área de Armazenamento de Matrizes, invalidando a sua posição salva. Observe o caso perfeitamente legal de uma primeira Variável simples e uma segunda Variável simples recém-criada, também é rejeitado.

Endereço . . . . . 6477H

Este é o manipulador da instrução “ERASE”. Primeiro, SUBFLG é colocado em 01H, para controlar a rotina de procura de Variável, e a Matriz localizada (5EA4H). Todas as Matrizes seguintes serão movidas para baixo e STREND colocado em seu valor novo, inferior. O texto do programa será, então, verificado e, caso siga uma vírgula, o controle será devolvido ao início do manipulador.

Endereço . . . . . 64A7H

Esta rotina verifica se o caractere cujo endereço é fornecido pelo par de registradores HL é maiúsculo, e, em caso afirmativo, devolve o Indicador NC.

Endereço . . . . . 64AFH

Este é o manipulador da instrução "CLEAR". Caso nenhum operando esteja presente, o controle será transferido para a rotina RUN-CLEAR(62A1H), a fim de remover todas as Variáveis atuais. Caso contrário, o operando do espaço string será avaliado(4756H) e seguido pelo operando opcional do topo de memória(542FH). O valor do topo de memória é verificado e será gerado um erro de "Chamada ilegal de uma função"(475AH), caso seja menor do que 8000H ou maior do que F380H. O espaço requerido pelos buffers de E/S (267 bytes cada) e a Área de Armazenamento de String serão subtraídos do valor do topo da memória e será gerado um erro de "Falta de memória"(6275H), caso haja menos do que 160 bytes sobrando na base da Área de Armazenamento da Variável. Estando tudo em ordem, HIMEM, MEMSIZ e STKTOP serão colocados em seus novos valores e os apontadores de armazenamento restantes limpos, por meio da rotina RUN-CLEAR(62A1H). O armazenamento no buffer de E/S é re-alocado(7E6BH) e o manipulador termina.

Infelizmente, a computação de MEMSIZ e STKTOP, quando é especificado um novo topo de memória, está incorreta, resultando na colocação do topo da Área de Armazenamento de String um byte além do correto. Isto poderá ser visto com o seguinte exemplo, onde uma string ilegal é aceita:

```
10 CLEAR 200,&HF380
20 A$=STRING$(201,"A")
30 PRINT FRE("")
```

Deveria existir uma instrução DEC HL a mais em 64EBH. Por isso, os novos valores de MEMSIZ e STKTOP são inicialmente, colocados um byte mais acima. Quando a rotina RUN-CLEAR é chamada, MEMSIZ é copiado em FRETOP, o topo da Área de Armazenamento de String, ficando este também um byte muito acima. Apesar de MEMSIZ e STKTOP serem corretamente computados, quando os apontadores de arquivo são repostos, FRETOP permanece com seu valor incorreto. Quando a instrução "FREE" é executada na linha 30 e iniciada a coletânea do lixo string, FRETOP será restaurado ao seu valor correto mas, pelo fato da string extravasar a Área de Armazenamento de String em um byte, a quantidade de espaço livre apresentada é -1 bytes. Para atribuir corretamente todos os apontadores do sistema, qualquer alteração no topo da memória deveria ser imediatamente seguida por outra instrução CLEAR, sem operandos.

Endereço . . . . . 6520H

Esta rotina computa a diferença entre o conteúdo do par de registradores HL e

DE. É uma duplicação da pequena seção de código de 64ECH até 64F1H e é completamente inútil.

Endereço . . . . . 6527H

Este é o manipulador da instrução "NEXT". Havendo texto adicional na instrução, a Variável de laço é localizada(5EA4H); caso contrário será tomado um endereço default de zero. Em seguida, na pilha é procurado o bloco parâmetro "FOR" correspondente(3FE2H). Se nenhum bloco de parâmetro for encontrado, ou se um bloco de parâmetro "GOSUB" for encontrado primeiro, será gerado um erro "NEXT sem FOR"(405BH). Presumindo que o bloco seja encontrado, a seção entreposta da pilha, juntamente com quaisquer blocos "FOR" que possa conter, será descartada. O tipo de Variável de laço, em seguida, é tomado do bloco de parâmetro e examinado para determinar a precisão requerida durante as operações subseqüentes.

O valor de STEP é tomado do bloco de parâmetros e somado(3172H, 324EH ou 2697H) ao conteúdo atual da Variável de laço que, em seguida, é atualizada. Em seguida, o novo valor é comparado (2F4DH, 2F21H e 2F5CH) com o valor terminal do bloco de parâmetro, a fim de determinar se o laço terminou(65B6H). O laço termina para um STEP positivo, se o novo valor do laço for maior do que o valor terminal. O laço termina para um passo negativo, se o novo valor do laço for menor do que o valor terminal. Se o laço não tiver terminado, a posição de texto do programa original e o número de linha serão tomados do bloco de parâmetro e o controle será transferido à Ronda de Execução (45FDH). Se o laço tiver terminado, o bloco de parâmetros será descartado da pilha e, caso haja mais texto, o controle será transferido de volta ao início do manipulador, senão, o controle será transferido à Ronda de Execução para a execução da próxima instrução(4601H).

Endereço . . . . . 65C8H

Esta rotina é utilizada pelo Avaliador de Expressões para encontrar a relação (<=>) entre operandos string. O endereço do primeiro descritor de string é fornecido na pilha do Z80 e o endereço do segundo em DAC. O resultado é devolvido no registrador A e nos indicadores como nas rotinas de relação numérica:

String 1=String 2 . . . A=00H, Indicador Z, NC  
 String 1<String 2 . . . A=01H, Indicador NZ, NC  
 String 1>String 2 . . . A=FFH, Indicador NZ,C

A comparação se inicia no primeiro caractere de cada string e continua até que dois caracteres difiram, ou uma das strings seja exaurida. Em seguida, o controle retorna ao Avaliador de Expressões(4F57H) para colocar o resultado numérico verdadeiro ou falso em DAC.

Endereço ..... 65F5H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "OCT\$" em um operando contido em DAC. Primeiro o número é convertido para forma textual em FBUFFR(371EH) e, em seguida, a string resultante será criada(6607H).

Endereço ..... 65FAH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "HEX\$" em um operando contido em DAC. Primeiro o número é convertido à forma textual em FBUFFR(3722H) e, em seguida, a string resultante será criada(6607H).

Endereço ..... 65FFH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "BIN\$" em um operando contido em DAC. Primeiro o número será convertido para a forma textual em FBUFFR(371AH) e, em seguida, a string resultante será criada(6607H).

Endereço ..... 6604H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "STR\$" em um operando contido em DAC. Primeiro, o número é convertido para a forma textual em FBUFFR(3425H) e depois analisado para determinar o seu comprimento e endereço (6635H). Depois de verificar que existe espaço suficiente disponível(668EH), a string será copiada na Área de Armazenamento de String(67C7H) e o descritor resultante será criado(6654H).

Endereço ..... 6627H

Esta rotina primeiro verifica se há espaço suficiente na Área de Armazenamento de String para uma string cujo comprimento é fornecido no registrador A(668EH). O comprimento da string e o endereço em que a string será colocada na Área de Armazenamento de String será, em seguida, copiado para DSCTMP.

Endereço . . . . . 6636H

Esta rotina é utilizada pelo Avaliador de Fatores para analisar a string de caracteres, cujo endereço é fornecido no par de registradores HL. A string de caracteres é varrida até que um caractere terminal (OOH ou ") seja encontrado. O comprimento e o endereço inicial são, em seguida, colocados em DSCTMP(662AH) e o controle será colocado na rotina de criação do descritor.

Endereço . . . . . 6654H

Esta rotina é utilizada pelas funções string para criar um descritor de resultado. O descritor é copiado de DSCTMP à próxima posição disponível em TEMPST e seu endereço colocado em DAC. A não ser que TEMPST esteja lotado, quando é gerado um erro de "Fórmula de string muito complexa", TEMPST será aumentado em três bytes e a rotina terminará.

Endereço . . . . . 6678H

Esta rotina apresenta a mensagem, ou string, cujo endereço é fornecido no par de registradores HL. A string é analisada(6635H) e seu armazenamento liberado(67D3H). Em seguida, caracteres sucessivos serão tomados da string e apresentados, por meio da rotina padrão OUTDO, até que a string seja exaurida.

Endereço . . . . . 668EH

Esta rotina verifica se há espaço na Área de Armazenamento da String para acrescentar a string, cujo comprimento é fornecido pelo registrador A. No término, o par de registradores DE aponta para o endereço inicial na Área de Armazenamento de String, onde a string deverá ser colocada. Primeiro, o comprimento da string é subtraído da área livre atual contida em FRETOP. Isto será, em seguida, comparado com STKTOP, a posição mais baixa permitida para armazenamento de string, a fim de determinar se há espaço para a string. Em caso afirmativo, FRETOP é atualizado com a nova posição e a rotina termina. Caso o espaço seja insuficiente para a string, será iniciada uma coleta de lixo(66B6H) para tentar eliminar qualquer string desnecessária. Se, depois da coleta de lixo não houver espaço suficiente, será gerado um erro de "Falta de espaço para string".



Endereço . . . . . 66B6H

Este é o coletor de lixo string, cuja função é eliminar qualquer string desnecessária da Área de Armazenamento de String. O problema principal com as Variáveis string, em oposição às numéricas, é que seu comprimento varia. Se os corpos das string fossem armazenados com suas Variáveis na Área de Armazenamento de Variáveis, mesmo instruções aparentemente simples como  $A\$ = A\$ + "X"$  necessitariam movimentação de milhares de bytes de memória, o que diminuiria de forma acentuada a velocidade de execução. O método utilizado pelo Interpretador a fim de superar este problema é de manter os corpos das strings separados das Variáveis. Assim, strings são mantidas na Área de Armazenamento de Strings, e cada Variável contém um descritor de três bytes contendo o comprimento e endereço da string associada. Sempre que uma string é atribuída à uma Variável, ela simplesmente, é adicionada ao monte de strings existentes na Área de Armazenamento de String, e o descritor da Variável é alterado. Nenhuma tentativa é feita para eliminar qualquer string prévia, pertencente à Variável, reestruturando a área, uma vez que isto eliminaria qualquer ganho de rendimento.

Caso sejam feitas muitas atribuições de Variáveis é inevitável que a Área de Armazenamento de String fique repleta. Em um programa típico, muitas destas strings ficarão sem utilização, como resultado de atribuições anteriores. Coleta do lixo é o processo pelo qual estas strings desnecessárias são removidas. Cada Variável string na memória, incluindo Matrizes e as Variáveis locais presentes, durante a avaliação de funções definidas pelo usuário, é examinada até que seja encontrada aquela cuja string mais alta é armazenada no monte. Em seguida, esta string é movida ao topo da Área de Armazenamento de Strings e os conteúdos das Variáveis serão modificados para apontar para a nova posição. O proprietário da string seguinte ao mais elevado será então encontrado e o processo repetido, até que cada string pertencente a uma Variável tenha sido compactada.

Caso haja um grande número de Variáveis, a coleta do lixo poderá levar um tempo considerável. O processo pode ser visto em funcionamento no programa seguinte que, repetidamente, atribui a string "AAAA" a cada elemento da Matriz A\$. O programa será processado em alta velocidade nas primeiras duzentas e cinquenta atribuições e, depois fará uma pausa para eliminar cinquenta strings desnecessárias. Cinquenta atribuições adicionais poderão ser feitas, antes que seja necessária uma nova coleta de lixo:

```
10 CLEAR 1000
20 DIM A$(200)
30 FOR N=0 TO 200
40 A$(N)=STRING$(4,"A")
```

```
50 PRINT".";  
60 NEXT N  
70 GOTO 30
```

A Área de Armazenamento de String é utilizada, também, para conter as strings intermediárias produzidas durante a avaliação de uma expressão. Pelo fato de tantas funções strings tomarem múltiplos argumentos, ("MID\$" toma três, por exemplo), o gerenciamento de resultados intermediários é um problema maior. Para cuidar dele um procedimento padrão com resultados string foi adotado no Interpretador. O produtor de uma string, simplesmente, acrescenta o corpo da string ao monte na Área de Armazenamento de String; acrescenta o descritor ao monte de descritores em TEMPST e coloca o endereço do descritor em DAC. Cabe ao usuário do resultado liberar este armazenamento(67DOH), uma vez que tenha processado a string. Esta regra se aplica à todas as partes do sistema, desde os manipuladores de funções individuais (pelo Avaliador de Expressão), até os manipuladores de instruções, com apenas duas exceções.

A primeira exceção ocorre quando o Avaliador de Fatores encontra uma string explicitamente declarada, como "QUALQUER", no texto do programa. Neste caso, não será necessário copiar a string na Área de Armazenamento de String, uma vez que o original será suficiente. A segunda exceção ocorre quando o Avaliador de Fatores encontra uma referência a uma Variável. Neste caso, não será necessário colocar uma cópia do descritor em TEMPST, uma vez que já existe um no interior da Variável.

Endereço . . . . . 6787H

Esta rotina é utilizada pelo Avaliador de Expressões para concatenar dois operandos string. O controle será transferido para cá, quando for encontrado o símbolo "+" após um operando string, de modo que a primeira ação a ser tomada é pegar o segundo operando string por intermédio do Avaliador de Fatores(4DC7H). Os comprimentos são, em seguida, tomados dos dois descritores de string e somados para verificar o comprimento da string combinada. Caso seja maior que 255 caracteres, será gerado um erro de "String muito longa". Depois de verificar se há espaço disponível na Área de Armazenamento de String(6627H), o armazenamento dos dois operandos é liberado (67D6H). Em seguida, a primeira string será copiada na Área de Armazenamento de Strings(67BFH) e seguido pela segunda(67BFH). Será criado o descritor resultante (6654H) e o controle devolvido ao Avaliador de Expressão(4C73H).

Endereço . . . . . 67DOH

Esta rotina libera qualquer armazenamento ocupado pela string, cujo descritor

tem seu endereço contido em DAC. O endereço do descritor é tomado de DAC e examinado para determinar se é aquele do último descritor em TEMPST(67EEH); em caso negativo a rotina termina. Caso contrário, TEMPPT será reduzido em três bytes, eliminando este descritor de TEMPST. O endereço do corpo da string é tomado do descritor e comparado com FRETOP para ver se é a string mais baixa na Área de Armazenamento de String, e, se não for, a rotina terminará. Caso contrário, o comprimento da string será acrescentada ao FRETOP, que, sem seguida, é atualizado com este novo valor, liberando o armazenamento ocupado pelo corpo da string.

Endereço ..... 67FFH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LEN" a um operando contido em DAC. O armazenamento do operando é liberado(67DOH) e o comprimento da string tomado do descritor e colocado no DAC como um inteiro(4FCFH).

Endereço ..... 680BH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "ASC" em um operando contido em DAC. O armazenamento do operando é liberado e o comprimento da string examinado(6803H), e se for zero, será gerado um erro de "Chamada ilegal de uma função"(475AH). Caso contrário, o primeiro caractere é tomado da string e colocado em DAC como um inteiro(4FCFH).

Endereço ..... 681BH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CHR\$" em um operando contido em DAC. Depois de verificar se existe espaço suficiente disponível(6625H), o operando será convertido em um inteiro de um único byte(521FH). Este caractere é, em seguida, colocado na Área de Armazenamento de Strings e será criado o descritor do resultado(6654H).

Endereço ..... 6829H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "STRING\$". Depois de verificar a existência de um caractere de parênteses de abertura, o operando de comprimento é avaliado e colocado no registrador E(521CH). O segundo operando é, em seguida, avaliado(4C64H). Se for numérico, será convertido em um inteiro de um byte(521FH) e colocado no registrador A. Se for uma string, o primeiro caractere

dela será tomado e colocado no registrador A(680FH). Em seguida, o controle irá para a função "SPACE\$", a fim de criar a string resultante.

Endereço ..... 6848H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "SPACE\$" em um operando contido em DAC. Primeiro, o operando é convertido para um inteiro de um byte, no registrador E(521FH). Depois de verificar se existe espaço suficiente disponível(6627H), o número necessário de espaços será copiado na Área de Armazenamento de String e será criado o descritor do resultando(6654H).

Endereço ..... 6861H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LEFT\$". O endereço do descritor do primeiro operando e do segundo operando inteiro serão fornecidos pela pilha do Z80. O tamanho da fatia é tomado da pilha(68E3H) e comparado com o comprimento da string fonte. Se o comprimento da string fonte for menor que o tamanho da fatia, aquela passa a ser o comprimento a ser extraído. Depois de verificar se um espaço suficiente está disponível(668EH), o número necessário de caracteres é copiado do início da string fonte para a Área de Armazenamento de Strings(67C7H). O armazenamento da string fonte é, em seguida, liberado(67D7H) e o descritor do resultado será criado(6654H).

Endereço ..... 6891H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "RIGHT\$". O endereço do descritor do primeiro operando e do segundo operando inteiro são fornecidos pela pilha do Z80. O tamanho da fatia é tomado da pilha(68E3H) e subtraído do comprimento da string fonte, para determinar a posição inicial da fatia. Em seguida, o controle será transferido para a rotina "LEFT\$", a fim de retirar a fatia(6865H).

Endereço ..... 689AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "MID\$". O endereço do descritor do primeiro operando e do segundo operando inteiro são fornecidos pela pilha do Z80. A posição inicial é tomada da pilha(68E6H) e verificada, e se for zero será gerado um erro de "Chamada ilegal de uma função"(475AH). O tamanho

opcional de fatia, em seguida, é avaliado(69E4H) e o controle será transferido para a rotina "LEFT\$", para a extração de fatia(6869H).

Endereço . . . . . 68BBH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "VAL" em um operando contido em DAC. O comprimento da string é tomado do descritor (6803H) e verificado; Se for zero, será colocado em DAC como um inteiro(4FCFH). Em seguida, o comprimento será acrescentado ao endereço inicial do corpo da string, para fornecer a localização do caractere que o segue. Este será temporariamente substituído com um byte zero e a string será convertida para a forma numérica em DAC(3299H). O caractere original é, em seguida, restaurado e a rotina terminará. O byte zero temporário, usado como delimitador, é necessário porque as strings serão colocadas lado a lado na Área de Armazenamento de String. Sem ele, o conversor numérico continuaria a conversão nas strings seguintes.

Endereço . . . . . 68E3H

Esta rotina é utilizada pelos manipuladores das funções "LEFT\$", "MID\$" e "RIGHT\$", para certificar que o próximo caractere do texto de programa seja "(" e depois desempilhar um operando da pilha do Z80 para o par de registradores DE.

Endereço . . . . . 68EBH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "INSTR". O primeiro operando, que pode ser a posição inicial ou string fonte, é avaliado(4C62H) e o seu tipo é testado. Se for uma string fonte, será tomado um valor default para a posição inicial. Se for o operando da posição inicial, seu valor será verificado e o operando da string fonte avaliado(4C64H). Em seguida, a string alvo é avaliada(4C64H) e liberados os armazenamentos dos dois operandos(67DOH). O comprimento da string alvo é verificado, e se for zero, a posição inicial será colocada em DAC(4FCFH). Em seguida, a string padrão é verificada contra caracteres sucessivos da string fonte, a partir na posição inicial, até que haja uma concordância ou que a string fonte seja esgotada. Numa procura bem sucedida, a posição do caractere do sub-string é colocada em DAC como um inteiro(4FCFH); caso contrário será devolvido um resultado zero.

Endereço . . . . . 696EH

Este é o manipulador da instrução “MID\$”. Depois de verificar a existência do caractere de parênteses de abertura, a Variável de destino é localizada(5EA4H) e verificada para garantir que seja um tipo string(3058H). O endereço do corpo da string, em seguida, é tomado da Variável e examinado para determinar se está dentro da Área de Texto do Programa, como seria o caso de uma string explicitamente declarada. Se for o caso, o corpo da string é copiado na Área e Armazenamento de Strings(6611H) e um novo descritor copiado para a Variável(2EF3H). Isto é feito para evitar a modificação do texto do programa. Em seguida, a posição inicial é avaliada(521CH) e verificada. Se for zero, será gerado um erro de “Chamada ilegal de uma função”(475AH). O operando opcional de comprimento de fatia é avaliado(69E4H), seguido pela string de substituição(4C5FH) cujo armazenamento, em seguida, será liberado(67DOH). Em seguida, caracteres serão copiados da string de substituição à string de destino, até que o comprimento da fatia esteja completo ou que a string de substituição esteja esgotada.

Endereço . . . . . 69E4H

Esta rotina é utilizada por diversas funções strings para avaliar um operando opcional(521CH) e entregar o resultado no registrador E. Caso nenhum operando esteja presente, um valor default de 255 será devolvido.

Endereço . . . . . 69F2H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “FRE” em um operando contido em DAC. Se o operando for numérico, a diferença de simples precisão entre o Apontador de Pilha do Z80 e o conteúdo do STREND é colocada em DAC(4FC1H). Se o operando for um tipo string, seu armazenamento é liberado(67D3H) e iniciada a coleta do lixo(66B6H). A diferença de precisão simples entre o conteúdo de FRETOP e aqueles de STKTOP será, em seguida, colocada em DAC(4FC1H).

Endereço . . . . . 6A0EH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo, para analisar uma especificação de arquivo como “A:ARQUIVO.BAS”. A especificação do arquivo consiste em três partes: o dispositivo, o nome do arquivo e a extensão do tipo. No início, o par de registradores HL aponta para o início da especificação no texto do programa. Ao terminar, o registrador D contém o código do dispositivo, o nome do arquivo estará nas posições de

zero a sete de FILNAM e a extensão de tipo nas posições de oito a dez. Quaisquer posições não utilizadas serão preenchidas com espaços.

A string de especificação de arquivo é avaliada(4C64H) e o seu armazenamento liberado(67DOH), e se a string for de comprimento zero, um erro de "Nome de arquivo incorreto" será gerado(6E6BH). O nome do dispositivo é analisado(6F15H) e caracteres sucessivos serão tomados da especificação do arquivo e colocados em FILNAM, até que a string seja esgotada, um caractere "." seja encontrado ou FILNAM esteja lotado. Será gerado um erro de "Nome de arquivo incorreto"(6E6BH), caso a especificação do arquivo contenha algum caractere de controle, cujo valor é menor do que 20H. Caso a especificação do arquivo contenha a extensão de tipo será gerado um erro de "Nome de arquivo incorreto"(6E6BH), se esta extensão for maior do que três caracteres ou o nome do arquivo tiver mais do que oito caracteres. Se não estiver presente nenhuma extensão de tipo, o nome do arquivo poderá ter qualquer comprimento, com os caracteres extra sendo simplesmente ignorados.

Endereço ..... 6A6DH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para localizar FCB do buffer de E/S, cujo número é fornecido pelo registrador A. Primeiro, o número do buffer é comparado com MAXFIL e será gerado um erro de "Nome de arquivo incorreto" (6E7DH), se for muito grande. Caso contrário, o endereço requerido é tomado do bloco de apontadores de arquivo e colocado no par de registradores HL, e o modo do buffer tomado do byte 0 de FCB e colocado no registrador A.

Endereço ..... 6A9EH

Esta rotina é utilizada pelos manipuladores de E/S de arquivos, para avaliar um número de buffer de E/S e localizar seu FCB. Qualquer caractere "#" é pulado(4666H) e o número do buffer avaliado(521CH). O FCB é localizado(6A6DH) e é gerado um erro de "Arquivo não aberto"(6E77H), se o byte do modo buffer for zero. Caso contrário, o endereço do FCB será colocado em PTRFIL para redirecionar a saída do Interpretador.

Endereço ..... 6AB7H

Este é o manipulador da instrução "OPEN". A especificação do arquivo é analisada(6A0EH) e qualquer modo que vier após o nome será convertido ao byte do modo correspondente: "FOR INPUT"(01H), "FOR OUTPUT"(02H) e "FOR APPEND"(08H). Se nenhum modo for explicitamente declarado, será considerado o modo

aleatório(04H). Os caracteres "AS" são verificados e o número do buffer avaliado (521CH); e se este for zero será gerado um erro de "Número incorreto de arquivo" (6E7DH). Em seguida, FCB é localizado(6A6DH) e será gerado um erro de "Arquivo já aberto"(6E6EH), se o byte do modo do buffer for diferente de zero. O código do dispositivo é colocado no byte 4 de FCB, a função de abertura (OPEN) despachada (6F8FH) e a saída do Interpretador devolvida à tela(4AFFH).

Endereço . . . . . 6B24H

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para fechar o buffer de E/S, cujo número é fornecido no registrador A. FCB é localizado(6A6DH) e, desde que o buffer esteja em utilização, a função de fechamento (CLOSE) é despachada (6F8FH) e o buffer preenchido com zeros(6CEAH). PTRFIL e o byte de modo de FCB serão, em seguida, zerados, para repor a saída do Intérprete para a tela.

Endereço . . . . . 6B5BH

Este é o manipulador das instruções "LOAD", "MERGE" e "RUN arquivo". A especificação do arquivo é analisada(6A0EH) e depois, apenas para "LOAD" e "RUN" o texto da instrução é examinado, a fim de determinar se a opção "R" de autoprocessoamento foi especificada. O buffer 0 de E/S é aberto para entrada(6AFAH) e o primeiro byte de FILNAM colocado em FFH caso o autoprocessoamento seja requerido. Em seguida, apenas para "LOAD" e "RUN", qualquer texto de programa será limpo por meio do manipulador da declaração "NEW"7(6287H). Como isto repõe a saída do Interpretador à tela, o buffer do FCB é novamente localizado e colocado em PTRFIL(6AAAH). Em seguida, o controle será transferido diretamente à Ronda Principal do Interpretador(4134H) para que o texto do programa seja carregado como se fosse digitado do teclado. Observe que nenhuma verificação de qualquer tipo de erro é feita nos dados lidos.

Endereço . . . . . 6BA3H

Este é o manipulador da instrução "SAVE". A especificação do arquivo é analisada(6A0EH) e o texto do programa examinado, para determinar se o sufixo "A" ASCII está presente. Isto é relevante apenas com BASIC de Disco, não fazendo nenhuma diferença em uma máquina MSX padrão. O buffer 0 de E/S é aberto para saída(6AFAH) e o controle é transferido ao manipulador da instrução "LIST"(522EH), para dar saída ao texto do programa. Observe que nenhuma informação de verificação de erro de qualquer tipo acompanha o texto.



Endereço . . . . . 6BDAH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para devolver o código do dispositivo do buffer de E/S atualmente ativo. O endereço de FCB é tomado do PTRFIL e, em seguida, o código do dispositivo é tomado do byte 4 de FCB e colocado no registrador A.

Endereço . . . . . 6BE7H

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para executar uma operação em um número de buffers de E/S. O endereço da rotina relevante é fornecido no par de registradores BC e o número de buffers no registrador A. Por exemplo, se o par de registradores BC contiver 6B24H e o registrador A contiver 03H, os buffers 3, 2, 1 e 0 serão fechados. A rotina terá uma função ligeiramente diferente se for iniciada com o Indicador NZ. Neste caso, os números dos buffers de E/S serão tomados sequencialmente do texto do programa e avaliados(521CH), antes que a operação seja executada (um caso típico poderia ser “#1, #2”).

Endereço . . . . . 6C14H

Este é o manipulador da instrução “CLOSE”. O par de registradores BC fica valendo 6B24H, o registrador A é carregado com o conteúdo de MAXFIL e os buffers fechados(6BE7H).

Endereço . . . . . 6C1CH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para fechar todos os buffers de E/S. O par de registradores BC fica valendo 6B24H, o registrador A é carregado com o conteúdo de MAXFIL e todos os buffers fechados(6BE7H).

Endereço . . . . . 6C2AH

Este é o manipulador da instrução “LFILES”. PRTFLG é feito não-zero, a fim de direcionar a saída para a impressora, e o controle vai para o manipulador da instrução “FILES”.

Endereço . . . . . 6C2FH

Este é o manipulador da instrução "FILES", sendo gerado um erro de "Chamada ilegal de uma função"(475AH) numa máquina MSX padrão.

Endereço . . . . . 6C35H

O controle é transferido aqui dos manipuladores gerais "PUT" e "GET"(7758H), quando o texto do programa for qualquer coisa diferente de um átomo "SPRITE". Em uma máquina MSX padrão será gerado um erro de "Somente E/S seqüencial"(6E86H).

Endereço . . . . . 6C48H

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para dar uma saída seqüencial ao caractere fornecido pelo registrador A. O caractere é colocado no registrador C e a função de saída seqüencial (sequential output) despachada(6F8FH).

Endereço . . . . . 6C71H

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para permitir a entrada seqüencial de um caractere. A função de entrada seqüencial (sequential input) é despachada(6F8FH) e o caractere devolvido no registrador A; o Indicador C indica uma condição EOF (Fim de Arquivo).

Endereço . . . . . 6C87H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "INPUT\$". O texto do programa é verificado para comprovar a existência dos caracteres "\$" e "(" e o operando de comprimento é avaliado, o FCB é localizado(6A9EH) e o byte de modo examinado. É gerado um erro de "Entrada além do final" (input past end) (6E83H), se o buffer não for uma entrada ou se o for de modo aleatório. Depois de verificar se há espaço disponível, (6627H) o número requerido de caracteres entra seqüencialmente(6C71H), ou estes serão coletados por meio da rotina padrão CHGET e copiados para a Área de Armazenamento de Strings. Finalmente, o descritor do resultado é criado(6654H).

Endereço ..... 6CEAH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para preencher com 256 zeros o buffer cujo endereço de FCB está contido em PTRFIL.

Endereço ..... 6CFBH

Esta rotina é utilizada pelos manipuladores de E/S de arquivo para devolver, no par de registradores HL, o endereço inicial do buffer cujo endereço de FCB está contido em PTRFIL. Isto envolve apenas a soma de nove ao endereço de FCB.

Endereço ..... 6DO3H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LOC" ao buffer de E/S, cujo número está contido em DAC. FCB é localizado(6A6AH) e a função LOC despachada(6F8FH). Será gerado um erro da "Chamada ilegal de uma função"(475AH) em uma máquina MSX padrão.

Endereço ..... 6D14H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "LOF" ao buffer de E/S, cujo número está contido em DAC. FCB é localizado(6A6AH) e a função IOF despachada(6F8FH). Será gerado um erro de "Chamada ilegal de uma função"(475AH) em uma máquina MSX padrão.

Endereço ..... 6D25H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar função "EOF" ao buffer de E/S, cujo número está contido em DAC. FCB é localizado(6A6AH) e a função de fim-de-arquivo (LOF) é despachada.

Endereço ..... 6D39H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "FPOS" ao buffer de E/S, cujo número está contido em DAC. FCB é localizado(6A6AH) e a função FPOS é despachada(6F8FH). Será gerado um erro de "Chamada ilegal de uma função"(475AH) em uma máquina MSX padrão.

Endereço . . . . . 6D48H

O controle é transferido para esta rotina quando a Ronda Principal do Interpretador encontra uma instrução direta, sem número de linha. A rotina padrão ISFLIO é utilizada, primeiro, para determinar se uma declaração "LOAD" está ativa. Se a entrada estiver sendo originada do teclado, o controle será transferido ao ponto de execução da Ronda de Execução(4640H), a fim de executar a instrução. Se a entrada estiver vindo do cassete, o buffer 0 é fechado(6B24H) e um erro de "Instrução direta no arquivo" será gerado(6E71H). Isto poderia ocorrer em uma máquina MSX padrão por meio de um erro de cassete ou pela tentativa de carregar um arquivo de texto sem números de linha.

Endereço . . . . . 6D57H

Esta rotina é utilizada pelos manipuladores das instruções "INPUT", "LINE INPUT" e "PRINT", para verificar a presença de um "#" no texto do programa. Se for encontrado um número de buffer de E/S, ele é avaliado(521BH). FCB localizado e seu endereço colocado em PTRFIL(6AAAH). O byte de modo de FCB é, em seguida, comparado com o número de modo fornecido pelo manipulador de instruções no registrador C, e caso não combinem, será gerado um erro "Número incorreto de arquivo"(6E7DH). Com "PRINT", os modos permitidos são de saída, aleatório e anexação (APPEND). Com "INPUT" e "LINE INPUT", os modos permitidos são entrada e aleatório. Observe que em uma máquina MSX padrão, nem todos estes modos são aceitáveis em níveis inferiores. Algum tipo de erro será gerado posteriormente, para estes modos ilegais.

Endereço . . . . . 6D83H

Esta rotina é utilizada pelo manipulador da instrução "INPUT", para permitir a entrada de uma string do buffer de E/S. Primeiro, um retorno será preparado para o manipulador das instruções "READ/INPUT"(4BF1H). Os caracteres que delimitam a string de entrada, vírgula e espaços para uma Variável numérica e apenas vírgula para uma Variável string, são colocados nos registradores D e E e o controle é transferido à rotina "LINE INPUT"(6DA3H).

Endereço . . . . . 6D8FH

Este é o manipulador da instrução "LINE INPUT" quando a entrada for de um buffer de E/S. O número do buffer é avaliado, FCB localizado e o modo verificado(6D55H). A variável que deve ser referenciada é, em seguida, localizada(5EA4H) e o seu

tipo verificado para garantir o tipo string(3058H). Um retorno será preparado ao manipulador da instrução "LET"(487BH) para executar a atribuição, e a string de entrada será coletada.

Os caracteres entram sequencialmente(6C71H) e são colocados no BUF, até que o delimitador correto seja encontrado, EOF seja alcançado ou o BUF fique lotado (6E41H). Quando a condição terminal é alcançada e a atribuição é para uma Variável numérica, a string é convertida para a forma numérica em DAC(3299H). Quando a atribuição é para uma Variável string, a string é analisada e o descritor do resultado é criado(6638H).

Para "LINE INPUT" todos os caracteres são aceitos, até que um código CR é alcançado. Observe que se este código CR é precedido por um código LF, ele não vai funcionar como um delimitador e será aceito como parte da string. Para o "INPUT" de uma Variável numérica, os espaços anteriores são eliminados e os caracteres seguintes são aceitos até que um código CR, um espaço ou uma vírgula sejam encontrados. Observe que da mesma forma que para "LINE INPUT", um código CR não funcionará como um delimitador se precedido por um LF. Neste caso, entretanto, o código CR não será colocado em BUF, mas ignorado. Para o "INPUT" uma Variável string, os espaços anteriores são eliminados e os caracteres seguintes são aceitos, até que um CR ou vírgula sejam encontrados. Observe que da mesma forma que para "LINE INPUT", um código CR não funcionará como um delimitador quando precedido por um código LF. Neste caso, entretanto, nenhum dos códigos será colocado no BUF; ambos serão ignorados. Um modo alternativo é ligado quando o primeiro caractere lido, após qualquer número de espaços, for um caractere do tipo aspas. Neste caso, todos os caracteres são aceitos, e armazenados em BUF, até que outra aspas seja lida.

Assim que a string de entrada tiver sido aceita, o delimitador terminal será examinado para verificar se alguma ação especial, relativa a caracteres posteriores, será necessária. Caso a string de entrada tenha sido delimitada por aspas ou um espaço, qualquer espaço subsequente será lido e ignorado, até que um caractere não-espaço seja encontrado. Se este caractere for uma vírgula ou um CR, será aceito e ignorado. Caso contrário, uma função de reposição (putback) é despachada(6F8FH) para devolver o caractere ao buffer de E/S. Se a string de entrada for delimitada por um código CR, o próximo caractere será lido e verificado. Se for um código LF, ele será aceito mas ignorado. Se não for um código LF, uma função de reposição será despachada(6F8FH) para devolver o caractere ao buffer de E/S.

Endereço . . . . . 6E6BH

Este é um grupo de dez geradores de erro relativos à E/S de arquivo. O registrador E é carregado com o código de erro relevante e o controle é transferido ao manipulador de erro(406FH):

| <u>ADDR</u> | <u>ERRO</u>                 |
|-------------|-----------------------------|
| 6E6BH       | Nome incorreto de arquivo   |
| 6E6EH       | Arquivo já aberto           |
| 6E71H       | Instrução direta no arquivo |
| E674H       | Arquivo não encontrado      |
| 6E77H       | Arquivo não aberto          |
| 6E7AH       | Extravasamento de campo     |
| 6E7DH       | Número incorreto de arquivo |
| 6E80H       | Erro interno                |
| 6E83H       | Entrada passou o fim        |
| 6E86H       | Somente E/S seqüencial      |

Endereço . . . . . 6E92H

Este é o manipulador da instrução "BSAVE". A especificação do arquivo é analisada(6A0EH) e o endereço inicial avaliado(6F0BH). Em seguida, o endereço de parada é avaliado(6F0BH) e colocado em SAVEND, seguido pelo endereço de ponto de entrada opcional(6F0BH) que é colocado em SAVENT. Se não existir nenhum endereço de entrada, o endereço inicial será tomado em seu lugar. O código do dispositivo é verificado para garantir CAS, e, se não for, será gerado um erro de "Nome incorreto de arquivo"(6E6BH), e os dados escritos no cassete(6FD7H). Observe que nenhum buffer está envolvido, os dados são escritos diretamente no cassete, e nenhuma informação de erro acompanha os dados.

Endereço . . . . . 6EC6H

Este é o manipulador da instrução "BLOAD". A especificação do arquivo é verificada(6A0EH) e RUNBNF feito não-zero, se a opção "R" de autoprocessoamento estiver presente no texto do programa. O deslocamento (Offset) do carregamento opcional, com um valor default de zero, é, em seguida, avaliado(6F0BH) e o código do dispositivo verificado para garantir CAS, e, se não for, será gerado um erro de "Nome incorreto de arquivo"(6E6BH). Em seguida, os dados são lidos diretamente do cassete(7014H), e, da mesma forma que com "BSAVE", nenhum buffer ou verificação de erro será envolvido.

Endereço ..... 6EF4H

O controle será transferido para esta rotina quando o manipulador da instrução "BLOAD" terminar de carregar dados na memória. Se RUNBNF for zero o buffer 0 será fechado(6B24H) e o controle devolvido à Ronda de Execução. Caso contrário, o buffer 0 será fechado(6B24H), um endereço de retorno 6CF3H é estabelecido (esta rotina simplesmente recoloca o apontador do texto do programa no par de registradores HL e volta à Ronda de Execução) e o controle será transferido ao endereço contido em SAVENT.

Endereço ..... 6FOBH

Esta rotina é utilizada pelos manipuladores de "BLOAD" e "BSAVE" para avaliar um operando de endereço, e o resultado é devolvido no par de registradores DE. O operando é avaliado(4C64H) e, em seguida, convertido para um inteiro(5439H).

Endereço ..... 6F15H

Esta rotina é utilizada pelo analisador da especificação do arquivo para fazer a análise sintática do nome de um dispositivo como "CAS:". No início, o par de registradores HL aponta ao início da string de especificação de arquivo e o registrador E contém seu comprimento. Se nenhum nome de dispositivo estiver presente, o código de dispositivo default (CAS=FFH) será devolvido no registrador A com o Indicador Z. Se um nome legal de dispositivo estiver presente, seu código será devolvido no registrador A com o Indicador NZ.

A especificação do arquivo é examinada até que um caractere ":" seja encontrado e o nome é comparado com cada um dos nomes legais de dispositivos, na tabela de dispositivos em 6F76H. Se houver uma concordância, o código do dispositivo será tomado da tabela e devolvido no registrador A. Se nenhuma concordância for, o controle será transferido para a rotina de busca da ROM externa(55F8H). Observe que qualquer caractere minúsculo é convertido em maiúsculo para a comparação. Assim crt ou CRT, por exemplo, indicam o mesmo dispositivo.

Endereço ..... 6F76H

Esta tabela é utilizada pelo analisador sintático do nome do arquivo e contém quatro nomes de dispositivos e códigos, disponíveis em uma máquina MSX padrão:

CAS ... FFH    LPT ... FEH    CRT ... FDH    GRP ... FCH

Endereço ..... 6F87H

Esta tabela é utilizada pelo despachante de função(6F8FH), e contém o endereço das tabelas de decodificação de função para cada um dos quatro dispositivos padrões MSX:

CAS ... 71C7H    LPT ... 72A6H    CRT ... 71A2H    GRP ... 7182H

Endereço ..... 6F8FH

Este é o despachante de funções de E/S de arquivo. Juntamente com a estrutura de buffers do Interpretador, ele fornece um método consistente, independente de dispositivo, de entrada e saída de dados. O código da função requerida é fornecido pelo registrador A e o endereço de FCB do buffer é fornecido no par de registradores HL.

O código do dispositivo é tomado do byte 4 de FCB e examinado, a fim de determinar se é um dos quatro dispositivos padrão. Em caso negativo, o controle será transferido para o despachante de funções na ROM externa(564AH). Caso contrário, o endereço da tabela de decodificação do dispositivo será tomado da tabela em 6F87H, o endereço da função requerida tomado dela e o controle transferido ao manipulador de função relevante.

Endereço ..... 6FB7H

Este é o manipulador da declaração "CSAVE". O nome do arquivo é avaliado(7098H), seguido pelo operando de frequência baud opcional(7A2DH). Em seguida, o bloco de identificação é escrito no cassete(7125H) com um byte de tipo de arquivo igual a D3H. O conteúdo da Área de Texto do Programa é escrito diretamente no cassete, em um único bloco de dados(713EH). Observe que nenhuma informação de verificação de erro acompanha os dados.

Endereço ..... 6FD7H

O controle é transferido para esta rotina, do manipulador da instrução "BSAVE", para escrever um bloco de memória no cassete. Primeiro, o bloco de identificação é escrito no cassete(7125H) com um byte de tipo-de-arquivo igual a D0H. Em seguida, o motor é ligado e um pequeno cabeçalho escrito no cassete(72F8H). O endereço inicial é retirado da pilha do Z80 e escrito no cassete com LSB primeiro, MSB, em seguida(7003H). O endereço de parada é tomado de SAVEND e escrito no cassete com LSB primeiro, MSB, em seguida(7003H). O elemento do ponto de entrada é tomado



de SAVENT e escrito no cassete com LSB primeiro, e MSB em seguida(7003H). A área requerida da memória é escrita no cassete, um byte por vez(72DEH), e o motor do cassete desligado por meio da rotina padrão TAPOOF. Observe que nenhuma informação de verificação de erro acompanha estes dados.

Endereço ..... 7003H

Esta rotina escreve o conteúdo do par de registradores HL no cassete, com o registrador L em primeiro lugar(72DEH) e o registrador H em segundo lugar(72DEH).

Endereço ..... 700BH

Esta rotina lê dois bytes do cassete e coloca o primeiro no registrador L(72D4H), o segundo no registrador H(72D4H).

Endereço ..... 7014H

O controle é transferido para esta rotina do manipulador da instrução "BLOAD", para carregar dados do cassete na memória. O cassete é lido, até que um bloco de identificação com um tipo de arquivo D0H e o nome de arquivo correto sejam encontrados(70B8H). O cabeçalho do bloco de dados, em seguida, é localizado no cassete(72E9H). O valor do deslocamento do endereço é retirado da pilha do Z80 e somado ao endereço inicial lido do cassete(700BH). O endereço de parada é lido do cassete(700BH) e o deslocamento acrescentado também a este. O endereço do ponto entrada é lido do cassete(700BH) e colocado em SAVENT se um autoprocessamento for requerido. Bytes de dados sucessivos são em seguida, lidos do cassete(72D4H) e colocados na memória, inicialmente no endereço de partida, até que o endereço de parada seja atingido. Finalmente, o motor é desligado por meio da rotina padrão TAPIOF e o controle transferido para o ponto de terminação BLOAD(6EF4H).

Endereço ..... 703FH

Esté é o manipulador das instruções "CLOAD" e "CLOAD?". Primeiro o texto do programa é verificado quanto a um "PRINT", na parte final(91H) que é como o caractere "?" é atomizado. Em seguida, o nome do arquivo é avaliado(708CH) e o cassete é lido, até que um bloco de identificação com um tipo de arquivo D3H e o nome de arquivo correto sejam encontrados(70B8H). Para "CLOAD" uma operação "NEW" é executada, em seguida(6287H) para cancelar o texto do programa atual. Para "CLOAD?"

todos os apontadores na Área de Texto do Programa são convertidos em números de linha(54EAH) para combinar com os dados do cassete.

O cabeçalho do bloco de dados é localizado no cassete e bytes de dados sucessivos são lidos do cassete e colocados na memória ou comparados com o conteúdo da memória atual(715DH). Quando o bloco de dados for completamente lido, a mensagem "OK" será apresentada(6678H) e o controle será transferido diretamente ao final da Ronda Principal do Interpretador(4237H) para repor os apontadores de armazenamento de Variáveis. Para "CLOAD?" a leitura do bloco de dados terminará, se o byte do cassete não for o mesmo que o byte do texto do programa na memória. Se o endereço onde isto ocorreu estiver acima do fim da Área de Texto do Programa, o manipulador terminará com uma mensagem "OK", como antes. Caso contrário, será gerado um "Erro de verificação".

Endereço . . . . . 708CH

Esta rotina é utilizada pelos manipuladores das instruções "CLOAD" e "CSAVE" para avaliar um nome de arquivo no texto do programa. Os dois manipuladores utilizam pontos de entrada diferentes, de modo que um nome de arquivo nulo será permitido para "CLOAD", mas não para "CSAVE". O string do nome do arquivo é avaliado(4C64H), seu armazenamento liberado(680FH) e os seis primeiros caracteres copiados em FILNAM. Se o nome do arquivo for maior do que seis caracteres, o excesso será ignorado. Se o nome do arquivo for menor do que seis caracteres, FILNAM será preenchido com espaços.

Endereço . . . . . 70B8H

Esta rotina é utilizada pelos manipuladores das instruções "CLOAD" e "BLOAD" e para a função de abertura (OPEN) do despachante (quando o dispositivo é CAS e o modo é entrada) para identificar um bloco de identificação no cassete. No início, o nome do arquivo está em FILNAM e o tipo do arquivo no registrador C, D3H para um arquivo BASIC (CLOAD) atomizado, D0H para um arquivo binário (BLOAD) e EAH para um arquivo ASCII (LOAD ou de dados).

O motor do cassete é ligado e o cassete é lido, até que um cabeçalho seja encontrado(72E9H). Cada bloco de identificação é prefixado por dez caracteres de tipo de arquivo, de modo que caracteres sucessivos são lidos do cassete(72D4H) e comparados ao tipo-de-arquivo requerido. Se os caracteres de tipo de arquivo não concordarem, o controle será devolvido ao início da rotina, a fim de encontrar o próximo cabeçalho. Caso contrário, os próximos seis caracteres serão lidos(72D4H) e colocados em FILNM2. Se

FILNAM estiver cheio de espaços, nenhuma comparação de nome de arquivo será realizada e o bloco de identificação é considerado encontrado. Caso contrário, o conteúdo de FILNAM e FILNM2 são comparados para determinar se este é o arquivo requerido. Se a concordância não for bem sucedida e o Interpretador estiver no modo direto, a mensagem "PULEI" ("Skip:") é apresentada(710DH) seguida do nome do arquivo. Em seguida, o controle é transferido de volta ao início da rotina para tentar o próximo cabeçalho. Se a concordância for bem sucedida, e o Interpretador estiver no modo direto, a mensagem "ACHEI" ("Found:") é apresentada(710DH) seguida do nome do arquivo e a rotina terminará.

Endereço . . . . . 70FFH

Esta é a mensagem "Found:", terminada por um byte zero.

Endereço . . . . . 7106H

Esta é a mensagem "Skip:", terminada por um byte zero.

Endereço . . . . . 710DH

A não ser que CURLIN mostre que o Interpretador está no modo de programa, esta rotina apresenta (6678H) primeiro a mensagem cujo endereço é fornecido pelo par de registradores HL, seguido pelos seis caracteres contidos em FILNM2.

Endereço . . . . . 7125H

Esta rotina é utilizada pelos manipuladores das instruções "CSAVE" e "BSAVE" e para a função de abertura (Open) do despachante (quando o dispositivo é CAS e o modo é de saída) escrever um bloco de identificação no cassete. No início, o nome do arquivo está em FILNAM e o tipo do arquivo no registrador A, D3H para um arquivo BASIC (CSAVE) atomizado, D0H para um arquivo binário (BSAVE) e EAH para um arquivo ASCII (SAVE ou dados). O motor do cassete é ligado e um cabeçalho grande é escrito no cassete(72F8H). O byte do tipo de arquivo, em seguida, é escrito no cassete(72DEH) dez vezes, seguido pelos primeiros seis caracteres de FILNAM(72DEH). O motor do cassete é desligado por meio da rotina padrão TAPOOF e a rotina termina.

Endereço . . . . . 713EH

Esta rotina é utilizada pelo manipulador da instrução "CSAVE" para escrever a

Área de Texto do Programa no cassete como um único bloco de dados. Todos os apontadores no texto do programa são reconvertidos em números de linha(54EAH), para tornar o endereço do texto independente. O motor do cassete é ligado e um pequeno cabeçalho será escrito no cassete(72F8H). Toda a Área de Texto do Programa é, em seguida, escrita no cassete um byte por vez(72DEH) e seguida por sete bytes zero(72DEH) como terminador. Em seguida, o motor do cassete é desligado por meio da rotina padrão TAPOOF e a rotina termina.

Endereço . . . . . 715DH

Esta rotina é utilizada pelos manipuladores das instruções "CLOAD" e "CLOAD?" para ler um único bloco de dados para a Área de Texto do Programa ou para compará-lo com o conteúdo atual. No início, o registrador A contém um indicador para distinguir entre as duas declarações, 00H para "CLOAD" e FFH para "CLOAD?". O motor do cassete é ligado e o primeiro cabeçalho localizado(72E9H). Caracteres sucessivos são lidos do cassete(72D4H) e colocados na Área de Texto do Programa ou comparados com o conteúdo atual. Caso a instrução atual seja "CLOAD?" a rotina terminará com o indicador NZ, se o caractere do cassete não for igual ao caractere da memória. Caso contrário, os dados serão lidos até que dez zeros sucessivos sejam encontrados. Esta seqüência de zeros é formada pelo caractere de fim de linha da última linha do programa, o elo final e os sete zeros terminais acrescentados por "CSAVE". Observe que a rotina, provavelmente, terminará durante esta seqüência, ao ser utilizada por "CLOAD?", uma vez que a comparação de memória continua sendo feita. Isto explica a codificação, de certa forma, peculiar das condições de término do manipulador "CLOAD?".

Endereço . . . . . 7182H

Esta tabela é utilizada pelo despachante ao decodificar códigos de função para o dispositivo GRP. Contém o endereço do manipulador para cada um dos códigos de função, a maioria, de fato, geradores de erro:

| <b>PARA</b> | <b>FUNÇÃO</b>         |
|-------------|-----------------------|
| 71B6H       | 0, abrir              |
| 71C2H       | 2, fechar             |
| 6E86H       | 4, aleatório          |
| 7196H       | 6, saída seqüencial   |
| 475AH       | 8, entrada seqüencial |
| 475AH       | 10, loc               |

| <u>PARA</u> | <u>FUNÇÃO</u> |
|-------------|---------------|
| 475AH       | 12, lof       |
| 475AH       | 14, eof       |
| 475AH       | 16, fpos      |
| 475AH       | 18, devolver  |

Endereço . . . . . 7196H

Esta é a rotina de saída sequencial do despachante para o dispositivo GRP. Primeiro, SCRMOD é verificado e é gerado um erro de “Chamada ilegal de uma função” (475AH), se a tela estiver no modo texto. O caractere da saída é tomado do registrador C e o controle é transferido para a rotina padrão GRPPRT.

Endereço . . . . . 71A2H

Esta tabela é utilizada pelo despachante ao decodificar códigos de função para o dispositivo CRT. Contém o endereço do manipulador para cada um dos códigos de função, a maioria, de fato, geradores de erro:

| <u>PARA</u> | <u>FUNÇÃO</u>         |
|-------------|-----------------------|
| 71B6H       | 0, abrir              |
| 71C2H       | 2, fechar             |
| 6E86H       | 4, aleatório          |
| 71C3H       | 6, saída sequencial   |
| 475AH       | 8, entrada sequencial |
| 475AH       | 10, loc               |
| 475AH       | 12, lof               |
| 475AH       | 14, eof               |
| 475AH       | 16, fpos              |
| 475AH       | 18, devolver          |

Endereço . . . . . 71B6H

Esta é a rotina de abertura (Open) do despachante para os dispositivos CRT, LPT e GRP. O modo requerido, no registrador E, é verificado e gerado um erro de “Nome incorreto de arquivo”(6E6BH) para entrada ou anexação. O endereço de FCB é, em seguida, colocado em PTRFIL, o modo no byte 0 de FCB e a rotina termina. Observe que a instrução RET do Z80, no final desta rotina (71C2H), é a rotina de fechamento (Close) do despachante para os dispositivos CRT, LPT e GRP.

Endereço . . . . . 71C3H

Esta é a rotina de saída seqüencial do despachante para o dispositivo CRT. O caractere para a saída é tomado do registrador C e o controle é transferido para a rotina padrão CHPUT.

Endereço . . . . . 71C7H

Esta tabela é utilizada pelo despachante ao decodificar códigos de função para o dispositivo CAS. Contém o endereço do manipulador para cada um dos códigos de função, grande parte sendo geradores de erro:

| PARA  | FUNÇÃO                |
|-------|-----------------------|
| 71DBH | 0, abrir              |
| 7205H | 2, fechar             |
| 6E86H | 4, aleatório          |
| 722AH | 6, saída seqüencial   |
| 723AH | 8, entrada seqüencial |
| 475AH | 10, loc               |
| 475AH | 12, lof               |
| 726DH | 14, eof               |
| 475AH | 16, fpos              |
| 727AH | 18, devolver          |

Endereço . . . . . 71DBH

Esta é a rotina de abertura do despachante para o dispositivo CAS. A posição no buffer de E/S atual, contida no byte 6 do FCB, e CASPRV, que contém qualquer caractere de devolução, são ambos zerados. O modo requerido, fornecido no registrador E, é examinado e é gerado um erro de "Nome incorreto de arquivo"(6E6BH) para os modos aleatório e anexação. Para o modo de saída o bloco de identificação é, então, escrito no cassete(7125H), enquanto que para o modo de entrada o bloco de identificação correto é localizado no cassete(70B8H). Em seguida, o endereço de FCB é colocado em PTRFIL, o modo no byte 0 de FCB e a rotina termina.

Endereço . . . . . 7205H

Esta é a rotina de fechamento, do despachante, para o dispositivo CAS. O byte 0 do FCB é examinado e, se o modo for "entrada", CASPRV é zerado e a rotina termina. Caso contrário, o restante do buffer de E/S é preenchido com caracteres de fim de

arquivo(1AH) e o conteúdo do buffer de E/S escrito no cassete(722FH). Em seguida, CASPRV é zerado e a rotina termina.

Endereço . . . . . 722AH

Esta é a rotina de saída sequencial, do despachante, para o dispositivo CAS. O caractere para a saída é tomado do registrador C e colocado na próxima posição livre do buffer de E/S(728BH). O byte 6 de FCB, a posição no buffer de E/S, é, em seguida, incrementada. Se a posição do buffer de E/S voltar para zero, isto significa que há 256 caracteres no buffer de E/S e ele tem que ser escrito no cassete. O motor do cassete é ligado, um pequeno cabeçalho é escrito no cassete(72F8H), seguido pelo conteúdo do buffer de E/S(72DEH), e o motor é desligado por meio da rotina padrão TAPOOF.

Endereço . . . . . 723FH

Esta é a rotina de entrada sequencial, do despachante, para o dispositivo CAS. Em primeiro lugar, CASPRV é verificado(72BEH) para determinar se ele contém um caractere devolvido, quando seu conteúdo será não-zero. Neste caso, a rotina termina com o caractere no registrador A. Caso contrário, a posição no buffer de E/S é verificada (729BH) para determinar se contém algum caractere. Se o buffer de E/S estiver vazio, o motor do cassete é ligado e o cabeçalho é localizado(72E9H). Em seguida, são lidos 256 caracteres(72D4H), o motor do cassete é desligado por meio da rotina padrão TAPION e a posição do buffer de E/S reposta em zero. Em seguida, o caractere é tomado atual da posição do buffer de E/S e a posição incrementada. Finalmente, o caractere é verificado para sabermos se é o caractere de fim de arquivo(1AH). Se não for, a rotina terminará com o caractere no registrador A e Indicador NC. Caso contrário, o caractere de fim de arquivo será colocado em CASPRV, de modo que solicitações posteriores de entradas sequenciais sempre voltarão para a condição de fim de arquivo, e a rotina terminará com o Indicador C.

Endereço . . . . . 726DH

Esta é a rotina eof, do despachante, para o dispositivo CAS. O próximo caractere é lido(723FH) e colocado em CASPRV. Em seguida, é testado quanto ao código de fim de arquivo(1AH) e o resultado colocado em DAC como um inteiro, zero para falso, e FFFFH para verdadeiro.

Endereço . . . . . 727CH

Esta é a rotina de devolução, do despachante, para o dispositivo CAS. O caractere é, simplesmente, colocado em CASPRV para ser tomado na próxima solicitação de entrada seqüencial.

Endereço . . . . . 7281H

Esta rotina é utilizada pela função de fechamento, do despachante, para verificar se há algum caractere no buffer de E/S e depois zerar o byte de posição do buffer de E/S no FCB.

Endereço . . . . . 728BH

Esta rotina é utilizada pela função de saída seqüencial do despachante, para colocar o caractere contido no registrador A no buffer de E/S, na posição atual, que será, em seguida, incrementada.

Endereço . . . . . 729BH

Esta rotina é utilizada pela função de entrada seqüencial do despachante, para coletar o caractere na posição atual do buffer de E/S, que é, em seguida, incrementada.

Endereço . . . . . 72A6H

Esta tabela é utilizada pelo despachante ao decodificar códigos de função para o dispositivo LPT. Contém o endereço do manipulador para cada um dos códigos de função, a maioria, de fato, geradores de erro:

| <b>PARA</b> | <b>FUNÇÃO</b>         |
|-------------|-----------------------|
| 71B6H       | 0, abrir              |
| 71C2H       | 2, fechar             |
| 6E86H       | 4, aleatório          |
| 72BAH       | 6, saída seqüencial   |
| 475AH       | 8, entrada seqüencial |
| 475AH       | 10, loc               |
| 475AH       | 12, lof               |
| 475AH       | 14, eof               |
| 475AH       | 16, fpos              |
| 475AH       | 18, devolver          |



Endereço ..... 72BAH

Esta é a rotina de saída seqüencial, do despachante, para o dispositivo LPT. O caractere a ser enviado é tomado do registrador C e o controle é transferido para a rotina padrão OUTDLP.

Endereço ..... 72BEH

Esta rotina é utilizada pela função de entrada seqüencial do despachante para verificar se existe um caractere devolvido em CASPRV, e, em caso negativo, devolver o Indicador Z. Caso contrário, CASPRV é zerado e o caractere testado para ver se é o caractere de fim de arquivo(1AH). Em caso negativo, ele volta com o caractere no registrador A e Indicador NZ, NC. Caso contrário, o caractere de fim de arquivo é colocado de volta em CASPRV e a rotina volta com o Indicador Z, C.

Endereço ..... 72CDH

Esta rotina é utilizada por diversas funções do despachante para verificar se o modo no Registrador E é anexação. Em caso afirmativo, será gerado um erro "Nome incorreto de arquivo"(6E6BH).

Endereço ..... 72D4H

Esta rotina é utilizada por diversas funções do despachante, para ler um caractere do cassete. O caractere é lido, por meio da rotina padrão TAPIN, e será gerado um "Erro de E/S de dispositivo", se o Indicador C for devolvido(73B2H).

Endereço ..... 72DEH

Esta rotina é utilizada por diversas funções do despachante para escrever um caractere no cassete. O caractere será escrito por meio da rotina padrão TAPOUT e um "Erro de E/S de dispositivo" será gerado(73B2H), se o Indicador C for devolvido.

Endereço ..... 72E9H

Esta rotina é utilizada por diversas funções do despachante para ligar o motor do cassete para entrada. O motor é ligado por meio da rotina padrão TAPION e o "Erro de E/S de dispositivo" será gerado(73B2H), se o Indicador C for devolvido.

Endereço . . . . . 72F8H

Esta rotina é utilizada por diversas funções do despachante para ligar o motor do cassete para saída, e o controle será simplesmente, transferido para a rotina padrão do TAPOON.

Endereço . . . . . 7304H

Esta rotina é utilizada pelo ponto "OK" da Ronda Principal do Interpretador, pelo manipulador da instrução "END" e pela rotina RUN-CLEAR, para desligar a impressora. Primeiro, PRTFLG é zerado e LPTPOS testado para ver se algum caractere saiu mas ficou preso no buffer de linha de impressora. Em caso afirmativo, a sequência CR, LF é acionada para mover a impressora e LPTPOS é zerado.

Endereço . . . . . 7323H

Esta rotina envia uma sequência CR, LF ao dispositivo de saída atual, por meio da rotina padrão OUTDO. LPTPOS ou TTYPOS será zerado, dependendo se a impressora ou a tela estiver ativa.

Endereço . . . . . 7347H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "INKEY\$". O estado do buffer do teclado é examinado, por meio da rotina padrão CHSNS. Se o buffer estiver vazio, o endereço de um descritor de string nulo sem significado é devolvido em DAC. Caso contrário, o próximo caractere será lido do buffer do teclado, por meio da rotina padrão CHGET. Depois de verificar se há espaço suficiente disponível(6625H) o caractere será copiado na Área de Armazenamento de Strings e será criado o descritor do resultado(6821H).

Endereço . . . . . 7367H

Esta rotina é utilizada pelo manipulador da instrução "LIST" para permitir a saída de um caractere ao dispositivo de saída atual, por meio da rotina padrão OUTDO. Se o caractere for um código LF, será enviado também um CR.

Endereço . . . . . 7374H

Esta rotina é utilizada pela Ronda Principal do Interpretador para coletar uma linha de texto, quando a entrada vem de um buffer de entrada/saída, em vez do teclado, isto é, quando uma instrução "LOAD" está ativa. A entrada dos caracteres é seqüencial (6C71H) e eles são colocados em BUF até que este fique lotado, um CR seja detectado ou o fim de arquivo seja atingido. Todos os caracteres são aceitos, a menos dos códigos LF que são filtrados. Caso BUF fique lotado ou um CR seja detectado, a rotina simplesmente, devolve a linha à Ronda Principal. Se o fim de arquivo for atingido, enquanto alguns caracteres estão em BUF, a linha será devolvida ao Laço-principal. Quando o fim de arquivo for atingido, sem nenhum caractere em BUF, então o buffer 0 de E/S será fechado(6D7BH) e FILNAM verificado, para determinar se um autoprocessoamento é requerido. Em caso negativo, o controle volta ao ponto "OK" do Interpretador(411EH). Caso contrário, o sistema é limpo(629AH) e o controle é transferido à Ronda de Execução(4601H) para executar o programa.

Endereço . . . . . 73B2H

Este é o gerador do "Erro de E/S do dispositivo".

Endereço . . . . . 73B7H

Este é o manipulador da instrução "MOTOR". Se nenhum operando estiver presente, o controle será transferido para a rotina padrão STMOTR com FFH no registrador A. Caso siga o átomo "OFF"(EBH) o controle será transferido com 00H no registrador A. Caso siga o átomo "ON"(95H), o controle será transferido com 01H no registrador A.

Endereço . . . . . 73CAH

Este é o manipulador da instrução "SOUND". O operando numérico do registrador, que deverá ser menor do que quatorze, será avaliado(521CH) e colocado no registrador A. O operando de dado será avaliado(521CH), o bit 7 ligado e o bit 6 desligado, para evitar a alteração dos modos da porta de E/S auxiliar do PSG. O operando do dado será colocado no registrador E e o controle será transferido para a rotina padrão WRTPSG.

Endereço . . . . . 73E4H

Este é um simples espaço ASCII utilizado pelo manipulador da instrução “PLAY”, para substituir um operando string nulo por uma string igual a um espaço em branco.

Endereço . . . . . 73E5H

Este é o manipulador da instrução “PLAY”. O endereço da tabela de comando “PLAY” em 752EH é colocado em MCLTAB para o analisador sintático de linguagem macro e PRSCNT será zerado. O primeiro operando string, que é obrigatório, será avaliado(4C64H), seu armazenamento liberado(67DOH) com seu comprimento e endereço colocados nos bytes 2, 3 e 4 de VCBA. O apontador da pilha do canal é inicializado com VCBA + 33 e colocado em VCBA nos bytes 5 e 6. Se existir texto adicional presente na declaração, este processo será repetido para as vozes B e C, até que um máximo de três operandos tenha sido avaliado, após o que é gerado um “Erro de sintaxe”(4055H). Se houver menos do que três operandos string presentes, uma marca de fim de fila(FFH) será colocada na fila(7507H) para cada voz não utilizada. Em seguida, o registrador A é zerado, para selecionar a voz A, e o controle irá para a Ronda Principal do PLAY.

Endereço . . . . . 744DH

Esta é a Ronda Principal do PLAY. O número de bytes livres na fila atual é verificado(7521H) e, se sobra menos do que oito bytes, a próxima voz será selecionada (74D6H) para se evitar esperar que a fila fique vazia. Em seguida, o comprimento restante da string do operando é tomado do buffer de voz atual e, caso sobre zero byte para ser analisado, o laço pula novamente para a próxima voz(74D6H). Caso contrário, o comprimento da string atual e o endereço são tomados do buffer de voz e colocados em MCLLEN e MCLPTR para o analisador sintático de linguagem macro. O antigo conteúdo da pilha é copiado do buffer de voz para a pilha de Z80(6253H), MCLFLG é feito não-zero e o controle transferido ao analisador sintático de linguagem macro(56A2H).

O analisador sintático de linguagem macro, normalmente, varre a string, utilizando os manipuladores de comando da declaração “PLAY”, até que a string seja esgotada. Entretanto, se uma fila musical é preenchida durante a geração de uma nota, uma conclusão anormal será forçada sobre o Laço-principal de música(748EH), de modo que a próxima voz possa ser processada sem precisar esperar que a fila se esgote. Quando o controle volta normalmente, será colocada uma marca de fim de fila na fila atual(7507H) e PRSCNT incrementado para mostrar o número de strings completados. Se o controle

voltar de forma anormal, então, qualquer coisa deixada na pilha do Z80 é copiada no buffer de voz atual(6253H). Devido a natureza recursiva no analisador sintático de linguagem macro, onde um comando "X" está envolvido, poderá haver um número de descritores de string de quatro bytes, marcando o ponto em que a string original foi suspensa. Os descritores são deixados na pilha de Z80 na terminação. Salvar o conteúdo da pilha do buffer de voz significa que poderão ser restaurados quando o laço voltar para aquela voz, novamente. Observe que como há apenas dezesseis bytes disponíveis em cada buffer de voz, será gerado um erro de "Chamada ilegal de uma função", (475AH) caso muitos dados permaneçam na pilha. Isto ocorrerá quando uma fila ficar lotada e existirem múltiplos comandos "X" aninhados, por exemplo:

```
10 A$="XB$;"
20 B$="XC$;"
30 C$="XD$;"
40 D$=STRING$(150,"A")
50 PLAY A$
```

Parece haver um problema nesta seção, uma vez que são permitidos apenas quinze bytes de dados empilhados, em vez de dezesseis, antes que um erro seja gerado.

Quando o controle volta do analisador sintático, o registrador A é incrementado para selecionar a próxima voz a ser processada. Quando todas as três vozes forem processadas INTFLG será verificado e, se CTRL-STOP for detectado pelo manipulador de interrupção, o controle será transferido para a rotina padrão GICINI para interromper toda música e terminar. Presumindo que o bit 7 de PRSCNT indique uma primeira passagem pela Ronda, isto é, que nenhuma voz tenha sido temporariamente suspensa, devido a uma fila lotada, PLYCNT é incrementado e será iniciada a interrupção do cancelamento da fila pela rotina padrão STRTMS. Se todas as três strings de operando forem completadas, o manipulador termina; caso contrário, o controle é transferido de volta ao início da Ronda Principal do PLAY para tentar novamente cada uma das vozes.

Endereço ..... 7507H

Esta rotina é utilizada pelo manipulador da instrução "PLAY" para colocar uma marcação de fim de fila(FFH) na fila atual, por meio da rotina padrão PUTQ. Se a fila estiver cheia, ele aguarda até que haja espaço disponível.

Endereço ..... 7521H

Esta rotina é utilizada pelo manipulador da instrução "PLAY" para verificar

quanto espaço sobra na fila atual, por meio da rotina padrão LFTQ. Caso permaneçam menos que oito bytes (o maior pacote de dados musicais possíveis é sete bytes de comprimento) o Indicador C é devolvido.

Endereço ..... 752EH

Esta tabela contém as letras de comando válidas e seus endereços associados para os comandos da instrução "PLAY". Aqueles comandos que levam um parâmetro, e conseqüentemente têm o bit 7 ligado, são indicados, na tabela, com um asterisco:

COMANDO    ENDEREÇO

|    |       |
|----|-------|
| A  | 763EH |
| B  | 763EH |
| C  | 763EH |
| D  | 763EH |
| E  | 763EH |
| F  | 763EH |
| G  | 763EH |
| M* | 759EH |
| V* | 7586H |
| S* | 75BEH |
| N* | 7621H |
| O* | 75EFH |
| R* | 75FCH |
| T* | 75E2H |
| L* | 75C8H |
| X  | 5782H |

Endereço ..... 755FH

Esta tabela é utilizada pelo manipulador dos comandos "A" até "G" da instrução "PLAY", para traduzir o número de uma nota de zero a quatorze, em um deslocamento (Offset) na tabela de divisores de tom em 756EH. A nota em si, em vez do número da nota, é indicada abaixo com cada valor de deslocamento:

|        |          |
|--------|----------|
| 16 ... | A-       |
| 18 ... | A        |
| 20 ... | A+ or B- |
| 22 ... | B or C-  |
| 00 ... | B+       |
| 00 ... | C        |
| 02 ... | C+ or D- |
| 04 ... | D        |
| 06 ... | D+ or E- |
| 08 ... | E or F-  |
| 10 ... | E+       |
| 10 ... | F        |
| 12 ... | F+ or G- |
| 14 ... | G        |
| 16 ... | G+       |

Endereço ..... 756EH

Esta tabela contém as doze constantes do divisor do PSG necessárias para produzir os tons da oitava um. Para cada constante, a nota e frequência correspondentes são mostrados:

|      |     |    |          |
|------|-----|----|----------|
| 3421 | ... | C  | 32.698Hz |
| 3228 | ... | C+ | 34.653Hz |
| 3047 | ... | D  | 36.712Hz |
| 2876 | ... | D+ | 38.895Hz |
| 2715 | ... | E  | 41.201Hz |
| 2562 | ... | F  | 43.662Hz |
| 2419 | ... | F+ | 46.243Hz |
| 2283 | ... | G  | 48.997Hz |
| 2155 | ... | G+ | 51.908Hz |
| 2034 | ... | A  | 54.995Hz |
| 1920 | ... | A+ | 58.261Hz |
| 1812 | ... | B  | 61.773Hz |

Endereço ..... 7586H

Este é o manipulador do comando "V" da declaração "PLAY". O parâmetro, com um valor default de oito, é colocado no byte 18 do buffer de voz atual, sem alterar o bit 6 do conteúdo atual. Nenhum dado musical será gerado.

Endereço ..... 759EH

Este é o manipulador do comando "M" da declaração "PLAY". O parâmetro, com um valor default de 255, é comparado com o período de modulação existente, contido nos bytes 19 e 20 do buffer de voz atual. Se forem os mesmos, a rotina termina sem ação. Caso contrário, o novo período de modulação será colocado no buffer de voz e o bit 6 será colocado no byte 18 do buffer de voz, para indicar que o novo valor terá que ser incorporado no próximo pacote de dados musicais produzido. Nenhum dado musical será gerado.

Endereço ..... 75BEH

Este é o manipulador do comando "S" da declaração "PLAY". O parâmetro é colocado no byte 18 do buffer de voz atual e o bit 4 do mesmo byte é ligado, para indicar que o novo valor terá que ser incorporado no próximo pacote de dados musicais produzido. Nenhum dado musical é gerado. Devido às características do PSG, os parâmetros de forma e volume são mutuamente exclusivos, de modo que o mesmo byte dos buffers de voz será utilizado para os dois.

Endereço . . . . . 75C8H

Este é o manipulador do comando "L" da instrução "PLAY". O parâmetro, com um valor default de quatro, é colocado no byte 16 do buffer de voz atual e será utilizado na computação das durações das próximas notas. Nenhum dado musical será gerado.

Endereço . . . . . 75E2H

Este é o manipulador do comando "T" da declaração "PLAY". O parâmetro, com um valor default de 120, é colocado no byte 17 do buffer de voz atual e será utilizado na computação das durações das próximas notas. Nenhum dado musical será gerado.

Endereço . . . . . 75EFH

Este é o manipulador do comando "O" da instrução "PLAY". O parâmetro, com o valor default de quatro, é colocado no byte 15 do buffer de voz atual onde será utilizado na computação das frequências das próximas notas. Nenhum dado musical será gerado.

Endereço . . . . . 75FCH

Este é o manipulador do comando "R" da instrução "PLAY". O parâmetro de comprimento, com um valor default de quatro, é deixado no par de registradores DE e um valor de divisor de tom zero é colocado no par de registradores HL. O valor do volume existente é tomado do byte 18 do buffer de voz atual, temporariamente substituído por um valor zero e o controle será transferido ao gerador de notas(769CH).

Endereço . . . . . 7621H

Este é o manipulador do comando "N" da instrução "PLAY". Primeiro, o parâmetro obrigatório é examinado, e se for zero será gerada uma pausa(760BH). Se for maior do que 96, será gerado um erro de "Chamada de função ilegal"(475AH). Caso contrário, a constante doze é subtraída repetidamente do número da nota até que haja sub-extravasamento, para obtenção de um número de oitava de um a nove no registrador E e um número de nota de zero a onze no registrador C. O controle será, em seguida, transferido ao gerador de notas(7673H).



Endereço . . . . . 763EH

Este é o manipulador dos comandos de “A” à “G”, da instrução “PLAY”. Primeiro, a letra da nota é convertida em um número de nota de zero a quatorze, sendo esta faixa de extensão necessária devido a redundância implícita na notação. A tabela em 755FH é, então, utilizada para a obtenção do deslocamento na tabela de divisão de tom e a constante de divisão para a nota colocada no par de registradores DE. O valor da oitava é tomado do byte 15 do buffer de voz atual e a constante do divisor é dividido por dois, até que a oitava correta seja atingida. Em seguida, o operando string é examinado diretamente(56EEH) para determinar se existe um parâmetro de comprimento de nota posterior. Em caso afirmativo, ele é convertido(572FH) e colocado no registrador C. Se não existir nenhum parâmetro, o comprimento default é tomado do byte 16 do buffer de voz atual. A duração da nota é, então, computada da seguinte forma:

$$\text{Duração (tiques de interrupção)} = 12000/(\text{COMPR} * \text{TEMPO})$$

Com o valor de comprimento normal (4) e valor de tempo (120) isto fornece uma duração para a nota de 25 tiques de interrupção de 20mS cada ou 0.5 segundos. Em seguida, o operando string é examinado(56EEH) para checar os caracteres “.” posteriores e, para cada um, a duração é multiplicada por um e meio. Finalmente, a duração resultante é verificada e, se for menor do que cinco tiques de interrupção, será substituída por um valor cinco. Assim, a nota mais curta que poderá ser gerada, em uma máquina inglesa, é 0.10 segundos, seja qual for o tempo ou comprimento da nota.

O pacote de dados musicais, que terá três, cinco ou sete bytes de comprimento, será, em seguida, montado nos bytes 8 a 14 do buffer de voz atual, antes de ser colocado na fila. A duração é colocada nos bytes 8 e 9 do buffer de voz. O byte de volume é tomado do byte 18 e colocado no byte 10 do buffer de voz, tendo o bit 7 ligado para indicar uma alteração de volume para a rotina de cancelamento de fila por interrupção. Se o bit 6 do byte de volume for ativado, então, o período de modulação é tomado dos bytes 19 e 20 e acrescentado ao pacote de dados, nos bytes 11 e 12. Se o valor do divisor de tom for não-zero, então, será acrescentado ao pacote de dados nos bytes 11 e 12 (sem período de modulação) ou bytes 13 e 14 (com período de modulação). Finalmente, a contagem de bytes é mascarada nos três bits superiores do byte 8 do buffer de voz, para completar a preparação do pacote de dados musicais.

Se o valor do divisor de tom for zero, indicando uma pausa, o conteúdo de SAVVOL é restaurado ao byte 18 do buffer estático. O pacote de dados musicais será, em seguida, colocado na fila atual, por meio da rotina padrão PUTQ e o número de bytes

livres remanescentes verificados(7521H). Caso sobrem menos do que oito bytes, o controle será transferido diretamente ao manipulador da instrução "PLAY"(748EH); caso contrário, o controle retorna normalmente ao analisador sintático de linguagem macro.

Endereço . . . . . 7754H .

Esta é a constante de simples precisão igual a 12000, utilizada na computação da duração de uma nota.

Endereço . . . . . 7758H

Este é o manipulador da instrução "PUT". O registrador B é colocado em 80H e o controle vai para o manipulador da instrução "GET".

Endereço . . . . . 775BH

Este é o manipulador da instrução "GET". O registrador B é zerado, para distinguir "GET" de "PUT", e o próximo átomo do programa será examinado. Em seguida, o controle é transferido para o manipulador da instrução "PUT SPRITE" (7AAFH), ou para o manipulador da instrução "GET/PUT" do BASIC de Disco(6C35H).

Endereço . . . . . 7766H

Este é o manipulador da instrução "LOCATE". Se uma coordenada de coluna estiver presente, ela será avaliada(521CH) e colocada no registrador D; caso contrário, a coluna atual é tomada de CSRS. Se uma coordenada de linha estiver presente, ela será avaliada(521CH) e colocada no registrador E; caso contrário, a linha atual é tomada de CSRY. Se existir um operando de comutação do cursor ele será avaliado(521CH) e o registrador A carregado com 78H para um operando zero (OFF) e 79H para um operando não-zero (ON). O cursor será, então, mudado enviando "ESC", 78H/79H, "5", por meio da rotina padrão OUTDO. As coordenadas de linha e coluna serão colocadas no par de registradores HL e a posição do cursor estabelecida, por meio da rotina padrão POSIT.

Endereço . . . . . 77A5H

Este é o manipulador das instruções "STOP ON/OFF/STOP". O endereço do byte de status do dispositivo na TRPTBL é colocado no par de registradores HL e o controle é transferido para a rotina "ON/OFF/STOP"(77CFH).

Endereço ..... 77ABH

Este é o manipulador das instruções “SPRITE ON/OFF/STOP”. O endereço do byte de status do dispositivo na TRPTBL será colocado no par de registradores HL e o controle será transferido para a rotina “ON/OFF/STOP”(77CFH).

Endereço ..... 77B1H

Este é o manipulador das instruções “INTERVAL ON/OFF/STOP”. Como não há nenhum átomo específico para “INTERVAL” (o controle é transferido para cá quando um símbolo “INT” é encontrado) primeiro é feita uma verificação no texto do programa dos caracteres “S” e “R” e depois do átomo “VAL”(94H). O endereço do byte de status do dispositivo na TRPTBL será colocado no par de registradores HL e o controle será transferido para a rotina “ON/OFF/STOP”(77CFH).

Endereço ..... 77BFH

Este é o manipulador das instruções “STRIG ON/OFF/STOP”. O número do gatilho (Trigger), de zero a quatro, será avaliado(7C08H) e o endereço de byte de status do dispositivo na TRPTBL será colocado no par de registradores HL. O átomo “ON/OFF/STOP” será examinado e o byte de status na TRPTBL será modificado de acordo (77FEH). Em seguida, o controle será transferido, diretamente, à Ronda de Execução (4612H) para evitar o teste de interrupção pendentes, até o final da próxima instrução.

Endereço ..... 77D4H

Este é o manipulador das instruções “KEY(n) ON/OFF/STOP”. O número da tecla, de um a dez, é avaliado(521CH) e o endereço do byte de status do dispositivo na TRPTBL colocado no par de registradores HL. O átomo “ON/OFF/STOP” é examinado e o byte de status na TRPTBL é modificado de acordo(77FEH). O bit 0 do byte de status na TRPTBL, o bit ON, é, então, copiado na entrada correspondente de FNKFLG, para ser utilizado durante a varredura de teclas por interrupção, e o controle é transferido diretamente à Ronda de Execução(4612H).

Endereço ..... 77FEH

Esta rotina verifica a presença de um dos átomos de comutação de interrupção e transfere o controle para a rotina adequada: “ON”(631BH), “OFF”(632BH) ou “STOP” (6331H). Caso nenhum símbolo esteja presente será gerado um “Erro de sintaxe”(4055H).

Endereço ..... 7810H

Esta rotina é utilizada pelo manipulador das instruções “ON DISPOSITIVO GOSUB”(490DH) para verificar se o texto do programa contém uma senha de dispositivo. A não ser que nenhum dos átomos de dispositivo esteja presente, sendo o Indicador C devolvido, o número de entrada do dispositivo na TRPTGL é devolvido no registrador B e o número máximo de operandos de número de linha permitido é devolvido no registrador C:

| DISPOSITIVO | NÚMERO NA TRPTBL# | NÚMEROS DE LINHA |
|-------------|-------------------|------------------|
| KEY         | 00                | 10               |
| STOP        | 10                | 01               |
| SPRITE      | 11                | 01               |
| STRIG       | 12                | 05               |
| INTERVAL    | 17                | 01               |

Além disso, apenas para “INTERVAL”, o operando do intervalo é avaliado(542FH) e colocado em INTVAL e INTCNT.

Endereço ..... 785CH

Esta rotina é utilizada pelo manipulador das instruções “ON DISPOSITIVO GOSUB”(490DH) para colocar o endereço de uma linha do programa em TRPTBL. O número de entrada na TRPTBL, fornecido pelo registrador B, será multiplicado por três e acrescentado à base da tabela para apontar para a entrada relevante. O endereço, fornecido no par de registradores DE, será, em seguida, colocado ali, primeiro o LSB e depois o MSB.

Endereço ..... 786CH

Este é o manipulador da instrução “KEY”. Se o caractere seguinte for alguma outra coisa, que não o átomo “LIST”(93H), o controle será transferido ao manipulador da instrução “KEY n”(78AEH). Cada uma das dez strings das teclas de função será, em seguida, tomada de FNKSTR e apresentada através da rotina padrão OUTDO com CR, LF(7328H) depois de cada uma. O caractere DEL(7FH), ou qualquer caractere de controle menor do que 20H será substituído por um espaço.

Endereço ..... 78AEH

Este é o manipulador das instruções “KEY n”, “KEY(n) ON/OFF/STOP”, “KEY ON” e “KEY OFF”. Se o próximo caractere do texto de programa for “(”, o

controle será transferido para o manipulador da instrução "KEY(n) ON/OFF/STOP" (77D4H). Se for um átomo "ON"(95H), o controle será transferido para a rotina padrão DSPFNK, e, se for um átomo "OFF"(EBH), para a rotina padrão ERAFNK. Caso contrário, o número da tecla de função será avaliado(521CH) e o endereço FNKSTR da tela colocado no par de registradores DE. O operando string será avaliado(4C64H) e seu armazenamento liberado(67DOH). Até quinze caracteres são copiados da string para FNKSTR e posições não utilizadas são preenchidas com bytes zero. Caso um byte zero seja encontrado em um operando string, será gerado um erro de "Chamada ilegal de uma função"(475AH). Em seguida, o controle será transferido para a rotina padrão FNKSB, a fim de atualizar o display das teclas de função caso este esteja ativado.

Endereço ..... 7900H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "TIME". O conteúdo de JIFFY será colocado em DAC como um número de simples precisão(3236H).

Endereço ..... 790AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CSRLIN". O conteúdo de CSRY será decrementado e colocado em DAC como um inteiro(2E9AH).

Endereço ..... 7911H

Este é o manipulador da instrução "TIME". O operando será avaliado(542FH) e colocado em JIFFY.

Endereço ..... 791BH

Esta rotina será utilizada pelo Avaliador de Fatores para aplicar a função "PLAY". O operando de seleção de canal numérico é avaliado(7CO8H). Se for zero, o conteúdo de MUSICF será colocado em DAC, como um inteiro de valor zero ou FFFFH. Caso contrário, o número de canal será utilizado para selecionar o bit apropriado de MUSICF e este é, em seguida, convertido para um inteiro como antes.

Endereço ..... 7940H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "STICK"

em um operando contido em DAC. O número do joystick será verificado(521FH) e passado para a rotina padrão GTSTCK no registrador A. O resultado será colocado em DAC como um inteiro(4FCFH).

Endereço . . . . . 794CH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “STRIG” em um operando contido em DAC. O número do gatilho será verificado (521FH) e passado para a rotina padrão GTTRIG no registrador A. O resultado será colocado em DAC como um inteiro de valor zero ou FFFFH.

Endereço . . . . . 795AH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “PDL” em um operando contido em DAC. O paddle tem seu número verificado(521FH) e passado para a rotina padrão GTPDL no registrador A. O resultado será colocado em DAC como um inteiro(4FCFH).

Endereço . . . . . 7969H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “PAD” em um operando contido em DAC. O número do pad será verificado(521FH) e passado para a rotina padrão GTPAD no registrador A. O resultado será colocado em DAC como um inteiro para os pads 1, 2, 5, e 6. Para os pads, 0, 3, 4 ou 7 o resultado será colocado em DAC como um inteiro de valor zero ou FFFFH.

Endereço . . . . . 7980H

Este é o manipulador da instrução “COLOR”. Caso exista um operando para a cor do primeiro plano, ele será avaliado(521CH) e colocado no registrador E; caso contrário a cor atual do primeiro plano será tomada de FORCLR. Se existir um operando para a cor de fundo, ele será avaliado(521CH) e colocado no registrador D; caso contrário a cor atual de fundo é tomada de BAKCLR. Se existir um operando para a cor de contorno, ele será avaliado(521CH) e colocado no BDRCLR. A cor de primeiro plano será colocada em FORCLR e ATRBYT, a cor de fundo em BAKCLR e o controle será transferido para a rotina padrão CHGCLR, a fim de modificar o VDP.

Endereço ..... 79CCH

Este é o manipulador da instrução "SCREEN". Caso exista um operando de modo, ele será avaliado(521CH) e passado para a rotina padrão CHGMOD no registrador A. Caso exista um operando de tamanho do sprite, ele será avaliado(521CH) e colocado nos bits 0 e 1 de RG1SAV, a cópia da Área de Trabalho do Registrador Modo 1 do VDP. Os parâmetros de sprite do VDP serão, em seguida, limpos por meio da rotina padrão CLRSPR. Caso exista um operando de click de tecla, ele será avaliado(521CH) e colocado em CLIKSW, zero para desativar o click e não-zero para a sua ativação. Caso exista um operando de frequência de baud, ele será avaliado e a frequência de baud ativada(7A2DH). Caso exista um operando de modo de impressora ele será avaliado(521CH) e colocado em NTMSXP, zero para uma impressora MSX e não-zero para uma impressora de utilização geral.

Endereço ..... 7A2DH

Esta rotina é utilizada para estabelecer a frequência baud do cassete. O operando é avaliado(521CH) e cinco bytes serão copiados de CS1200 ou CS2400 para LOW, conforme o caso.

Endereço ..... 7A48H

Este é o manipulador da instrução "SPRITE". Se o próximo caractere for qualquer coisa diferente de "\$", o controle será transferido para o manipulador das instruções "SPRITE ON/OFF/STOP"(77ABH). Em seguida, SCRMOD será verificado e será gerado um erro de "Chamada ilegal de uma função"(475AH), se a tela estiver no Modo Texto 40x24. O número da imagem do sprite será avaliado e sua localização na Tabela de imagens de Sprites da VRAM será obtida(7AAOH). Em seguida, o operando string é avaliado(4C5FH) e seu armazenamento liberado(67DOH). O tamanho do sprite, obtido por meio da rotina padrão GSPSIZ, é comparado com o comprimento da string e, se a string for mais curta do que o sprite, a entrada da Tabela de Padrão Sprite será primeiro preenchida com zeros por meio da rotina padrão FILVRM. Em seguida, os caracteres serão copiados do corpo da string para a Tabela de Imagens de Sprites, por meio da rotina padrão LDIRVM, até que a string seja esgotada ou o sprite esteja lotado. Caso a string seja maior do que o tamanho do sprite, quaisquer caracteres em excesso serão ignorados.

Endereço ..... 7A84H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função

“SPRITE\$”. O número da imagem do sprite é avaliado e é obtida a sua localização na Tabela de Imagens de Sprites da VRAM(7A9FH). O tamanho do sprite, obtido por meio da rotina padrão GSPSIZ, é, em seguida, colocado no par de registradores BC para controlar o número de bytes copiados. Depois de verificar se há espaço suficiente disponível na Área de Armazenamento de Strings(6627H), a imagem do sprite será copiada da VRAM, por meio da rotina padrão LDIRMV e o descritor de resultado será criado(6654H). Observe que, se nenhuma verificação for feita no modo de tela, durante esta função poderão ser encontrados alguns efeitos secundários interessantes, conforme abaixo.

Endereço . . . . . 7A9FH

Esta rotina é utilizada pela instrução e pela função “SPRITE\$” para localizar uma imagem de sprite na Tabela de Imagens de Sprites na VRAM. O operando do número de imagem será avaliado(7C80H) e passado para a rotina padrão CALPAT no registrador A. O endereço da imagem será colocado no par de registradores DE e a rotina terminará.

Observe que nenhuma verificação é feita quanto à validade do número da imagem para o tamanho selecionado. Números de imagens até 255 são aceitos, mesmo no modo sprite 16x16, quando o número máximo deveria ser 63. Como resultado, um endereço da VRAM maior do que 3FFFH será produzido, que dará a volta até um endereço abaixo. A instrução “SPRITE\$”, neste caso, estragará a Tabela de Geração de Caracteres, por exemplo:

```
10 SCREEN 3,2
20 SPRITE$(0)=STRING$(32,255)
30 PUT SPRITE 0,(0,0),,0
40 SPRITE$(65)=STRING$(32,255)
50 GOTO 50
```

O programa acima coloca um sprite real no topo esquerdo da tela e depois utiliza uma instrução ilegal na linha 40 para corromper a VRAM logo à sua direita. A função “SPRITE\$”, também, poderá ser manipulada desta forma e, como não há verificação do modo de tela, até um total de 32 bytes da Tabela de Nomes poderão ser lidos no Modo Texto 40x24, por exemplo:

```
10 SCREEN 0,2
20 PRINT "QUALQUER COISA"
30 A$=SPRITE$(64)
40 PRINT A$
```



Endereço . . . . . 7AAFH

Este é o manipulador da instrução “GET/PUT SPRITE”, com o controle transferido para cá, a partir do manipulador da instrução “GET/PUT”(775BH) Primeiro, o registrador B é verificado para garantir que a declaração é “PUT” e, em caso contrário, será gerado um erro de “Chamada ilegal de uma função” (475AH). Em seguida, o modo SCRMOD será verificado e será gerado um erro de “Chamada ilegal de uma função” (475AH), se a tela estiver no Modo Texto 40x24. O operando do número do sprite, de zero a 32, será avaliado(521CH) e passado para a rotina padrão CALATR para a localização do bloco de atributo de quatro bytes na Tabela de Atributos dos Sprites. Se existir um operando de coordenada ele será avaliado, a coordenada X sendo colocada no par de registradores BC, e a coordenada Y, no par de registradores DE(579CH).

O LSB da coordenada Y é escrito no byte 0 do bloco de atributo na VRAM, por meio da rotina padrão WRTVRM. O bit 7 da coordenada X será examinado, a seguir, para determinar se é negativa, isto é, se passou da tela pelo lado esquerdo. Em caso afirmativo, 32 será somado à coordenada X e o registrador B colocado em 80H para ligar o bit “adianta” (early clock) do bloco de atributo. Por exemplo, uma coordenada X de -1(FFFFH) seria alterada para +31 com um “adianta”. O LSB da coordenada X será, então, escrito no byte 1 do bloco de atributos por meio da rotina padrão WRTVRM. O byte 3 do bloco de atributo será lido por meio da rotina padrão RDVRM, o novo bit “adianta” mascarado e o byte será reescrito na VRAM, por meio da rotina WRTVRM.

Se um operando de cor estiver presente, ele será avaliado(521CH), o byte 3 do bloco de atributo será lido, por meio da rotina padrão RDVRM, o novo código de cor mascarado nos quatro bits inferiores e o byte será reescrito na VRAM, por meio da rotina padrão WRTVRM. Se existir, um operando de número de imagem, ele será avaliado (521CH) e será verificada a sua grandeza, em comparação com o tamanho do sprite atual, fornecido pela rotina padrão GSPSIZ. O número máximo permissível é 255 para sprites 8x8 e 63 para sprites 16x16. O número da imagem será escrito no byte 2 do bloco de atributo; por meio da rotina padrão WRTVRM e o manipulador terminará.

Endereço . . . . . 7B37H

Este é o manipulador da instrução “VDP”. O operando do número do registrador, de zero a sete, será avaliado(7CO8H), seguido pelo operando do dado (521CH). O número do registrador será colocado no registrador C, o valor dos dados no registrador B e o controle transferido para a rotina padrão WRTVDP.

Endereço . . . . . 7B47H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "VDP". O operando do número do registrador, de zero a oito, será avaliado(7C08H) e acrescentado a RG0SAV, para localizar a imagem do registrador correspondente na Área de Espaço de Trabalho. A imagem do registrador VDP, em seguida, será lida e colocada em DAC, como um inteiro(4FCFH).

Endereço . . . . . 7B5AH

Este é o manipulador da instrução "BASE". O operando do número da tabela VDP, de zero a dezenove, será avaliado(7C08H), seguido pelo operando do endereço base(4C64H). Depois de verificar se o endereço de base é menor do que 4000H(7BFEH), o número da tabela VDP será utilizado para localizar a entrada associada na tabela de máscaras em 7BA3H. É aplicada a função AND entre o endereço de base e a máscara e será gerado um erro de "Chamada ilegal de uma função(475AH), se quaisquer bits ilegais estiverem ligados. O número da tabela VDP será, em seguida, somado ao TXTNAM para localizar o endereço de base atual na Área de Trabalho e o novo endereço de base será colocado ali. O número da tabela VDP será dividido por cinco, para determinar a qual dos quatro modos de tela a tabela pertence. Se este for o mesmo do modo de tela atual, o novo endereço de base também será escrito no VDP(7B99H).

Endereço . . . . . 7B99H

Esta rotina é utilizada pelo manipulador da instrução "BASE", para atualizar os endereços de base VDP. O modo de tela atual, no registrador A, será examinado e o controle transferido à rotina padrão SETTXT, SETT32, SETGRP ou SETMLT, conforme apropriado. Observe que isto não é uma inicialização completa do VDP e que os quatro endereços de tabela atuais (NAMBAS, CGPBAS, PATBAS e ATRBAS), que são aqueles realmente utilizados pelas rotinas de tela, não são atualizados. Isto poderá ser demonstrado com o seguinte programa, no qual o Interpretador continuará enviando para a antiga Tabela de Nomes da VRAM:

```
10 SCREEN 0
20 BASE(0)=&H400
30 PRINT"QUALQUER COISA"
40 FOR N=1 TO 2000:NEXT
50 BASE(0)=0
```

Observe também que esta rotina contém um erro. Enquanto SETTXT é utilizado corretamente para o Modo Texto 40x24, SETGRP é utilizado para o Modo Texto 32x24 e SETMLT para o Modo Gráfico e Modo Multicolorido. Qualquer instrução “BASE” deveria, portanto, ser seguida imediatamente por uma instrução “SCREEN” para uma inicialização completa.

Endereço ..... 7BA3H

Esta tabela de máscaras é utilizada pelo manipulador da instrução “BASE”, para garantir que apenas endereços de base VDP legais sejam aceitos. O número da tabela e a variável da Área de Trabalho correspondente será mostrado com sua máscara:

| <u>MASK</u> | <u>TABLE</u> |
|-------------|--------------|
| 03FFH       | 00, TXTNAM   |
| 003FH       | 01, TXTCOL   |
| 07FFH       | 02, TXTCGP   |
| 007FH       | 03, TXTATR   |
| 07FFH       | 04, TXTPAT   |
| 03FFH       | 05, T32NAM   |
| 003FH       | 06, T32COL   |
| 07FFH       | 07, T32CGP   |
| 007FH       | 08, T32ATR   |
| 07FFH       | 09, T32PAT   |
| 03FFH       | 10, GRPNAM   |
| 1FFFH       | 11, GRPCOL   |
| 1FFFH       | 12, GRPCGP   |
| 007FH       | 13, GRPATR   |
| 07FFH       | 14, GRPPAT   |
| 03FFH       | 15, MLTNAM   |
| 003FH       | 16, MLTCOL   |
| 07FFH       | 17, MLTCGP   |
| 007FH       | 18, MLTATR   |
| 07FFH       | 19, MLTPAT   |

Endereço ..... 7BCBH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “BASE”. O operando do número da tabela do VDP, de zero a dezenove será avaliado(7C08H) e acrescentado a TXTNAM para localizar o endereço de base da Área de Trabalho requerida. Isto é, em seguida, colocado em DAC, como um número de simples precisão(3236H).

Endereço ..... 7BE2H

Este é o manipulador da instrução “VPOKE”. O operando de endereço da VRAM será avaliado(4C64H) e verificado para garantir que seja menor que 4000H (7BFEH). Em seguida, o operando de dado será avaliado(521CH) e passado para a rotina padrão WRTVRM no registrador A, a fim de escrever no endereço requerido.

Endereço . . . . . 7BF5H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “VPEEK” em um operando contido em DAC. O operando do endereço da VRAM será verificado para garantir de que seja menor do que 4000H(7BFEH). Em seguida, a VRAM é lida por meio da rotina padrão RDVRAM e o resultado será colocado em DAC como um inteiro(4FCFH).

Endereço . . . . . 7BFEH

Esta rotina converte um operando numérico em DAC em um inteiro(2F8AH) e o devolve no par de registradores HL. Se o operando for igual ou maior do que 4000H, e assim fora da faixa permitida para a VRAM, será gerado um erro de “Chamada ilegal de uma função(475AH).

Endereço . . . . . 7C08H

Esta rotina avalia(521CH) um operando numérico com parênteses e o devolve como um inteiro no registrador A. Se o operando for maior do que o valor permissível máximo, inicialmente fornecido no registrador A, será gerado um erro de “Chamada ilegal de uma função”(475AH).

Endereço . . . . . 7C16H

Este é o manipulador da instrução “DSKO\$”. Será gerado um erro “Chamada ilegal de uma função”(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C1BH

Este é o manipulador da instrução “SET”. Será gerado um erro “Chamada ilegal de uma função”(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C20H

Este é o manipulador da instrução “NAME”. Será gerado um erro “Chamada ilegal de uma função”(475AH), em uma máquina MSX padrão.

Endereço ..... 7C25H

Este é o manipulador da instrução "KILL". Será gerado um erro "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C2AH

Este é o manipulador da instrução "IPL". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C2FH

Este é o manipulador da instrução "COPY". Será gerado um erro "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C34H

Esta é o manipulador da instrução "CMD". Será gerado um erro "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C39H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "DSKF" em um operando contido em DAC. Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C3EH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "DSKI\$". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço ..... 7C43H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "ATTR\$". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C48H

Este é manipulador da instrução "LSET". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C4DH

Este é manipulador da instrução "RSET". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C52H

Este é manipulador da instrução "FIELD". Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C57H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "MKI\$" em um operando contido em DAC. Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C5CH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "MKD\$" em um operando contido em DAC. Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C61H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "MKD\$" em um operando contido em DAC. Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C66H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função "CVI" em um operando contido em DAC. Será gerado um erro de "Chamada ilegal de uma função"(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C6BH

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “CVS” em um operando contido em DAC. Será gerado um erro de “Chamada ilegal de uma função”(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C70H

Esta rotina é utilizada pelo Avaliador de Fatores para aplicar a função “CVD” em um operando contido em DAC. Será gerado um erro de “Chamada ilegal de uma função”(475AH), em uma máquina MSX padrão.

Endereço . . . . . 7C76H

Esta rotina completa a inicialização de partida (Power-up). Neste momento, toda a Área de Trabalho está zerada e unicamente EXPTBL e SLTTBL foram inicializados. Uma pilha temporária é estabelecida em 7376H e todos os 112 ganchos (560 bytes) são preenchidos com instruções RET do Z80 (C9H). HIMEN ficará valendo F380H e será encontrada a posição mais baixa da RAM(7D5DH) e colocada em BOTTOM. Os 144 bytes de dados, iniciando em 7F27H serão copiados na Área de Trabalho de F380H até F40FH. As strings das teclas de função serão inicializadas por meio das rotinas padrões INIFNK, ENDBUF e NLOONLY e serão zerados, uma vírgula será colocada em BUFMIN e um ponto-e-vírgula em KBFMIN. O endereço do conjunto de caracteres MSX em ROM é tomado das posições 0004H e 0005H e colocado em CGPNT + 1 e PRMPRV é apontado para PRMSTK. Valores, sem significado, serão colocados em STKTOP, MEMSIZ e VARTAB (os seus valores corretos não são ainda conhecidos), é alocado um buffer de E/S(7E6BH) é atribuído ao SP do Z80(62E5H). Um byte zero será colocado na base da RAM, TXTTAB será colocado nas posições seguintes e um “NEW” será executado(6287H).

Em seguida, o VDP é inicializado por meio das rotinas padrões INITIO, INIT32 e CLRSPR, as coordenadas do cursor são colocadas na linha 11, na coluna 10 a mensagem inicial “sistema MSX etc...” é apresentada(6678H). Após uma demora de três segundos, possíveis ROMs de extensão são procuradas(7D75H) e um “NEW” adicional é executado (6287H), se um programa BASIC tiver sido processado da ROM. Finalmente, a mensagem de identificação “MSX BASIC etc.” é apresentada(7D29H) e o controle é transferido ao ponto “OK” da Ronda Principal do Interpretador(411FH).

Endereço . . . . . 7D29H

Esta rotina é utilizada durante a partida para ativar a apresentação das teclas de função, colocar a tela no Modo Texto 40x24, por meio da rotina padrão INITXT, e apresentar(6678H) e mensagem de identificação “MSX BASIC etc.”. A quantidade de memória livre será, em seguida, computada pela subtração do conteúdo de VARTAB, do conteúdo de STKTOP e apresentada(4312H), seguida, pela mensagem de “Bytes livres”.

Endereço . . . . . 7D5DH

Esta rotina é utilizada durante a partida para encontrar a posição da RAM mais baixa. Iniciando em EF00H cada byte é testado, até que seja encontrado um que não possa ser escrito ou o endereço 8000H seja atingido. O endereço de base arredondado para cima até a página de 256 bytes mais próxima, será devolvido no par de registradores HL.

Endereço . . . . . 7D75H

Esta rotina é utilizada durante a partida para procurar uma ROM de extensão. As páginas 1 e 2(4000H até BFFFH) de cada conector são examinadas e os resultados colocados em SLTATR. Uma ROM de extensão em dois caracteres de identificação “AB” nos dois primeiros bytes, para distingui-la da RAM. A informação a respeito de suas propriedades, também está presente nos primeiros dezesseis bytes como segue:

| RESERVADO                        |            |
|----------------------------------|------------|
| MSB do Endereço de texto BASIC   | Byte 10-15 |
| LSB do Endereço de texto BASIC   | Byte 9     |
| MSB do Endereço de DISPOSITIVO   | Byte 8     |
| LSB do Endereço de DISPOSITIVO   | Byte 7     |
| MSB do Endereço de INSTRUÇÕES    | Byte 6     |
| LSB do Endereço de INSTRUÇÕES    | Byte 5     |
| MSB do Endereço de INICIALIZAÇÃO | Byte 4     |
| LBS do Endereço de INICIALIZAÇÃO | Byte 3     |
|                                  | Byte 2     |
|                                  | Byte 1     |
|                                  | Byte 0     |
| 42H                              |            |
| 41H                              |            |

Figura 48 Cabeçalho da ROM.

Cada página em um dado conector é examinada pela leitura dos dois primeiros bytes(7E1AH) e é verificada a presença dos caracteres “AB”. Se uma ROM estiver presente, o endereço de inicialização será lido(7E1AH) e o controle será passado a ele, por meio da rotina padrão CALSTL. Em uma ROM de jogos poderá não haver retorno ao



BASIC deste ponto. O endereço do manipulador da instrução de extensão "CALL" será lido, em seguida(7E1AH) e o bit 5 do registrador B ligado se for válido, isto é, não-zero. O endereço do manipulador de dispositivos de extensão é lido(7E1AH) e o bit 6 do registrador B ligado se for válido. Finalmente, o endereço do texto de programa BASIC, será lido(7E1AH) e o bit 7 do registrador B será ligado se for válido. Em seguida, o registrador B é copiado para a posição relevante em SLTATR e a procura continuará, até que não haja mais nenhum conector.

Em seguida SLTATR, é examinado quanto a qualquer ROM de extensão indicada como contendo texto de programa BASIC. Se for encontrada uma, sua posição no SLTATR será convertida em uma identificação de conector(7E2AH) e a ROM permanentemente comutada por meio da rotina padrão ENASLT. VARTAB será colocada em COOOH, uma vez que não sabemos qual é o tamanho da Área de Texto do Programa, TXTTAB será colocado em 8008H e BASROM feito não-zero para desativar a tecla CTRL-STOP. O sistema será limpo(629AH) e o controle será transferido à Ronda de Execução (4601H) para executar o programa BASIC.

Endereço ..... 7E1AH

Esta rotina é utilizada para ler dois bytes de posições sucessivas em uma ROM de extensão. O endereço inicial é fornecido pelo par de registradores HL e o Identificador do conector pelo registrador C. Os bytes são lidos, por meio da rotina padrão RDSLt e devolvidos no par de registradores DE. Se os dois forem zero, o Indicador Z será devolvido.

Endereço ..... 7E2AH

Esta rotina converte a posição SLTATR fornecida pelo registrador B na Identificação de conector correspondente no registrador C e o endereço de base da ROM no registrador HL. A posição é primeiro modificada de modo que varie de 0 a 63, em vez de 64 a 1, fazendo com que a informação requerida esteja presente na forma:

| 7 | 6 | 5                    | 4 | 3                      | 2 | 1            | 0 |
|---|---|----------------------|---|------------------------|---|--------------|---|
| 0 | 0 | Nº conector primário |   | Nº conector secundário |   | Nº da página |   |

**Figura 49**

Os bit 0 e 1 são deslocados para os bits superiores do registrador H, para formar o endereço. Os bits 4 e 5 são deslocados para os bits 0 e 1 do registrador C, para formar o

---

número de Conector Primário. Os bits 2 e 3 são deslocados para os bits 2 e 3 do registrador C, para formar o número do Conector Secundário e o bit 7 da entrada na EXPTBL correspondente copiado no bit 7 do registrador C.

Endereço . . . . . 7E4BH

Este é o manipulador da instrução "MAXFILES". O controle é transferido para cá quando um átomo "MAX"(CDH) é delectado, o texto do programa é verificado primeiro quanto à existência de um átomo "FILES" final(B7H). Em seguida, o operando de número de buffers, de zero a quinze, será avaliado(521CH) e qualquer buffer existente será fechado(6C1CH). O número requerido de buffers de E/S será alocado(7E6BH), o sistema será limpo(62A7H) e o controle transferido diretamente à Ronda de Execução (4601H).

Endereço . . . . . 7E6BH

Esta é a rotina de alocação do buffer de E/S. É utilizada durante a partida e pelos manipuladores das declarações "MAXFILES" e "CLEAR", para alocar armazenamento para o número de buffers de E/S fornecidos pelo registrador A. São subtraídos 267 bytes do conteúdo de HIMEM para cada buffer, a fim de fornecer um novo valor para MEMSIZ. O tamanho da Área de Armazenamento de Strings existentes (inicialmente duzentos bytes) é computado pela subtração do antigo conteúdo de STKTOP, do antigo conteúdo de MEMSIZ, que por sua vez é subtraído do novo valor de MEMSIZ para produzir o novo valor de STKTOP. 140 bytes adicionais são subtraídos para a pilha do Z80 e um erro de "Falta de memória" será gerado(6275H), se este endereço for menor do que o início da Área de Armazenamento de Variáveis. Caso contrário, o número de buffers será colocado em MAXFIL e MEMSIZ e STKTOP e atualizados com seus novos valores. O endereço do chamador é retirado da pilha, o SP do Z80 passa a apontar para a nova posição e o endereço de retorno colocado de volta na pilha. Em seguida, FILTAB passa a apontar para o começo do bloco de apontador do buffer de E/S e cada apontador colocado de forma a apontar para o FCB associado. Finalmente, o endereço do buffer 0 de E/S, o buffer "SAVE" e "LOAD" do Interpretador, será colocado em NULBUF e a rotina terminará.

Endereço . . . . . 7ED8H

Esta é a mensagem de texto "sistema MSX" terminando em um byte zero.

Endereço ..... 7EE4H

Esta é a mensagem de texto “versão 1.0” CR, LF terminando em um byte zero.

Endereço ..... 7EF2H

Esta é a mensagem de texto “MSX BASIC” terminando em um byte zero.

Endereço ..... 7EFDH

Esta é a mensagem de texto “Copyright 1983 by Microsoft” CR, LF terminando em um byte zero.

Endereço ..... 7F1BH

Esta é a mensagem de texto “Bytes livres” terminando em um byte zero.

Endereço ..... 7F27H

Este bloco de 144 bytes de dados é utilizado para inicializar a Área de Trabalho de F380H até F40FH.

Endereço ..... 7FB7H

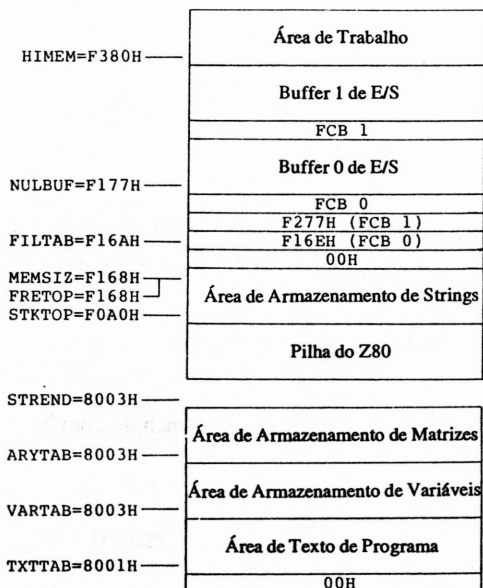
Este remendo de sete bytes conserta um problema de rotina de análise sintática de dispositivos externos(55F8H). Se há um nome de dispositivo de comprimento zero no registrador A ele é alterado para um.

Endereço ..... 7FBEH

Esta seção da ROM não é utilizada e preenchida com bytes zero.

**MAPEAMENTO DA MEMÓRIA**

Um máximo de 32KB de RAM está disponível para o Interpretador BASIC, a fim de guardar o texto do programa, as Variáveis BASIC, a pilha do Z80, os buffers de E/S e a Área de Trabalho. Um mapa de memória destas áreas no estado de partida, é mostrado abaixo:

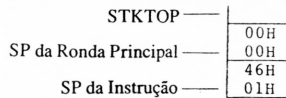


**Figura 50** Mapa da Memória de 8000H até FFFFH.

A Área de Texto do Programa é formada por linhas de programa atomizadas, armazenadas por ordem de número de linha e terminadas por um elo terminal zero. No estado “NEW”, apenas o elo terminal está presente. O byte zero em 8000H é um caractere de fim-de-linha necessário apenas para sincronizar a Ronda de Execução no início do programa.

As Áreas de Armazenamento de Matrizes e de Variáveis são formadas por Variáveis e Matrizes numéricas ou string armazenadas na ordem em que são encontradas no texto do programa. A velocidade de execução de um programa melhora acentuadamente quando as Variáveis são declaradas antes das Matrizes, após isto reduz a quantidade de memória que deve ser movida para cima.

A pilha do Z80 é posicionada imediatamente abaixo da Área de Armazenamento de Strings, com a estrutura do topo da pilha mostrada a seguir:



**Figura 51** Topo da Pilha do Z80.

Sempre que a posição da pilha é alterada, como resultado de uma instrução “CLEAR” ou “MAXFILES”, dois bytes zero são empurrados primeiro para funcionar como um terminador, durante a busca de blocos de parâmetros “FOR” ou “GOSUB”. Considerando que nenhum bloco de parâmetro esteja presente, o SP do Z80 estará, portanto, com STKTOP-2 na Ronda Principal do Interpretador e com STKTOP-4, quando o controle for transferido da Ronda Principal a um manipulador de instrução.

A Área de Armazenamento de String é formada por corpos string atribuídos à Variáveis ou Matrizes. Durante a avaliação de uma expressão, os resultados de várias strings intermediárias poderão também estar temporariamente presentes entre as strings permanentes. O byte zero que segue a Área de Armazenamento de String é um delimitador temporário para a função “VAL”.

A região entre a Área de Armazenamento de Strings e HIMEM é utilizada para armazenamento de buffer de E/S. O buffer 0 de E/S, o buffer de “SAVE” e “LOAD”, estará sempre presente, mas o número de buffers de usuário será determinado pela instrução “MAXFILES”. Cada buffer de E/S consiste em um FCB de 9 bytes, cujo endereço está contido na tabela abaixo do FCB 0, seguido por um buffer de dados de 256 bytes. O FCB contém o status do buffer de E/S conforme segue:

| 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| MOD | 00H | 00H | 00H | DEV | 00H | POS | 00H | PPS |

**Figura 52** Bloco de Controle de Arquivo.

O byte MOD contém o modo do buffer, o byte DEV contém o código do dispositivo, o byte POS contém a posição atual no buffer (0 a 255) e o byte PSS contém a posição de "PRINT". O restante do FCB não é utilizado em uma máquina MSX padrão.

### Área de Espaço de Trabalho

A seção da Área de Trabalho de F380H até FD99H contém as variáveis do BIOS/Interpretador. Estas são relacionadas nas páginas a seguir, na forma de linguagem assembler:

```

F380H   RDPRIM:  OUT    (0A8H), A   ;Estabelece novo Conect. prim.
F382H           LD     E, (HL)      ;Lê memória
F383H           JR     WRPRM1       ;Restaura antigo con. primário

```

Esta rotina é utilizada pela rotina padrão RDSLTL para comutar Conectores Primários e ler um byte da memória. A nova opção do Registrador do Conector Primário é fornecida no registrador A, a antiga opção no registrador D e o byte lido devolvido no registrador E.

```

F385H   WRP1M:   OUT    (0A8H), A   ;Estabelece novo Co. Primário
F387H           LD     (HL), E       ;Escreve na memória
F388H   WRPRM1:  LD     A, D         ;Pega opção antiga
F389H           OUT    (0A8H), A     ;Restaura opção Prim. antigo

```

Esta rotina é utilizada pela rotina padrão WRSLTL para comutar Conectores Primários e escrever um byte na memória. A nova opção do Registrador de Conector Primário é fornecida no registrador A, a antiga opção no registrador D e o byte a ser escrito, no registrador E.

|       |         |      |           |                              |
|-------|---------|------|-----------|------------------------------|
| F38CH | CLPRIM: | OUT  | (0A8H), A | ;Estabelece nova opção Prim. |
| F38EH |         | EX   | AF, AF    | ;Permuta AF para chamar      |
| F38FH |         | CALL | CLPRM1    | ;Chama                       |
| F392H |         | EX   | AF, AF    | ;Permuta AF'                 |
| F393H |         | POP  | AF        | ;Pega opção antiga           |
| F394H |         | OUT  | (0A8H), A | ;Restaura opção Prim. antigo |
| F396H |         | EX   | AF, AF    | ;Permuta AF                  |
| F397H |         | RET  |           |                              |
| F398H | CLPRM1: | JP   | (IX)      |                              |

Esta rotina é utilizada pela rotina padrão CALSLT para comutar Conectores primários e chamar um endereço. A condição nova do Registrador do Conector Primário é fornecida pelo registrador A, a condição antiga pela pilha do Z80 e o endereço a ser chamado pelo par de registradores IX.

|       |         |      |       |        |
|-------|---------|------|-------|--------|
| F39AH | USRTAB: | DEFW | 475AH | ;USR 0 |
| F39CH |         | DEFW | 475AH | ;USR 1 |
| F39EH |         | DEFW | 475AH | ;USR 2 |
| F3A0H |         | DEFW | 475AH | ;USR 3 |
| F3A2H |         | DEFW | 475AH | ;USR 4 |
| F3A4H |         | DEFW | 475AH | ;USR 5 |
| F3A6H |         | DEFW | 475AH | ;USR 6 |
| F3A8H |         | DEFW | 475AH | ;USR 7 |
| F3AAH |         | DEFW | 475AH | ;USR 8 |
| F3ACH |         | DEFW | 475AH | ;USR 9 |

Estas dez variáveis contêm os endereços das funções "USR". Na partida, seus valores são os do gerador de erro "Chamada de uma função ilegal" do Interpretador. Eles são alterados apenas pela instrução "DEFUSR".

F3AEH LINL40: DEFB 37 (39 no HOT-BIT)

Esta variável contém a largura de tela do Modo Texto 40x24. O seu valor é estabelecido na partida e alterado apenas pela instrução "WIDTH".

F3AFH LINL32: DEBF 29

Esta variável contém a largura de tela do Modo Texto 32x24. O seu valor é estabelecido na partida e alterado apenas pela instrução "WIDTH".

F3BOH LINLEN: DEFB 37

Esta variável contém a largura de tela no Modo Texto atual. O seu valor é estabelecido a partir de LINL40 ou LINL32, sempre que o VDP é inicializado em um modo texto, por meio da rotina padrão INITXT ou INIT32.

F3B1H CRTCNT: DEFB 24

Esta variável contém o número de linhas na tela. O seu valor é estabelecido na partida e depois fica inalterado.

F3B2H CLMLST: DEBF 14

Esta variável contém o número mínimo de colunas que devem estar disponíveis em uma linha para que um item de dado seja impresso via PRINT. Se menos espaço estiver disponível, um CR, LF é acionado primeiro. O seu valor é estabelecido na partida e alterado apenas pelas instruções "WIDTH" e "SCREEN".

|       |              |       |                            |
|-------|--------------|-------|----------------------------|
| F3B3H | TXTNAM: DEFW | 0000H | ;Base da Tabela de Nomes   |
| F3B5H | TXTCOL: DEFW | 0000H | ;Base da Tabela de Cores   |
| F3B7H | TXTCGP: DEFW | 0800H | ;Base Imagem de Caracteres |
| F3B9H | TXATTR: DEFW | 0000H | ;Base de Atributos Sprite  |
| F3BBH | TXTPAT: DEFW | 0000H | ;Base de Imagens Sprite    |

Estas cinco variáveis contém os endereços de base do VDP no Modo Texto 40x24. Os seus valores são estabelecidos na partida e alterados exclusivamente pela instrução "BASE".

|       |              |       |                                |
|-------|--------------|-------|--------------------------------|
| F3BDH | T32NAM: DEFW | 1800H | ;Base da Tabela de Nomes       |
| F3BFH | T32COL: DEFW | 2000H | ;Base da Tabela de Cores       |
| F3C1H | T32CGP: DEFW | 0000H | ;Base de Imagens de Caracteres |
| F3C3H | T32ATR: DEFW | 1BOOH | ;Base de Atributos dos Sprites |
| F3C5H | T32PAT: DEFW | 3800H | ;Base de Imagens de Sprites    |

Estas cinco variáveis contém os endereços de base do VDP no Modo Texto 32x24. Os seus valores são atribuídos na partida e alterados exclusivamente pela instrução "BASE".



|       |              |       |                                |
|-------|--------------|-------|--------------------------------|
| F3C7H | GRPNAM: DEFW | 1800H | ;Base da Tabela de Nomes       |
| F3C9H | GRPCOL: DEFW | 2000H | ;Base da Tabela de Cores       |
| F3CBH | GRPCGP: DEFW | 0000H | ;Base de Imagens de Caracteres |
| F3CDH | GRPATR: DEFW | 1BOOH | ;Base de Atributos de Sprites  |
| F3CFH | GRPPAT: DEFW | 3800H | ;Base de Imagens de Sprites    |

Estas cinco variáveis contêm os endereços de base do VDP no Modo Gráfico. Os seus valores são atribuídos na partida e alterados exclusivamente pela instrução "BASE".

|       |              |       |                                 |
|-------|--------------|-------|---------------------------------|
| F3D1H | MLTNAM: DEFW | 0800H | ;Base da Tabela de Nomes        |
| F3D3H | MLTCOL: DEFW | 0000H | ;Base da Tabela de Cores        |
| F3D5H | MLTCGP: DEFW | 0000H | ;Base das Imagens de Caracteres |
| F3D7H | MLTATR: DEFW | 1BOOH | ;Base de Atributos dos Sprites  |
| F3D9H | MLTPAT: DEFW | 3800H | ;Base das imagens do Sprite     |

Estas cinco variáveis contêm os endereços de base do VDP no Modo Multicolorido. Os seus valores são atribuídos na partida e alterados exclusivamente pela instrução "BASE".

F3DBH CLIKSW: DEFB 01H

Esta variável controla o manipulador do click de tecla por interrupção: 00H=desligado, NZ=ligado. O seu valor é atribuído na partida e alterado exclusivamente pela declaração "SCREEN".

F3DCH CSRY: DEFB 01H

Esta variável contém a coordenada de linha (de 1 até CTRCNT) do cursor no Modo Texto.

F3DDH CSRX: DEFB 01H

Esta variável contém a coordenada da coluna (de 1 até LINLEN) do cursor no modo texto. Observe que, para a BIOS as coordenadas da posição de origem do cursor são 1,1, seja qual for a largura da tela.

F3DEH    CONSDFG: DEFB    FFH

Esta variável contém o estado atual da apresentação das teclas de função:  
00H=desligado, NZ=ligado.

|       |              |     |
|-------|--------------|-----|
| F3DFH | RG0SAV: DEFB | 00H |
| F3E0H | RG1SAV: DEFB | F0H |
| F3E1H | RG2SAV: DEFB | 00H |
| F3E2H | RG3SAV: DEFB | 00H |
| F3E3H | RG4SAV: DEFB | 01H |
| F3E4H | RG5SAV: DEFB | 00H |
| F3E5H | RG6SAV: DEFB | F4H |
| F3E6H | RG7SAV: DEFB | F4H |

Estas oito variáveis reproduzem o estado dos oito Registradores de Modo do VDP apenas de escrita (Write-only). Os valores mostrados são para o Modo Texto 40x24.

F3E7H    STATFL: DEFB    CAH

Esta variável é continuamente atualizada pelo manipulador de interrupção com o conteúdo do Registrador de Status do VDP.

F3E8H    TRGFLG: DEFB    F1H

Esta Variável é continuamente atualizada pelo manipulador de interrupção com o estado das quatro entradas dos gatilhos dos joystick mais a tecla de espaço.

F3E9H    FORCLR: DEFB    0FH            ;Branco

Esta variável contém a cor atual do primeiro plano. Seu valor é atribuído na partida e alterado exclusivamente pela instrução "COLOR". A cor do primeiro plano é utilizada pela rotina padrão CLRSPR para estabelecer a cor do sprite e pela rotina padrão CHGCLR para estabelecer a cor do pixel 1, no modo texto. Funciona também como a cor de tinta gráfica sendo copiada em ATRBYT pela rotina padrão GRPPRT e utilizada em todo o Interpretador como valor default para qualquer operando opcional de cor.

F3EAH    BAKCLR: DEFB    04H                    ;Azul escuro

Esta variável contém a cor atual de fundo. O seu valor é atribuído na partida e alterado exclusivamente pela instrução "COLOR". A cor de fundo é utilizada pela rotina padrão CLS para limpar a tela, nos modos gráficos, e pela rotina padrão CHGCLR para estabelecer a cor do pixel 0, nos modos texto.

F3EBH    BDRCLR: DEFB    04H                    ;Azul escuro

Esta variável contém a cor atual do contorno. O seu valor é estabelecido na partida e alterado exclusivamente pela instrução "COLOR". A cor de contorno é utilizada pela rotina padrão CHGCLR no Modo Texto 32x24, Modo Gráfico e Modo Multicolorido para estabelecer a cor de contorno.

F3ECH    MAXUPD: DEFB    C3H  
F3EDH                    DEFW    0000H

Estes dois bytes são preenchidos pelo manipulador da instrução "LINE" para formar um JP do Z80 para as rotinas padrões RIGHTC, LEFTC, UPC ou DOWNC.

F3EFH    MINUPD: DEFB    C3H  
F3F0H                    DEFW    0000H

Estes dois bytes são preenchidos pelo manipulador da instrução "LINE" para formar um JP do Z80 para as rotinas padrões RIGHTC, LEFTC, UPC ou DOWNC.

F3F2H    ATRBYT: DEFB    0FH

Esta variável contém a cor de tinta gráfica utilizada pelas rotinas padrões SETC e NSETCX;

F3F3H    QUEUES: DEFW    F959H

Esta variável contém o endereço dos blocos de controle para as três filas musicais. O seu valor é estabelecido na partida e permanece inalterado.

F3F5H    FRCNEW: DEFB    FFH

Esta variável contém um indicador para distinguir as duas instruções do manipulador das instruções “CLOAD/CLOAD?”: 00H=CLOAD, FFH=CLOAD?.

F3F6H    SCNCNT: DEFB    01H

Esta variável é utilizada como um contador pelo manipulador de interrupção para controlar a velocidade com que as varreduras do teclado são feitas.

F3F7H    REPCNT: DEFB    01H

Esta variável é utilizada como um contador pelo manipulador de interrupção para controlar a frequência de repetição da tecla.

F3F8H    PUTPNT: DEFW    FBFOH

Esta variável contém o endereço da posição de “colocar” em KEYBUF.

F3FAH    GETPNT: DEFW    FBFOH

Esta variável contém o endereço da posição “pegar” em KEYBUF.

|       |         |      |     |                                   |
|-------|---------|------|-----|-----------------------------------|
| F3FCH | CS1200: | DEFB | 53H | ;Ciclo LO primeira metade         |
| F3FDH |         | DEFB | 5CH | ;Ciclo LO segunda metade          |
| F3FEH |         | DEFB | 26H | ;Ciclo HI primeira metade         |
| F3FFH |         | DEFB | 2DH | ;Ciclo HI segunda metade          |
| F400H |         | DEFB | 0FH | ;Contagem dos ciclos do cabeçalho |

Estas cinco variáveis contém os parâmetros do cassete para 1200 baud. Os seus valores são estabelecidos na partida e permanecem inalterados.

|       |         |      |     |                                   |
|-------|---------|------|-----|-----------------------------------|
| F401H | CS2400: | DEFB | 25H | ;Ciclo LO primeira metade         |
| F402H |         | DEFB | 2DH | ;Ciclo LO segunda metade          |
| F403H |         | DEFB | 0EH | ;Ciclo HI primeira metade         |
| F404H |         | DEFB | 16H | ;Ciclo HI segunda metade          |
| F405H |         | DEFB | 1FH | ;Contagem dos ciclos do cabeçalho |

Estas cinco variáveis contém os parâmetros do cassete para 2400 baud. Os seus valores são estabelecidos na partida e permanecem inalterados.

|       |         |      |     |                                   |
|-------|---------|------|-----|-----------------------------------|
| F406H | LOW:    | DEFB | 53H | ;Ciclo LO primeira metade         |
| F407H |         | DEFB | 5CH | ;Ciclo LO segunda metade          |
| F408H | HIGH:   | DEFB | 26H | ;Ciclo HI primeira metade         |
| F409H |         | DEFB | 2DH | ;Ciclo HI segunda metade          |
| F40AH | HEADER: | DEFB | 0FH | ;Contagem dos ciclos do cabeçalho |

Estas cinco variáveis contêm os parâmetros atuais do cassete. O seus valores são colocados em 1200 baud na partida e alterados exclusivamente com as instruções "CSAVE" e "SCREEN".

F40BH ASPCT1: DEFW 0100H

Esta variável contém a recíproca da razão de aspecto default de "CIRCLE" multiplicada por 256. O seu valor é estabelecido na partida e permanece inalterado.

F40DH ASPCT2: DEFW 0100H

Esta variável contém a razão de aspecto default de "CIRCLE", multiplicada por 256. O seu valor é estabelecido na partida e permanece inalterado. A razão de aspecto está presente em duas formas para que o manipulador da declaração "CIRCLE" possa seleccionar a apropriada imediatamente, em vez de precisar examinar e possivelmente calcular a recíproca como seria o caso com um operando no texto do programa.

|       |         |      |     |
|-------|---------|------|-----|
| F40FH | ENDPRG: | DEFB | “;” |
| F410H |         | DEFB | 00H |
| F411H |         | DEFB | 00H |
| F412H |         | DEFB | 00H |
| F413H |         | DEFB | 00H |

Estes cinco bytes formam uma lista de programa sem efeito. Os seus valores são estabelecidos na partida e permanecem inalterados. A linha existe caso um erro ocorra na Ronda Principal do Interpretador, antes que qualquer texto atomizado esteja disponível em KBUF. Caso um "ON ERROR GOTO" esteja ativo neste momento, esta linha serve como um texto onde a instrução "RESUME" possa terminar.

F414H ERRFLG: DEFB 00H

Esta variável é utilizada pelo manipulador de erro do Interpretador para salvar o número do erro.

F415H LPTPOS: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "LPRINT", para conter a posição atual da cabeça da impressora.

F416H PRTFLG: DEFB 00H

Esta variável determina se a rotina padrão OUTDO deve direcionar sua saída para a tela ou para a impressora: 00H=tela, 01H=impressora.

F417H NTMSXP: DEFB 00H

Esta variável determina se a rotina padrão OUTDO substituirá caracteres gráficos prefixados e direcionados à impressora com espaços: 00H=Gráficos, NZ=espaços. O seu valor é atribuído na partida e alterado exclusivamente pela instrução "SCREEN".

F418H RAWPRT: DEFB 00H

Esta variável determina se a rotina padrão OUTDO modificará os caracteres gráficos prefixados e de controle direcionados à impressora: 00H=Modifica, NZ=Mantém. O seu valor é atribuído na partida e permanece inalterado.

F419H VLZADR: DEFW 0000H

F41BH VLZDAT: DEFB 00H

Estas variáveis contêm o endereço e valor de qualquer caractere temporariamente removido pela função "VAL".

F41CH CURLIN: DEFW FFFFH

Esta variável contém o número da linha atual do Interpretador. Um valor de FFFFH denota modo direto.

F41EH KBFMIN: DEFB ""

Este byte fornece um prefixo fictício ao texto atomizado contido em KBUF. A sua função é semelhante àquela de ENDPRG, mas este é utilizado quando ocorre um erro de uma instrução direta.

F41FH KBUF: DEFS 318

Este buffer guarda a forma atomizada da linha coletada pela Ronda Principal do Interpretador. Quando uma instrução direta é executada o conteúdo deste buffer, por si só, forma o texto do programa.

F55DH BUFMIN: DEFB “.”

Este byte fornece um prefixo fictício a texto contido em BUF. É utilizado para sincronizar o manipulador da instrução “INPUT”, assim que este começa a analisar o texto coletado.

F55EH BUF: DEFS 259

Este buffer contém o texto coletado do console pela rotina padrão INLIN.

F661H TTYPOS: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução “PRINT” para guardar a posição atual da tela (a abreviatura vem de Teletipo, em inglês, Teletype!).

F662H DIMFLG: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida pelo manipulador da instrução “DIM” para controlar a operação da rotina de busca de variáveis.

F663H VALTYP: DEFB 02H

Esta variável contém o código de tipo do operando atualmente contido no DAC: 2=Inteiro, 3=String, 4=Simple Precisão, 8=Dupla Precisão.

F664H DORES: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida para impedir a atomização de palavras-chave sem aspas, após um átomo “DATA”.

F665H DONUM: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida quando uma constante

numérica após uma palavra-chave GOTO, GOSUB, THEN etc., precisa ser atomizada na forma especial de operando de número de linha.

F666H    CONTXT: DEFW    0000H

Esta variável é utilizada pela rotina padrão CHRGTTR para salvar o endereço do caractere que vem após uma constante numérica no texto do programa.

F668H    CONSAV: DEFB    00H

Esta variável é utilizada pela rotina padrão CHRGTTR para guardar o átomo de uma constante numérica encontrada no texto do programa.

F669H    CONTYP: DEFB    00H

Esta variável é utilizada pela rotina padrão CHRGTTR para salvar o tipo de uma constante numérica encontrada no texto do programa.

F66AH    CONLO:  DEFS    8:

Este buffer é utilizado pela rotina padrão para salvar o valor de uma constante numérica encontrada no texto do programa.

F672H    MEMSIZ: DEFW    F168H

Esta variável contém o endereço do topo da Área de Armazenamento de Strings. Seu valor é estabelecido na partida e alterado exclusivamente pelas instruções "CLEAR" e "MAXFILES".

F674H    STKTOP: DEFW    F0AOH

Esta variável contém o endereço do topo da pilha do Z80. O seu valor é estabelecido na partida com MEMSIZ-200 e alterado exclusivamente pelas instruções "CLEAR" e "MAXFILES".

F676H    TXTTAB: DEFW    8001H

Esta variável contém o endereço do primeiro byte da Área de Texto do Programa. O seu valor é estabelecido na partida e permanece inalterado.



F678H TEMPPT: DEFW F67AH

Esta variável contém o endereço da próxima posição livre em TEMPST.

F67AH TEMPST: DEFS 30

Este buffer é utilizado para armazenar descritores de string. Funciona como uma pilha onde os produtores da string empilham seus resultados e de onde os consumidores da string as retiram.

F698H DSCTMP: DEFS 3

Este buffer é utilizado pelas funções string para guardar um descritor de resultado, enquanto estiver sendo construído.

F69BH FRETOP: DEFW F168H

Esta variável contém o endereço da próxima posição livre na Área de Armazenamento de Strings. Quando a área estiver vazia FRETOP será igual a MEMSIZ.

F69DH TEMP3: DEFW 0000H

Esta variável é utilizada para armazenamento temporário por várias partes do Interpretador.

F69FH TEMP8: DEFW 0000H

Esta variável é utilizada para armazenamento temporário por diversas partes do Interpretador.

F6A1H ENDFOR: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "FOR" para guardar o endereço do fim da instrução, durante a construção de um bloco de parâmetros.

F6A3H DATLIN: DEFW 0000H

Esta variável contém o número da linha do programa onde está o item "DATA" atual.

F6A5H SUBFLG: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida pelos manipuladores “ERASE”, “FOR” e “DEF FN” para controlar o processamento de índices da rotina de busca de variáveis.

F6A6H FLGINP: DEFB 00H

Esta variável contém um indicador para distinguir as duas instruções do manipulador das instruções “READ/INPUT”: 00H=INPUT, NZ=READ.

F6A7H TEMP: DEFW 0000H

Esta variável é utilizada para armazenamento temporário por diversas partes do Interpretador.

F6A9H PTRFLG: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida se qualquer operando de número de linha na Área de Texto do Programa for convertido em apontadores.

F6AAH AUTFLG: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida quando o modo “AUTO” é ligado.

F6ABH AUTLIN: DEFW 0000H

Esta variável contém o número da linha “AUTO” atual.

F6ADH AUTINC: DEFW 0000H

Esta variável contém o incremento atual do número de linha “AUTO”.

F6AFH SAVTXT: DEFW 0000H

Esta variável é atualizada pela Ronda de Execução, no início de cada instrução, com a posição atual no texto do programa. É utilizada durante a recuperação de erro para

atualizar ERRTXT para o manipulador da declaração "RESUME" e OLDTXT para o manipulador da declaração "CONT".

F6B1H SAVSTK: DEFW F09EH

Esta variável é atualizada pela Ronda de Execução, no início de cada instrução com o SP atual visando a recuperação de erro.

F6B3H ERRLIN: DEFW 0000H

Esta variável é utilizada pelo manipulador de erro para guardar o número da linha do programa que gerou o erro.

F6B5H DOT: DEFW 0000H

Esta variável é atualizada pela Ronda Principal e pelo manipulador de erro com o número de linha atual, para ser utilizada com o parâmetro ".".

F6B7H ERRTXT: DEFW 0000H

Esta variável é atualizada de SAVTXT pelo manipulador de erro, para ser utilizada pelo manipulador da instrução "RESUME".

F6B9H ONELIN: DEFW 0000H

Esta variável é estabelecida pelo manipulador da instrução "ON ERROR GOTO" com o endereço da linha do programa que deve ser executada quando ocorre um erro.

F6BBH ONEFLG: DEFB 00H

Esta variável é normalmente zero, mas é estabelecida pelo manipulador de erro quando o controle é transferido para uma instrução “ON ERROR GOTO”. Isto visa evitar que um loop seja criado, se outro erro ocorrer dentro das próprias instruções de tratamento de erro.

F6BCH TEMP2: DEFW 0000H

Esta variável é utilizada para armazenamento temporário por diversas partes do Interpretador.

F6BEH OLDLIN: DEFW 00000H

Esta variável contém o número da linha terminadora do programa. Ela é atualizada pelos manipuladores das instruções “END” e “STOP” para ser utilizado com a instrução “CONT”.

F6COH OLDTXT: DEFW 0000H

Esta variável contém o endereço da instrução terminadora do programa.

F6C2H VARTAB: DEFW 8003H

Esta variável contém o endereço do primeiro byte da Área de Armazenamento de Variáveis.

F6C4H ARYTAB: DEFW 8003H

Esta variável contém o endereço do primeiro byte da Área de Armazenamento de Matrizes.

F6C6H STREND: DEFW 0803H

Esta variável contém o endereço do byte, após a Área de Armazenamento de Matrizes.

F6C8H    DATPTR:   DEFW   8000H

Esta variável contém o endereço do item "DATA" atual no texto do programa.

|       |         |      |     |    |
|-------|---------|------|-----|----|
| F6CAH | DEFTBL: | DEFB | 08H | ;A |
| F6CBH |         | DEFB | 08H | ;B |
| F6CCH |         | DEFB | 08H | ;C |
| F6CDH |         | DEFB | 08H | ;D |
| F6CEH |         | DEFB | 08H | ;E |
| F6CFH |         | DEFB | 08H | ;F |
| F6D0H |         | DEFB | 08H | ;G |
| F6D1H |         | DEFB | 08H | ;H |
| F6D2H |         | DEFB | 08H | ;I |
| F6D3H |         | DEFB | 08H | ;J |
| F6D4H |         | DEFB | 08H | ;K |
| F6D5H |         | DEFB | 08H | ;L |
| F6D6H |         | DEFB | 08H | ;M |
| F6D7H |         | DEFB | 08H | ;N |
| F6D8H |         | DEFB | 08H | ;O |
| F6D9H |         | DEFB | 08H | ;P |
| F6DAH |         | DEFB | 08H | ;Q |
| F6DBH |         | DEFB | 08H | ;R |
| F6DCH |         | DEFB | 08H | ;S |
| F6DDH |         | DEFB | 08H | ;T |
| F6DEH |         | DEFB | 08H | ;U |
| F6DFH |         | DEFB | 08H | ;V |
| F6E0H |         | DEFB | 08H | ;W |
| F6E1H |         | DEFB | 08H | ;X |
| F6E2H |         | DEFB | 08H | ;Y |
| F6E3H |         | DEFB | 08H | ;Z |

Estas vinte e seis variáveis contêm o tipo default para cada grupo de variáveis BASIC. Os seus valores são atribuídos com "dupla precisão" na partida, "NEW" e "CLEAR" e alterados unicamente pelo grupo de instruções "DEF".

F6E4H    PRMSTK:   DEFW   0000H

Esta variável contém o endereço base do bloco de parâmetro "FN", anterior na

pilha do Z80. É utilizada durante a coleta de lixo string para percorrer bloco a bloco na pilha.

F6E6H PRMLN: DEFW 0000H

Esta variável contém o comprimento do bloco de parâmetro "FN" atual em PARM1.

F6E8H PARM1: DEFS 100

Este buffer contém as Variáveis locais pertencentes à função "FN", que atualmente está sendo avaliada.

F74CH PRMPRV: DEFW F6E4H

Esta variável contém o endereço do bloco de parâmetro "FN" anterior. Na realidade é uma constante utilizada para garantir que a coleta do lixo string inicie com o bloco de parâmetro atual, antes de prosseguir aqueles da pilha.

F74EH PRMLN2: DEFW 0000H

Esta variável contém o comprimento do bloco de parâmetros "FN", que está sendo construído em PARM2.

F750H PARM2: DEFS 100

Este buffer é utilizado para construir as Variáveis locais pertencentes à função "FN" atual.

F7B4H PRMFLG: DEFB 00H

Esta variável é utilizada durante uma busca de Variáveis para indicar se Variáveis locais ou globais estão sendo examinadas.

F7B5H ARYTA2: DEFW 0000H

Esta variável é utilizada durante uma busca de Variável para conter o último endereço da área de armazenamento, que está sendo examinada.

F7B7H NOFUNS: DEFB 00H

Esta variável normalmente é zero, mas é estabelecida pelo manipulador da função “FN” para avisar à rotina de busca de variáveis que há Variáveis locais.

F7B8H TEMP9: DEFW 0000H

Esta variável é utilizada para armazenamento temporário por diversas partes do Interpretador.

F7BAH FUNACT: DEFW 0000H

Esta variável contém o número de funções “FN” atualmente ativas.

F7BCH SWPTMP: DEFS 8

Este buffer é utilizado para conter o primeiro operando de uma instrução “SWAP”.

F7C4H TRCFLG: DEFB 00H

Esta variável é normalmente zero, mas é ligada pelo manipulador da declaração “TRON” para ativar a facilidade de rastreamento (trace).

F7C5H FBUFFR: DEFS 43

Este buffer é utilizado para conter o texto produzido durante uma conversão de saída numérica.

F7F0H DECTMP: DEFW 0000H

Esta variável é utilizada para armazenamento temporário pela rotina de divisão de dupla precisão.

F7F2H DECTM2: DEFW 0000H

Esta variável é utilizada para armazenamento temporário pela rotina de divisão de dupla precisão.

F7F4H DECCNT: DEFB 00H

Esta variável é utilizada pela rotina de divisão de dupla precisão, para conter o número de bytes não-zero na mantissa do segundo operando.

F7F6H DAC: DEFS 16

Este buffer funciona como o acumulador primário do Interpretador, durante a avaliação de uma expressão.

F806H HOLD8: DEFS 65

Este buffer é utilizado pela rotina de multiplicação de dupla precisão, para conter os múltiplos do primeiro operando.

F847H ARG: DEFS 16

Este buffer funciona como o acumulador secundário do Interpretador, durante a avaliação de uma expressão.

F857H RNDX: DEFS 8

Este buffer contém o atual número aleatório de precisão dupla.

F85FH MAXFIL: DEFB 01H

Esta variável contém o número de buffers de E/S de usuário atualmente alocados. Seu valor é colocado em 1 na partida e alterado exclusivamente pela declaração "MAXFILES".

F860H FILTAB: DEFW F16AH

Esta variável contém o endereço da tabela de apontadores para os FCBs dos buffers de E/S.

F862H NULBUF: DEFW F177H

Esta variável contém o endereço do primeiro byte do buffer de dados pertencente ao buffer de E/S.



F864H PTRFIL: DEFW 0000H

Esta variável contém o endereço do FCB do buffer de E/S atualmente ativo.

F866H FILNAM: DEFS 11

Este buffer contém um nome de arquivo especificado pelo usuário. Tem um tamanho de onze caracteres, para permitir especificação de arquivo em disco como "ARQUIVOS BAS".

F871H FILNM2: DEFS 11

Este buffer contém um nome de arquivo lido de um dispositivo de E/S para ser comparado com o conteúdo de FILNAM.

F87CH NLONLY: DEFB 00H

Esta variável normalmente é zero, mas é ativada durante um programa "LOAD". O Bit 0 é utilizado para impedir que o buffer 0 de E/S seja fechado durante o carregamento e o bit 7 utilizado para impedir que os buffers de E/S do usuário sejam fechados, caso um autoprocessamento seja solicitado.

F87DH SAVEND: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "BSAVE" para conter o endereço final do bloco de memória que deve ser salvo.

F87FH FNKSTR: DEFS 160

Este buffer contém as dez strings das teclas de função de dezesseis caracteres. Os seus valores são estabelecidos na partida e alterados pela instrução "KEY".

|       |        |      |       |                            |
|-------|--------|------|-------|----------------------------|
| F91FH | CGPNT: | DEFB | 00H   | ;Identificação do conector |
| F920H |        | DEFW | 1BBFH | ;Endereço                  |

Estas variáveis contêm a posição do conjunto de caracteres copiados no VDP pelas rotinas padrões INITXT e INIT32. Os seus valores são apontados para o conjunto de caracteres no ROM do MSX na partida e permanecem inalterados.

F992H    NAMBAS: DEFW    0000H

Esta variável contém o endereço de base da Tabela de Nomes do VDP do Modo Texto atual. O seu valor é colocado de TXTNAM ou T32NAM, sempre que VDP for inicializado em um Modo Texto, por meio das rotinas padrões INITXT ou INIT32.

F924H    CGPBAS: DEFW    0800H

Esta variável contém o endereço de base da Tabela de Imagens de Caracteres do VDP do modo texto atual. O seu valor é colocado de TXTCGP ou T32CGP, sempre que o VDP for inicializado em um modo texto, por meio das rotinas padrões INITXT ou INIT32.

F926H    PATBAS: DEFW    3800H

Esta variável contém o endereço de base da Tabela de Imagens de Sprites do VDP. Seu valor é colocado de T32PAT, GRPPAT ou MLTPAT, sempre que o VDP for inicializado por meio das rotinas padrões INIT32, INIGRP ou INIMLT.

F928H    ATRBAS: DEFW    1B00H

Esta variável contém o endereço de base da Tabela de Atributos de Sprites do VDP. O seu valor é colocado de T32ATR, GRPATR ou MLTATR, sempre que o VDP for inicializado por meio das rotinas padrões INIT32, INIGRP ou INIMLT.

|       |        |      |       |                       |
|-------|--------|------|-------|-----------------------|
| F92AH | CLOC:  | DEFW | 0000H | ;Localização do Pixel |
| F92CH | CMASK: | DEFB | 80H   | ;Máscara do Pixel     |

Estas variáveis contém o endereço físico do pixel atual, utilizado pelas rotinas padrões RIGHTC, LEFTC, UPC, TUPC, DOWNC, TDOWNC, FETCHC, STOREC, READC, SETC, NSETCX, SCANR e SCANL. CLOC contém o endereço do byte contendo o pixel atual e CMASK define o pixel dentro do byte.

F92DH    MINDEL: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "LINE" para conter a diferença mínima entre os pontos extremos da linha.

F92FH    MAXDEL: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "LINE" para conter a diferença máxima entre os pontos terminais da linha.

F931H    ASPECT: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE" para conter a razão atual. Esta é armazenada como uma fração binária em um único byte, de modo que uma razão de aspecto de 0.75 ficaria 00C0H. O MSB será necessário apenas, se a razão de aspecto for exatamente 1.00, isto é 0100H.

F933H    CENCNT: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE" para conter a contagem de pontos do ângulo final.

F935H    CLINEF: DEFB    00H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE" para conter os dois indicadores de linha. O Bit 0 será ligado, se uma linha for requerida a partir do ângulo inicial ao centro e o bit 7 será ligado, caso uma linha seja requerida a partir do ângulo final.

F936H    CNPNTS: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE" para conter o número de pontos dentro de um segmento de quarenta e cinco graus.

F938H    CPLOTF: DEFB    00H

Esta variável é normalmente zero, mas é ligada pelo manipulador da instrução "CIRCLE" se o ângulo final for menor do que o ângulo inicial. É utilizada para determinar se os pixels devem ser colocados "dentro" ou "fora" dos ângulos.

F939H    CPCNT: DEFW    0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE", para conter

a contagem de pontos dentro do segmento atual de quarenta e cinco graus, sendo de fato a coordenada Y.

F93BH CPCNT8: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE", para conter a contagem total de pontos da posição atual.

F93DH CRCSUM: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE", como contador de computação de pontos.

F93FH CSTCNT: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE", para conter a contagem de pontos do ângulo inicial.

F941H CSCLXY: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "CIRCLE" como um indicador, para determinar em que direção a compressão elíptica deve ser feita: 00H=Y, 01H=X.

F942H CSAVEA: DEFW 0000H

Esta variável é utilizada para armazenamento temporário pela rotina padrão SCANR.

F944H CSAVEM: DEFB 00H

Esta variável é utilizada para armazenamento temporário pela rotina padrão SCANR.

F945H CXOFF: DEFW 0000H

Esta variável é utilizada para armazenamento temporário pelo manipulador da instrução "CIRCLE".

F947H CYOFF: DEFW 0000H

Esta variável é utilizada para armazenamento temporário pelo manipulador da instrução "CIRCLE".

F949H LOHMSK: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter a posição mais à esquerda da excursão LH.

F94AH LOHDIR: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter a nova direção de pintura requerida pela excursão LH.

F94BH LOHADR: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter a posição mais à esquerda de uma excursão LH.

F94DH LOHCNT: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter o tamanho da excursão LH.

F94FH SKPCNT: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter a contagem de pulso devolvida pela rotina padrão SCANR.

F951H MOVCNT: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter a contagem de movimento devolvida pela rotina padrão SCANR.

F953H PDIREC: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "PAINT", para conter o sentido da pintura atual: 40H=Para baixo, COH=Para cima, 00H=Terminar.

F954H    LEPROG: DEFB    00H

Esta variável é normalmente zero, mas é ligada pelo manipulador da instrução “PAINT” se houver qualquer progresso para à esquerda.

F955H    RTPROG: DEFB    00H

Esta variável é normalmente zero, mas é ligada pelo manipulador da instrução “PAINT” se houver algum progresso para à direita.

F956H    MCLTAB: DEFW    0000H

Esta variável contém o endereço da tabela de comando que deve ser utilizada pelo analisador sintático de linguagem macro. A tabela “DRAW” está em 5D83H e a tabela “PLAY” está em 752EH.

F958H    MCLFLG: DEFB    00H

Esta variável é zero se o analisador sintático de linguagem macro estiver sendo utilizado pelo manipulador da instrução “DRAW”, e não-zero se estiver sendo utilizado pelo manipulador da instrução “PLAY”.

|       |              |       |                             |
|-------|--------------|-------|-----------------------------|
| F959H | QUETAB: DEFB | 00H   | ;AQ Posição para colocar    |
| F95AH | DEFB         | 00H   | ;AQ Posição para pegar      |
| F95BH | DEFB         | 00H   | ;AQ Indicação para devolver |
| F95CH | DEFB         | 7FH   | ;AQ Tamanho                 |
| F95DH | DEFW         | F975H | ;AQ Endereço                |

|       |      |       |                             |
|-------|------|-------|-----------------------------|
| F95FH | DEFB | 00H   | ;BQ Posição para colocar    |
| F960H | DEFB | 00H   | ;BQ Posição para pegar      |
| F961H | DEFB | 00H   | ;BQ Indicação para devolver |
| F962H | DEFB | 7FH   | ;BQ Tamanho                 |
| F963H | DEFW | F9F5H | ;BQ Endereço                |

|       |      |       |                             |
|-------|------|-------|-----------------------------|
| F965H | DEFB | 00H   | ;CQ Posição para colocar    |
| F966H | DEFB | 00H   | ;CQ Posição para pegar      |
| F967H | DEFB | 00H   | ;CQ Indicação para devolver |
| F968H | DEFB | 7FH   | ;CQ Tamanho                 |
| F969H | DEFW | FA75H | ;CQ Endereço                |

|       |      |       |                             |
|-------|------|-------|-----------------------------|
| F969H | DEFB | 00H   | ;RQ Posição para colocar    |
| F96CH | DEFB | 00H   | ;RQ Posição para pegar      |
| F96DH | DEFB | 00H   | ;RQ Indicação para devolver |
| F96EH | DEFB | 00H   | ;RQ Tamanho                 |
| F96FH | DEFW | 0000H | ;RQ Endereço                |

Estas vinte e quatro variáveis formam os blocos de controle para as três filas musicais (VOICAQ, VOICBQ e VOICCQ) e a fila da RS232. Os três blocos de controle musical são inicializados pela rotina padrão GICINI e, a partir dali, mantidos pelo manipulador de arquivo e pela rotina padrão PUTQ. O bloco de controle RS232 não é utilizado na ROM MSX atual.

|       |              |     |                            |
|-------|--------------|-----|----------------------------|
| F971H | QUEBAK: DEFB | 00H | ;AQ caractere de devolução |
| F972H | DEFB         | 00H | ;BQ caractere de devolução |
| F973H | DEFB         | 00H | ;CQ caractere de devolução |
| F974H | DEFB         | 00H | ;RQ caractere de devolução |

Estas quatro variáveis são utilizadas para manter qualquer caractere indesejável devolvido a uma fila associada. Apesar da facilidade de devolução estar implementada na ROM do MSX, atualmente, não é utilizada.

|       |              |     |                |
|-------|--------------|-----|----------------|
| F975H | VOICAQ: DEFS | 128 | ;Fila da voz A |
| F975H | VOICQB: DEFS | 128 | ;Fila da voz B |
| FA75H | VOICCQ: DEFS | 128 | ;Fila da voz C |
| FAF5H | RS2IQ: DEFS  | 64  | ;Fila RS232    |

Estes quatro buffers contêm as três filas musicais e a fila RS232. A última não é utilizada.

FB35H PRSCNT: DEFB 00H

Esta variável é utilizada pelo manipulador da instrução "PLAY" para contar o número de strings de operando completados. O Bit 7 também será ligado, após cada um dos três operandos ter sido analisado, para impedir uma ativação repetida da rotina padrão STRTMS.

FB36H SAVSP: DEFW 0000H

Esta variável é utilizada pelo manipulador da instrução "PLAY", para salvar o SP do Z80 antes do controle ser transferido para o analisador sintático de linguagem

macro. O seu valor é comparado com o SP no retorno, para determinar se qualquer dado foi deixado na pilha devido a uma terminação de fila-cheia pelo analisador sintático.

FB38H VOICEN: DEFB 00H

Esta variável contém o número da voz atual que está sendo processado pelo manipulador da instrução "PLAY". Os valores 0, 1 e 2 correspondem aos canais A, B e C do PSG.

FB39H SAVVOL: DEFW 0000H

Esta variável é utilizada pelo manipulador do comando "R" da instrução "PLAY" para salvar o volume atual, enquanto uma causa de amplitude zero é gerada.

FB3BH MCLLEN: DEFB 00H

Esta variável é utilizada pelo analisador sintático de linguagem macro, para conter o comprimento do operando string que está sendo analisado.

FB3CH MCLPTR: DEFW 0000H

Esta variável é utilizada pelo analisador sintático de linguagem macro, para conter o endereço do operando string que está sendo analisado.

FB3EH QUEUEN: DEFB 00H

Esta variável é utilizada pelo manipulador de interrupção, para conter o número da fila musical, atualmente, sendo processada. Os valores 0, 1 e 2 correspondem aos canais A, B e C do PSG.

FB3FH MUSICF: DEFB 00H

Esta variável contém três bits indicadores ligados pela rotina padrão STRTMS, para iniciar o processamento de uma fila musical pelo manipulador de interrupção. Os bits 0, 1 e 2 correspondem a VOICAQ, VOICBQ e VOICCQ.

FB40H PLACNT: DEFB 00H

Esta variável é utilizada pela rotina padrão STRTMS, para conter o número das



seqüências da instrução “PLAY”, atualmente mantidas na filas musicais. Ela é examinada quando todas as três marcas de fim de fila forem encontradas para uma seqüência, a fim de determinar se a eliminação de fila deve ser reiniciada.

|       |       |      |       |                             |
|-------|-------|------|-------|-----------------------------|
| FB41H | VCBA: | DEFW | 0000H | ;Contador de duração        |
| FB43H |       | DEFB | 00H   | ;Comprimento da string      |
| FB44H |       | DEFW | 0000H | ;Endereço da string         |
| FB46H |       | DEFW | 0000H | ;Endereço de dados da pilha |
| FB48H |       | DEFB | 00H   | ;Tamanho do pacote musical  |
| FB49H |       | DEFS | 7     | ;Pacote musical             |
| FB50H |       | DEFB | 04H   | ;Oitava                     |
| FB51H |       | DEFB | 04H   | ;Comprimento                |
| FB52H |       | DEFB | 78H   | ;Tempo                      |
| FB53H |       | DEFB | 88H   | ;Volume                     |
| FB54H |       | DEFW | 00FFH | ;Período da envoltória      |
| FB56H |       | DEFS | 16    | ;Espaço para dados da pilha |

Este buffer de trinta e sete bytes é utilizado pelo manipulador da instrução “PLAY”, para conter os parâmetros atuais para a voz A.

FB66H VCBB: DEFS 37

Este buffer é utilizado pelo manipulador da instrução “PLAY”, para conter os parâmetros atuais para a voz B, com uma estrutura igual a VCBA.

FB8BH VCBC: DEFS 37

Este buffer é utilizado pelo manipulador da instrução “PLAY”, para conter os parâmetros atuais para a voz C, com uma estrutura igual a VCBA.

FBB0H ENSTOP: DEFB 00H

Esta variável determina se o manipulador de interrupção executará uma partida aquecida para o Interpretador, ao detectar as teclas CODE, GRPH, CTRL e SHIFT operadas juntas: 00H=Desativado, NZ=Ativado.

FBB1H BASROM: DEFB 00H

Esta variável determina se as rotinas padrões ISCNTC e INLIN responderão à

tecla CTRL+STOP: 00H=Ativar, NZ=Desativar. Ela é utilizada para evitar a conclusão de uma ROM em BASIC, localizada durante a pesquisa de ROM na partida.

FBB2H LINTTB: DEFS 24

Cada uma destas vinte e quatro variáveis é normalmente não-zero, mas será zerada se o conteúdo da linha da tela correspondente avançou na linha seguinte. São atualizadas pelo BIOS, mas apenas, realmente, utilizadas pela rotina padrão INLIN (o editor de tela) para discriminar entre linhas físicas e lógicas.

FBCAH FSTPOS: DEFW 0000H

Esta variável é utilizada para manter as coordenadas do cursor, durante a entrada para a rotina padrão INLIN. A sua função é a de restringir a extensão do retrocesso executado, quando o texto é coletado da tela na terminação.

FBCCH CURSAV: DEFB 00H

Esta variável é utilizada para conter o caractere de tela substituído pelo cursor de texto.

FBCDH FNKSWI: DEFB 00H

Esta variável é utilizada pela rotina padrão CHSNS para determinar se as teclas de função com "SHIFT" estão sendo atualmente apresentadas ou não: 00H=Com SHIFT, 01H=Sem SHIFT.

FBCEH FNKFLG: DEFS 10

Cada uma destas dez variáveis normalmente é zero, mas será colocada em 01H se a tecla de função associada foi ligada por uma instrução "KEY(n) ON". São utilizadas pelo manipulador de interrupção para determinar se, apenas no modo de programa, deve ser devolvido uma string, ou se a entrada correspondente de TRPTBL deve ser atualizada.

FBD8H ONGSBF: DEFB 00H

Esta variável normalmente é zero, mas será incrementada pelo manipulador de interrupção sempre que um dispositivo adquiriu a condição necessária para gerar uma

interrupção de programa. É utilizada pela Ronda de Execução para determinar se alguma interrupção de programa está pendente, sem precisar vasculhar TRPTBL.

FBD9H CLIKFL: DEFB 00H

Esta variável é utilizada internamente pelo manipulador de interrupção para impedir clicks de tecla empúrios, ao devolver múltiplos caracteres gerador por uma única tecla, com uma tecla de função.

FBD4H OLDKEY: DEFS 11

Este buffer é utilizado pelo manipulador de interrupção para guardar o estado anterior da matriz do teclado, em que cada byte contém uma linha de teclas iniciando com a linha 0.

FB5H NEWKEY: DEFS 11

Este buffer é utilizado pelo manipulador de interrupção para guardar o estado atual da matriz do teclado. As transições das teclas são detectadas por comparação com o conteúdo de OLDKEY. Depois ONDKEY é atualizado com o estado atual.

FBF0H KEYBUF: DEFS 40

Este buffer contém os caracteres de tecla do decodificados, produzidos pelo manipulador de interrupção. Observe que o buffer é organizado como uma fila circular acionado por GETPNT e PUTPNT e, conseqüentemente, não tem nenhum ponto de origem fixo.

FC18H LINWRK: DEFS 40

Este buffer é utilizado pelo BIOS, para conter uma linha completa de caracteres de tela.

FC40H PATWRK: DEFS 8

Este buffer é utilizado por BIOS, para conter uma imagem de pixel 8x8.

FC48H    BOTTOM: DEFW    8000H

Esta variável contém o endereço da posição inferior da RAM utilizada pelo Interpretador. Seu valor é colocado na partida e permanece inalterado.

FC4AH    HIMEN:    DEFW    F380H

Esta variável contém o endereço do byte, após a posição mais alta da RAM utilizada pelo Interpretador. O seu valor é estabelecido na partida e alterado exclusivamente pela instrução "CLEAR".

|       |         |      |   |            |
|-------|---------|------|---|------------|
| FC4CH | TRPTBL: | DEFS | 3 | ;KEY 1     |
| FC4FH |         | DEFS | 3 | ;KEY 2     |
| FC52H |         | DEFS | 3 | ;KEY 3     |
| FC55H |         | DEFS | 3 | ;KEY 4     |
| FC58H |         | DEFS | 3 | ;KEY 5     |
| FC5BH |         | DEFS | 3 | ;KEY 6     |
| FC5EH |         | DEFS | 3 | ;KEY 7     |
| FC61H |         | DEFS | 3 | ;KEY 8     |
| FC64H |         | DEFS | 3 | ;KEY 9     |
| FC67H |         | DEFS | 3 | ;KEY 10    |
| FC6AH |         | DEFS | 3 | ;STOP      |
| FC6DH |         | DEFS | 3 | ;SPRITE    |
| FC70H |         | DEFS | 3 | ;STRIG 0   |
| FC73H |         | DEFS | 3 | ;STRIG 1   |
| FC76H |         | DEFS | 3 | ;STRIG 2   |
| FC79H |         | DEFS | 3 | ;STRIG 3   |
| FC7CH |         | DEFS | 3 | ;STRIG 4   |
| FC7FH |         | DEFS | 3 | ;INTERVAL  |
| FC82H |         | DEFS | 3 | ;Não Usado |
| FC85H |         | DEFS | 3 | ;Não Usado |
| FC88H |         | DEFS | 3 | ;Não Usado |
| FC8BH |         | DEFS | 3 | ;Não Usado |
| FC8EH |         | DEFS | 3 | ;Não Usado |
| FC91H |         | DEFS | 3 | ;Não Usado |
| FC94H |         | DEFS | 3 | ;Não Usado |
| FC97H |         | DEFS | 3 | ;Não Usado |

Estas vinte e seis variáveis de três bytes contém o estado atual dos dispositivos

de geração de interrupção. O primeiro byte de cada entrada contém o status do dispositivo (bit 0 = Ligado, bit 1 = Parado, bit 2 = Evento ativo) e é atualizado pelo manipulador de interrupção, pelo processador de interrupção da Ronda de Execução e pelos manipuladores das instruções “DISPOSITIVO ON/OFF/STOP” e “RETURN”. Os dois bytes restantes de cada entrada são ativados pelo manipulador da instrução “ON DISPOSITIVO GOSUB” e contém o endereço da linha de programa a ser executada, em uma interrupção do programa.

```
FC9AH  RTYCNT: DEFB  00H
```

Esta variável não é utilizada pela ROM do MSX atual.

```
FC9BH  INTFLG: DEFB  00H
```

Esta variável é normalmente zero, mas é colocada em 03H ou 04H se a tecla CTRL-STOP ou STOP são detectados pelo manipulador de interrupção.

```
FC9CH  PADY:   DEFB  00H
```

Esta variável contém a coordenada Y do último ponto detectado pelo “touchpad”.

```
FC9DH  PADX:   DEFB  00H
```

Esta variável contém uma coordenada X do último ponto detectores pelo “touchpad”.

```
FC9EH  JIFFY:   DEFW  0000H
```

Esta variável é continuamente incrementada pelo manipulador de interrupção. Seu valor pode ser atribuído ou lido pela função ou instrução “TIME”.

```
FCA0H  INTVAL: DEFW  0000H
```

Esta variável contém a duração do intervalo estabelecido pelo manipulador da instrução “ON INTERVAL”.

FCA2H INTCNT: DEFW 0000H

Esta variável é continuamente decrementada pelo manipulador de interrupção. Quando o zero for atingido, seu valor será repostado de INTVAL e, se aplicável, será gerada uma interrupção de programa. Observe que esta variável sempre conta independente de uma instrução "INTERVAL ON" estar ativa.

FCA4H LOWLIM: DEFB 31H

Esta variável é utilizada para conter a duração mínima permissível para o bit de partida, conforme determinado pela rotina padrão TAPION.

FCA5H WINWID: DEFB 22H

Esta variável é utilizada para conter a duração de discriminação do ciclo LO/HI, conforme determinado pela rotina padrão TAPION.

FCA6H GRPHED: DEFB 00H

Esta variável normalmente é zero, mas é colocada em 01H pela rotina padrão CNVCHR ao detectar um código de cabeçalho gráfico.

FCA7H ESCCNT: DEFB 00H

Esta variável é utilizada pelo processador de seqüências ESC da rotina padrão CHPUT, para contar parâmetros de escape.

FCA8H INSFLG: DEFB 00H

Esta variável é normalmente zero, mas é colocada em FFH pela rotina padrão INLIN, quando o modo de inserção está ligado.

FCA9H CSRSW: DEFB 00H

Se esta variável for zero o cursor será apresentado apenas enquanto a rotina padrão CHGET está esperando por uma caractere de teclado. Se for não-zero o cursor será permanentemente apresentado, por meio da rotina padrão CHPUT.

FCAAH CSTYLE: DEFB 00H

Esta variável determina o estilo do cursor: 00H = Bloco, NZ = Sub-alinhamento.

FCABH CAPST: DEFB 00H

Esta variável é utilizada pelo manipulador de interrupção para conter o status atual do caps lock: 00H = Desligado, NZ = Ligado.

FCACH KANAST: DEFB 00H

Esta variável é utilizada para guardar o status do “Kana Lock” do teclado das máquinas japonesas e o status da tecla DED em máquinas européias.

FCADH KANAMD: DEFB 00H

Esta variável contém um modo de teclado apenas em máquinas japonesas.

FCAEH FLBMEM: DEFB 00H

Esta variável é usada apenas pelos geradores de erro de E/S de arquivo.

FCAFH SCRMOD: DEFB 00H

Esta variável contém o modo de tela atual: Modo de Texto 0 = 40x24, Modo Texto 1 = 32x24, Modo Gráfico = 2 e Modo Multicolorido = 3.

FCB0H OLDSCR: DEFB 00H

Esta variável contém o modo de tela do último conjunto de modo texto.

FCB1H CASPRV: DEFB 00H

Esta variável é utilizada para manter qualquer caractere devolvido a um buffer de E/S pela função de devolução do cassete.

FCB2H BDRATR: DEFB 00H

Esta variável contém a cor de contorno para o manipulador da instrução "PAINT". Seu valor é colocado pela rotina padrão PNTINI e utilizado pelas rotinas padrões SCANR e SCANL.

FCB3H GXPOS: DEFW 0000H

Esta variável é utilizada para armazenamento temporário de uma coordenada X gráfica.

FCB5H GYPÓS: DEFW 0000H

Esta variável é utilizada para armazenamento temporário de uma coordenada Y gráfica.

FCB7H GRPACX: DEFW 0000H

Esta variável contém a coordenada X gráfica atual para a rotina padrão GRPPRT.

FCB9H GRPACY: DEFW 0000H

Esta variável contém a coordenada Y gráfica atual para a rotina padrão GRPPRT.

FCBBH DRWFLG: DEFB 00H

Os bits 6 e 7 desta variável são usadas pelos manipuladores "N" e "B" da instrução "DRAW" para ligar o modo associado.

FCBCH DRWSCL: DEFB 00H

Esta variável é utilizada pelo manipulador do comando "S" da instrução "DRAW" para conter o fator de escala atual.

FCBDH DRWANG: DEFB 00H

Esta variável é utilizada pelo manipulador do comando "A" da instrução "DRAW" para conter o ângulo atual.



```
FCBEH  RUNBNF: DEFW  00H
```

Esta variável normalmente é zero, mas será ligada pelo manipulador da declaração “BLOAD” quando for especificado um parâmetro “R” de autoprocessamento.

```
FCBFH  SAVENT: DEFW  0000H
```

Esta variável contém os endereços do ponto de entrada para “BSAVE” e “BLOAD”.

```
FCC1H  EXPTBL: DEFB  00H      ;Conector primário 0
FCC2H           DEFB  00H      ;Conector primário 1
FCC3H           DEFB  00H      ;Conector primário 2
FCC4H           DEFB  00H      ;Conector primário 3
```

Cada uma destas quatro variáveis normalmente é zero, mas serão colocadas em 80H, durante a busca da RAM na partida, se o Conector Primário associado estiver expandido.

```
FCC5H  SLTTBL: DEFB  00H      ;Conector primário 0
FCC6H           DEFB  00H      ;Conector primário 1
FCC7H           DEFB  00H      ;Conector primário 2
FCC8H           DEFB  00H      ;Conector primário 3
```

Estas quatro variáveis duplicam o conteúdo dos quatro possíveis Registradores de Conector Primário. O conteúdo de cada variável deveria apenas ser visto como válido, se EXPTBL mostrar o Conector Primário associado como sendo expandido.

|       |         |      |   |           |
|-------|---------|------|---|-----------|
| FCC9H | SLTATR: | DEFS | 4 | ;PS0, SS0 |
| FCCDH |         | DEFS | 4 | ;PS0, SS1 |
| FCD1H |         | DEFS | 4 | ;PS0, SS2 |
| FCD5H |         | DEFS | 4 | ;PS0, SS3 |
|       |         |      |   |           |
| FCD9H |         | DEFS | 4 | ;PS1, SS0 |
| FCDDH |         | DEFS | 4 | ;PS1, SS1 |
| FCE1H |         | DEFS | 4 | ;PS1, SS2 |
| FCE5H |         | DEFS | 4 | ;PS1, SS3 |
|       |         |      |   |           |
| FCE9H |         | DEFS | 4 | ;PS2, SS0 |
| FCEDH |         | DEFS | 4 | ;PS2, SS1 |
| FCF1H |         | DEFS | 4 | ;PS2, SS2 |
| FCF5H |         | DEFS | 4 | ;PS2, SS3 |
|       |         |      |   |           |
| FCF9H |         | DEFS | 4 | ;PS3, SS0 |
| FCFDH |         | DEFS | 4 | ;PS3, SS1 |
| FD01H |         | DEFS | 4 | ;PS3, SS2 |
| FD05H |         | DEFS | 4 | ;PS3, SS3 |

Estas sessenta e quatro variáveis contêm os atributos de quaisquer ROM de extensão encontrada durante a procura de ROM na partida. As características de cada ROM de 16KB são codificadas em um único byte, de modo que quatro bytes são requeridos para cada conector possível. A codificação é:

Bit 7 ativado = Programa BASIC

Bit 6 ativado = Manipulador de dispositivo

Bit 5 ativado = Manipulador de instrução

Observe que as entradas para a página 0 (0000H até 3FFFH) e a página 3 (C000H até FFFFH) serão sempre zero, pois apenas a página 1 (4000H até 7FFFH) e a página 2 (8000H até BFFFH) são, de fato, examinadas. A conversão do MSX é a de que ROMs de extensão em código de máquina são colocadas na página 1 e ROMs de programa BASIC na página 2.

FDO9H SLTWRK: DEFS 128

Este buffer fornece dois bytes de espaço de trabalho local para cada uma das sessenta e quatro extensões de ROM possíveis.

FD89H PROCNM: DEFS 16

Este buffer é utilizado para conter um dispositivo ou um nome de instrução, para ser examinado por uma ROM de extensão.

FD99H DEVICE: DEFB 00H

Esta variável é utilizada para passar um código de dispositivo de 0 a 3, para uma ROM de extensão.

## Os Ganchos

Esta seção da Área de Trabalho de FD9AH até FFC9H contém cento e doze ganchos, cada um dos quais é preenchido com cinco instruções RET do Z80 na partida. Estes são chamados a partir de localizações estratégicas dentro do BIOS/Interpretador, de modo que a ROM possa ser estendida, particularmente, de modo que possa ser ampliada para o BASIC de Disco. Cada gancho tem espaço suficiente para conter uma chamada distante (For call) para qualquer conector:

```
RST 30H
DEFB Identificação de conector
DEFW Endereço
RET
```

Os ganchos são relacionados nas páginas seguintes, juntamente, com os endereços de onde são chamados e uma pequena nota referente à sua função.

|        |         |      |   |                                        |
|--------|---------|------|---|----------------------------------------|
| FD9AH  | HKEYI:  | DEFS | 5 | ;Manipulador de Interrupção 0C4AH      |
| FD9FH  | HTIMI:  | DEFS | 5 | ;Manipulador de Interrupção 0C53H      |
| FDA4H  | HCHPU:  | DEFS | 5 | ;Rotina padrão CHPUT 08COH             |
| FDA9H  | HDSPC:  | DEFS | 5 | ;Mostra cursor 09E6H                   |
| FDAEH  | HERAC:  | DEFS | 5 | ;Apaga cursor 0A33H                    |
| FDB3H  | HDSPF:  | DEFS | 5 | ;Rotina padrão DSPFNK 0B2BH            |
| FDB8H  | HERAF:  | DEFS | 5 | ;Rotina padrão ERAFNK 0B15H            |
| FDBDH  | HTOTE:  | DEFS | 5 | ;Rotina padrão TOTEXT 0842H            |
| FDC2H  | HCHGE:  | DEFS | 5 | ;Rotina padrão CHGET 10CEH             |
| FDC7H  | HINIP:  | DEFS | 5 | ;Copia conj. caracteres p/ o VDP 071EH |
| FDCCH  | HKEYC:  | DEFS | 5 | ;Decodificador de teclado 1025H        |
| FDD1H  | HKEYA:  | DEFS | 5 | ;Decodificador de teclado 0F10H        |
| FDD6H  | HNMI:   | DEFS | 5 | ;Rotina padrão NMI 1398H               |
| FDDBH  | HPINL:  | DEFS | 5 | ;Rotina padrão PINLIN 23BFH            |
| FDEOH  | HQINL:  | DEFS | 5 | ;Rotina padrão QINLIN 23CCH            |
| FDESH  | HINLI:  | DEFS | 5 | ;Rotina padrão INLIN 23D5H             |
| FDEAH  | HONGO:  | DEFS | 5 | ;“ON DISPOSITIVO GOSUB” 7810H          |
| FDEFH  | HDSKO:  | DEFS | 5 | ;“DSKO\$” 7C16H                        |
| FDF4H  | HSETS:  | DEFS | 5 | ;“SET” 7C1BH                           |
| FDF9H  | HNAME:  | DEFS | 5 | ;“NAME” 7C20H                          |
| FD FEH | HKILL:  | DEFS | 5 | ;“KILL” 7C25H                          |
| FE03H  | HIPL:   | DEFS | 5 | ;“IPL” 7C2AH                           |
| FE08H  | HCOPY:  | DEFS | 5 | ;“COPY” 7C2FH                          |
| FE0DH  | HCMD:   | DEFS | 5 | ;“DSKF” 7C39H                          |
| FE17H  | HDSKI:  | DEFS | 5 | ;“DSKI\$” 7C3EH                        |
| FE1CH  | HATTR:  | DEFS | 5 | ;“ATTR\$” 7C43H                        |
| FE21H  | HLSET:  | DEFS | 5 | ;“LSET” 7C48H                          |
| FE26H  | HRSET:  | DEFS | 5 | ;“RSET” 7C4DH                          |
| FE2BH  | HFIEL:  | DEFS | 5 | ;“FIELD” 7C52H                         |
| FE30H  | HMK1\$: | DEFS | 5 | ;“MKI\$” 7C57H                         |
| FE35H  | HMK\$S: | DEFS | 5 | ;“MKS\$” 7C5CH                         |
| FE3AH  | HMKD\$: | DEFS | 5 | ;“MKD\$” 7C61H                         |
| FE3FH  | HCVI:   | DEFS | 5 | ;“CVI” 7C66H                           |
| FE44H  | HCVS:   | DEFS | 5 | ;“CVS” 7C6BH                           |
| FE49H  | HCVD:   | DEFS | 5 | ;“CVD” 7C70H                           |
| FE4EH  | HSETP:  | DEFS | 5 | ;Localizar FCB 6A93H                   |
| FE53H  | HSETP:  | DEFS | 5 | ;Localizar FCB 6AB3H                   |
| FE58H  | HNOFO:  | DEFS | 5 | ;“OPEN” 6AF6H                          |
| FE5DH  | HNULO:  | DEFS | 5 | ;“OPEN” 6B0FH                          |
| FE62H  | HNTFL:  | DEFS | 5 | ;Fecha buffer 0 de E/S 6B3BH           |
| FE67H  | HMERGE: | DEFS | 5 | ;“MERGE/LOAD” 6B63H                    |
| FE6CH  | HSAVE:  | DEFS | 5 | ;“SAVE” 6BA6H                          |
| FE71H  | HBINS:  | DEFS | 5 | ;“SAVE” 6BCEH                          |
| FE76H  | HBINL:  | DEFS | 5 | ;“MERGE/LOAD” 6BD4H                    |
| FE7BH  | HFILE:  | DEFS | 5 | ;“FILES” 6C2FH                         |

|       |        |      |   |                                                |
|-------|--------|------|---|------------------------------------------------|
| FE80H | HDGET: | DEFS | 5 | ;“GET/PUT” 6C3BH                               |
| FE85H | HFIL0: | DEFS | 5 | ;Saída sequencial 6C51H                        |
| FE8AH | HINDS: | DEFS | 5 | ;Entrada sequencial 6C79H                      |
| FE8AH | HRSLT: | DEFS | 5 | ;“INPUT\$” 6CD8H                               |
| FE94H | HSAVD: | DEFS | 5 | ;“LOC” 6D03H, “LOF” 6D14H                      |
| FE99H | HLOC:  | DEFS | 5 | ;“LOC” 6D0FH                                   |
| FE9EH | HLOF:  | DEFS | 5 | ;“LOF” 6D20H                                   |
| FEA3H | HEOF:  | DEFS | 5 | ;“EOF” 6D33H                                   |
| FEA8H | HFPOS: | DEFS | 5 | ;“FPOS” 6D43H                                  |
| FEADH | HBAKU: | DEFS | 5 | ;“LINE INPUT#” 6E36H                           |
| FEB2H | HPADR: | DEFS | 5 | ;Analisar nome do dispositivo 6F15H            |
| FEB7H | HNODE: | DEFS | 5 | ;Analisar nome do dispositivo 6F33H            |
| FEBCH | HPOSD: | DEFS | 5 | ;Analisar nome do dispositivo 6F37H            |
| FEC1H | HDEVN: | DEFS | 5 | ;Este gancho não é utilizado                   |
| FEC6H | HGEND: | DEFS | 5 | ;Despachador de função E/S 6F8FH               |
| FECBH | HRUNC: | DEFS | 5 | ;Processar-liberar 629AH                       |
| FED0H | HCLEA: | DEFS | 5 | ;Processar-liberar 62A1H                       |
| FED5H | HLOPD: | DEFS | 5 | ;Processar-liberar 62AFH                       |
| FEDAH | HSTKE: | DEFS | 5 | ;Repor pilha 62F0H                             |
| FEDFH | HISFL: | DEFS | 5 | ;Rotina padrão ISFLIO 145FH                    |
| FEE4H | HOUTD: | DEFS | 5 | ;Rotina padrão OUTDO 1B46H                     |
| FEE9H | HCRDO: | DEFS | 5 | ;CR, LF para OUTDO 7328H                       |
| FEEEH | HDSCC: | DEFS | 5 | ;Linha de entrada laço-principal 7274H         |
| FEF3H | HDOGR: | DEFS | 5 | ;Traçado de linha 593CH                        |
| FEF8H | HPRGE: | DEFS | 5 | ;Fim de programa 4039H                         |
| FEFDH | HERRP: | DEFS | 5 | ;Manipulador de erro 40DCH                     |
| FF02H | HERRF: | DEFS | 5 | ;Manipulador de arquivo 40FDH                  |
| FF07H | HREAD: | DEFS | 5 | ;“OK” do laço principal 4128H                  |
| FF0CH | HMAIN: | DEFS | 5 | ;Laço principal 4134H                          |
| FF11H | HDIRD: | DEFS | 5 | ;“Declaração direta do laço principal<br>41A8H |
| FF16H | HFINI: | DEFS | 5 | ;Término do laço-principal 4237H               |
| FF1BH | HFINE: | DEFS | 5 | ;Término do laço-principal 4247H               |
| FF20H | HCRUN: | DEFS | 5 | ;Simbolizar 42B9H                              |
| FF25H | HCRUS: | DEFS | 5 | ;Simbolizar 4353H                              |
| FF2AH | HISRE: | DEFS | 5 | ;Simbolizar 437CH                              |
| FF2FH | HNTFN: | DEFS | 5 | ;Simbolizar 43A4H                              |
| FF34H | HNOTR: | DEFS | 5 | ;Simbolizar 44EBH                              |
| FF39H | HSNGF: | DEFS | 5 | ;“FOR” 45D1H                                   |
| FF3EH | HNEWS: | DEFS | 5 | ;Nova declaração do laço-principal 4601H       |
| FF43H | HGONE: | DEFS | 5 | ;Executar laço-de-processamento 4646H          |
| FF48H | HCHRG: | DEFS | 5 | ;Rotina padrão CHRGR 4666H                     |
| FF4DH | HRETU: | DEFS | 5 | ;“RETURN” 4821H                                |
| FF52H | HPRTF: | DEFS | 5 | ;“PRINT” 4A5EH                                 |
| FF57H | HCOMP: | DEFS | 5 | ;“PRINT” 4A94H                                 |

|       |        |      |   |                                       |
|-------|--------|------|---|---------------------------------------|
| FF5CH | HFINP: | DEFS | 5 | ;“PRINT” 4AFFH                        |
| FF61H | HTRMN: | DEFS | 5 | ;Erro “READ/INPUT” 4B4DH              |
| FF66H | HFRME: | DEFS | 5 | ;Avaliador de expressão 4C6DH         |
| FF6BH | HNTPL: | DEFS | 5 | ;Avaliador de expressão 4CA6H         |
| FF70H | HEVAL: | DEFS | 5 | ;Avaliador de fatores 4DD9H           |
| FF75H | HOKNO: | DEFS | 5 | ;Avaliador de fatores 4F2CH           |
| FF7AH | HFING: | DEFS | 5 | ;Avaliador de fatores 4F3EH           |
| FF7FH | HISMI: | DEFS | 5 | ;Executar laço-de-processamento 51C3H |
| FF84H | HWIDT: | DEFS | 5 | ;“WIDTH” 51CCH                        |
| FF89H | HLIST: | DEFS | 5 | ;“LIST” 522EH                         |
| FF8EH | HBUFL: | DEFS | 5 | ;De-simbolizar 532DH                  |
| FF93H | HFRQI: | DEFS | 5 | ;Converter a inteiro 543FH            |
| FF98H | HSCNE: | DEFS | 5 | ;Número de linha para apontador 5514H |
| FF9DH | HFRET: | DEFS | 5 | ;Liberar descritor 67EEH              |
| FFA2H | HPTRG: | DEFS | 5 | ;Procura de Variável 5EA9H            |
| FFA7H | HPHYD  | DEFS | 5 | ;Rotina padrão PHYDIO 148AH           |
| FFACH | HFORM: | DEFS | 5 | ;Rotina padrão FORMAT 148EH           |
| FFB1H | HERRO: | DEFS | 5 | ;Manipulador de erro 406FH            |
| FFB6H | HLPTO: | DEFS | 5 | ;Rotina padrão LPTOUT 085DH           |
| FFBBH | HLPTS: | DEFS | 5 | ;Rotina padrão LPTSTT 0884H           |
| FFC0H | HSCRE: | DEFS | 5 | ;“SCREEN” 79CCH                       |
| FFC5H | HPLAY: | DEFS | 5 | ;Declaração “PLAY” 73E5H              |

O espaço da Área de Trabalho de FFCAH até FFFFH não é utilizado.

## PROGRAMAS EM ASSEMBLER

Este capítulo contém alguns programas em código de máquina para ilustrar a utilização dos recursos do sistema MSX. Apesar de ter sido preparado com o Assembler ZEN\*, eles foram projetados para serem executados a partir do BASIC e, se necessário, poderão ser digitados na forma hexadecimal utilizando o carregador mostrado a seguir. Em seguida, o código deveria ser salvo em cassete antes que qualquer tentativa de execução acontecesse.

```
10 CLEAR 200,&HE000
20 E=&HE000
30 PRINT RIGHT$("000"+HEX$(E),4);
40 INPUT D$
50 POKE E,VAL("&H"+D$)
60 E=E+1
70 GOTO 30
```

Todos os programas iniciam no endereço E000H e são executados nesse mesmo endereço. A não ser que seja dito o contrário, nenhum parâmetro precisa ser passado aos programas. A execução poderá, portanto, ser iniciada com a instrução DEFUSR=&HE000:USR(0).

\* Na edição brasileira usamos o Assembler GEN, que tem apenas uma particularidade digna de nota: constantes hexadecimais são indicadas com um sinal #. Assim a constante 0E02CH, por exemplo, ficaria #E02C.

## Matriz do Teclado

Hisoft GEN Assembler. Page

1.

Pass 1 errors: 00

```

E000      10      ORG  #E000
E000      20      ENT  #E000
          30      ;-----
          40      ;--  ROTINAS DA BIOS  --
          50      ;-----
006C      60      INITXT: EQU  #006C
00A2      70      CHPUT: EQU  #00A2
0141      80      SNSMAT: EQU  #0141
00B7      90      BREAKX: EQU  #00B7
          100     ;-----
          110     ;--  VARIAVEIS DA AREA DE  --
          120     ;--  TRABALHO  --
          130     ;-----
FC9B      140     INTFLG: EQU  #FC9B
          150     ;-----
          160     ;--  CARACTERES DE CONTROLE  --
          170     ;-----
000A      180     LF:      EQU  10
000B      190     HOME:    EQU  11
000D      200     CR:      EQU  13
          210
          220
          230
E000 CD6C00      240 MATRIZ: CALL INITXT          ;screen 0
E003 3E0B        250 MAT1:  LD  A,HOM
E005 CDA200      260      CALL CHPUT
E008 AF          270      XOR  A          ;cursor para o topo
E009 F5          280 MAT2:  PUSH AF          ;A=linha atual
E00A CD4101      290      CALL SNSMAT
E00D 060B        300      LD  B,B          ;le a linha atual
E00F 07          310 MAT3:  RLCA
E010 F5          320      PUSH AF          ;seleciona a coluna
E011 E601        330      AND  1
E013 C630        340      ADD  A,"0"
E015 CDA200      350      CALL CHPUT          ;mostra a coluna
E018 F1          360      POP  AF
E019 10F4        370      DJNZ MAT3
E01B 3E0D        380      LD  A,CR          ;termina linha
E01D CDA200      390      CALL CHPUT
E020 3E0A        400      LD  A,LF
E022 CDA200      410      CALL CHPUT
E025 F1          420      POP  AF
E026 3C          430      INC  A          ;prox. linha
E027 FE0B        440      CP  11          ;acabou ?
E029 20DE        450      JR  NZ,MAT2
E02B CDB700      460      CALL BREAKX
E02E 30D3        470      JR  NC,MAT1          ;apertou CTRL-STOP?
E030 AF          480      XOR  A
E031 329BFC      490      LD  (INTFLG),A          ;limpa o STOP
E034 C9          500      RET
E035            510      END

```

Pass 2 errors: 00

Table used: 154 from 300

Executes: 57344

Este programa apresenta a matriz do teclado na tela, de modo que a operação das teclas possa ser observada diretamente. O programa pode ser terminado pressio-



nando-se CTRL e STOP. Observe que operações espúrias de teclas podem ser produzidas sob determinadas circunstâncias, caso mais de três ou quatro teclas sejam pressionadas ao mesmo tempo. Esta é uma característica de todos os teclados do tipo matriz.

### Texto Gráfico de 40 Colunas

Este programa imprime texto na tela no Modo Gráfico com quarenta caracteres por linha. A string a ser apresentada é passada como o parâmetro da chamada USR, por exemplo, A\$=USR ("alguma coisa"). Não há necessidade de abrir um arquivo GRP antecipadamente, a única exigência do programa é que a tela esteja no modo correto. O coração do programa é funcionalmente equivalente à rotina padrão GRPPTR, mas exclusivamente as primeiras seis colunas de ponto de uma dada imagem de caractere são colocadas na tela, em vez de oito. Da mesma forma que com GRPPTR a imagem é colocada na posição gráfica corrente, e o único caractere de controle reconhecido é ASCII GR (13) que funciona como um GR, LF combinado. Ao contrário da rotina padrão GRPPTR, caracteres impressos em coordenadas negativas, mas tocando a tela, serão corretamente apresentadas. O programa é corretamente preparado para executar uma alimentação automática de linha depois da coluna 239, desta forma, fornecendo exatamente quarenta caracteres por linha. Caso seja requerido, isso poderá ser mudado, por meio da constante na sub-rotina RMDCOL, de modo que a largura total da tela seja utilizável.

Uma vez carregado, o programa pode ser testado com o seguinte programa BASIC:

```
10 SCREEN 2
20 LINE (10,10)-(100,50)
30 PSET (50,50)
40 DEFUSR=4HE000
50 U$=USR("TESTE DE IMPRESSAO")
60 GOTO 60
```

Pass 1 errors: 000

```

E000      10      ORG  #E000
E000      20      ENT  #E000
          30      ;-----
          40      ;--  ROTINAS DA BIOS  --
          50      ;-----
000C      60      RDSLT: EQU  #000C
00AB      70      CNVCHR: EQU  #00AB
0111      80      MAPXYC: EQU  #0111
0120      90      SETC:  EQU  #0120
          100     ;-----
          110     ;--  VARIAVEIS DA AREA DE  -
          120     ;--  TRABALHO  -
          130     ;-----
F3E9      140     FORCLR: EQU  #F3E9
F3F2      150     ATRBYT: EQU  #F3F2
F91F      160     CGPNT: EQU  #F91F
FC40      170     PATWRK: EQU  #FC40
FCAF      180     SCRMOD: EQU  #FCAF
FCB7      190     GRPACX: EQU  #FCB7
FCB9      200     GRPACY: EQU  #FCB9
          210     ;-----
          220     ;--  CARACTERES DE CONTROLE  -
          230     ;-----
000D      240     CR:    EQU  13
          250
          260
          270
E000 FE03      280     TGRAF: CP  3                      ;string ?
E002 C0        290     RET  NZ
E003 3AAFFC    300     LD  A, (SCRMOD)
E006 FE02      310     CP  2                      ;modo grafico ?
E008 C0        320     RET  NZ
E009 EB        330     EX  DE,HL                      ;HL->descriptor
E00A 46        340     LD  B,(HL)                      ;B=compr. do string
E00B 23        350
E00C 5E        360     LD  E,(HL)
E00D 23        370     INC  HL
E00E 56        380     LD  D,(HL)                      ;DE=endereco do string
E00F 04        390     INC  B
E010 05        400     TG2: DEC  B                      ;acabou?
E011 C8        410     RET  Z
E012 1A        420     LD  A,(DE)                      ;A=carac.do string
E013 CD19E0    430     CALL IMPGRF
E016 13        440     INC  DE
E017 18F7      450     JR   TG2
          460     ;
          470     ; ROTINA IMPGRF - IMPRIME O CARACTERE
          480     ; NA TELA GRAFICA
          490     ;
E019 F5        500     IMPGRF: PUSH AF
E01A C5        510     PUSH BC
E01B D5        520     PUSH DE
E01C E5        530     PUSH HL
E01D FDE5      540     PUSH IY
E01F ED48B7FC  550     LD  BC, (GRPACX)                ;BC=coord.X
E023 ED58B9FC  560     LD  DE, (GRPACY)                ;DE=coord.Y
E027 CD39E0    570     CALL GDC
          575     *E
E02A ED43B7FC  580     LD  (GRPACX),BC                ;novo X
E02E ED53B9FC  590     LD  (GRPACY),DE                ;novo Y
E032 FDE1      600     POP  IY
E034 E1        610     POP  HL
E035 D1        620     POP  DE
E036 C1        630     POP  BC
E037 F1        640     POP  AF
E038 C9        650     RET
          660     ;
          670     ; ROTINA GDC - DECODIFICA CARACTERE

```

```

E039 CDAB00      680 ;
E03C D0          690 GDC: CALL CNVCHR
E03D 2007        700 RET NC ;volta se grafico
E03F FE0D        710 JR NZ,GD2 ;NZ=convertido
E041 2873        720 CP CR ;e' CR?
E043 FE20        730 JR Z,GCRLF
E045 D8          740 CP #20
E046 6F          750 RET C ;volta se controle
E047 2600        760 LD L,A
E049 29          770 LD H,0
E04A 29          780 ADD HL,HL
E04B 29          790 ADD HL,HL
E04C C5          800 ADD HL,HL ;HL=ASC(carac)*8
E04D D5          810 PUSH BC ;coord X
E04E ED5B20F9    820 PUSH DE ;coord Y
E052 19          830 LD DE,(CGPNT+1) ;conj.caracteres
E053 1140FC      840 ADD HL,DE ;HL=imagem
E056 0608        850 LD DE,PATWRK ;HL=buffer
E058 C5          860 LD B,8 ;oito linhas
E059 D5          870 GD3: PUSH BC
E05A 3A1FF9      880 PUSH DE
E05D CD0C00      890 LD A,(CGPNT)
E060 FB          900 CALL RDSLT ;le imagem
E061 D1          910 EI
E062 C1          920 POP DE
E063 12          930 POP BC
E064 13          940 LD (DE),A ;poe no buffer
E065 23          950 INC DE
E066 10F0        960 INC HL
E068 D1          970 DJNZ GD3 ;prox.
E069 C1          980 POP DE
E06A 3AE9F3      990 POP BC
E06D 32F2F3      1000 LD A,(FORCLR)
E070 FD2140FC    1010 LD (ATRBVT),A ;cor de frente
E074 D5          1020 LD IY,PATWRK
E075 2608        1030 PUSH DE ;IY->imagem
E077 CB7A        1040 LD H,8
E079 202A        1050 BIT 7,D
E07B CDBFE0      1060 JR NZ,GD8
E07E 382B        1070 CALL ULTLIN ;esta' na ult. linha?
E080 C5          1080 JR C,GD9
E081 2E06        1090 PUSH BC
E083 FD7E00      1100 LD L,6
E083 3          1110 LD A,(IY+0)
E086 CB78        1115 *E
E088 2015        1120 GD5: BIT 7,B
E08A CDCBE0      1130 JR NZ,GD6
E08D 3815        1140 CALL ULTCOL ;esta' na ult.col.?
E08F CB7F        1150 JR C,GD7
E091 2B0C        1160 BIT 7,A
E093 F5          1170 JR Z,GD6
E094 D5          1180 PUSH AF
E095 E5          1190 PUSH DE
E096 CD1101      1200 PUSH HL
E099 CD2001      1210 CALL MAPXYC
E09C E1          1220 CALL SETC ;liga o ponto
E09D D1          1230 POP HL
E09E F1          1240 POP DE
E09F 07          1250 POP AF
E0A0 03          1260 GD6: RLCA
E0A1 2D          1270 INC BC
E0A2 20E2        1280 DEC L ;acabou colunas?
E0A4 C1          1290 JR NZ,GD5
E0A5 FD23        1300 POP BC
E0A7 13          1310 POP BC ;scord.X inicial
E0A8 25          1320 INC IY ;prox.byte imagem
E0A9 20CC        1330 INC DE ;incr. Y
E0AB D1          1340 DEC H ;acabou linhas?
E0AC 210600      1350 JR NZ,GD4
E0AC 210600      1360 POP DE ;Y inicial
E0AC 210600      1360 LD HL,6

```

```

E0AF 09      1370      ADD HL,BC          ; X=X+6
E0B0 44      1380      LD B,H
E0B1 4D      1390      LD C,L          ; BC=novo X
E0B2 CDC8E0 1400      CALL ULTCOL
E0B5 D0      1410      RET NC
E0B6 010000 1420      GCRLF: LD BC,0      ; X=0
E0B9 210B00 1430      LD HL,B
E0BC 19      1440      ADD HL,DE
E0BD EB      1450      EX DE,HL        ; Y=Y+B
E0BE C9      1460      RET
      4          1465 *E
      1470 ;
      1480 ; ROTINA ULTLIN - CHECA SE ESTA'
      1490 ; NA ULTIMA LINHA DA TELA
      1500 ;
E0BF E5      1510      ULTLIN: PUSH HL
E0C0 21BF00 1520      LD HL,191        ; ult.linha
E0C3 B7      1530      OR A
E0C4 ED52    1540      SBC HL,DE        ; compara
E0C6 E1      1550      POP HL
E0C7 C9      1560      RET
      1570 ;
      1580 ; ROTINA ULTCOL - CHECA SE ESTA'
      1590 ; NA ULTIMA COLUMNA DA TELA
      1600 ;
E0C8 E5      1610      ULTCOL: PUSH HL
E0C9 21EF00 1620      LD HL,239        ; ult.coluna
E0CC B7      1630      OR A
E0CD ED42    1640      SBC HL,BC        ; compara
E0CF E1      1650      POP HL
E0D0 C9      1660      RET
      1670
E0D1          1680      END

```

Pass 2 errors: 00

Table used: 326 from 445

Executes: 57344

## "Bubble Sort" de Strings

Este programa vai classificar o conteúdo de uma matriz string na ordem alfabética ascendente. A posição da matriz é passada como parâmetro da chamada USR, por exemplo, V=USR(VARPTR(A\$(0))). Não há nenhuma restrição quanto ao tamanho da Matriz ou em seu conteúdo, mas ela poderá ter apenas uma dimensão. O programa é baseado no algoritmo clássico de classificação tipo "bubble", onde pares de strings são comparados e as suas posições permutadas quando o segundo é menor do que o primeiro. Uma Matriz de 250 elementos randomicamente gerados será classificada em aproximadamente 2,5 segundos. O programa BASIC equivalente leva mais do que seis minutos.

O programa seguinte é um exemplo de aplicação desta rotina:

```

10 DIM A$(25)
20 FOR N=0 TO 25:A$(N)=CHR$(70-N):NEXT
30 FOR N=0 TO 25:PRINT A$(N);" ";NEXT
40 DEFUSR=&HE000
50 U=USR(VARPTR(A$(0)))
60 PRINT:PRINT
70 FOR N=0 TO 25:PRINT A$(N);" ";NEXT

```

Pass 1 errors: 00

|      |        |     |        |                               |
|------|--------|-----|--------|-------------------------------|
| E000 | 10     | ORG | #E000  |                               |
| E000 | 20     | ENT | #E000  |                               |
|      | 30     |     |        |                               |
|      | 40     |     |        |                               |
|      | 50     |     |        |                               |
| E000 | FE02   | 60  | SORT:  | CP 2 ;inteiro?                |
| E002 | C0     | 70  | RET    | NZ                            |
| E003 | 23     | 80  | INC    | HL                            |
| E004 | 23     | 90  | INC    | HL ;HL->DAC+2                 |
| E005 | 5E     | 100 | LD     | E, (HL)                       |
| E006 | 23     | 110 | INC    | HL                            |
| E007 | 56     | 120 | LD     | D, (HL)                       |
| E008 | EB     | 130 | EX     | DE, HL ;HL->A\$(0)            |
| E009 | E5     | 140 | PUSH   | HL                            |
| E00A | DDE1   | 150 | POP    | IX                            |
| E00C | DD7EFB | 160 | LD     | A, (IX-8)                     |
| E00F | FE03   | 170 | CP     | 3 ;vetor string?              |
| E011 | C0     | 180 | RET    | NZ                            |
| E012 | DD7EFD | 190 | LD     | A, (IX-3)                     |
| E015 | 3D     | 200 | DEC    | A ;so' uma dimensao?          |
| E016 | C0     | 210 | RET    | NZ                            |
| E017 | DD4EFE | 220 | LD     | C, (IX-2)                     |
| E01A | DD46FF | 230 | LD     | B, (IX-1) ;BC=no. deelementos |
| E01D | C5     | 240 | SR2:   | PUSH BC                       |
| E01E | E5     | 250 | PUSH   | HL ;HL->descriptor(N)         |
| E01F | 46     | 260 | SR3:   | LD B, (HL) ;B=LEN(N)          |
| E020 | 23     | 270 | INC    | HL                            |
| E021 | 5E     | 280 | LD     | E, (HL)                       |
| E022 | 23     | 290 | INC    | HL                            |
| E023 | E5     | 300 | PUSH   | HL                            |
| E024 | 56     | 310 | LD     | D, (HL) ;DE->string(N)        |
| E025 | 23     | 320 | INC    | HL ;HL->descriptor(N+1)       |
| E026 | 4E     | 330 | LD     | C, (HL) ;C=LEN(N+1)           |
| E027 | 23     | 340 | INC    | HL                            |
| E028 | 7E     | 350 | LD     | A, (HL)                       |
| E029 | 23     | 360 | INC    | HL                            |
| E02A | E5     | 370 | PUSH   | HL                            |
| E02B | 66     | 380 | LD     | H, (HL)                       |
| E02C | 6F     | 390 | LD     | L, A                          |
| E02D | EB     | 400 | EX     | DE, HL ;HL->(N), DE->(N+1)    |
| E02E | 04     | 410 | INC    | B                             |
| E02F | 0C     | 420 | INC    | C                             |
| E030 | 05     | 430 | SR4:   | DEC B ;compr. de N            |
| E031 | 2825   | 440 | JR     | Z, PROX ;pula se (N)<=(N+1)   |
| E033 | 0D     | 450 | DEC    | C ;compr. de N+1              |
| E034 | 2808   | 460 | JR     | Z, TROCA ;pula se (N)>(N+1)   |
| E036 | 1A     | 470 | LD     | A, (DE)                       |
| E037 | BE     | 480 | CP     | (HL) ;compara 2 caracs.       |
| E038 | 13     | 490 | INC    | DE                            |
| E039 | 23     | 500 | INC    | HL                            |
| E03A | 28F4   | 510 | JR     | Z, SR4 ;volta se iguais       |
| E03C | 301A   | 520 | JR     | NC, PROX                      |
| E03E | E1     | 530 | TROCA: | POP HL ;HL->descr(N+1)        |
| E03F | D1     | 540 | POP    | DE ;DE->descr(N)              |
| E040 | 0603   | 550 | LD     | B, 3                          |

```

2          555 *E
E042 1A      560 TR2: LD A, (DE)          ;troca os descritores
E043 4E      570 LD C, (HL)
E044 77      580 LD (HL), A
E045 79      590 LD A, C
E046 12      600 LD (DE), A
E047 1B      610 DEC DE
E048 2B      620 DEC HL
E049 10F7    630 DJNZ TR2
E04B DDE5    640 PUSH IX
E04D E1      650 POP HL          ;HL->A$(0)
E04E B7      660 OR A
E04F ED52    670 SBC HL, DE      ;esta' no inicio?
E051 3007    680 JR NC, PR2
E053 1B      690 DEC DE
E054 1B      700 DEC DE
E055 EB      710 EX DE, HL
E056 18C7    720 JR SR3
E058 E1      730 PROX: POP HL      ;tira lixo da pilha
E059 E1      740 POP HL
E05A E1      750 PR2: POP HL      ;HL->descriptor(N)
E05B C1      760 POP BC          ;BC=no.de elementos
E05C 23      770 INC HL
E05D 23      780 INC HL
E05E 23      790 INC HL        ;prox.descritor
E05F 0B      800 DEC BC
E060 7B      810 LD A, B
E061 B1      820 OR C
E062 20B9    830 JR NZ, SR2     ;acabou?
E064 C9      840 RET
E065          850 END

```

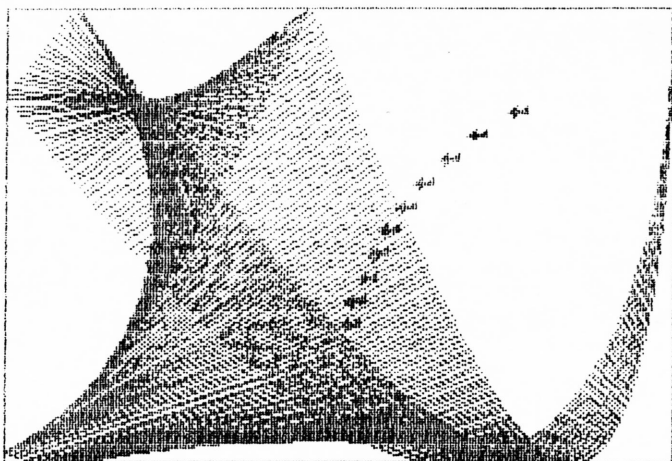
Pass 2 errors: 00

Table used: 99 from 261

Executes: 57344

## Um Dump de Tela Gráfica

Este programa faz um “dump” do conteúdo da tela, em qualquer modo, para a impressora. Quando é ativado pela primeira vez, por meio da chamada *USR*, o programa apenas se liga ao gancho da varredura de teclado do manipulador de interrupção. Uma vez instalado, o programa se torna, efetivamente, uma extensão do manipulador de interrupção, e um “dump” de tela pode assim ser iniciado de qualquer parte do sistema simplesmente pressionando-se a tecla *ESC*. Caso seja necessário, o dump pode ser encerrado pressionando-se as teclas *CTRL* e *STOP*. Um exemplo de uma tela de Modo Gráfico é apresentado a seguir. Os “passarinhos” são sprites.



O método mais simples para gerar um dump de tela é copiar todos os códigos de caracteres da Tabela de Nomes para a impressora. Entretanto, isto funcionaria apenas nos dois modos texto, os sprites não poderiam ser apresentados e o resultado refletiria o conjunto de caracteres internos da impressora em lugar do conjunto de caracteres do VDP. O programa, portanto, reproduz a tela como uma imagem de 240/256x192 bits da impressora em todos os modos, sendo que cada ponto de imagem será derivado do código de cor do ponto correspondente da tela. Nenhum ponto para as cores 0 a 7 e um ponto para as cores 8 a 15.

O código de cores para um determinado ponto é obtido examinando primeiro os trinta e dois sprites em sequência, para determinar se algum se sobrepõe. Se todos os sprites são transparentes no ponto, então o plano de caracteres é examinado. Isto é feito utilizando as coordenadas do ponto para localizar a entrada correspondente na Tabela de Nomes, em seguida, por meio do código de caracteres, para isolar o bit relevante na imagem associada. Caso o código de cor do bit é encontrado como sendo transparente a cor de fundo é devolvida.

Observe que as sequências de códigos de controle utilizadas no programa são para a impressora FX80 da Epson. Estas são marcadas na listagem, caso outra impressora seja utilizada. Uma sequência é utilizada para inserir o modo de imagem de bit no início de uma linha de 240/256 bytes (cada byte define oito pontos verticais) e uma sequência é utilizada para iniciar uma alimentação de papel no final da linha. O programa foi otimizado quanto à velocidade, em vez de para código mínimo, e leva aproximadamente cinco segundos mais o tempo da impressora para produzir os 46.080/49.152 pontos na imagem.

Pass 1 errors: 00

```

E000      10      ORG  #E000
E000      20      ENT  #E000
          30      ;-----
          40      ; -      ROTINAS DA BIOS      -
          50      ;-----
004A      60      RDVRM: EQU  #004A
00B7      70      CALATR: EQU  #00B7
00A5      80      LPTOUT: EQU  #00A5
          90      ;-----
          100     ; -      VARIAVEIS DA AREA      -
          110     ; -      DE TRABALHO      -
          120     ;-----
F3BF      130     T32COL: EQU  #F3BF
F3C7      140     GRPNAM: EQU  #F3C7
F3C9      150     GRPCOL: EQU  #F3C9
F3CB      160     GRPCGP: EQU  #F3CB
F3D1      170     MLTNAM: EQU  #F3D1
F3D5      180     MLTCGP: EQU  #F3D5
F3E0      190     RG1SAV: EQU  #F3E0
F3E6      200     RG7SAV: EQU  #F3E6
F922      210     NAMBAS: EQU  #F922
F924      220     CGPBAS: EQU  #F924
F926      230     PATBAS: EQU  #F926
F928      240     ATRBAS: EQU  #F928
FC4F      250     SCRMOD: EQU  #FC4F
FDCC      260     HKEYC: EQU  #FDCC
          270     ;-----
          280     ; -      CARACTERES DE CONTROLE      -
          290     ;-----
000D      300     CR: EQU  13
001B      310     ESC: EQU  27
          320
          330
          340
          350 ;
          360 ; ROTINA INICIO - INSTALA UMA CHAMADA PARA
          370 ; A ROTINA DUMP NO GANCHO
          380 ;
E000      390     INICIO: LD  A, (HKEYC)      ;gancho
E003      400     CP    #C9                  ;pode ser usado?
E005      410     RET  NZ
E006      420     LD    HL, DUMP              ;endereço para ir
E009      430     LD    (HKEYC+1), HL
E00C      440     LD    A, #CD                ;codigo de CALL
E00E      450     LD    (HKEYC), A
E011      460     RET
          470 ;
          480 ; ROTINA DUMP -- ROTINA PRINCIPAL
          490 ;
E012      500     DUMP: CP    #3A              ;e' tecla ESC?
E014      510     RET  NZ
E015      520     PUSH AF
E016      530     PUSH BC
E017      540     PUSH DE
E018      550     PUSH HL
          560 *E
E019      570     LD    (PILBRK), SP          ;CTRL-STOP
E01D      580     LD    C, 0                  ;C=linha
E01F      590     DU1: LD    A, (SCRMOD)
E022      600     OR    A
E023      610     LD    HL, 240              ;pontos p/linha
E026      620     LD    DE, 6*256+40
E029      630     JR    Z, DU2
E02B      640     LD    HL, 256
E02E      650     LD    DE, 8*256+32
E031      660     DU2: LD    A, ESC            ;codigos EPSON
E033      670     CALL IMPR
E036      680     LD    A, "K"

```



```

E038 CD8DE0 690 CALL IMPR
E03B 7D 700 LD A,L
E03C CD8DE0 710 CALL IMPR
E03F 7C 720 LD A,H
E040 CD8DE0 730 CALL IMPR
E043 0600 740 LD B,0
E045 CD97E0 750 DU3: CALL BLOC0
E048 05 760 PUSH DE
E049 C5 770 PUSH BC
E04A 2151E2 780 LD HL,BUFFB ;HL->cores
E04D 42 790 LD B,D
E04E 110800 800 LD DE,8
E051 C5 810 DU4: PUSH BC
E052 E5 820 PUSH HL
E053 060B 830 LD B,B ;B=linhas
E055 7E 840 DU5: LD A,(HL) ;A=cor
E056 FE0B 850 CP B ;cor clara?
E058 3F 860 CCF ;clara=imprima
E059 CB11 870 RL C
E05B 19 880 ADD HL,DE ;prox linha
E05C 10F7 890 DJNZ DU5
E05E 79 900 LD A,C
E05F CD8DE0 910 CALL IMPR
E062 E1 920 POP HL
E063 C1 930 POP BC
E064 23 940 INC HL ;prox coluna
E065 10EA 950 DJNZ DU4
E067 C1 960 POP BC
E068 D1 970 POP DE
E069 04 980 INC B
E06A 78 990 LD A,B
E06B BB 1000 CP E ;fim de linha?
E06C 20D7 1010 JR NZ,DU3
E06E 3E0D 1020 LD A,CR
E070 CD8DE0 1030 CALL IMPR
E073 3E1B 1040 LD A,ESC ;Codigo EPSON
E075 CD8DE0 1050 CALL IMPR ;de avanço de
E078 3E4A 1060 LD A,"J" ;1/9 poi.
E07A CD8DE0 1070 CALL IMPR
E07D 3E18 1080 LD A,24
E07F CD8DE0 1090 CALL IMPR
E082 0C 1100 INC C
E083 79 1110 LD A,C
E084 FE18 1120 CP 24 ;acabou tela?
E086 2097 1130 JR NZ,DU1
3 1135 *E
E088 E1 1140 DU6: POP HL
E089 D1 1150 POP DE
E08A C1 1160 POP BC
E08B F1 1170 POP AF
E08C C9 1180 RET
1200 ;
1210 ; ROTINA IMPR - IMPRIME E TESTA
1220 ; CTRL-STOP
1230 ;
E08D CDA500 1240 IMPR: CALL LPTOUT ;imprime
E090 D0 1250 RET NC ;CTRL-STOP?
E091 ED7B4FE2 1260 LD SP,(PILBRK) ;restaura pilha
E095 1BF1 1270 JR DU6
1280 ;
1290 ; ROTINA BLOC0 - CRIA BLOC0
1300 ;
E097 C5 1310 BLOC0: PUSH BC
E098 D5 1320 PUSH DE
E099 E5 1330 PUSH HL
E09A FDE5 1340 PUSH IY
E09C 2151E2 1350 LD HL,BUFFB
E09F 3E40 1360 LD A,64
E0A1 3600 1370 BL1: LD (HL),0 ;transparente
E0A3 23 1380 INC HL

```

|      |        |      |      |             |                     |
|------|--------|------|------|-------------|---------------------|
| E0A4 | 3D     | 1390 | DEC  | A           |                     |
| E0A5 | 20FA   | 1400 | JR   | NZ,BL1      |                     |
| E0A7 | 3AFFF  | 1410 | LD   | A,(SCRMOD)  |                     |
| E0AA | B7     | 1420 | OR   | A           | ;modo T40?          |
| E0AB | F5     | 1430 | PUSH | AF          |                     |
| E0AC | C5     | 1440 | PUSH | BC          |                     |
| E0AD | C469E1 | 1450 | CALL | NZ,SPRITES  | ;testa sprites      |
| E0B0 | C1     | 1460 | POP  | BC          |                     |
| E0B1 | 69     | 1470 | LD   | L,C         |                     |
| E0B2 | 2600   | 1480 | LD   | H,0         |                     |
| E0B4 | 29     | 1490 | ADD  | HL,HL       |                     |
| E0B5 | 29     | 1500 | ADD  | HL,HL       |                     |
| E0B6 | 29     | 1510 | ADD  | HL,HL       | ;HL=linha*8         |
| E0B7 | 5D     | 1520 | LD   | E,L         |                     |
| E0B8 | 54     | 1530 | LD   | D,H         | ;DE=HL              |
| E0B9 | 29     | 1540 | ADD  | HL,HL       |                     |
| E0BA | 29     | 1550 | ADD  | HL,HL       | ;HL=linha*32        |
| E0BB | F1     | 1560 | POP  | AF          |                     |
| E0BC | F5     | 1570 | PUSH | AF          |                     |
| E0BD | 2001   | 1580 | JR   | NZ,BL2      | ;modo T40?          |
| E0BF | 19     | 1590 | ADD  | HL,DE       | ;HL=linha*40        |
| E0C0 | 58     | 1600 | LD   | E,B         | ;DE=coluna          |
| E0C1 | 19     | 1610 | ADD  | HL,DE       |                     |
| E0C2 | EB     | 1620 | EX   | DE,HL       |                     |
| E0C3 | D602   | 1630 | SUB  | 2           |                     |
| E0C5 | 79     | 1640 | LD   | A,C         | ;A=linha            |
| E0C6 | 010000 | 1650 | LD   | BC,0        | ;BC=incr. da LGPIAB |
| E0C9 | 2A24F9 | 1660 | LD   | HL,(CGPBAS) |                     |
| E0CC | E5     | 1670 | PUSH | HL          |                     |
| E0CD | 2A22F9 | 1680 | LD   | HL,(NAMBAS) |                     |
| E0D0 | 3B19   | 1690 | JR   | C,BL4       | ;C=T40 ou T32       |
|      | 4      |      |      |             |                     |
| E0D2 | 200C   | 1695 | JR   | NZ,BL3      | ;NZ=MLT             |
| E0D4 | 2ACBF3 | 1710 | LD   | HL,(GRPCGP) | ;senao e' GRP       |
| E0D7 | E3     | 1720 | EX   | (SP),HL     |                     |
| E0D8 | 2AC7F3 | 1730 | LD   | HL,(GRPNAM) |                     |
| E0DB | E618   | 1740 | AND  | #18         |                     |
| E0DD | 47     | 1750 | LD   | B,A         |                     |
| E0DE | 180B   | 1760 | JR   | BL4         |                     |
| E0E0 | 2AD5F3 | 1770 | LD   | HL,(MLTCGP) |                     |
| E0E3 | E3     | 1780 | EX   | (SP),HL     |                     |
| E0E4 | 2AD1F3 | 1790 | LD   | HL,(MLTNAM) |                     |
| E0E7 | 07     | 1800 | RLCA |             |                     |
| E0E8 | E606   | 1810 | AND  | 6           | ;linha*2            |
| E0EA | 4F     | 1820 | LD   | C,A         |                     |
| E0EB | 19     | 1830 | ADD  | HL,DE       | ;HL->NANTAB         |
| E0EC | CD4A00 | 1840 | CALL | RDVRM       |                     |
| E0EF | 6F     | 1850 | LD   | L,A         |                     |
| E0F0 | 2600   | 1860 | LD   | H,0         |                     |
| E0F2 | 29     | 1880 | ADD  | HL,HL       |                     |
| E0F3 | 29     | 1890 | ADD  | HL,HL       |                     |
| E0F4 | 29     | 1900 | ADD  | HL,HL       | ;HL=carac*8         |
| E0F5 | 09     | 1910 | ADD  | HL,BC       |                     |
| E0F6 | EB     | 1920 | EX   | DE,HL       | ;DE=pos. na tabela  |
| E0F7 | FDE1   | 1930 | POP  | IY          | ;IY=base tabela     |
| E0F9 | FD19   | 1940 | ADD  | IY,DE       | ;IY->imagem         |
| E0FB | 2AC9F3 | 1950 | LD   | HL,(GRPCOL) |                     |
| E0FE | 19     | 1960 | ADD  | HL,DE       |                     |
| E0FF | 0F     | 1970 | RRCA |             |                     |
| E100 | 0F     | 1980 | RRCA |             |                     |
| E101 | 0F     | 1990 | RRCA |             |                     |
| E102 | E61F   | 2000 | AND  | #1F         |                     |
| E104 | 4F     | 2010 | LD   | C,A         |                     |
| E105 | 0600   | 2020 | LD   | B,0         |                     |
| E107 | 3AE6F3 | 2030 | LD   | A,(RG7SAV)  |                     |
| E10A | 57     | 2040 | LD   | D,A         |                     |
| E10B | E60F   | 2050 | AND  | #0F         |                     |
| E10D | 5F     | 2060 | LD   | E,A         | ;E=cor fundo        |
| E10E | F1     | 2070 | POP  | AF          |                     |
| E10F | E5     | 2080 | PUSH | HL          |                     |

```

E110 3D      2090      DEC  A
E111 200B    2100      JR   NZ,BL5          ;Z=T32
E113 2ABFF3  2110      LD   HL,(T32COL)
E116 09      2120      ADD  HL,BC
E117 CD4A00  2130      CALL RDVRM
E11A 57      2140      LD   D,A
E11B 2151E2  2150 BL5: LD   HL,BUFFB
E11E 0608    2160      LD   B,B
E120 FDE5    2170 BL6: PUSH  IY
E122 E3      2180      EX   (SP),HL          ;HL->imagem
E123 CD4A00  2190      CALL RDVRM
E126 4F      2200      LD   C,A          ;C=imagem
E127 E1      2210      POP  HL
E128 FD23    2220      INC  IY
E12A 3AAFFC  2230      LD   A,(SCRMOD)
E12D D602    2240      SUB  2
E12F 3B15    2250      JR   C,BL8          ;C=T40 ou T32
E131 2B0C    2260      JR   Z,BL7          ;Z=GRP
E133 51      2270      LD   D,C
E134 0EF0    2280      LD   C,#F0
E136 7B      2290      LD   A,B
E137 FE05    2300      CP   5
E139 2B0B    2310      JR   Z,BL8
E13B FD2B    2320      DEC  IY
E13D 1B07    2330      JR   BL8
E13F E3      2340 BL7: EX   (SP),HL          ;HL->cores GRP
E140 CD4A00  2350      CALL RDVRM
E143 57      2360      LD   D,A          ;D=cores
E144 23      2370      INC  HL
E145 E3      2380      EX   (SP),HL
5      2390 *E
E146 C5      2400 BL8: PUSH  BC
E147 060B    2410      LD   B,B
E149 CB11    2420 BL9: RL   C
E14B 34      2430      INC  (HL)
E14C 35      2440      DEC  (HL)          ;BUFFB limpo?
E14D 200D    2450      JR   NZ,BL12
E14F 7A      2460      LD   A,D
E150 3004    2470      JR   NC,BL10
E152 0F      2480      RRCA
E153 0F      2490      RRCA
E154 0F      2500      RRCA
E155 0F      2510      RRCA          ;cor 1
E156 E60F    2520 BL10: AND  #0F
E158 2001    2530      JR   NZ,BL11          ;Z=transparente
E15A 7B      2540      LD   A,E
E15B 77      2550 BL11: LD   (HL),A
E15C 23      2560 BL12: INC  HL
E15D 10EA    2570      DJNZ BL9          ;prox coluna
E15F C1      2580      POP  BC
E160 10BE    2590      DJNZ BL6          ;prox linha
E162 E1      2600      POP  HL
E163 FDE1    2610      POP  IY
E165 E1      2620      POP  HL
E166 D1      2630      POP  DE
E167 C1      2640      POP  BC
E168 C9      2650      RET
2660 ;
2670 ; ROTINA SPRTES - TESTA SPRITES
2680 ;
E169 7B      2690 SPRTES: LD   A,B          ;A=coluna
E16A 07      2700      RLCA
E16B 07      2710      RLCA
E16C 07      2720      RLCA          ;A=coord.X
E16D C607    2730      ADD  A,7          ;mais 7 a direita
E16F 47      2740      LD   B,A          ;B=X
E170 79      2750      LD   A,C          ;A=linha
E171 07      2760      RLCA
E172 07      2770      RLCA
E173 07      2780      RLCA          ;A=coord.Y

```

```

E174 C607      2790      ADD  A,7           ;embaixo
E176 4F        2800      LD   C,A           ;C=Y
E177 AF        2810      XOR   A
E178 CD8700    2820  SS1:  CALL  CALATR      ;HL->atributos
E17B 57        2830      LD   D,A
E17C CD4A00    2840      CALL  RDVRM
E17F FED0      2850      CP    200         ;fim?
E181 C8        2860      RET   Z
E182 D5        2870      PUSH  DE
E183 C5        2880      PUSH  BC
E184 CD8FE1    2890      CALL  SPRITE      ;manda sprite
E187 C1        2900      POP   BC
E188 F1        2910      POP   AF
E189 3C        2920      INC   A
E18A FE20      2930      CP    32
E18C 20EA      2940      JR    NZ,SS1      ;prox sprite
E18E C9        2950      RET
6             2960  *E
             2970 ;
             2980 ; ROTINA SPRITE - COLOCA SPRITE NO BLOCO
             2990 ;
E18F 91        3000  SPRITE: SUB   C
E190 2F        3010      CPL
E191 FE27      3020      CP    39           ;sobrepós?
E193 D0        3030      RET   NC
E194 4F        3040      LD   C,A
E195 23        3050      INC   HL
E196 CD4A00    3060      CALL  RDVRM
E199 5F        3070      LD   E,A
E19A 78        3080      LD   A,B           ;A=coord.X
E19B 93        3090      SUB   E
E19C 5F        3100      LD   E,A
E19D 9F        3110      SBC   A,A
E19E 57        3120      LD   D,A
E19F 23        3130      INC   HL
E1A0 CD4A00    3140      CALL  RDVRM      ;no. imagem
E1A3 47        3150      LD   B,A
E1A4 23        3160      INC   HL
E1A5 CD4A00    3170      CALL  RDVRM
E1A8 CB7F      3180      BIT   7,A
E1AA 2805      3190      JR    Z,SP1
E1AC 212000    3200      LD   HL,32
E1AF 19        3210      ADD   HL,DE
E1B0 EB        3220      EX    DE,HL
E1B1 14        3230  SP1:  INC   D
E1B2 15        3240      DEC   D
E1B3 C0        3250      RET   NZ           ;NZ=fora do bloco
E1B4 E60F      3260      AND   #0F        ;cor
E1B6 C8        3270      RET   Z           ;Z=transparente
E1B7 57        3280      LD   D,A
E1B8 3AE0F3    3290      LD   A,(RG1SAV)
E1BB CB4F      3300      BIT   1,A
E1BD 0F        3310      RRCA
E1BE 3E08      3320      LD   A,B
E1C0 3001      3330      JR    NC,SP2
E1C2 87        3340      ADD   A,A
E1C3 2805      3350  SP2:  JR    Z,SP3
E1C5 CB80      3360      RES   0,B
E1C7 CB88      3370      RES   1,B
E1C9 87        3380      ADD   A,A
E1CA 6F        3390  SP3:  LD   L,A
E1CB C606      3400      ADD   A,6
E1CD B9        3410      CP    C
E1CE D8        3420      RET   C           ;sprite acima
E1CF BB        3430      CP    E
E1D0 D8        3440      RET   C           ;sprite 'a esq.
E1D1 79        3450      LD   A,C
E1D2 D607      3460      SUB   7
7             3470  *E
E1D4 4F        3480      LD   C,A

```

```

E1D5 7D      3490      LD  A,L          ;A=tamanho do sprite
E1D6 2600    3500      LD  H,B
E1D8 3808    3510      JR  C,SP5      ;C=abaixo do topo
E1DA 91      3520      SUB  C
E1DB FE09    3530      CP   9
E1DD 3802    3540      JR  C,SP4
E1DF 3E08    3550      LD  A,B
E1E1 67      3560 SP4:  LD  H,A
E1E2 7B      3570 SP5:  LD  A,E
E1E3 D607    3580      SUB  7
E1E5 5F      3590      LD  E,A
E1E6 7D      3600      LD  A,L
E1E7 2E08    3610      LD  L,B
E1E9 3808    3620      JR  C,SP7
E1EB 93      3630      SUB  E
E1EC FE09    3640      CP   9
E1EE 3802    3650      JR  C,SP6
E1F0 3E08    3660      LD  A,B
E1F2 6F      3670 SP6:  LD  L,A
E1F3 FD2151E2 3680 SP7: LD  IY,BUFFB
E1F7 D5      3690 SP8:  PUSH DE
E1F8 CB79    3700      BIT  7,C
E1FA 2048    3710      JR  NZ,SP15     ;alcançou sprite?
E1FC E5      3720      PUSH HL
E1FD FDE5    3730      PUSH IY
E1FF CB7B    3740 SP9:  BIT  7,E
E201 2038    3750      JR  NZ,SP14     ;alcançou sprite?
E203 FD7E00  3760      LD  A,(IY+0)   ;BUFFB
E206 B7      3770      OR   A
E207 2032    3780      JR  NZ,SP14     ;transparente?
E209 C5      3790      PUSH BC
E20A D5      3800      PUSH DE
E20B E5      3810      PUSH HL
E20C 3AE0F3  3820      LD  A,(R6ISAV)
E20F 0F      3830      RRCA
E210 3004    3840      JR  NC,SP10
E212 CB39    3850      SRL  C
E214 CB3B    3860      SRL  E
E216 CB5B    3870 SP10: BIT  3,E
E218 2804    3880      JR  Z,SP11
E21A CB9B    3890      RES  3,E
E21C CBE1    3900      SET  4,C
E21E 68      3910 SP11: LD  L,B
E21F 2600    3920      LD  H,0        ;HL=no. imagem
E221 44      3930      LD  B,H
E222 29      3940      ADD  HL,HL
E223 29      3950      ADD  HL,HL
E224 29      3960      ADD  HL,HL     ;imagem*8
E225 09      3970      ADD  HL,BC
E226 ED4B26F9 3980      LD  BC,(PATBAS)
E22A 09      3990      ADD  HL,BC     ;HL->imagem
E22B CD4A00  4000      CALL RDVRM
E22E 1C      4010      INC  E
      8      4015 *E
E22F 07      4020 SP12: RLCA
E230 1D      4030      DEC  E
E231 20FC    4040      JR  NZ,SP12
E233 3003    4050      JR  NC,SP13     ;NC=pixel 0
E235 FD7200  4060      LD  (IY+0),D
E238 E1      4070 SP13: POP  HL
E239 D1      4080      POP  DE
E23A C1      4090      POP  BC
E23B FD23    4110 SP14: INC  IY
E23D 1C      4120      INC  E        ;p/direita
E23E 2D      4130      DEC  L
E23F 20BE    4140      JR  NZ,SP9
E241 FDE1    4150      POP  IY
E243 E1      4160      POP  HL
E244 110800  4170 SP15: LD  DE,8
E247 FD19    4180      ADD  IY,DE

```

```

E249 D1      4190      POP DE
E24A 0C      4200      INC C           ;p/baixo
E24B 25      4210      DEC H
E24C 20A9    4220      JR NZ,SP8      ;acabou?
E24E C9      4230      RET
           4240
           4250
E24F 0000    4260 PILBRK: DEFW 0      ;pilha p/break
           4270
           4280
           4290 ;-----
           4300 ;- O BUFFER A SEGUIR GUARDA -
           4310 ;- UM BLOCO, ISTO E', OS 64 -
           4320 ;- CODIGOS DE COK DE UMA -
           4330 ;- MATRIZ 8x8 -
           4340 ;-
           4350 ;- CCCCCCCC - BYTES 00-07 -
           4360 ;- CCCCCCCC - BYTES 08-15 -
           4370 ;- CCCCCCCC - BYTES 16-23 -
           4380 ;- CCCCCCCC - BYTES 24-31 -
           4390 ;- CCCCCCCC - BYTES 32-39 -
           4400 ;- CCCCCCCC - BYTES 40-47 -
           4410 ;- CCCCCCCC - BYTES 48-55 -
           4420 ;- CCCCCCCC - BYTES 56-63 -
           4430 ;-
           4440 ;-
           4450 ;-----
           4460
           4470
E251 4480    4480 BUFFB: DEFS 64      ;buffer de bloco
           4490
           4500
E291 4510      END

```

Pass 2 errors: 00

Table used: 700 from 925

Executes: 57344

## Editor de Caracteres

Este programa permite que as imagens dos caracteres do MSX sejam modificadas. Quando o programa é executado pela primeira vez ele copia o conjunto de caracteres de 2KB de sua posição atual (normalmente a ROM do MSX) para o buffer TABCAR (E2A3H a EAA2H).

O programa tem dois níveis de operação: comando e edição, com a tecla RETURN sendo utilizada para alternar entre eles. No modo comando, as quatro teclas de seta são utilizadas para selecionar o caractere para edição. Este é marcado por um grande cursor e é também apresentado em forma ampliada do lado direito da tela. A tecla "Q" sai do programa e volta ao BASIC. A tecla "A" é utilizada para adotar o conjunto de

caracteres, isto é, para torná-lo o conjunto de caracteres do sistema. Quando o conjunto de caracteres é adotado, ele é copiado na parte mais alta da memória (EB80H até F37FH) e a sua Identificação de Conector e seu endereço colocados em CGPNT.

No modo edição, as quatro teclas de seta são utilizadas para selecionar o ponto para ser editado, marcado pelo cursor pequeno. A barra de espaços cancela o ponto atual e a tecla “.” o liga. À medida que a imagem é modificada, o menu de caracteres do lado esquerdo da tela é atualizado.

O conjunto de caracteres em TABCAR poderá ser salvo no cassete utilizando uma instrução “BSAVE” e posteriormente recarregado com uma instrução “BLOAD”. A sub-rotina ADOTA deve ser salva com as imagens e executada no recarregamento, de modo que o sistema adote o novo conjunto de caracteres. Alternativamente, o conjunto de caracteres sozinho poderá ser salvo e a sua Identificação de Conector e endereço colocados em CGPNT no recarregamento utilizando instruções BASIC. Observe que a alteração das imagens de caracteres não afeta a operação do sistema MSX em nada.

Pass 1 errors: 00

```

E000      10      ORG #E000
E000      20      ENT #E000
          30      ;-----
          40      ;--      ROTINAS DA BIOS      --
          50      ;-----
000C      60      RDSLT: EQU #000C
004A      70      RDVRM: EQU #004A
004D      80      WRTVRM: EQU #004D
0056      90      FILVRM: EQU #0056
0072     100      INIGRP: EQU #0072
009C     110      CHSNS: EQU #009C
009F     120      CHGET: EQU #009F
0111     130      MAPXYC: EQU #0111
0114     140      FETCHC: EQU #0114
0138     150      RSLREG: EQU #0138
          160      ;-----
          170      ;--      VARIÁVEIS DA ÁREA      --
          180      ;--      DE TRABALHO      --
          190      ;-----
F3C9     200      GRPCOL: EQU #F3C9
F3E9     210      FORCLR: EQU #F3E9
F3EA     220      BAKCLR: EQU #F3EA
F91F     230      CGPNT: EQU #F91F
FCC1     240      EXPTBL: EQU #FCC1
FCC5     250      SLTTBL: EQU #FCC5
          260      ;-----
          270      ;--      CARACTERES DE CONTROLE      --
          280      ;-----
000D     290      CR: EQU 13
001C     300      DIR: EQU 28
001D     310      ESQ: EQU 29
001E     320      CIMA: EQU 30
001F     330      BAIXO: EQU 31
          340
          350
          360
E000     370      EDITOR: CALL INIC
E003     380      ETI: CALL AUMCAR
E006     390      CFEE1: CALL CARXY

```

;partida fria

```

E009 160B      400      LD      D,8
E00B CD2FE2    410      CALL   TECLA
E00E FE51      420      CP      "Q"
E010 C8        430      RET     Z
E011 2103E0    440      LD      HL,ET1      ;retorno
E014 E5        450      PUSH    HL
E015 FE41      460      CP      "A"
E017 CA6EE2    470      JP      Z,ADOTA
E01A FE0D      480      CP      CR
E01C 2B1F      490      JR      Z,EDITA
E01E 0E01      500      LD      C,1
E020 FE1C      510      CP      DIR
E022 2B11      520      JR      Z,ET2
E024 0EFF      530      LD      C,#FF
E026 FE1D      540      CP      ESQ
E02B 2B0B      550      JR      Z,ET2
2
E02A 0EF0      555 *E    LD      C,#F0
E02C FE1E      560      CP      CIMA
E02E 2B05      580      JR      Z,ET2
E030 0E10      590      LD      C,16
E032 FE1F      600      CP      BAIXO
E034 C0        610      RET     NZ
E035 3AA1E2    620 ET2:   LD      A,(NUMCAR)
E038 B1        630      ADD     A,C
E039 32A1E2    640      LD      (NUMCAR),A      ;novo carac.
E03C C9        650      RET
660 ;
670 ; ROTINA EDITA - EDITA UM CARACTERE
680 ;
E03D CDE6E1    690 EDITA: CALL   PONTO
E040 1602      700      LD      D,2
E042 CD2FE2    710      CALL   TECLA
E045 FE0D      720      CP      CR
E047 C8        730      RET     Z
E04B 213DE0    740      LD      HL,EDITA      ;retorno
E04B E5        750      PUSH    HL
E04C 0100FE    760      LD      BC,#FE00
E04F FE20      770      CP      " "
E051 2B24      780      JR      Z,ED3
E053 0C        790      INC     C
E054 FE2E      800      CP      " "
E056 2B1F      810      JR      Z,ED3
E05B FE1C      820      CP      DIR
E05A 2B11      830      JR      Z,ED2
E05C 0EFF      840      LD      C,#FF
E05E FE1D      850      CP      ESQ
E060 2B0B      860      JR      Z,ED2
E062 0EF8      870      LD      C,#F8
E064 FE1E      880      CP      CIMA
E066 2B05      890      JR      Z,ED2
E06B 0E0B      900      LD      C,B
E06A FE1F      910      CP      BAIXO
E06C C0        920      RET     NZ
E06D 3AA2E2    930 ED2:   LD      A,(NUMPT)
E070 B1        940      ADD     A,C
E071 E63F      950      AND     63
E073 32A2E2    960      LD      (NUMPT),A      ;novo num. ponto
E076 C9        970      RET
E077 CD1EE2    980 ED3:   CALL   POSIMA
E07A 3AA2E2    990      LD      A,(NUMPT)
E07D F5        1000     PUSH    AF
E07E 0F        1010     RRCA
E07F 0F        1020     RRCA
E080 0F        1030     RRCA
E081 E607      1040     AND     7
E083 5F        1050     LD      E,A
E084 1600      1060     LD      D,0      ;DE=linha
E086 FD19      1070     ADD     IV,DE      ;IV->linha
E08B F1        1080     POP     AF

```



```

E089 E607 1090 AND 7
E08B 3C 1100 INC A
E08C CB0B 1110 ED4: RRC B
      3 1115 *E
E08E CB09 1120 RRC C
E090 3D 1130 DEC A ;conta colunas
E091 20F9 1140 JR NZ,ED4
E093 FD7E00 1150 LD A,(IY+0) ;A=imagem
E096 A0 1160 AND B ;remove bit
E097 B1 1170 OR C ;novo bit
E098 FD7700 1180 LD (IY+0),A ;repoe
E09B CDBDE0 1190 CALL AUMCAR
      1200 ;
      1210 ; ROTINA IMPCAR - IMPRIME UM CARACTERE
      1220 ;
E09E CD1EE2 1230 IMPCAR: CALL POSIMA
E0A1 CDFEE1 1240 CALL CARXY
E0A4 CDA3E1 1250 CALL MAPA
E0A7 060B 1260 LD B,B
E0A9 05 1270 IC1: PUSH DE
E0AA E5 1280 PUSH HL
E0AB 3E0B 1290 LD A,B
E0AD FD5E00 1300 LD E,(IY+0)
E0B0 CDC4E1 1310 CALL LINHA
E0B3 E1 1320 POP HL
E0B4 D1 1330 POP DE
E0B5 CDB8E1 1340 CALL PBAIXO
E0B8 FD23 1350 INC IY
E0BA 10ED 1360 DJNZ IC1
E0BC C9 1370 RET
      1380 ;
      1390 ; ROTINA AUMCAR - AUMENTA O CARACTERE
      1400 ;
E0BD CD1EE2 1410 AUMCAR: CALL POSIMA
E0C0 0EBF 1420 LD C,191 ;X inicial
E0C2 1E07 1430 LD E,7 ;Y inicial
E0C4 CDA3E1 1440 CALL MAPA
E0C7 060B 1450 LD B,B
E0C9 0E05 1460 AC1: LD C,5
E0CB C5 1470 AC2: PUSH BC
E0CC 05 1480 PUSH DE
E0CD E5 1490 PUSH HL
E0CE 060B 1500 LD B,B
E0D0 FD7E00 1510 LD A,(IY+0)
E0D3 07 1520 AC3: RLCA ;testa bit
E0D4 F5 1530 PUSH AF
E0D5 9F 1540 SBC A,A ;bit 0=00,1=FF
E0D6 5F 1550 LD E,A
E0D7 3E05 1560 LD A,5
E0D9 CDC4E1 1570 CALL LINHA
E0DC CDAEE1 1580 CALL PDIR
E0DF CDAEE1 1590 CALL PDIR ;pula grade
E0E2 F1 1600 POP AF
E0E3 10EE 1610 DJNZ AC3
E0E5 E1 1620 POP HL
E0E6 D1 1630 POP DE
E0E7 C1 1640 POP BC
      4 1645 *E
E0E8 CDB8E1 1650 CALL PBAIXO
E0EB 0D 1660 DEC C
E0EC 20DD 1670 JR NZ,AC2
E0EE CDB8E1 1680 CALL PBAIXO ;pula grade
E0F1 FD23 1690 INC IY
E0F3 10D4 1700 DJNZ AC1
E0F5 C9 1710 RET
      1720 ;
      1730 ; ROTINA INIC - INICIALIZA
      1740 ;
E0F6 01000B 1750 INIC: LD BC,204B
E0F9 11A3E2 1760 LD DE,TABCAR

```

```

E0FC 2A20F9 1770 LD HL, (CGPNT+1)
E0FF C5 1780 IN1: PUSH BC
E100 D5 1790 PUSH DE
E101 3A1FF9 1800 LD A, (CGPNT)
E104 CD0C00 1810 CALL RDSLT ;le carac
E107 FB 1820 EI
E108 D1 1830 POP DE
E109 C1 1840 POP BC
E10A 12 1850 LD (DE),A ;poe no buffer
E10B 13 1860 INC DE
E10C 23 1870 INC HL
E10D 0B 1880 DEC BC
E10E 7B 1890 LD A,B
E10F B1 1900 OR C
E110 20ED 1910 JR NZ, IN1
E112 CD7200 1920 CALL INIGRP ;SCREEN 2
E115 3AE9F3 1930 LD A, (FORCLR)
E118 07 1940 RLCA
E119 07 1950 RLCA
E11A 07 1960 RLCA
E11B 07 1970 RLCA
E11C 4F 1980 LD C,A
E11D 3AEAF3 1990 LD A, (BAKCLR)
E120 B1 2000 OR C ;mistura cores
E121 010018 2010 LD BC, 6144
E124 2AC9F3 2020 LD HL, (GRPCOL)
E127 CD5600 2030 CALL FILVRM
E12A 210BB1 2040 LD HL, #B10B ;177*256+11
E12D 010AFF 2050 LD BC, #FF0A ;#FF*256+10
E130 1E06 2060 LD E, 6
E132 3E11 2070 LD A, 17
E134 CD62E1 2080 CALL GRADE ;grade caracs.
E137 210631 2090 LD HL, #3106 ;49*256+6
E13A 01BEAA 2100 LD BC, #AABE ;#AA*256+190
E13D 1E06 2110 LD E, 6
E13F 3E09 2120 LD A, 9
E141 CD62E1 2130 CALL GRADE ;grade carac.grande
E144 213031 2140 LD HL, #3130 ;49*256+48
E147 01BEFF 2150 LD BC, #FFBE ;#FF*256+190
E14A 1E06 2160 LD E, 6
E14C 3E02 2170 LD A, 2
E14E CD62E1 2180 CALL GRADE ;caixa carac.
E151 AF 2190 XOR A
E152 32A2E2 2200 LD (NUMPT),A
5 2205 *E
E155 21A1E2 2210 LD HL, NUMCAR
E158 77 2220 LD (HL),A ;zera NUMPT e NUMCAR
E159 E5 2230 IN2: PUSH HL
E15A CD9EE0 2240 CALL IMPCAR ;imprime carac.
E15D E1 2250 POP HL
E15E 34 2260 INC (HL) ;proximo carac.
E15F 20FB 2270 JR NZ, IN2
E161 C9 2280 RET
2290 ;
2300 ; ROTINA GRADE - DESENHA GRADE
2310 ;
E162 F5 2320 GRADE: PUSH AF
E163 C5 2330 PUSH BC
E164 E5 2340 PUSH HL
E165 CDA3E1 2350 CALL MAPA
E168 C1 2360 POP BC
E169 F1 2370 POP AF
E16A 5F 2380 LD E,A
E16B F1 2390 POP AF
E16C F5 2400 PUSH AF
E16D D5 2410 PUSH DE
E16E E5 2420 PUSH HL
E16F F5 2430 GR1: PUSH AF
E170 C5 2440 PUSH BC
E171 D5 2450 PUSH DE

```

```

E172 E5      2460      PUSH HL
E173 78      2470      LD A,B
E174 CDC4E1  2480      CALL LINHA      ;linha horizontal
E177 E1      2490      POP HL
E178 D1      2500      POP DE
E179 CDB8E1  2510 GR3:  CALL PBAIXO
E17C 0D      2520      DEC C          ;vai descendo
E17D 20FA    2530      JR NZ,GR3
E17F C1      2540      POP BC
E180 F1      2550      POP AF
E181 3D      2560      DEC A          ;todas as linhas?
E182 20EB    2570      JR NZ,GR1
E184 E1      2580      POP HL
E185 D1      2590      POP DE
E186 F1      2600      POP AF
E187 F5      2610 GR4:  PUSH AF
E188 C5      2620      PUSH BC
E189 D5      2630      PUSH DE
E18A E5      2640      PUSH HL
E18B 3E01    2650 GR5:  LD A,1
E18D CDC4E1  2660      CALL LINHA
E190 CDB8E1  2670      CALL PBAIXO
E193 10F6    2680      DJNZ GR5      ;compr.vertical
E195 E1      2690      POP HL
E196 D1      2700      POP DE
E197 CDAEE1  2710 GR6:  CALL PDIR
E19A 0D      2720      DEC C          ;continua p/dir.
E19B 20FA    2730      JR NZ,GR6
E19D C1      2740      POP BC
E19E F1      2750      POP AF
E19F 3D      2760      DEC A
E1A0 20E5    2770      JR NZ,GR4
E1A2 C9      2780      RET
6          2785 *E
          2790 ;
          2800 ; ROTINA MAPA - MAPEIA COORDENADAS
          2810 ;
E1A3 0600    2820 MAPA:  LD B,0
E1A5 50      2830      LD D,B
E1A6 CD1101  2840      CALL MAPXYC
E1A9 CD1401  2850      CALL FETCHC      ;HL=CLOC
E1AC 57      2860      LD D,A          ;D=CMASK
E1AD C9      2870      RET
          2880 ;
          2890 ; ROTINA PDIR - MOVE P/DIREITA
          2900 ;
E1AE CB0A    2910 PDIR:  RRC D
E1B0 D0      2920      RET NC          ;fica no bloco
E1B1 C5      2930 PD1:  PUSH BC
E1B2 010800  2940      LD BC,8
E1B5 09      2950      ADD HL,BC      ;HL=prox.bloco
E1B6 C1      2960      POP BC
E1B7 C9      2970      RET
          2980 ;
          2990 ; ROTINA PBAIXO - MOVE P/BAIXO
          3000 ;
E1B8 23      3010 PBAIXO: INC HL
E1B9 7D      3020      LD A,L
E1BA E607    3030      AND 7
E1BC C0      3040      RET NZ          ;mesmo bloco
E1BD C5      3050      PUSH BC
E1BE 01FB00  3060      LD BC,00FB
E1C1 09      3070      ADD HL,BC      ;HL=prox bloco
E1C2 C1      3080      POP BC
E1C3 C9      3090      RET
          3100 ;
          3110 ; ROTINA LINHA - ESCRIVE LINHA
          3120 ;
E1C4 C5      3130 LINHA: PUSH BC
E1C5 47      3140      LD B,A

```

```

E1C6 CD4A00 3150 LI1: CALL RDVRM ;pega imagem
E1C9 4F 3160 LI2: LD C,A
E1CA 7A 3170 LD A,D
E1CB 2F 3180 CPL
E1CC A1 3190 AND C ;remove bit
E1CD CB03 3200 RLC E
E1CF 3001 3210 JR NC,LI3 ;pixel 0
E1D1 B2 3220 OR D
E1D2 05 3230 LI3: DEC B
E1D3 200C 3240 JR Z,L14 ;terminou?
E1D5 CB0A 3250 RRC D
E1D7 30F0 3260 JR NC,L12 ;mesmo bloco
E1D9 CD4D00 3270 CALL WRTVRM
E1DC CDB1E1 3280 CALL PD1 ;prox bloco
E1DF 18E5 3290 JR LI1
E1E1 CD4D00 3300 LI4: CALL WRTVRM
E1E4 C1 3310 POP BC
E1E5 C9 3320 RET
7 3325 *E
3330 ;
3340 ; ROTINA PONTO - COORDENADAS DE UM PONTO
3350 ;
E1E6 3AA2E2 3360 PONTO: LD A,(NUMPT)
E1E9 F5 3370 PUSH AF
E1EA E607 3380 AND 7 ;coluna
E1EC 07 3390 RLCA
E1ED 4F 3400 LD C,A
E1EE 07 3410 RLCA
E1EF 81 3420 ADD A,C ;A=col*6
E1F0 C6BF 3430 ADD A,191
E1F2 4F 3440 LD C,A ;C=coord.X
E1F3 F1 3450 POP AF
E1F4 E638 3460 AND 038 ;linha*8
E1F6 0F 3470 RRCA
E1F7 5F 3480 LD E,A
E1F8 0F 3490 RRCA
E1F9 83 3500 ADD A,E ;A=linha*6
E1FA C607 3510 ADD A,7
E1FC 5F 3520 LD E,A ;E=coord.Y
E1FD C9 3530 RET
3540 ;
3550 ; ROTINA CARXY - COORDENADAS DO CARACTERE
3560 ;
E1FE 3AA1E2 3570 CARXY: LD A,(NUMCAR)
E201 F5 3580 PUSH AF
E202 CD14E2 3590 CALL MULT11
E205 C60C 3600 ADD A,12
E207 4F 3610 LD C,A ;C=coord.X
E208 F1 3620 POP AF
E209 0F 3630 RRCA
E20A 0F 3640 RRCA
E20B 0F 3650 RRCA
E20C 0F 3660 RRCA
E20D CD14E2 3670 CALL MULT11
E210 C608 3680 ADD A,B
E212 5F 3690 LD E,A ;E=coord.Y
E213 C9 3700 RET
3710 ;
3720 ; ROTINA MULT11 - MULTIPLICA A POR 11
3730 ;
E214 E60F 3740 MULT11: AND 00F
E216 57 3750 LD D,A
E217 07 3760 RLCA
E218 47 3770 LD B,A
E219 07 3780 RLCA
E21A 07 3790 RLCA
E21B 80 3800 ADD A,B
E21C 82 3810 ADD A,D
E21D C9 3820 RET

```

```

8          3825 *E
          3830 ;
          3840 ; ROTINA POSIMA - POSICAO DA IMAGEM
          3850 ;
E21E 3AA1E2 3860 POSIMA: LD A,(NUMCAR)
E221 6F      3870 LD L,A
E222 2600    3880 LD H,0
E224 29      3890 ADD HL,HL
E225 29      3900 ADD HL,HL
E226 29      3910 ADD HL,HL
E227 EB      3920 EX DE,HL ;DE=NUMCAR*8
E228 FD21A3E2 3930 LD IY,TABCAR
E22C FD19    3940 ADD IY,DE ;IY->pos.imagem
E22E C9      3950 RET
          3960 ;
          3970 ; ROTINA TECLA - PEGA TECLA
          3980 ;
E22F 0600    3990 TECLA: LD B,0
E231 C5      4000 TE1:  PUSH BC
E232 D5      4010 PUSH DE
E233 CD50E2  4020 CALL INVERT ;muda cursor
E236 D1      4030 POP DE
E237 C1      4040 POP BC
E238 04      4050 INC B ;pisca
E239 21401F  4060 LD HL,8000
E23C CD9C00  4070 TE2:  CALL CHSNS
E23F 2007    4080 JR NZ,TE3 ;NZ=tem tecla
E241 2B      4090 DEC HL
E242 7C      4100 LD A,H
E243 B5      4110 OR L
E244 20F6    4120 JR NZ,TE2
E246 18E9    4130 JR TE1
E248 CB40    4140 TE3:  BIT 0,B
E24A C450E2  4150 CALL NZ,INVERT ;tira cursor
E24D C39F00  4160 JP CHGET ;pega carac
          4170 ;
          4180 ; ROTINA INVERT
          4190 ;
E250 D5      4200 INVERT: PUSH DE
E251 CDA3E1  4210 CALL MAPA
E254 F1      4220 POP AF
E255 47      4230 LD B,A
E256 5F      4240 LD E,A
E257 D5      4250 IV1:  PUSH DE
E258 E5      4260 PUSH HL
E259 CD4A00  4270 IV2:  CALL RDVRM ;le
E25C AA      4280 XOR D ;inverte
E25D CD4D00  4290 CALL WRTVRM ;escreve
E260 CDAEE1  4300 CALL PDIR
E263 1D      4310 DEC E ;todos dir?
E264 20F3    4320 JR NZ,IV2
E266 E1      4330 POP HL
E267 D1      4340 POP DE
E268 CD88E1  4350 CALL PBAIXO
E26B 10EA    4360 DJNZ IV1 ;todos baixo?
E26D C9      4370 RET
          4375 *E
9          4380 ;
          4390 ; ROTINA ADOTA - ADOTA O CONJUNTO DE CARACTERES
          4400 ;
E26E 01000B  4410 ADOTA: LD BC,2048 ;tamanho
E271 1180EB  4420 LD DE,0EB80 ;destino
E274 ED5320F9 4430 LD (CGPNT+1),DE
E27B 21A3E2  4440 LD HL,TABCAR ;fonte
E27B ED80    4450 LDIR
E27D CD3801  4460 CALL RSLREG
E280 07      4470 RLCA
E281 07      4480 RLCA
E282 E603    4490 AND 3 ;seleciona pag.3
E284 4F      4500 LD C,A

```

|      |        |      |         |           |                        |
|------|--------|------|---------|-----------|------------------------|
| E285 | 0600   | 4510 | LD      | B,0       |                        |
| E287 | 21C1FC | 4520 | LD      | HL,EXPTBL |                        |
| E28A | 09     | 4530 | ADD     | HL,BC     |                        |
| E28B | CB7E   | 4540 | BIT     | 7,(HL)    | ;expandido?            |
| E28D | 280E   | 4550 | JR      | Z,AD1     | ;Z=nao                 |
| E28F | 21C5FC | 4560 | LD      | HL,SLTIBL |                        |
| E292 | 09     | 4570 | ADD     | HL,BC     |                        |
| E293 | 7E     | 4580 | LD      | A,(HL)    |                        |
| E294 | 07     | 4590 | RLCA    |           |                        |
| E295 | 07     | 4600 | RLCA    |           |                        |
| E296 | 07     | 4610 | RLCA    |           |                        |
| E297 | 07     | 4620 | RLCA    |           |                        |
| E298 | E60C   | 4630 | AND     | #0C       | ;A=num.conec.sec.pag.3 |
| E29A | B1     | 4640 | OR      | C         | ;junta c/prim.         |
| E29B | CBFF   | 4650 | SET     | 7,A       |                        |
| E29D | 321FF9 | 4660 | LD      | (CGPNT),A |                        |
| E2A0 | C9     | 4670 | RET     |           |                        |
|      |        | 4680 |         |           |                        |
|      |        | 4690 |         |           |                        |
|      |        | 4700 |         |           |                        |
| E2A1 | 00     | 4710 | NUMCAR: | DEFB 0    | ;num.carac.atual       |
| E2A2 | 00     | 4720 | NUMPT:  | DEFB 0    | ;num.ponto.atual       |
| E2A3 |        | 4730 | TABCAR: | DEFS 204B | ;conj.carac.           |
|      |        | 4740 |         |           |                        |
| EAA3 |        | 4760 | END     |           |                        |

Pass 2 errors: 00

Table used: 786 from 932  
Executes: 57344

## ÍNDICE ANALÍTICO

### A

Ângulos, 183  
 Apontadores, 38, 141, 166  
 Área de trabalho, 37, 250  
 Arg, 109, 270  
 Armazenamento de Matriz, 186, 194, 250  
 Arquivo, bloco de controle (FCB), 79, 251  
 ARYTAB, 165, 250, 268  
 "ASB", 117  
 "ASC", 201  
 ASPCT1, 93, 259  
 ASPCT2, 93, 259  
 ASPECT, 179, 273  
 "ATN", 112  
 Átomos (Token), 132, 140, 141, 144, 164  
 ATRBAS, 40, 45, 48, 275  
 ATRBYT, 83, 86, 92, 173, 257  
 "ATTR\$", 243  
 AUTFLG, 139, 190, 264  
 AUTINC, 139, 264  
 AUTLIN, 139, 264  
 "AUTO", 150

### B

BAKCLR, 46, 48, 257  
 "BASE", 240, 241

BASIC, palavras-chaves do, 132  
 BASROM, 37, 103, 279  
 Baud, frequência, 95, 197  
 BDRATR, 93, 286  
 BDRCLR, 48, 257  
 "BEEP", 72  
 "BINS", 128, 197  
 "BLOAD", 212  
 BOTTOM, 245, 282  
 "BSAVE", 212  
 BUF, 102, 152, 211, 261

### C

"CALL", 144, 168  
 Caps LED, 5, 68  
 CAPST, 68, 285  
 CASPRV, 220, 285  
 CASSETE  
   Entrada de, 27, 97  
   Motor de, 5, 77, 94  
   Saída de, 5, 96  
 "CDBL", 121  
 CENCNT, 179, 273  
 CGPBAS, 40, 272  
 CGPNT, 46, 245, 271, 309  
 "CHR\$, 201  
 "CINT", 120

"CIRCLE", 93, 179  
 "CLEAR", 195  
 CLIKFL, 69, 281  
 CLIKSW, 69, 237, 255  
 CLINEF, 182, 273  
 CLMLST, 151, 254  
 "CLOAD", 215  
 CLOC, 84, 272  
 "CLOSE", 206  
 CLPRIM, 35, 252  
 "CLS", 49  
 CMASK, 84, 92, 272  
 "CMD", 243  
 CNPNTS, 179, 273  
 CNSDFG, 59, 256  
 Códigos de controle, 52  
 Coincidência, 9, 62  
 "COLOR", 236  
 Corrector.  
     ID – Identificação de 'Slot ID)(Espaço ID),  
         34, 168, 246  
     Primário, 1, 34, 36, 79  
     Secundário (espaço secundário), 3, 34, 36  
 Conjunto de caracteres, 46, 100, 308  
 CONLO, 144, 262  
 CONSAV, 144, 262  
 "CONT", 138, 193  
 CONTXT, 144, 262  
 CONTYP, 144, 262  
 Coordenadas, gráficas, 84, 172  
 Coordenadas de texto, 50, 56, 60  
 "COPY", 243  
 Cores, 13, 46, 48, 93  
 Corte (Clipping, 84  
 "COS", 111  
 CPCNT8, 182, 274  
 CPLOT8, 179, 273  
 (Crash), quebra do sistema, 34  
 CRCSUM, 180, 274  
 CRTCNT, 56, 61, 254  
 "CSAVE", 214  
 CSAVEA, 93, 179, 274  
 CSAVEM, 93, 179, 274  
 CSCLXY, 179, 274  
 CS 1200, 237, 258

CS 2400, 237, 258  
 "CSNG", 120  
 "CSRLIN", 235  
 CSRSW, 54, 284  
 CSRSX, 52, 54, 255  
 CSRY, 52, 54, 255  
 CSTCNT, 179, 274  
 CSTYLE, 54, 102, 285  
 CTRL-STOP, 37, 38, 49, 68, 71, 95  
 CURLIN, 67, 137, 138, 260  
 CURSAV, 55, 280  
 Cursor, 14, 38, 51, 55  
 "CVD", 245  
 "CVI", 244  
 "CVS", 245  
 CXOFF, 181, 274  
 CYOFF, 181, 275

## D

DAC, 109, 113, 127, 158, 201  
 Dados, áreas de, 30  
 "DATA", 148  
 DATLIN, 137, 153, 263  
 DATPTR, 190, 267  
 "DEFDBL", 145  
 "DEFFN", 161  
 "DEFINT", 145  
 "DEFSNG", 145  
 "DEFSTR", 145  
 "DEFSUR", 161  
 "DEFTBL", 145, 267  
 "DELETE", 165  
 Desempilhamento, (Retirar pacotes da fila)  
     73  
 DEVICE, 170, 289  
 Digitalizador (touchpad), 27, 77  
 "DIM", 185  
 DIMFLG, 185, 261  
 Dispositivo, 168, 204, 213  
 DOT, 139, 265  
 "DRAW", 170, 183  
 DRWANG, 184, 286  
 DRWFLG, 184, 286



DRWSCL, 185, 286

"DSKF", 243

"DSKIS", 243

"DSKOS", 242

## E

Edição, teclas de, 102

Editor, 107

"ELSE", 148, 150

"END", 193

ENDFOR, 142, 263

ENSTOP, 63, 279

"EOF", 209

"ERASE", 194

"ERL", 156

"ERR", 156

ERRFLG, 138, 156, 259

ERRLIN, 138, 149, 156, 265

ERRO

geradores de, 138, 211

manipulador de, 138

mensagens de, 135

"ERROR", 149

ERRTXT, 138, 149, 265

E/S,

buffer de, 79, 150, 156, 205

despachante de, 214

ESCCNT, 53, 284

Espaço da pilha (stack space), 190

Espaço -ID- Identificação de corrector,  
34, 168, 246

Estado de espera (wait state), 97

"EXP", 113

Expansores, 3

Expressão, avaliador de, 122, 135, 154

EXPTBL, 3, 37, 287

Extensão, ROM de, 35, 168, 213, 246

## F

Fatores, avaliador de, 117, 134, 155

FBUFR, 127, 164, 269

"FIELD", 244

Fila, 72, 81, 82, 226

"FILES", 207

Filespec, 204

FILNAM, 205, 216, 271

FILNM2, 217, 271

FILTAB, 250, 270

"FIX", 121

FLGINP, 152, 264

"FN", 161

FNKFLG, 66, 280

FNKSTR, 78, 271

"FOR", 142

FORCLR, 48, 256

"FPOS", 209

"FRE", 204

Frequência de Baud, 95, 197

FRETOP, 190, 250, 263

FSTPOS, 101, 106, 280

FUNACT, 161, 269

FUNACT, 191, 269

Função(ões),

Apresentação das teclas de,

Endereços das,

Teclas de, 66, 69, 78

## G

Ganchos, 34, 245, 289

"GET", 232

GETPNT, 38, 71, 258

"GOSUB", 147

"GOTO", 142, 146

Gráfica, saída, 83

Gráficos, caracteres, 51

GRPACX, 83, 172, 286

GRPACY, 83, 172, 286

GRPATR, 42, 255

GRPHED, 51, 284

GRPNAM, 42, 255

GRPPAT, 51, 284

GXPOS, 172, 286

GYPOS, 172, 286

## H

HEADER, 95, 259

"HEX\$", 129, 197

HIMEM, 195, 245, 248, 250, 282

## I

ID - Espaço (Identificação de corrector)

34, 168, 246

"IF", 150

Impressora, 49, 99

Inflexões de fronteira, 177

"INKEY\$", 224

"INP", 136

"INPUT", 152

"INPUT\$", 208

INSFLG, 102, 234

"INSTR", 203

Instruções,

endereços das, 131, 163

"INT", 121, 183

INTCNT, 62, 284

Interpretador,

Saída do, 99

Interrupção, modo de, 37

Interrupções, 9, 62, 67, 192

"INTERVAL", 62, 233

INTFLG, 38, 68, 283

INTVAL, 62, 283

"IPL", 243

## J

Japonês (esas), 2, 30, 70, 106

JIFEY, 62, 235, 283

Joystick, 27, 62, 75

## K

KANAMD, 39, 285

KANAST, 68, 285

Kansas City, 95

KBUF, 139, 142, 261

"KEY", 234

KEYBUF, 38, 67, 281

"KILL", 243

## L

"LED Caps", 5, 68

"LEFT\$", 202

"LEN", 201

"LET", 142, 153

"LFILES", 207

"LINE", 172

"LINE INPUT", 151

Linhas, números das, 140, 141, 143, 146

Elos, 140

LINL 32, 40, 253

LINL 40, 40, 163, 253

LINLEN, 40, 163, 254

LINTTB, 46, 101, 280

LINWRK, 55, 281

"LIST", 164

"LLIST", 164

"LOAD", 206

"LOC", 209

"LOCATE", 232

"LOF", 209

"LOG", 112

LOWLIN, 97, 284

"LPOS", 159

"LPRINT", 150

LPTPOS, 99, 151, 159, 260

"LSET", 244

## M

Macro, analisador gramatical (Macroparser),  
170

Matemáticas(os),

Constantes, 116

Operadores, 134

MAXDEL, 176, 273

MAXFIL, 205, 270

"MAXFILES", 248

MAXUPD, 176, 257

MCLFLG, 170, 226, 276

MCLLEN, 170, 226, 278

MCLPTR, 170, 226, 278

MCLTAB, 170, 226, 276

MEMS12, 190, 195, 250, 262

“MERGE”, 206  
 “MIDS”, 202, 203  
 MINDEL, 176, 272  
 MINUPD, 176, 257  
 “MKD\$”, 244  
 “MKI\$”, 244  
 “MK\$\$”, 244  
 MLTATR, 43, 255  
 MLTNAM, 42, 255  
 MLTPAT, 43, 255  
 “MOTOR”, 225  
 MOVcnt, 179, 275  
 Musical, pacote, 72, 73, 232  
 MUSICF, 62, 74, 278

## N

NAMBAS, 40, 45, 272  
 “NAME”, 242  
 “NEW”, 190  
 NEWKEY, 63, 67, 281  
 Newton-Raphson, 112  
 “NEXT”, 196  
 NLONLY, 190, 271  
 NOFUNS, 162, 187, 269  
 NTMSXP, 100, 237, 260  
 NULBUF, 250, 270  
 Numérica.(os),  
     Saída, 127  
     Tipos, 126

## O

“OCT\$”, 128, 197  
 OLDKEY, 38, 63, 281  
 OLDLIN, 138, 193, 266  
 OLDSCR, 40, 163, 285  
 OLDTXT, 71, 138, 190, 193, 266  
 “ON”, 148  
 ONEFLG, 137, 148, 190, 266  
 ONELIN, 138, 148, 190, 265  
 ONG SBF, 67, 191, 280  
 “OPEN”, 205  
 “OUT”, 137

## P

“PAD”, 236  
 Paddle, 28, 76  
 PADX, 77, 283  
 PADY, 77, 283  
 Página, 1  
 “PAINT”, 93, 179  
 PARM1, 161, 187, 268  
 PARM2, 161, 268  
 Partida (Power-up), 38, 245  
     a quente, 63  
 PATBAS, 40, 44, 272  
 PATWRK, 83, 281  
 “PDL”, 236  
 “PEEK”, 165  
 “PLAY”, 39, 226, 235  
 PLYCNT, 75, 227, 278  
 “POINT”, 173  
 “POKE”, 163  
 Polinômio, 115  
 “POS”, 159  
 Precedência, 134, 155  
 “PRESET”, 173  
 “PRINT”, 150  
 PRMLEN, 191, 268  
 PRMLN2, 191, 268  
 PRMSTK, 191, 267  
 PROCNM, 168, 289  
 Programa, armazenamento de, 140, 250  
 PRSCNT, 226, 277  
 PRTFLG, 99, 150, 260  
 “PSET”, 173  
 PSG, 22, 39, 72, 229  
 PTRFIL, 79, 150, 271  
 “PUT”, 232  
 PUTPNT, 38, 63, 64, 69, 258

## Q

QUEBAK, 81, 277  
 Quebra do sistema (Crash), 34  
 QUETAB, 39, 81, 277  
 QUEVEN, 75, 278

QUEVES, 81, 257

## R

RDPRIM, 34, 252  
 "READ", 137, 153  
 "REM", 148  
 "RENUM", 166  
 REPENT, 63, 258  
 "RESTORE", 192  
 "RESUME", 137, 149  
 "RETURN", 148  
 RGOSAV, 41, 256  
 "RIGHT\$", 202  
 "RND", 113  
 RNDX, 113, 190, 270  
 Ronda,  
     de execução, 143  
     principal, 138  
 Rotinas-padrão, 29  
 "RSET", 244  
 "RUN", 146  
 RUNBFN, 212, 287

## S

"SAVE", 206  
 SAVENT, 215, 287  
 SAVSTK, 138, 143, 190, 265  
 SAVTXT, 143, 264  
 SCNCNT, 62, 258  
 "SCREEN", 237  
 SCRMOD, 30, 40, 51, 163, 285  
 "SET", 242  
 "SGN", 117  
 "SIN", 183  
 SKPCNT, 179, 275  
 SLTATR, 168, 247, 288  
 SLTTBL, 37, 287  
 "SOUND", 225  
 "SPACES", 202  
 "SPRITE", 233, 237, 238  
 Sprites, 9, 20, 44, 238  
 "SQR", 112

Stack space (Espaço da pilha), 190  
 STATFL, 62, 256  
 "STICK", 235  
 STKTOP, 191, 195, 250-251, 262  
 "STOP", 193, 233  
 "SRTR\$", 197  
 STREND, 165, 250, 266  
 "STRING", 233, 236  
 "STRINGS", 201  
     armazenamento de, 160, 195, 198, 200  
 SUBFLG, 186, 264  
 "SWAP", 194  
 SWPTMP, 194, 269

## T

"TAN", 112  
 Tecla(s)-Chave(s),  
     click de, 5, 68, 69  
     morta, 66, 68  
     números de, 64  
 Teclado, 4, 63, 66, 79  
     entrada de, 64, 71  
 Tela, saída na (screen-out), 51, 83  
 TEMPPT, 190, 263  
 TEMPST, 148, 263  
 "TIME", 235  
 Token (átomos), 132, 140, 141, 144, 164  
 TRCFLG, 143, 269  
 TRGFLG, 63, 256  
 "TROFF", 194  
 "TRON", 193  
 TRPTBL, 38, 63, 67, 191, 232, 282  
 Truques de programação (Assembler), 108  
 T32ATR, 40, 254  
 T32GCP, 40, 254  
 T32NAM, 40, 254  
 T32PAT, 40, 254  
 TTYPOS, 51, 151, 159, 261  
 TXTCGP, 40, 254  
 TXTNAM, 40, 41, 254  
 TXTTAB, 140, 245, 250, 262  
 Tipos (Types), 168

**U**

“USR”, 159

USRTAB, 159, 253

**V**

“VAL”, 203

VALTYP, 119, 121, 126, 155, 168, 261

Variáveis, armazenamento de, 162, 186, 251

“VARPTR”, 156

VARTAB, 165, 250, 266

VCBA, 39, 74, 279

“VDP”, 239, 240

modos do, 10, 13, 40, 48

registrador de endereço do, 8, 48

registrador de estado, 8, 62, 74

registrador de modo, 9, 39

temporização do, 8, 62, 74

“VPEEK”, 242

“VPOKE”, 241

VLZADR, 138, 260

VLZDAT, 138, 260

**W**

“WAIT”, 137

Wait state (estado de espera), 97

“WIDTH”, 163

WINWID, 97, 284

WRPRIM, 34, 252

**Z**

Z80, clock do, 97

## OUTROS LIVROS NA ÁREA

- Burd** – Simulações no MSX
- Burd/Moreira** – MSX - Jogos - 3 volumes
- Burd/Moreira** – MSX - Comandos Básicos - Guia do Operador
- Bussab** – MSX Música
- Carvalho** – Assembler para o MSX
- Casari** – MSX com Disk Drive
- Hoffman** – MSX - Guia do Usuário