MANUAL DO USUÁRIO

HOME COMPUTER HOTBIT HB-8000

INSTRUÇÕES DE USO

HOTBIT

APRESENTAÇÃO

Parabéns! Você acaba de adquirir um poderoso e versátil auxiliar, para suas tarefas diárias que temos absoluta certeza o ajudará em seus projetos de estudo, trabalho e diversão.

Permita-nos apresentar a você, usuário, o HB-8000, nosso microcomputador pessoal, que se tornará o seu companheiro, inseparável e juntos, lado a lado, irão trilhar e desvendar o maravilhoso caminho da informática.

Este manual tem a finalidade de introduzí-lo na operação do HB-8000, sendo que o seu interesse em pesquisá-lo é que o tornará um hábil operador.

No 1º capítulo são apresentados conceitos gerais para conhecimento do produto e cuidados para com o mesmo. O 2º lhe fornece uma introdução ao Basic, que é a linguagem mais simples e difundida para os computadores pessoais; e finalmente o 3º e último capítulo constitui-se de apêndices, onde toda a informação está condensada de forma natural e de fácil acesso.

Sendo assim, leia atentamente cada capítulo deste manual, pesquise, conheça-o e depois não hesite, divirta-se!

Índice

1.1. INTRODUÇÃO

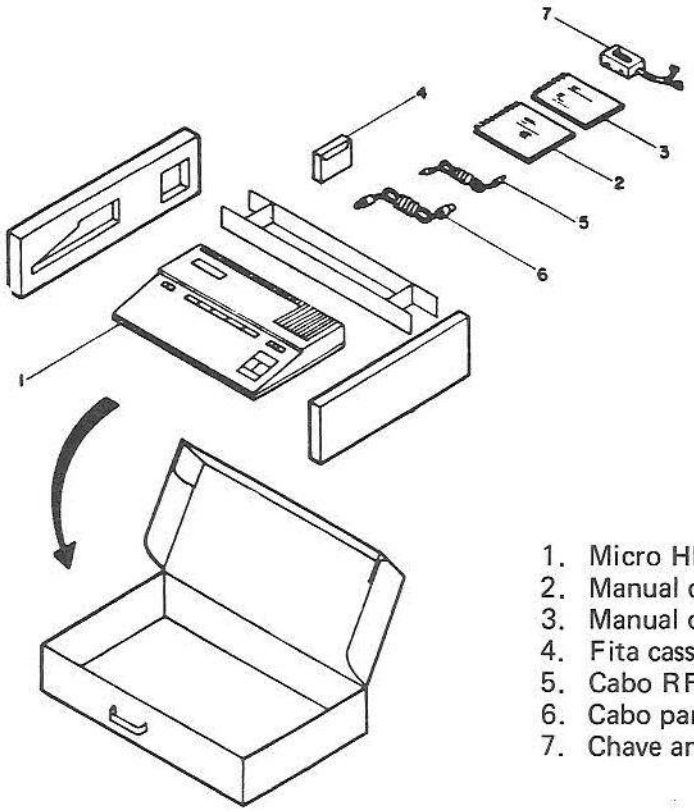
A partir de agora você começará a familiarizar-se com o seu microcomputador HB-8000.

A seguir regras gerais de uso, apresentação do computador e cuidados com o o novo equipamento, que devem ser seguidas para permitir sua máxima satisfação com o uso deste produto.

1.1.1. DESEMBALAGEM

A figura 1 mostra a embalagem onde está acomodado o seu computador e periféricos.

Verifique se todos os ítens descritos na figura encontram-se na mesma.



1. Micro HB-8000
2. Manual de Programação
3. Manual de Instruções
4. Fita cassete
5. Cabo RF
6. Cabo para gravador
7. Chave antena

1.2. DESCRIÇÃO E INSTALAÇÃO DO PRODUTO

1.2.1. MICROCOMPUTADOR

Descrição:

PARTE SUPERIOR (Fig. 2)

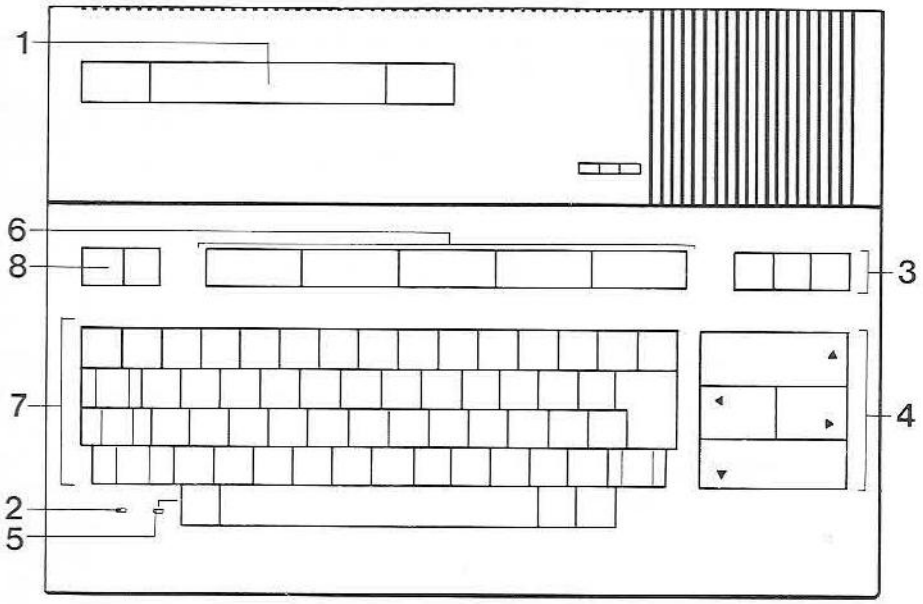


Fig. 2: Parte superior do produto

1 - SLOT — Fenda de inserção de cartuchos (jogos, expansão de memória, etc.).

2 - LÂMPADA DA FONTE — A lâmpada é acesa ao ligarmos o computador através do interruptor.

3 - TECLAS DE EDIÇÃO — Teclas que servem para realizar correções ou modificações nos caracteres presentes no vídeo.

4 - TECLAS DO CURSOR — Usadas para movimentar o cursor para cima, para direita, esquerda e para baixo.

5 - LÂMPADA DE CAPS — Quando essa lâmpada está acesa, as letras escritas são maiúsculas. Para apagá-la, pressione a tecla CAPS

6 - TECLAS DE FUNÇÃO — Pressionando uma dessas teclas é introduzido no computador uma cadeia de caracteres pré-definida.

7 - TECLAS DE CARACTERES — Utilizadas para introduzir os comandos e dados das instruções.

8 - TECLA STOP — Detém o programa.

PARTE TRASEIRA (Fig. 3)

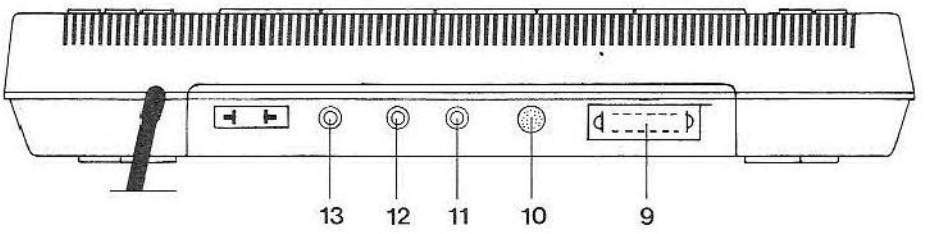


Fig. 3: Parte traseira do equipamento

9- CONECTOR PARA IMPRESSORA — Ponto para conexão de uma impressora (usada para listar programas ou dados).

10- CONECTOR PARA GRAVADOR — Ponto para conexão do computador com um gravador de áudio comum para entrada ou saída de dados (carregar programas).

11- TERMINAL PARA AMPLIFICADOR — Terminal de saída de som do computador para o monitor de vídeo ou para um amplificador de alta fidelidade.

12- TERMINAL PARA MONITOR DE VÍDEO — Terminal de saída para um monitor de vídeo.

13- TERMINAL DE SAÍDA RF — Ligação entre o computador e um aparelho de televisão branco e preto ou colorido. (O aparelho de TV serve como monitor de vídeo, para a emissão de desenhos, caracteres e sons).

VISTAS LATERAIS (Fig. 4)

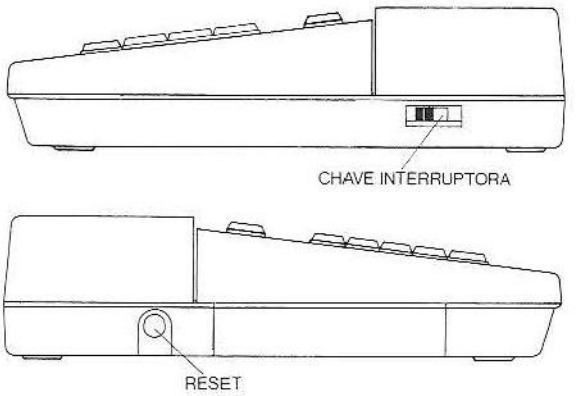


Fig. 4: Vistas laterais

VISTA **FRONTAL** (Fig. 5)

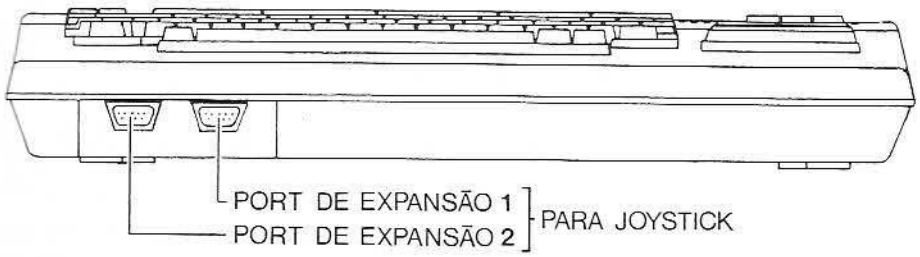


Fig. 5: Vista frontal

PARTE INFERIOR (Fig. 6)

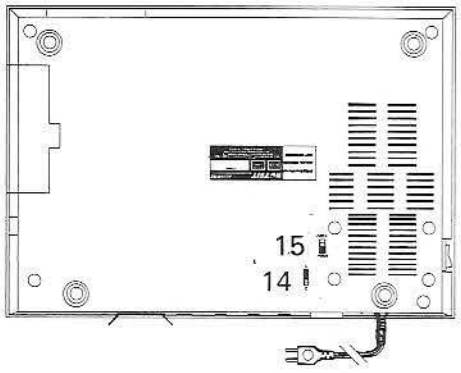


Fig. 6: Parte inferior

14- CHAVE SELETORA DE CANAL RF — Essa chave posiciona o canal de saída de RF (3 ou 4).

15- CHAVE SELETORA DE TENSÃO — Essa chave permite escolher a tensão de trabalho em 120 ou 220 Volts. Verificar a tensão da rede local antes de começar a trabalhar.

1.2.2. INSTALAÇÃO DO MICROCOMPUTADOR E CONEXÕES

Precauções ao instalar o seu computador:

- Não colocá-lo perto de uma fonte de calor ou exposto aos raios solares.
- Não instalar o equipamento num ambiente muito úmido ou com poeira.
- Não manipular o computador com violência, nem submetê-lo a vibrações.
- Não depositar objetos pesados sobre o aparelho.
- O computador está preparado para trabalhar com 120 ∿ 220 Volts e uma potência de 24 Watts.
- O computador poderá produzir interferências em um aparelho de rádio ou de televisão instalado em local muito próximo. Assim também um dispositivo com um forte campo elétrico ou magnético poderá perturbar o funcionamento normal do equipamento. Caso sejam verificadas essas anomalias, será conveniente consultar a Assistência Técnica sobre a melhor maneira de isolar os aparelhos.

CONEXÃO AO TELEVISOR

Existem duas formas de conectar o computador ao aparelho de TV. Primeiro, você conecta o computador diretamente ao terminal de entrada de vídeo do televisor e segundo, a conexão é feita no terminal de antena.

Os dois procedimentos descritos são explicados a seguir:

A) CONEXÃO NO TERMINAL DE ENTRADA DE VÍDEO

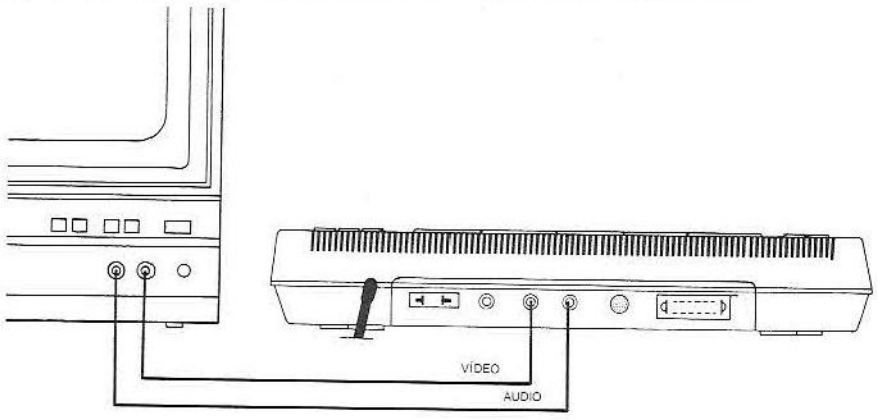


Fig. 7: Conexão no terminal de entrada de vídeo

Os aparelhos que vêm preparados para utilização como monitor de vídeo para computador, possuem um terminal de entrada de vídeo. Nestes casos, conecta-se o cabo de vídeo entre o terminal de entrada de vídeo do TV (vídeo IN) e o terminal de saída de vídeo do computador. Da mesma forma conecta-se um cabo extra entre o terminal de entrada de áudio ao TV e o terminal de saída de áudio do computador. (Fig. 7)

B) CONEXÃO NO TERMINAL DE ANTENA

A. PARA TV QUE UTILIZA FIO DE ANTENA TIPO "FIO PARALELO".

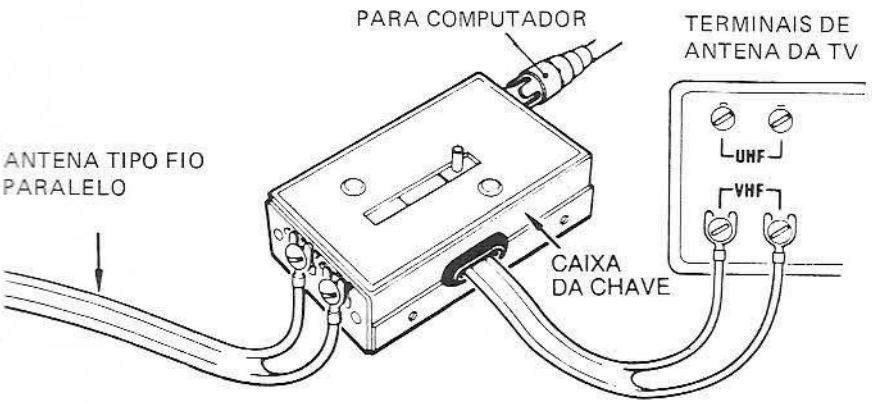
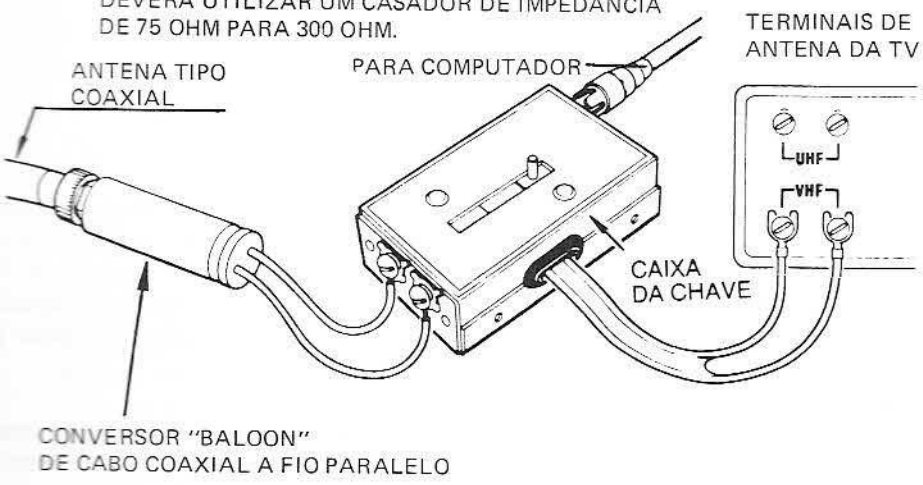


Fig. 8: Conexão no terminal de antena

B. CASO VOCÊ QUEIRA UTILIZAR O CABO TIPO COAXIAL, DEVERÁ UTILIZAR UM CASADOR DE IMPEDÂNCIA DE 75 OHM PARA 300 OHM.



NOTA: NENHUMA MUDANÇA É NECESSÁRIA PARA A CONEXÃO DA ANTENA UHF EXISTENTE.

Fig. 9: Conexão do terminal de antena — cabo tipo coaxial

1.3. NOÇÕES GERAIS SOBRE MICRO-COMPUTADOR

"Pode um computador pensar?" – "Que tipo de problema estará capacitado para resolver?"

São perguntas que surgem freqüentemente ao assistir um filme de ficção-científica, no qual um computador, às voltas com um problema muito complicado, começa a soltar fumaça e explode em pedaços. De fato, isso não poderá acontecer jamais, porque o computador não possui nenhuma função mediante a qual possa pensar.

Na evolução dos computadores, desde o primeiro a válvulas até os computadores atuais de quarta ou quinta geração, tem-se obtido progressos imensos no campo da velocidade de operação e na redução das dimensões de um computador. Todavia ainda não foi criada uma função capaz de fazer o computador pensar e é improvável que isto venha a ser conseguido nos próximos anos. Se de fato isso acontecer, a máquina resultante não poderá mais ser chamada de computador. Então, qual será a resposta à pergunta: "O que é um computador?". Suscintamente, é uma máquina utilizada para fazer cálculos e memorizar dados. Em outras palavras, não é nem mais nem menos do que um instrumento elétrico, formado por uma calculadora programável e um bloco de anotações. Geralmente, para escrever as instruções nesse bloco de anotações, o computador é dotado de um teclado e um monitor de vídeo, a fim de controlar os resultados dos cálculos e das elaborações.

Todos os dados que podem ser convertidos em sinais elétricos, tais como sons, luzes, etc., podem ser introduzidos ou retirados do computador, através de aparelhos ou dispositivos a ele ligados. A parte do computador que funciona como uma calculadora programável, nada mais é do que um circuito integrado chamado de CPU (Unidade Central de Processamento). Pode-se afirmar, sem incorrer em exagero, que a CPU é o computador.

O nosso microcomputador utiliza como CPU um circuito integrado que recebe o nome de Z-80. (Fig. 23)

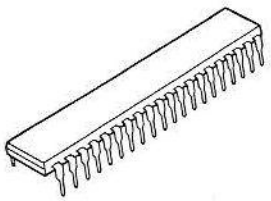


Fig. 23: Circuito integrado Z-80

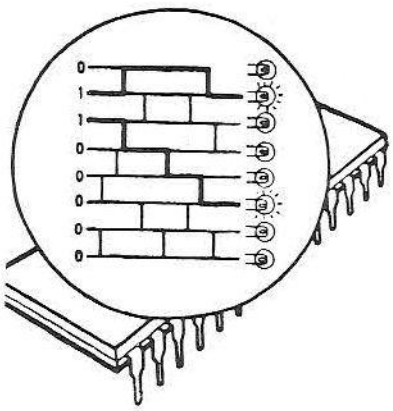


Fig. 24: Circuito integrado tipo LSI

Um circuito integrado desse tipo (LSI) possui uma grande quantidade de pinos, cada um dos quais serve para um uso bem específico. Por exemplo quando uma certa tensão é aplicada entre dois desses pinos, um terceiro pino poderá ser ativado. (Fig. 24)

A parte do computador que funciona como bloco de anotações é chamada de memória (nela poderão ser armazenados diversos tipos de instruções e dados). Memorizar dados significa indicar quais dos pinos e em qual seqüência deverão ser alimentados com corrente elétrica. Ao ligarmos a CPU, esta recebe a ordem de ler o conteúdo da memória. Se não houver instrução anterior na memória, a CPU não realizará nenhuma operação. Os tipos dessas operações são muito limitados. Mesmo os resultados

dessas operações também são muito limitados (como por exemplo, mover uma cifra e levar esse resultado a um dos pinos de saída do circuito). Contudo a combinação de um grande número de instruções simples possibilita a execução de operações do tipo: introduzir pelo teclado "?10 + 10" e obter como resposta no vídeo "20".

Um grupo de instruções desse tipo é chamado de programa em linguagem de máquina. Não é fácil escrever essas instruções na memória, e ninguém terá interesse em combinar centenas ou milhares de instruções para obter como resultado uma simples soma. Para resolver este problema, o computador possui na memória um programa escrito em linguagem de máquina que especifica que a CPU deverá operar seguindo uma linguagem chamada BASIC. Em outras palavras, o BASIC ordena que, ao ser pressionada uma tecla, o computador deve preparar-se para enviar uma mensagem para o vídeo, e que a instrução A = 3 indica que deve preparar a variável A e atribuir-lhe o valor 3.

É somente neste ponto que, a instrução BASIC, de fácil compreensão para o usuário, será transferida para a CPU. (Fig. 25)

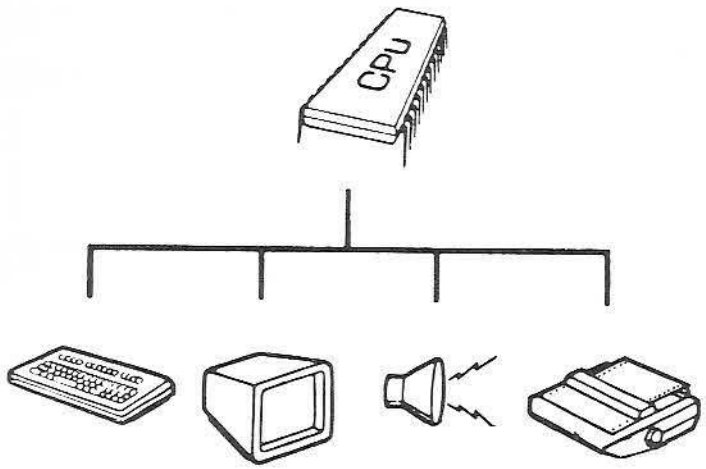


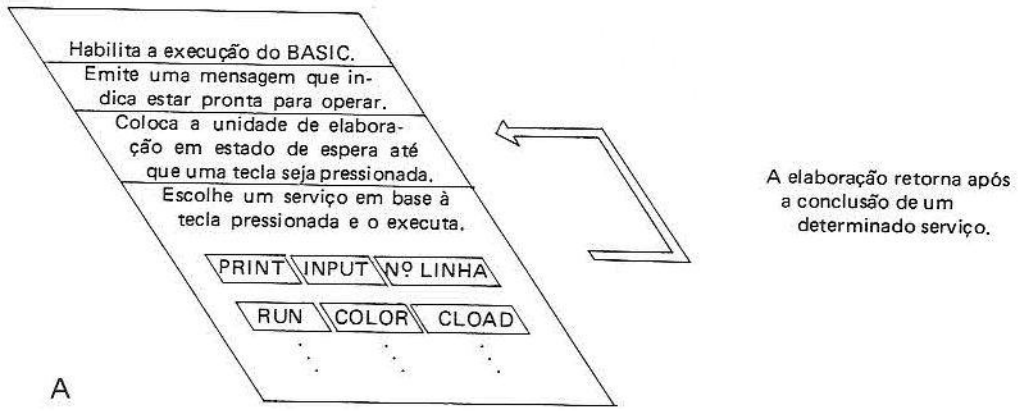
Fig. 25: Mensagem emitida para o vídeo e impressora

Como já dissemos o programa (interpretador BASIC) que traduz as instruções BASIC escritas pelo usuário, para linguagem de máquina, está escrito em uma memória ROM (read-only-memory, isto é, memória somente para leitura). O conteúdo de uma memória ROM pode ser comparado a caracteres esculpidos numa pedra: uma vez gravado na fábrica, o conteúdo não pode ser alterado, mesmo cortando a corrente elétrica desligando o computador ou tentando seguir uma instrução para modificá-lo. Em outras palavras, nenhuma operação poderá apagar ou cancelar o interpretador BASIC, que é o coração do nosso computador.

O contrário acontece com a memória RAM (random-access-memory, isto é, memória de acesso aleatório). Nessa memória os dados podem ser escritos e apagados a vontade.

O conteúdo da RAM será perdido cada vez que o computador for desligado. A RAM é indispensável para escrever os programas, já que cada usuário escreverá nela as instruções de cálculo e os programas BASIC de acordo com as exigências do problema que desejar resolver. (Fig. 26)

ROM — Programa em linguagem de máquina, de grande extensão (interpretador BASIC). O conteúdo dessa memória não é perdido quando o computador é desligado.



RAM — O conteúdo dessa memória é perdido quando o computador é desligado.

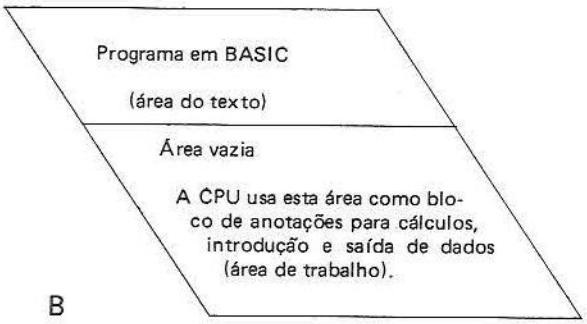


Fig. 26 A e B : Dispositivo dos métodos de execução de cada comando e função

1.4. O TECLADO

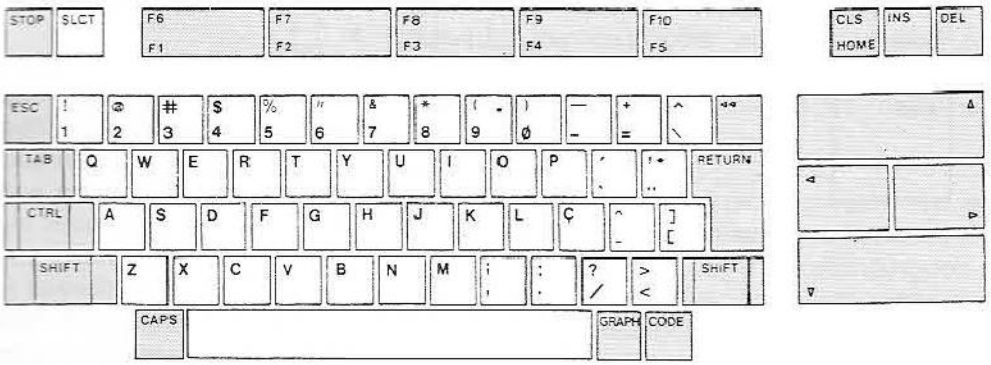


Fig. 10

As teclas de cor escura na figura são chamadas de teclas especiais, para diferenciá-la das outras teclas, que são as dos caracteres normais. Descrevemos a seguir, essas teclas:

1.4.1. TECLAS ESPECIAIS

SHIFT

(MAIÚSCULAS)

Usa-se a tecla SHIFT para escrever letras maiúsculas ou para escrever os caracteres impressos na parte superior das teclas. Esta tecla encontra-se repetida, nos lados direito e esquerdo do teclado, podendo-se usar tanto uma quanto a outra. No apêndice G deste manual pode-se ver a figura referente ao grupo de caracteres disponíveis ao pressionar a tecla SHIFT. Pressionando simultaneamente as teclas SHIFT + CODE ou SHIFT + GRAPH obteremos outros grupos de caracteres e símbolos gráficos, todos eles mostrados no apêndice G deste manual. (Teclados)

CAPS

(BLOQUEIO DE MAIÚSCULAS)

Quando esta tecla é pressionada, logo após o microcomputador ser ligado, todas as letras introduzidas a seguir aparecerão em minúscula. Se a tecla for pressionada novamente, todas as letras introduzidas a seguir serão maiúsculas.

GRAPH

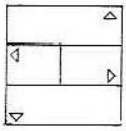
(GRÁFICA)

Essa tecla não influi sobre os caracteres impressos na parte superior das teclas. (Sempre que a lâmpada CAPS estiver acesa, as letras serão impressas em maiúsculas).
Usa-se essa tecla para obter a impressão no vídeo dos símbolos gráficos mantendo-a pressionada, digita-se no teclado a correspondente ao símbolo desejado. No apêndice G (Teclados) encontra-se a correspondência entre as teclas e os símbolos gráficos disponíveis.

CTRL

(CONTROLE)

A tecla CTRL pressionada junto com as outras teclas, serve para obter funções especiais e caracteres especiais de controle. (No apêndice D – tabela de contrôle)



Usam-se essas teclas para movimentar o cursor para cima, para baixo, para esquerda e direita. No manual as teclas serão indicadas como [↑], [↓], [←] e [→].

INS

Pressionando a tecla [INS], o teclado funcionará no modo INSERÇÃO e modificará a forma do cursor. Nesse modo, ao apertar uma tecla, o caracter correspondente será inserido entre o texto e a posição do cursor e todos os caracteres seguintes ao cursor, serão afastados para a direita uma posição. Para sair do modo inserção, deve-se pressionar novamente a tecla [INS], ou então movimentar qualquer um dos cursores, ou ainda pressionar a tecla [RETURN]. Em qualquer um desses casos, o microcomputador volta ao modo normal.

DEL

Ao pressionar a tecla [DEL], obter-se-á o cancelamento do caracter sobre o qual esteja posicionado o cursor, e todos os caracteres da linha serão deslocados em uma posição à esquerda.

◁◁ (BACKSPACE)

A tecla [◁◁] provoca o cancelamento do caracter situado à esquerda do cursor, e todos os caracteres da linha são deslocados em uma posição para a esquerda.

CODE (CÓDIGO)

A tecla [CODE] pressionada junto com qualquer outra tecla, serve para gerar símbolos e caracteres especiais. (Ver no apêndice G (Teclado) as opções de caracteres oferecidas pela tecla CODE).

TAB

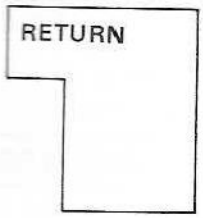
Pressionando a tecla [TAB] o cursor se desloca à direita até a próxima tabulação.
Os valores normais das posições de tabulação são 1, 9, 17 e 25, isto é, cada 8 colunas, estas colunas são preenchidas com espaços.

CLS
HOME

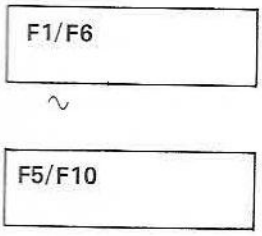
Pressionando essa tecla, o cursor se posicionará na primeira posição do vídeo, na parte superior à esquerda, sendo esta a posição [HOME].
Pressionando essa tecla junto com a tecla [SHIFT], obtém-se a função [CLS] que significa CLEAR SCREEN, isto é, LIMPAR A TELA. Nesse caso o que tiver na tela será apagado e o cursor ficará na posição [HOME].

STOP

A tecla [STOP] detém o programa BASIC em fase de execução. Pressionando-a novamente a execução continuá a partir do ponto da interrupção. Durante a interrupção motivado pela tecla [STOP] não será possível introduzir dados. Para isso, a execução deverá ser efetivada pressionando simultâneamente [CTRL] + [STOP].
Uma vez completada a introdução, a execução continuará ao ser digitado [C], [O], [N], [T] e pressionando [RETURN].



Essa tecla é uma das teclas mais importantes e mais utilizadas.
Ela é usada quando desejamos mandar dados, comandos ou instruções ao computador. Após a digitação dos dados, deve-se pressionar [RETURN] a fim de gravá-los na memória do computador.



Pressionar uma tecla de função programável equivale a introduzir o comando correspondente à tecla pressionada. Ao ligar o computador, na parte inferior do vídeo aparecem visualizados os comandos relativos às teclas, [F1] até [F5]. Os comandos referentes às teclas de função [F6] até [F10] são obtidas pressionando-se a tecla [SHIFT].

1.4.2. TECLAS DE CARACTERES

São chamadas assim as demais teclas.

Neste manual acharemos repetidas vezes duas ou mais teclas de caracteres e/ou de função, separadas por um sinal de mais (+) ou por uma vírgula (,). A seguir a explicação do significado dessa notação:

[CTRL] + [A]

Significa que deverá ser pressionada a tecla [CTRL] e mantê-la assim ao digitar a tecla [A].

[A] , [B] , [C]

Indica que se deve digitar sucessivamente as teclas [A], [B] e [C] (liberando cada uma ao se pressionar a seguinte).

TECLAS	CARACTERES
[E]	e (minúsculo)
[SHIFT] + [E]	E (maiúsculo)
[CAPS] + [E]	E (maiúsculo)
[CODE] + [E]	π
[GRAPH] + [E]	▼ (símbolo gráfico)
[GRAPH] + [SHIFT] + [E]	▲ (símbolo gráfico)

Fig. 11

TECLAS	CARACTERES
[: .]	.
[SHIFT] + [: .]	:
[GRAPH] + [: .]	≥

Fig. 12

[! 1]	1
[SHIFT] + [! 1]	!
[CAPS] + [! 1]	1 (e não !)
[GRAPH] + [! 1]	1/4 (gráfico)

Fig. 13

Fig. 11, 12 e 13: Combinação de teclas para obtenção de caracteres

1.4.3. FUNÇÃO DE REPETIÇÃO AUTOMÁTICA

Ao se pressionar uma tecla por mais do que 0,4 segundos, o caracter correspondente será introduzido continuamente no computador. Experimente manter pressionada a tecla [A] e veja no vídeo o resultado, análogo ao da figura 14.

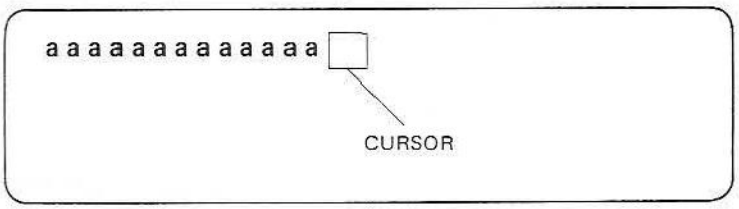


Fig. 14

Essa função é chamada de função de repetição automática, e aplica-se não só às teclas de caracteres, mas também às de função e às teclas especiais, como a de movimento do cursor ou a tecla [DEL] .

1.4.4. CORREÇÃO DE DADOS INTRODUZIDOS

A) INTRODUÇÃO DE CARACTERES

Pressione uma das teclas de caracteres; o mesmo será apresentado no vídeo, na posição onde estava o cursor. Tente agora introduzir todos os caracteres, do A até o Z (*).

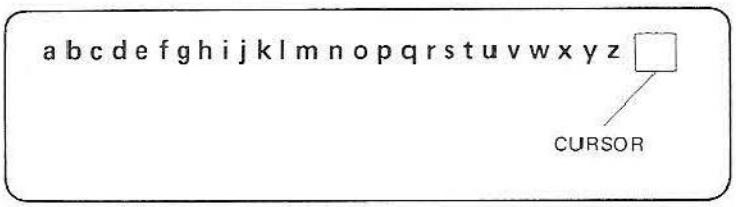


Fig. 15

(*) NOTA: Ao ligar o computador, todas as letras introduzidas serão maiúsculas. Para escrever as minúsculas deverá primeiro pressionar a tecla [CAPS] .

B) SUBSTITUIÇÃO DE UM CARACTER

Passaremos agora dos caracteres minúsculos aos maiúsculos.

- Movimente o cursor para a esquerda até colocá-lo sobre a letra d , usando para isto a tecla [←] .
- Digitar sucessivamente [D] , [E] , [F] juntamente com a tecla [SHIFT] .

Repare no vídeo:

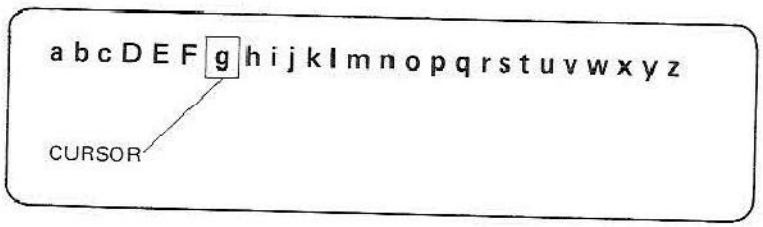


Fig. 16: Substituição de um caracter

Quando se deseja modificar (substituir) um caracter, deve-se sobrepor o cursor ao caracter a ser modificado e digitar o novo caracter. (Fig. 16)

C) CANCELAMENTO DE UM CARACTER

Para cancelar três caracteres, por exemplo: g h i , pressiona-se a tecla [DEL] (delete = apagar). (Fig. 17)

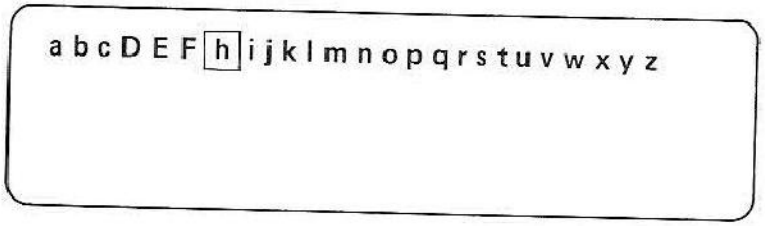
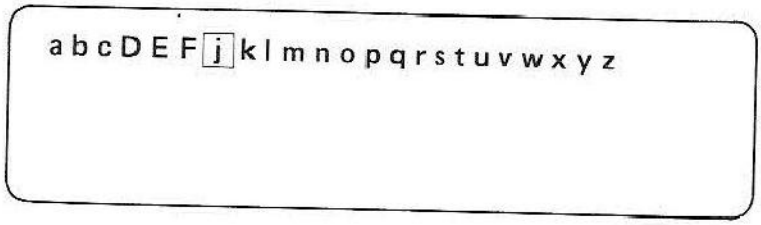


Fig 17: Cancelamento de um caracter

Pressionando a tecla [DEL] outras duas vezes, apagará também as letras h i . (Fig 18: Cancelamento de caracteres)



Pára apagar os caracteres à direita do cursor basta pressionar a tecla [DEL] .

Vamos agora apagar as letras maiúsculas [D E F]. Pressiona-se para isso a tecla [◁◁] (BACKSPACE, situada no teclado imediatamente acima da tecla [RETURN]). Veja como a letra F que precede ao cursor é cancelada. (Fig. 19)

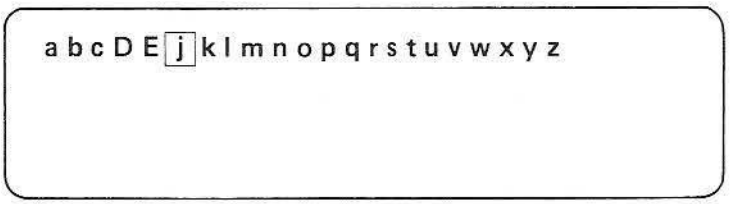


Fig. 19: Cancelamento à esquerda

Volte a pressionar duas vezes a tecla [◁◁] e apagará também as letras D E . (Fig. 20)

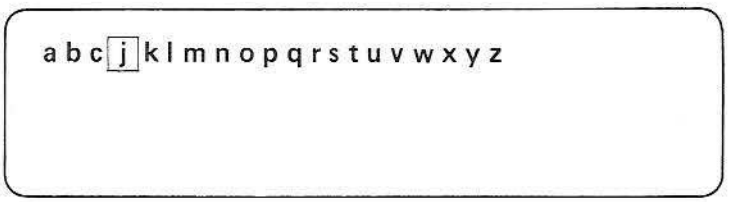


Fig. 20: Cancelamento à esquerda

D) INSERÇÃO DE CARACTERES

Vamos agora inserir 6 caracteres "d e f g h i " entre o "c" e o "j". Pressione primeiro a tecla [INS]. Veja como o cursor muda de forma no vídeo, passando de ■ para ▬ , ficando agora o vídeo no modo inserção.

Pressione a tecla [D] e a mesma será inserida antes do cursor no vídeo. (Fig. 21)

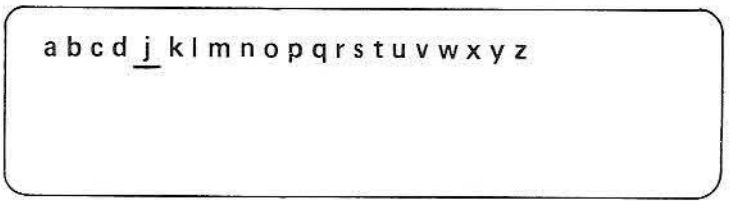


Fig. 21: Inserção de caracter

Tente introduzir "e f g h i" entre o "d" e o "j".

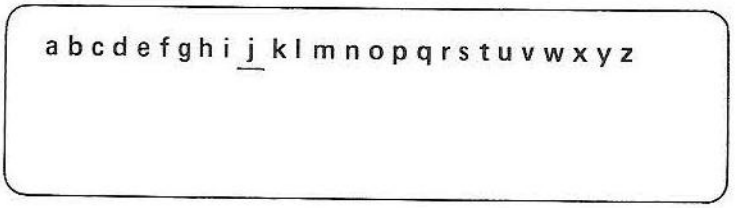


Fig. 22: Inserção de caracteres

Ao pressionar novamente a tecla [INS] (ou ainda movendo o cursor ou pressionando [RETURN]) será abandonado o modo inserção e o cursor volta ao seu tamanho original.

1.5. CRIAÇÃO E EXECUÇÃO DE UM PROGRAMA

Ao ligar o computador o vídeo exibe uma mensagem (a palavra vídeo ou display indica por exemplo o seu monitor de televisão ligado ao computador).

A mensagem é:

```
HOTBIT
HB-8000
EPCOM [ Z.F. MANAUS ] 1985
```

Logo após, o computador verifica se foi a ele ligado algum cartucho. Caso contrário, aparece no vídeo uma segunda mensagem:

```
HOT-BASIC versão X.X
EPCOM [ Z.F. MANAUS ] 1985
Mem. Livre 28815
OK
```

O pequeno retângulo que aparece abaixo do O K é o "cursor". Ele indica o ponto no qual será escrito o texto introduzido pelo teclado.

O número 28815 indica as posições livres na memória do computador.

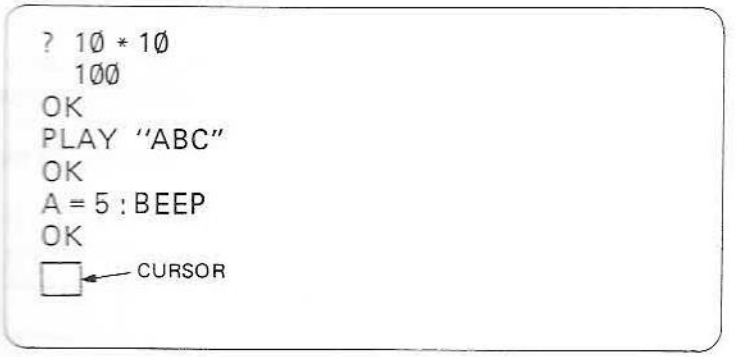
A mensagem "O K" aparece no vídeo cada vez que o BASIC está no nível de comandos, ou seja, quando está pronto para receber instruções.

Sendo assim, seguimos trabalhando em qualquer um dos dois modos: "direto" ou "indireto".

No modo direto, o comando é introduzido e o computador o executa imediatamente.

Vejamos o exemplo na tela 1.

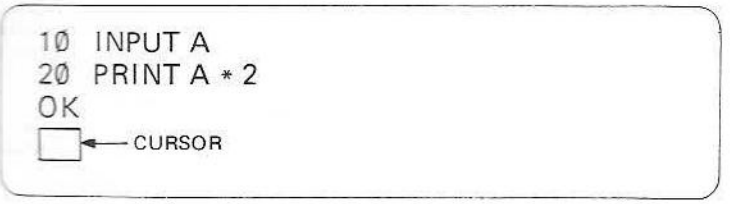
Tela 1



Cada vez que é pressionada a tecla RETURN esses comandos são transferidos para a CPU, processados, gravados e o computador volta ao estado de espera de introdução de novos comandos, através da mensagem "O K".

A outra possibilidade de operação, ou seja, no modo "indireto", é a criação de um programa, introduzindo um número (no início da linha, seguido de uma instrução válida, em BASIC).

Vejamos exemplo na tela 2:



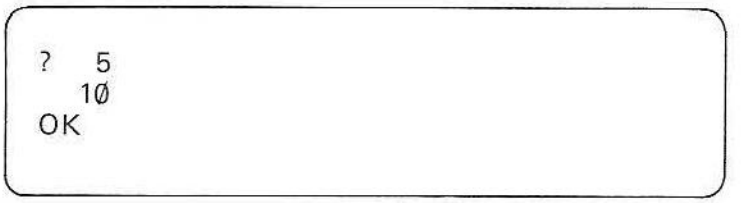
Tela 2

Ao pressionarmos a tecla RETURN a instrução não será executada e sim memorizada para a sua posterior utilização. A mensagem "O K" não aparecerá no vídeo.

Uma vez o programa criado, deve ser dado o comando RUN para executá-lo.

```
RUN
?
```

O programa está agora seguindo a instrução INPUT da linha 10. Enquanto não colocarmos um dado, a CPU ficará esperando. Digitemos, por exemplo: 5 , RETURN .



Tela 3

Isto permite a execução da próxima instrução, a da linha 20: PRINT A * 2, e como a seguir não tem mais instruções, a execução do programa termina, comparecendo novamente no vídeo a mensagem "O K". (Vide exemplo tela 3)

Este modo de operação será explicado com mais detalhes num outro setor deste manual. Por enquanto somente ilustraremos alguns conceitos fundamentais:

Toda a memória do computador está configurada e repartida segundo indica a ilustração seguinte:

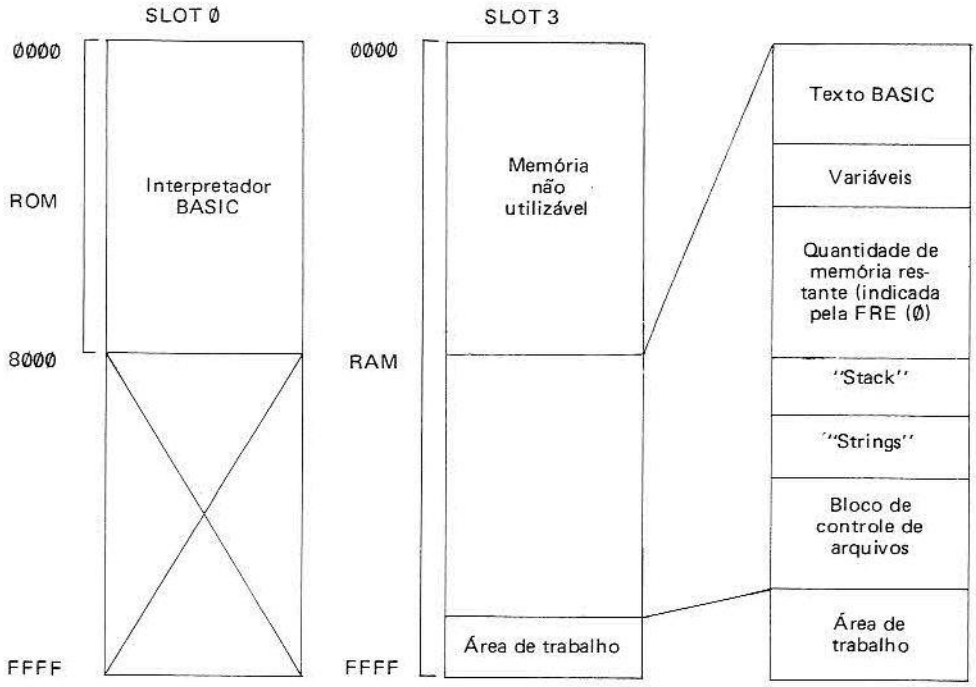


Fig. 27: Ilustração da memória

A área que vai de "Texto BASIC" até "Strings" pode ser usada livremente em BASIC. Imediatamente após, temos a área de carregamento interno do computador, na qual não poderá o usuário carregar nem programa, nem variáveis. Neste ponto o valor de FRE (0) (área de memória que pode ser utilizada pelo usuário) é de 28815.

Quando introduzirmos um programa, tal como o do exemplo anterior, ele será memorizado na área "Texto BASIC".

Se através da função PEEK tentássemos ler o conteúdo dos 20 bytes seguintes ao endereço &HØØØØ, acharíamos o seguinte resultado:

Ø, 9, 128, 1Ø, ...

Esses valores constituem o texto do programa BASIC e tem um aspecto muito diferente daquele que aparece no comando LIST, que é:

```
10 INPUT A
20 PRINT A * 2
```

Todavia, o interpretador BASIC memorizado em ROM considera-os como dados.

Quando se opera em modo direto, o interpretador controla a "ÁREA DE TRABALHO", isto quer dizer, enquanto o usuário introduz dados através do teclado, o interpretador escreve esses dados na "área de trabalho" e simultâneamente os visualiza no monitor de vídeo.

Ao ser pressionada a tecla [RETURN] , o interpretador analisa se anteriormente foi introduzido um comando ou uma instrução. Se foi um comando, executa-o; se foi uma linha de programa, escreve-a na "área de texto".

O texto assim criado está composto por códigos intermediários. Esses códigos estão bem definidos, por exemplo: 133 é o código correspondente a INPUT, 145 é o correspondente a PRINT , e assim por diante.

A função dessa codificação é a de poupar tempo e espaço na memória na hora de ler ou escrever uma instrução na "área de texto".

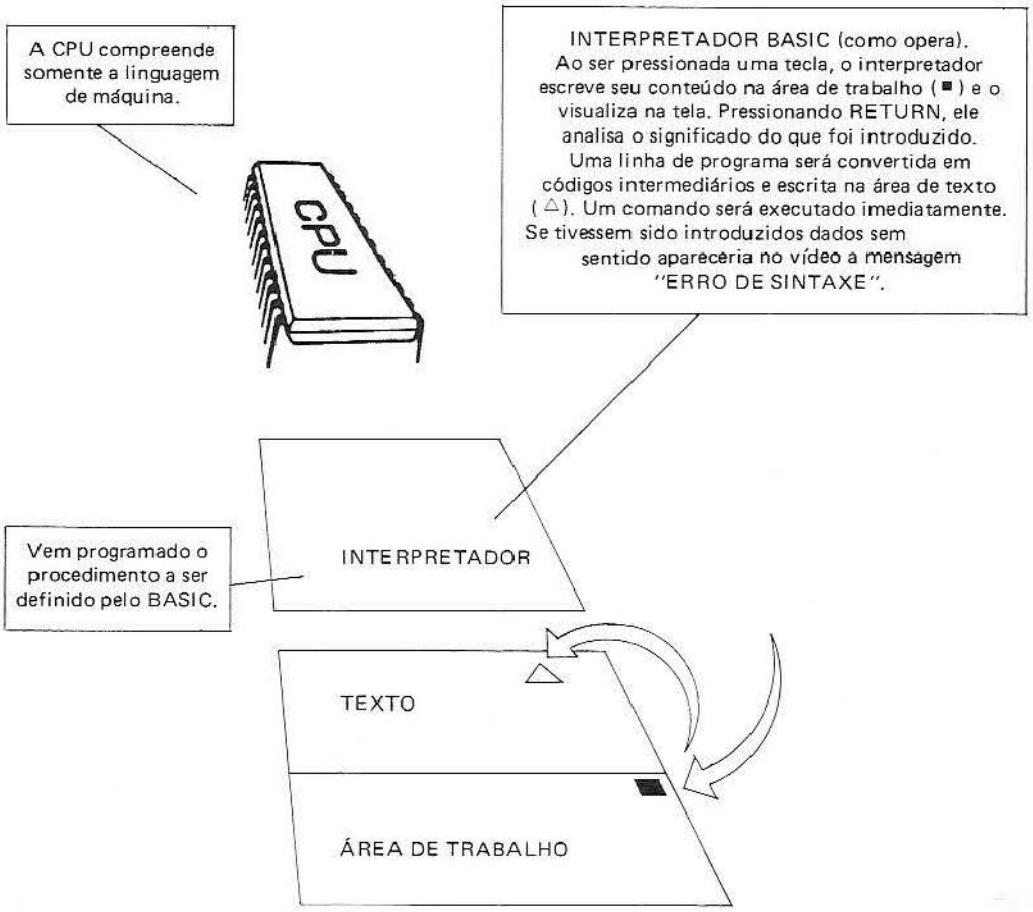
Finalmente quando se introduz um comando RUN, o interpretador executa o programa seguindo o texto a partir do início e fornece, quando lhe é solicitado, as variáveis encontradas na "área de variáveis", que segue a "area de texto".

Quando encontra uma instrução STOP ou END, termina a execução e volta ao modo direto.

Pressionando simultâneamente as teclas [CTRL] e [STOP] voltará ao modo direto.

Veja a seguir o diagrama que resume o explicado até o momento (vide também o capítulo: A MEMÓRIA).

Fig. 28: Modo direto



Execução de um programa. (Fig. 29)

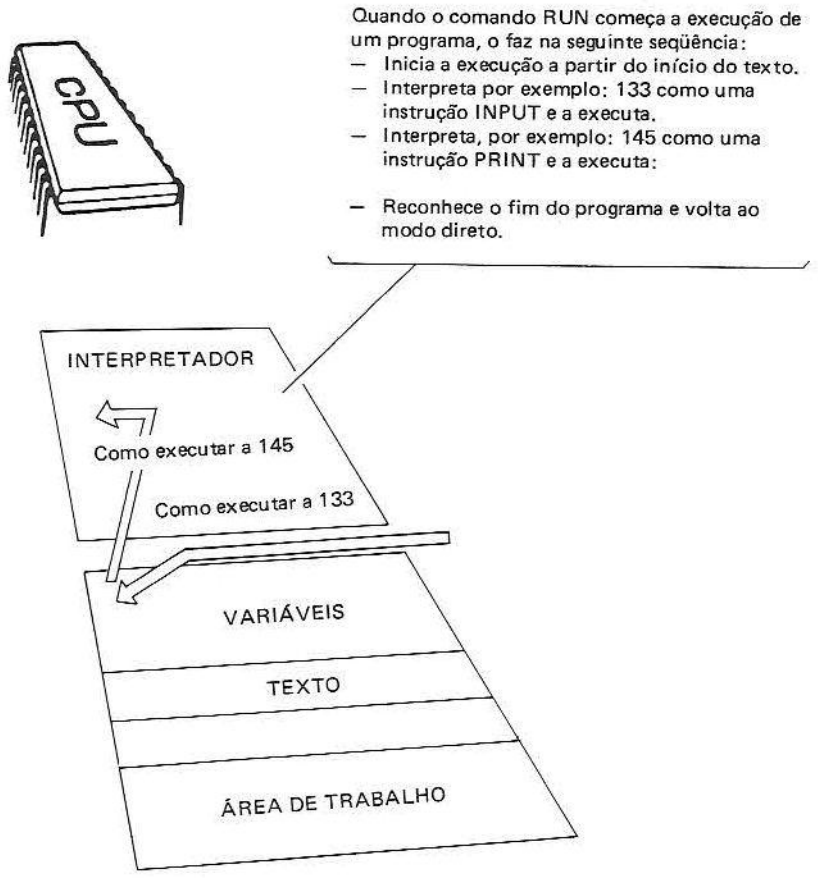


Fig. 29: Execução de um programa

2. Linguagem Basic

2.1. INTRODUÇÃO

As pessoas comunicam-se entre si através de uma linguagem (português, inglês, etc.).

Esta linguagem é composta de letras e palavras que possuem regras e funções gramaticais.

Assim também, se faz necessário a existência de uma linguagem ou meio de comunicação entre o homem e o computador.

O BASIC é uma dessas linguagens e a utilizada por nosso computador.

Para uma descrição detalhada de cada uma das "palavras" utilizadas pelo "BASIC" e suas funções deve ser consultado o manual de BASIC.

2.1.1. MODOS DE EXECUÇÃO

O computador executa as "instruções" fornecidas de duas formas diferentes:

— Modo direto, na qual a instrução é executada assim que é introduzida e logo após ter pressionado a tecla [RETURN]. Por exemplo PRINT "OLA" e dando RETURN , a instrução PRINT será imediatamente executada e aparecerá na tela do vídeo a palavra OLA.

Um comando introduzido pelo teclado só será executado depois que a tecla [RETURN] for pressionada. Fig. 30

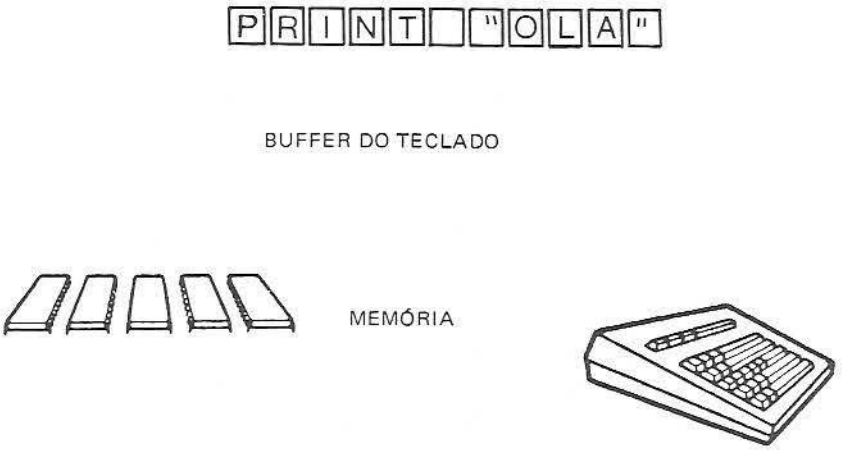


Fig. 30

— Modo indireto, ou também chamado Programa. Nela a instrução não é executada imediatamente após sua introdução (nem mesmo pressionando [RETURN]), mas é gravada e armazenada junto com outras instruções na memória do computador, para a sua posterior execução.

Isso se deve ao fato dos caracteres introduzidos serem mesmo usados no "buffer" do teclado, e não enviados diretamente para o CPU. Somente depois que a tecla [RETURN] for pressionada, o conteúdo do "buffer" do teclado será mandado para a CPU e interpretado. (Fig. 31)

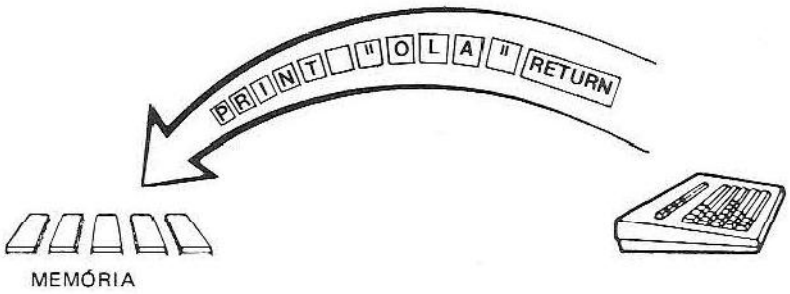


Fig. 31

Um comando enviado ao computador é executado e depois cancelado da memória. Para poder ser executado novamente, deverá ser reintroduzido através do teclado.

A seguir um exemplo de uso do computador no modo direto.

Calcular o resultado da seguinte expressão:

10 * 53 + 6 – 30/5

Introduzir através do teclado a seguinte forma:

[SHIFT] + [?/] , [1] , [Ø] , [SHIFT] + [(8] , [5] , [3] , [SHIFT] + [+=] , [6] , [_] , [3] , [Ø] , [?/] , [5] , [RETURN]

Após acionar a tecla [RETURN] , aparecerá na próxima linha do vídeo o resultado da expressão:

530

O modo direto permite assim, executar, de um modo simples, os cálculos elementares, mas não possibilita cálculos ou operações mais complexas. Para isso veremos agora o modo programa ou indireto.

- MODO PROGRAMA OU INDIRETO

Quando um comando é precedido de um número, será memorizado na área de texto BASIC. As instruções não recebidas na memória pela ordem em que são introduzidas no teclado, mas segundo a ordem dos números que as precedem. Exemplo:

```
30 NEXT I
20 S = S + 1
40 PRINT S
10 FOR I = 1 TO 1Ø
```

As instruções serão armazenadas na memória da seguinte forma:

```
10 FOR I = 1 TO 10
20 S = S + 1
30 NEXT I
40 PRINT S
```

Um grupo de instruções armazenadas na memória dessa forma, é chamado de programa.

Um programa é executado através do comando RUN, e não é apagado da memória após a sua execução. Por isso poderá ser executado sempre que for necessário (enquanto o computador não for desligado).

- **LINHA E Nº DE LINHA**

Em um programa, cada linha se inicia com um número e termina quando é pressionada a tecla [RETURN]. Uma linha de programa está composta então, por um número, e uma ou mais instruções.

10	INPUT A	:	PRINT A
Nº de linha	Instrução		Instrução

As instruções são separadas entre si por dois pontos (:).

A linha do exemplo é chamada de linha 1Ø.

O número da linha indica que ela faz parte de um programa (não é executada imediatamente e sim memorizada), e especifica ao mesmo tempo, a ordem de execução numa vez que for dado o comando RUN.

Uma linha de programa poderá conter até 255 caracteres.

2.1.2. DIFERENÇA ENTRE COMANDOS, INSTRUÇÕES E FUNÇÕES

– Diferença entre comandos e instruções.

Os comandos e as instruções podem ser usadas tanto no modo direto como no modo programa. A instrução faz com que o programa seja inteiramente executado, enquanto que o comando faz com que o BASIC volte sempre ao nível de comandos.

Tente introduzir o seguinte exemplo:

```
1Ø PRINT 3+1
2Ø LIST
3Ø PRINT 6-3
RUN
```

"LIST" é um comando BASIC que é utilizado para conseguir uma listagem de todas as linhas do programa da memória, na tela do vídeo. O usuário poderá, através dessa listagem examinar o programa e identificar possíveis erros, modificar, anexar ou ainda apagar linhas.

Por ser "LIST" um comando ao encontrá-lo durante a execução de um programa, o BASIC voltará ao nível de comandos logo após a execução da linha 2Ø. Isto é, a linha 3Ø não será executada.

No manual de BASIC pode-se ver o uso de cada um dos comandos e instruções.

— Diferença entre instruções e funções.

Uma função deverá ser sempre precedida por uma instrução. Um exemplo de função é "FRE (Ø)", utilizada para perguntar ao BASIC quantos bytes ainda estão disponíveis na memória do computador.

Tente o seguinte exemplo:

O comando NEW é utilizado para apagar da memória do computador todas as linhas introduzidas anteriormente:

```
NEW
10 FRE (Ø)
RUN
```

Poderá observar que aparece na tela a mensagem de erro: ERRO SINTAXE EM 1Ø. Isso acontecerá sempre que for utilizada uma função sem uma instrução correspondente.

Modifique agora o exemplo da forma seguinte:

```
1Ø PRINT FRE (Ø)
RUN
```

Agora sim o BASIC lhe informará quantos bytes, tem ainda disponíveis para continuar com os seus programas, pois foi colocada a instrução "PRINT" antes da função "FRE (Ø)".

2.2. O "HOT-BASIC"

2.2.1. COMANDOS E INSTRUÇÕES DO HOT-BASIC

Após ser introduzido um comando do BASIC e ser pressionada a tecla [RETURN], o comando é enviado ao computador, que o executa.

A instrução "PRINT" mostra no vídeo (imprime) os caracteres ou os números que são especificados na própria instrução.

Vejamos o seguinte exemplo:

Pressione [SHIFT] + [CLS HOME] para limpar a tela do vídeo.

Digite depois [P], [R], [I], [N], [T], [ESPAÇO], [1], [Ø] e [RETURN], nessa ordem.

O que você verá na tela é:

```
PRINT  1Ø  .. Comando introduzido
  1Ø  ..... Resposta do computador
OK  ..... Computador novamente pronto
□  ..... Cursor
```

A instrução PRINT 1Ø ordena ao computador a impressão na tela do número 1Ø. O resultado é apresentado na linha seguinte à do comando. O fim da operação está representado pela palavra O K que aparece abaixo do resultado; isto é, a máquina está pronta para continuar trabalhando. Não esquecer de pressionar [RETURN] logo após a introdução de cada comando ou instrução. Somente após o [RETURN] é que o comando é enviado ao sistema para a sua execução.

Os comandos e instruções do BASIC podem ser introduzidos tanto em letra maiúscula como minúscula.

2.2.2. VISUALIZAÇÃO DE CARACTERES

Se através da instrução PRINT deseja-se imprimir uma cadeia de caracteres, deve-se colocar a string sempre entre aspas (").

Por exemplo, para imprimir a palavra BASIC na tela digita-se: PRINT "BASIC" e depois RETURN .

```
PRINT "BASIC"
BASIC
OK
□
```

2.2.3. VISUALIZAÇÃO DO VALOR DE UMA EXPRESSÃO

Através da instrução PRINT pode-se também obter a visualização do cálculo de uma expressão numérica. Isto permite o uso do computador como se fosse uma calculadora.

Vejamos alguns exemplos:

```
? 10+5 ..................  10+5
 15
O K
? 15*3+8/4 ..............  15 x 3 + 8 ÷ 4
 47
O K
? 100-5^2 ...............  100 − 5²
 75
O K
? (10+2)*SQR(100) .......  (10+2) x √100
 120
```

Nesses exemplos a instrução PRINT foi substituída pelo símbolo ? (o computador por sua vez interpreta o símbolo ? como PRINT).

2.2.4. REINTRODUÇÃO DE UM COMANDO OU UMA INSTRUÇÃO

Para executar novamente um comando que já foi introduzido e executado através do [RETURN], devemos redigitá-lo.

Todavia, se o comando ainda estiver impresso na tela, poderemos reutilizá-lo sem necessidade de uma nova digitação, bastando para isso,

movimentar o cursor sobre a linha do vídeo que contém o comando, e pressionar [RETURN].

No caso de modificação da expressão, mesmo que parcialmente, podemos fazê-lo usando as teclas e a técnica indicada no capítulo 1 (1.3.4. Correção de caracteres).

Vejamos o exemplo:

```
PRINT 10 + 5 * 2
  20
O K
□—— CURSOR
```

Desejamos calcular agora 10 + 6 * 2, movemos então o cursor até a posição do 5:

```
PRINT 10 + [5] * 2
  20
O K
          |
       CURSOR
```

Digitamos o número 6 e pressionamos [RETURN]. Aparecerá no vídeo o seguinte resultado:

```
PRINT 10 + 6 * 2 ..... Expressão modificada
  22 ................ Novo resultado
O K
□
```

2.2.5. PROGRAMA EM BASIC

Introduzindo um comando BASIC e pressionando depois RETURN, podem-se realizar somente cálculos simples. Para realizar, cálculos e operações mais longas e complexas, será necessário introduzir as instruções na forma de programa BASIC e depois executá-lo.

Para isso, deveremos enumerar as instruções com um valor compreendido entre 1 e 65529. Todas as instruções deverão pertencer a uma linha. Em uma linha de programa poderemos colocar mais de uma instrução, separadas entre si por dois pontos (:).

Por exemplo:

```
10 PRINT "B" [RETURN]
```

A instrução será memorizada pelo computador mas não será executada, nem mesmo depois de pressionar [RETURN].

Continuamos agora com as seguintes instruções:

```
20 PRINT "A" [RETURN]
30 PRINT "S" [RETURN]
40 PRINT "I" [RETURN]
50 PRINT "C" [RETURN]
```

Todas as instruções foram memorizadas pelo computador. O número colocado antes da instrução, chamado de número de linha, é utilizado para ordenar a seqüência de armazenamento das linhas, e a ordem, em que serão executadas.

Não poderão existir duas linhas com o mesmo número. O grupo de instrução do exemplo acima constituem um "programa BASIC".

2.2.6. LISTAR UM PROGRAMA

Para obter no vídeo a impressão do grupo completo de instruções que formam o programa BASIC, usa-se o comando LIST.

Digitando o comando LIST, aparecerá no vídeo:

```
10 PRINT "B"
20 PRINT "A"
30 PRINT "S"
40 PRINT "I"
50 PRINT "C"
Ok
□
```

2.2.7. EXECUÇÃO DE UM PROGRAMA

Existe um comando que permite finalmente executar todas as instruções armazenadas na memória. É o comando:

RUN

Pode-se dar esse comando digitando [R] , [U] , [N] , [RETURN] ou então pressionando a tecla de função [F5] (em cujo caso não precisará pressionar [RETURN]).

Dando RUN ao programa do nosso exemplo obteremos:

```
RUN
B
A
S
I
C
Ok
```

2.2.8. CORREÇÃO DE UM PROGRAMA

Durante o desenvolvimento de um programa poderá suceder que se cometam erros. Existem diversos tipos de erro, e será necessário corrigir o programa várias vezes.

Veremos agora como fazer essas correções e/ou modificações de um programa.

– INSERÇÃO DE UMA NOVA LINHA

O número da linha é de grande importância durante a correção de um programa. Na necessidade de inserção de uma linha de programa entre duas já existentes, deve-se atribuir a esta nova linha um número compreendido entre os dois em questão.

Entre as linhas 20 e 30 do programa, poderão ser inseridas 9 linhas, com números de 21 até 29.

Se quiséssemos por exemplo, introduzir uma linha no exemplo anterior, digitaríamos:

25 PRINT "I" [RETURN]

Com o comando LIST poderemos verificar a introdução da nova linha, na ordem certa.

```
LIST
10  PRINT "B"
20  PRINT "A"
25  PRINT "I"
30  PRINT "S"
40  PRINT "I"
50  PRINT "C"
Ok
□
```

– COMO CANCELAR UMA LINHA

Suponhamos agora o cancelamento de uma linha de programa introduzida anteriormente. Veja bem! Não é suficiente apagá-la do vídeo! Devemos lembrar que ela foi armazenada na memória. Para ser cancelada bastará digitar o número da linha e pressionar [RETURN] logo após.

Por exemplo para cancelar a linha 25 introduzida no exemplo anterior, fazemos:

25 [RETURN]

Também é possível usar o comando DELET E para cancelar várias linhas de um programa de uma só vez. Por exemplo, para cancelar todas as linhas entre a número 20 e a número 40, damos o comando:

DELETE 20 – 40 [RETURN]

Também podemos usar DELETE para:

DELETE 20 cancelar somente a linha 20
(equivale a digitar 20 [RETURN])

DELETE – 40 cancelar todas as linhas anteriores à de número 40 inclusive

Para cancelar da memória um programa completo, usa-se o comando NEW.

NEW RETURN

– SUBSTITUIÇÃO DE UMA LINHA

Caso desejar a substituição do conteúdo de uma linha, por exemplo a linha 10, basta introduzí-la novamente, com o mesmo número e com o novo conteúdo. A linha anterior é apagada da memória e substituída pela nova. A ordem de execução fica obviamente inalterada. Por esse motivo é que não podemos ter na memória duas linhas com o mesmo número.

Veja por exemplo. No mesmo programa anterior, digite:

```
10 PRINT "b"

LIST
10 PRINT "b"
20 PRINT "A"
30 PRINT "S"
40 PRINT "I"
50 PRINT "C"
```

A linha 10 foi modificada.

Se a linha do programa, como nesse caso, deve ser substituída por uma linha de conteúdo muito similar, poderá ser conveniente corrigí-la em lugar de substituí-la.

2.2.9. CONSTANTES E VARIÁVEIS

A memória do computador está capacitada para receber não somente linhas com comandos, instruções e funções, mas também outros elementos que desempenham um papel importante durante o exercício de um programa.

Esses elementos podem ser divididos em dois grupos: "constantes e variáveis".

– CONSTANTES

As constantes contém informações que não mudam durante a execução do programa. Por sua vez, as constantes podem ser de dois tipos: constantes alfanuméricas e numéricas.

Uma constante alfanumérica poderá conter um máximo de 255 caracteres e deverá começar e terminar com o símbolo de aspas (" ... ").

Exemplo:

"MARIA" ou "12345"

Embora a constante "12345" esteja formada por números, não poderá ser utilizada para realizar cálculos.

Uma constante numérica estará representada por uma cifra positiva ou negativa. No nosso BASIC, a vírgula decimal deverá ser substituída por um ponto.

Exemplo:

o número 123,45 deverá ser escrito como 123.45

O BASIC reconhece 6 tipos de constantes numéricas.

1. Constante inteira.

   Poderá ser um número inteiro qualquer compreendido entre − 32768 e 32767.

   Exemplo:

   13 ou − 45

2. Constante de ponto fixo.

   Números reais positivos ou negativos, isto é, números que contém casas decimais.

   Exemplo:

   15.73 ou − 178.3

3. Constante de ponto flutuante.

   Números positivos ou negativos representados na forma exponencial (similar a notação científica).

   A constante de ponto flutuante consiste de um número inteiro positivo ou negativo (ou seja, um número de ponto fixo), chamado de mantissa, seguido de uma letra E e um número inteiro (positivo ou negativo) chamado de expoente. As constantes de ponto flutuante estão compreendidas entre 10^{-64} e 10^{+63}.

   Exemplos:

   235.988 E − 7 = .0000235988

   2359 E6 = 2359000000

4. Constante hexadecimal.

   Números hexadecimais, identificados pelo prefixo &H.

   Exemplo:

&H 7F
&H 32F

5. Constante octal.

   Números octais, identificados pelo prefixo &0.

   Exemplo:

   &0 37

6. Constante Binária.

   Números binários identificados pelo prefixo &B.

   Exemplo:

   &B 11000110
   &B 00111001

- **PRECISÃO SIMPLES E DUPLA**

As constantes numéricas podem ser declaradas como números de precisão simples ou dupla.

Na forma de precisão simples as constantes numéricas são armazenadas com 6 dígitos, e impressas com até 6 dígitos de precisão.

Uma constante de precisão simples é qualquer constante numérica com uma das seguintes características.

1. Forma exponencial usando "E".

   Exemplo:

   3.50 E8

2. Um ponto de exclamação após o número.

   Exemplo:

   22.5 !

Na forma de precisão dupla as constantes numéricas são armazenadas com 14 dígitos de precisão e impressas com até 14 dígitos.

No HOT-BASIC, a precisão "default" para constantes, é a dupla.

Uma constante de dupla precisão é qualquer constante numérica com uma das seguintes características.

1. Qualquer número de dígitos sem expoente ou especificador de tipo.

   Exemplos:

   3489
   12879543.47

2. Forma exponencial usando "D".

   Exemplo:

   −1.09537 D−06

3. O símbolo (#) escrito após o número.

   Exemplo:

   1279.0 #

- VARIÁVEIS

As variáveis são nomes usados para representar valores dentro de um programa BASIC.

O valor de uma variável pode ser atribuído explicitamente pelo programador ou pode ser atribuído através de cálculos realizados pelo programa.

Antes de ser atribuído um valor qualquer a uma variável, ela assume o valor zero (Ø).

- NOMES DE VARIÁVEIS E CARACTERES DE DECLARAÇÃO

Os nomes das variáveis no nosso BASIC podem ter qualquer comprimento. Até 2 caracteres são significativos.

Os nomes das variáveis podem conter letras e números. Todavia, o primeiro caracter deverá ser uma letra. Existem ainda certos caracteres especiais de declaração de variáveis (vide abaixo).

O nome de uma variável não poderá ser uma das palavras reservadas. As palavras reservadas incluem todos os nomes de comandos, instruções, funções e operadores do BASIC (vide apêndice C).

As variáveis poderão representar um valor numérico ou uma cadeia alfanumérica (string). Os nomes de variáveis "string" são escritos colocando sempre o símbolo "$" como último caracter.

Exemplo: A$ = "vendas anuais".

O símbolo "$" é um dos caracteres especiais de declaração de variáveis, isto é, ele "declara" que a variável representará um "string".

Os nomes de variáveis numéricas poderão declarar valores inteiros, de simples precisão ou de dupla precisão.

Os caracteres de declaração do tipo da variável são:

%	Para declarar uma variável inteira. (Ex.: VALOR %)
!	Para declarar uma variável de simples precisão. (Ex.: M !)

Para declarar uma variável de dupla precisão. (Ex.: PI #)

O tipo "default" para uma variável numérica, é a dupla precisão.

Existe um outro método de declaração do tipo das variáveis. As instruções do BASIC: DEFINT, DEFSTR, DEFSNG e DEFDBL podem ser incluídas no programa a fim de declarar os tipos de determinados nomes de variáveis. (Vide Manual de BASIC para maiores detalhes sobre essas instruções)

- **VARIÁVEIS "ARRAY"**

"Array" é um grupo ou tabela de valores referenciados por um mesmo nome de variável. Sendo que cada elemento é identificado por um índice subscrito, que pode ser um número inteiro ou uma expressão inteira. O nome de uma "variável array" terá tantos índices subscritos quantas dimensões tiver o mesmo.

Por exemplo V (1Ø), faz referência a um array "unidimensional" que pode conter até 11 elementos. T (1,4) referencia um array bidimensional, e assim por diante. O máximo número possível de dimensões de um array é de 255. O número de elementos será determinado pela capacidade de memória.

- **REQUISITOS DE ESPAÇO DE MEMÓRIA**

A seguinte tabela lista o número de bytes ocupados pelos valores representados pelos nomes das variáveis.

VARIÁVEIS

TIPO	BYTES
Inteiras	2
Simples Precisão	4
Dupla Precisão	8

ARRAY

TIPO	BYTES
Inteiros	2 por elemento
Simples Precisão	4 por elemento
Dupla Precisão	8 por elemento

STRINGS

3 bytes mais o espaço requerido pelo conteúdo efetivo do "string".

- **CONVERSÃO DE TIPO**

Quando houver necessidade, o BASIC poderá converter uma constante numérica de um tipo para outro. Deverão ser lembradas as seguintes regras:

1. Se uma constante numérica de um tipo é igualada a uma variável numérica de um tipo diferente, o número será armazenado obedecendo ao tipo declarado no nome da variável. (Se uma variável numérica é igualada a um string, provocará um erro do tipo "TIPO DESIGUAL").

   Exemplo:

   ```
   10 A% = 27.25
   20 PRINT A%
   RUN
   27
   ```

2. Durante a avaliação de uma expressão, todos os operandos de uma operação aritmética ou de relação, são convertidos ao mesmo tipo de precisão, que é a do operando mais preciso.

   Assim também o resultado será proporcionado nesse mesmo grau de precisão.

   Exemplos:

   ```
   10 D = 6/7.!
   20 PRINT D
   RUN
   .85714285714286
   ```

   A operação aritmética foi realizada com dupla precisão e o resultado é também colocado em D com dupla precisão.

   ```
   10 D ! = 6/7
   20 PRINT D !
   RUN
   .857143
   ```

   A operação aritmética foi realizada com dupla precisão, e o resultado é colocado em D ! (variável de precisão simples) arredondado e impresso na tela como sendo um valor de precisão simples.

3. As operações lógicas convertem seus operandos em inteiros e fornecem um resultado inteiro. Os operandos deverão estar compreendidos entre — 32768 e 32767 caso contrário haverá um erro de "OVERFLOW".

4. Quando um valor de ponto flutuante é convertido em inteiro, a porção decimal é truncada.

   Exemplo:

   ```
   10 C% = 55.88
   20 PRINT C%
   RUN
   55
   ```

5. Se a uma variável de dupla precisão for atribuído um valor de simples precisão, somente os primeiros 6 dígitos do número convertido terão validade. Isto acontece porque com o valor de simples precisão foram fornecidos somente 6 dígitos exatos.

   Exemplo:

```
10 A! = SQR (2)
20 B = A!
30 PRINT A!, B
RUN
1.41421.        1.41421
```

- **CONTEÚDO DE UMA VARIÁVEL**

Pode ser atribuído um valor a uma variável usando a instrução "LET".

Exemplo:

```
10 LET A = 100
20 LET B$ = "BRASIL"
```

O uso da palavra LET não é imprescindível, basta se escrever deste modo:

```
10 A = 100
20 B$ = "BRASIL"
```

Também pode ser atribuído um valor a uma variável usando a instrução "INPUT". Quando o computador executa uma instrução "INPUT", fica à espera, até que seja introduzido um valor através do teclado. Somente após a pressão da tecla [RETURN], o valor é inserido na variável indicada na instrução "INPUT". Pode também ser introduzida uma frase interrogativa usando a instrução INPUT.

Exemplo:

```
NEW
10 INPUT "QUAL É O NÚMERO"; A
20 INPUT "SEU NOME É"; B$
RUN
QUAL É O NÚMERO?
```

O ponto de interrogação aparece automáticamente. Nesse programa exemplo, na linha número 10, o BASIC aceitará somente valores numéricos na linha número 20 podem ser usados caracteres alfanuméricos.

- **VARIÁVEIS ESPECIAIS**

Escrevendo um programa em BASIC é possível usar certas variáveis especiais. Elas são: "TIME", "VDP", "BASE" e "SPRITE $".

Exemplo:

```
10 PRINT "INÍCIO": TIME = Ø
20 FOR I% = 1 TO 1ØØØ : NEXT I%
30 PRINT "AGORA TRANSCORRERAM";
   TIME / 60; "SEGUNDOS"
RUN
INÍCIO
AGORA TRANSCORRERAM
.78333333333333 SEGUNDOS
```

O programa exemplo mostra que o computador levou Ø.7833... segundos para executar a instrução FOR ... NEXT da linha Nº 2Ø. As outras variáveis especiais são explicadas mais detalhadamente no Manual de BASIC.

2.2.1Ø. EXPRESSÕES E OPERAÇÕES

- **EXPRESSÕES**

Uma expressão é uma composição de tipo aritmético, formada por constantes e variáveis combinadas com operadores algébricos.

Exemplo:

```
1Ø * 5 – 2
1Ø + 3/2
6 + (5 + 2) * 3
```

Uma expressão produz como resultado um valor único.

Os operadores realizam operações matemáticas e operações lógicas.

Os operadores do BASIC podem ser classificados em 4 categorias:

1. Aritméticos
2. Relativos
3. Lógicos
4. Funcionais

1. OPERAÇÕES ARITMÉTICAS

Entende-se por operação aritmética, uma das seguintes operações:

1. Adição
2. Subtração
3. Multiplicação
4. Divisão
5. Potenciação

Os operadores (símbolos) utilizados são os seguintes:

Prioridade	Operador	Operação	Exemplo	Significado
1	^	Potenciação	5 ^ 3	5^3
2	–	Sinal	–5	–5
3	*	Multiplicação	5 * 3	5 x 3
3	/	Divisão	6/3	6 ÷ 3
4	+	Adição	9 + 3	9 + 3
4	–	Subtração	9 – 3	9 – 3

A prioridade indica a ordem de execução das operações respeitada pelo BASIC. Essa ordem pode ser alterada mediante o uso de parênteses:

```
10 + 5 * 6 + 3 / 2  =  41.5

10 + 5 * (6 + 3) / 2  = 28
```

Usando a barra (/) como operador na divisão o quociente será calculado como número decimal.

Exemplo:

```
? 16 / 5
  3.2
```

Podemos também obter como quociente, somente a parte inteira separada do resto da divisão. Usando para isso a barra invertida (\ ).

```
? 16 \ 5
  3
```

O resto da divisão poderá ser obtido mediante o operador MOD (módulo).

Exemplo:

```
? 16 MOD 5
  1
```

Nesse tipo de divisão, sem decimais no quociente, mesmo eventuais partes decimais nos operandos serão desprezados.

Exemplo:

```
? 23.80 \ 5  ( = 23 \ 5)
  4
```

Ainda no cálculo do resto através do operador MOD, a parte decimal será eliminada antes da execução do cálculo (efetua-se um truncamento e não um arredondamento !).

Exemplo:

```
? 23.80  MOD 5
  3
```

Ao se efetuar uma divisão com divisor igual a zero, aparecerá a mensagem de erro "DIVISÃO POR ZERO".

Um outro tipo de erro que poderá se verificar durante a execução dos cálculos, é o de "OVERFLOW", que indica que o resultado da operação é um número excessivamente grande.

Exemplo:

```
A % = 1000 * 1000
   ou
B # = 2 ^ 1000
```

2. OPERAÇÕES RELACIONAIS

Os operadores de relação são utilizados para comparar dois valores numéricos. O resultado poderá ser: verdadeiro (−1) ou falso (Ø).

Esses operadores são utilizados principalmente nas instruções de decisão (do tipo IF) para modificar o fluxo de um programa.

Os "operadores relacionais" são os seguintes:

Operador	Significado	Exemplo
=	Igual a	A = B
< >	Diferente de	A < > B
<	Menor que	A < B
>	Maior que	A > B
< =	Menor ou igual	A < = B
> =	Maior ou igual	A > = B

1Ø = 1Ø	Verdadeiro (−1)
1Ø > 5	Verdadeiro (−1)
5 > 1Ø	Falso (Ø)
1Ø < > 1Ø	Falso (Ø)

No caso de utilizar "operadores relacionais" junto com "operadores aritiméticos", primeiro serão calculadas as expressões aritméticas e depois serão usados os operadores.

Exemplo:

```
10 + 5 * 2 = 10 * 6 + 5
```

Primeiro serão calculados os valores à esquerda e à direita do operador:

```
20 = 65
```

e depois será obtido o resultado da expressão: FALSO (Ø).

Os "operadores relacionais" são freqüentemente utilizados nas expressões condicionais, isto é, nas instruções tipo IF do BASIC.

Exemplo:

```
IF A = B THEN 5Ø
IF I MOD J < > 5 THEN K = K + 1
```

Os operadores relacionais podem ser usados também em expressões e instruções de atribuição.

Exemplo:

X = (5 = 5)	(X = −1)	isto é verdadeiro.
Y = (5 < 3)	(Y = 0)	porque 5 não é menor que 3.
Z = (5 * 3 < > 15)	(Z = 0)	

OPERAÇÕES LÓGICAS

Os "operadores lógicos" são usados para determinar condições múltiplas ou para executar operações sobre um único bit ou para operações Booleanas.

Usam-se os seguintes operadores lógicos:

NOT, AND, OR, XOR, IMP, EQV

Vamos supor que X e Y são duas expressões lógicas que podem assumir os valores 1 (expressão verdadeira) ou Ø (expressão falsa).

A condição múltipla (poderá ser uma instrução do BASIC)

IF X OR Y THEN 200

indica que o programa deverá saltar à instrução 2ØØ, se X ou Y são expressões verdadeiras, isto é, se pelo menos uma das duas (mas não necessariamente todas as duas) é verdadeira, ou seja, quando X ou Y ou ambas valem 1.

O resultado de um OR é falso (vale Ø) somente quando ambas condições são falsas e valem zero.

Entendendo isso, as seguintes tabelas serão interpretadas da seguinte forma:

À esquerda achamos os valores de X e de Y (Ø e 1 que representam falso e verdadeiro respectivamente), à direita aparece o resultado de cada operação. Observando particularmente a tabela OR, acharemos o resultado Ø somente quando X e Y valem Ø, nas outras três combinações o resultado é sempre 1.

O resultado de uma operação AND é verdadeiro somente se ambos operandos são expressões verdadeiras (se X e Y valem 1), em todos os outros casos o resultado é falso.

XOR (chamado também de OR exclusivo) dá o resultado 1 quando X ou Y (mas não ambos) valem 1.

Os outros operadores (de uso menos freqüente), dão os resultados indicados na tabela.

NOT (negação lógica)

X	NOT X
1	Ø
Ø	1

AND

X	Y	X AND Y
1	1	1
1	Ø	Ø
Ø	1	Ø
Ø	Ø	Ø

OR

X	Y	X OR Y
1	1	1
1	Ø	1
Ø	1	1
Ø	Ø	Ø

XOR (OR exclusivo)

X	Y	X XOR Y
1	1	Ø
1	Ø	1
Ø	1	1
Ø	Ø	Ø

IMP (implicação)

X	Y	X IMP Y
1	1	1
1	Ø	Ø
Ø	1	1
Ø	Ø	1

EQV (equivalência)

X	Y	X EQV Y
1	1	1
1	Ø	Ø
Ø	1	Ø
Ø	Ø	1

Como já foi dito anteriormente, os "operadores lógicos" são freqüentemente utilizados em expressões condicionais do tipo IF para determinar condições múltiplas.

Exemplo:

```
IF X < Ø OR X > 99 THEN 1ØØ
```

Essa linha de programa manda seguir para a linha 1ØØ caso X seja menor que Ø ou então maior que 99. Obviamente as duas

condições não podem ser verdadeiras simultaneamente nesse caso, mas poderiam ser caso a instrução tivesse sido:

```
IF X > Ø AND X < 99 THEN 1ØØ
```

4. OPERAÇÕES COM STRINGS DE CARACTERES OU FUNCIONAIS

Os "strings" podem ser concatenados usando +

Exemplo:

```
1Ø A$ = "NOME" : B$ = ".DE." : C$ = "ARQUIVO"
2Ø PRINT A$ + B$ + C$
3Ø PRINT "NOVO." + A$ + B$ + C$
RUN
NOME.DE.ARQUIVO
NOVO.NOME.DE.ARQUIVO
```

Os "strings" podem também ser comparados usando os "operadores de relação" explicados anteriormente.

```
=  <>  <>  <=  >=
```

As comparações entre "strings" são feitas comparando os códigos ASCII dos caracteres, tomando um de cada vez de cada "string". Se os códigos ASCII diferirem, o número de código inferior precederá ao superior. Se durante a comparação dos strings, o fim de um dos dois for alcançado, o "string" mais curto será considerado menor. Os espaços em branco, precedendo ou sucedendo ao "string", são significativos.

Exemplos:

```
"AA" < "AB"
"NOME" = "NOME"
"X $" > "X #"
"CLI" > "CL"
"AN" > "AM"
"fg" > "FG"
B$ < "9/12/83"          onde B$ = "8/12/83"
```

Assim, a comparação entre "strings" pode ser usada para testar valores de "strings" ou para ordená-los segundo o alfabeto.

Todas as constantes "string" usadas em expressões de comparação devem ser escritas entre aspas.

2.2.11. MATRIZES

Quando em um programa são utilizados muitos dados constantes é aconselhável registrá-los em uma tabela, usando para isso a instrução "DATA". Os dados são chamados da tabela através da instrução "READ".

Exemplo:

```
NEW
1Ø READ X$ : PRINT X$
2Ø GO TO 1Ø
```

```
3Ø DATA JAN, FEV, MAR
RUN
JAN
FEV
MAR
Sem "DATA" em 1Ø
```

Os elementos de dados que acompanham a instrução DATA são separados entre si por uma vírgula.

A instrução "READ" retira da tabela (criada pela instrução DATA), os elementos de informação em forma sucessiva.

A mensagem de erro "Sem DATA em 1Ø" indica que não existem mais dados disponíveis na instrução DATA.

A instrução "RESTORE" força a leitura dos dados da instrução "DATA" a partir do primeiro elemento.

Se anexarmos ao programa acima, uma linha

```
15 RESTORE
```

o programa repetiria indefinidamente a palavra JAN até ser interrompido pela pressão simultânea das teclas [CTRL] + [STOP] .

- MATRIZES DE VARIÁVEIS

Em uma tabela é possível também memorizar variáveis. O uso dessas tabelas ou matrizes, requer em primeiro lugar, uma declaração do tipo DIM.

Ao encontrar uma instrução DIM (20) por exemplo, o BASIC cria uma tabela ou matriz unidimensional de 21 elementos, os quais poderão ser chamados por exemplo, de A (Ø), A (1), ..., A (2Ø).

As matrizes mais usadas são as unidimensionais, bidimensionais e tridimensionais. O computador pode trabalhar com até 255 dimensões, mas a capacidade de memória é quem determinará o número máximo de elementos. Da mesma forma que as variáveis ordinárias, as matrizes podem ser de diversos tipos:

A$ (3Ø) indica uma matriz de tipo "string" alfanumérica.
A (12) é uma matriz real de dupla precisão.
A# (7) também é uma matriz real de dupla precisão.
A! (8) é uma matriz de simples precisão.
A% (6) é uma matriz inteira (seus elementos são números inteiros).

É possível atribuir um valor a cada elemento de uma matriz.

```
1Ø  DIM A$ (20)
2Ø  A$ ( Ø ) = "M"
3Ø  A$ ( 1 ) = "A"
4Ø  A$ ( 2 ) = "T"
5Ø  A$ ( 3 ) = "R"
6Ø  A$ ( 4 ) = "I"
7Ø  A$ ( 5 ) = "Z"
8Ø  A$ ( 6 ) = "  "
9Ø  A$ ( 7 ) = "N"
1ØØ A$ ( 8 ) = "O"
11Ø A$ ( 9 ) = "V"
12Ø A$ (1Ø) = "A"
```

A$ (Ø)	A$ (1)	A$ (2)	A$ (3)	A$ (4)	A$ (5)
M	A	T	R	I	Z

A$ (6)	A$ (7)	A$ (8)	A$ (9)	A$ (1Ø)
	N	O	V	A

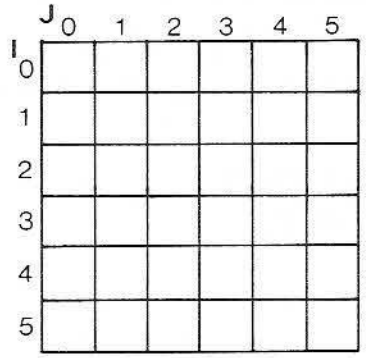
O número entre parênteses é chamado de índice do elemento, e pode ser um número ou uma variável numérica.

O uso de variáveis como índices permite escrever programas do tipo:

```
1Ø  DIM A$ (2Ø)
2Ø  DATA  M, A, T, R, I, Z, "  ", N, O, V, A
3Ø  FOR  I = 1 TO 11
4Ø  READ A$ (I)
5Ø  NEXT I
```

Note que essas 5 instruções equivalem às 12 anteriores. Quando a dimensão de uma matriz é menor ou igual a 10 a instrução DIM poderá ser omitida.

Uma matriz do tipo A (1Ø, 1Ø) é chamada de bidimensional. Quando declaramos uma matriz DIM A (5,5), o computador cria 36 elementos (e não 25!), dispostos da seguinte forma: A (I,J)



Uma matriz com 3 índices ou argumentos, é chamado de tridimensional, e assim por diante.

As matrizes criadas pela declaração DIM podem ser apagadas durante a execução do programa através da instrução ERASE.

2.2.12. DECISÕES

O fluxo de um programa pode ser modificado como já vimos no parágrafo referente a operações lógicas, utilizando a instrução "IF", como no exemplo seguinte.

```
1Ø  INPUT A, B
2Ø  IF A = B THEN PRINT "A = B"  ELSE PRINT "A <> B"
```

Nesse caso, na linha nº 20 são comparados os conteúdos das variáveis A e B. Com a instrução IF, é perguntado ao BASIC se uma determinada condição (neste caso a igualdade entre A e B) é verdadeira. Caso seja, é executada a instrução que segue à palavra "THEN". Se não for, o programa seguirá a instrução indicada pela palavra "ELSE". A palavra "ELSE" seguida de uma instrução na mesma linha de programa iniciado pela instrução IF, é facultativa. Se a linha não contiver a palavra ELSE, o programa segue automaticamente para a linha sucessiva, sempre que a condição não é verdadeira.

Se a palavra "THEN" é seguida pela instrução "GO TO", não se deve escrever a palavra THEN. Uma alternativa consiste em cancelar a palavra "GO TO", como ilustra o seguinte exemplo:

```
1Ø  INPUT A, B
2Ø  IF A = B THEN 5Ø  ELSE  GO TO 4Ø
3Ø  END
4Ø  PRINT "A <> B" : GO TO 3Ø
5Ø  PRINT "A = B" : GO TO 3Ø
```

- ITERAÇÕES

Vejamos agora o seguinte exemplo:

```
1Ø  CT = 1
2Ø  PRINT "TESTE"
3Ø  CT = CT + 1
4Ø  IF CT < 11  GO TO 2Ø
5Ø  END
```

Ao executar esse programa, a palavra "TESTE" será impressa na tela 1Ø vezes.

Uma outra forma, mais fácil, de obter o mesmo resultado seria:

```
NEW
1Ø  FOR CT = 1  TO 1Ø
2Ø  PRINT "TESTE"
3Ø  NEXT CT
4Ø  END
RUN
```

A instrução FOR CT = 1 TO 1Ø faz com que as linhas 2Ø e 3Ø do programa sejam repetidas enquanto CT for menor ou igual a 1Ø.

A instrução NEXT faz com que CT seja incrementado no valor 1 cada vez que o programa passa por ela. Se quisermos um incremento diferente de 1 deveremos adicionar na instrução FOR, a palavra "STEP" seguida do valor do incremento.

Os valores da instrução FOR podem também ser substituídos por uma variável.

Exemplo:

```
1Ø  A = 4
2Ø  FOR CT = 8 TO A STEP – 2
3Ø  PRINT "TESTE"
4Ø  NEXT CT
RUN
```

Nesse exemplo, o valor de CT é decrementado e não incrementado, por causa do valor negativo de STEP. Nesses casos é necessário que o valor inicial seja maior que o final.

- **ORDENAÇÃO**

Damos a seguir um exemplo de um programa de ordenação. Uma série de números são fornecidos pelo usuário, numa determinada seqüência. O usuário deverá informar ao computador qual vai ser a quantidade total de números introduzidos e logo a seguir digitar esses números.

```
10 DIMG(100)
20 INPUT"QUANTIDADE DE NUMEROS(ENTRE 2 E 100)";A
30 IF(A<2)OR(A>100)GOTO20
40 FORI=1TOA
50 PRINT"NUMERO";I;
60 INPUTG(I)
70 NEXTI
80 PRINT"ESTOU ORDENANDO"
90 FORI=2TOA
100 X1=1:X2=I
110 X3=X1+INT((X2-X1)/2)
120 IFG(I)<G(X3)THEN150
130 IFG(I)>G(X3)THEN170
140 X2=X3
150 IFX2=X3THEN190
160 X2=X3:GOTO110
170 IFX1=X3THEN190
180 X1=X3:GOTO110
190 H=G(I)
200 FORJ=ITOX2+1STEP-1
210 G(J)=G(J-1)
220 NEXTJ
230 G(X2)=H
240 NEXTI
250 PRINT"ESSES SÃO OS NUMEROS EM SEQUENCIA"
260 FORI=1TOA
270 PRINTG(I);
280 NEXTI
```

- PROGRAMAS COM MENÚ

Alguns programas têm a possibilidade de executar diversas funções, estando as opções listadas em um menú fornecido ao usuário do programa no início do mesmo.

Vejamos o exemplo:

```
10 PRINT"MENU DE OPÇOES"
20 PRINT"1=LINHA NUMERO 100"
30 PRINT"2=LINHA NUMERO 200"
40 PRINT"3=FIM DO PROGRAMA"
50 INPUT"FAÇA A SUA ESCOLHA";A
60 IF(A<1)OR(A>3)GOTO50
70 ONAGOTO100,200,300
100 PRINT"ESTA E A LINHA NUMERO 100":GOTO10
200 PRINT"ESTA E A LINHA NUMERO 200":GOTO10
300 END
```

Na linha 7Ø desse programa, a instrução ON A GO TO [lista de linhas] faz com que, quando a variável assuma o valor 1, o programa se desvie para a linha indicada no primeiro lugar da lista de linhas, com o valor 2 passe para o segundo endereço e assim por diante.

- O TRATAMENTO DOS ERROS

O BASIC tem a possibilidade de controlar e corrigir os erros verificados durante a execução de um programa.

Vejamos o exemplo:

```
10 ONERRORGOTO50
20 INPUT"NUMERO";A%
30 END
50 A$="O NUMERO DEVE ESTAR COMPREENDIDO
   ENTRE -32768 E +32767"
60 IFERR=6THENPRINTA$:RESUME0
70 ONERROR GOTO0
```

Nesse programa, a instrução ON ERROR GO TO, da linha 1Ø faz com que no caso de acontecer um erro, o programa seja desviado à linha 5Ø. Se o usuário introduzir um número menor que — 32768 ou maior que + 32767 como resposta à pergunta da linha 2Ø, o BASIC acusa um erro, já que A% é uma variável inteira. O erro de "OVERFLOW" recebe o número de erro 6 (vide tabela de Mensagem de Erro). Nesse caso aparecerá na tela a mensagem escrita na linha 5Ø.

A instrução RESUME da linha 6Ø, faz com que o programa seja continuado.

O HOT-BASIC permite também ao usuário definir seus próprios códigos, de erro. Nesse caso, é preciso definir códigos superiores aos usados pelo BASIC.

Vejamos o exemplo:

```
10 ONERRORGOTO100
20 INPUT"FIM DO PROGRAMA";A$
30 IFA$="SIM"THENERROR250
40 PRINT"EXECUTE PROGRAMA":GOTO20
50 END
100 IFERR=250THENINPUT"TEM CERTEZA";A$:IFA$=
    "SIM"THENRESUME50ELSERESUME20
110 ONERRORGOTO0
```

Na linha 30 é utilizado o código de erro 250 quando a variável A$ é igual a "SIM". Nesse caso é executado a linha 100, como vem indicado na linha 10.

Na linha 100 é perguntado se o uso da sub-rotina de erro foi provocado pelo erro 250. Caso o seja, pede uma confirmação ao usuário, após o qual o programa continua graças ao RESUME 50. Se a resposta foi negativa, o programa continuará com a linha 20 através do RESUME 20.

2.2.13. SUB-ROTINAS

Suponhamos ter carregado na memória o seguinte programa:

```
10 INPUTA:INPUTB
20 C=A*100/B:PRINTA;
30 PRINT"E 0";C;"% DE";
40 PRINTB
50 C=B*100/A:PRINTB;
60 PRINT"E 0";C;"% DE";
70 PRINTA
80 END
```

Sem dúvida, deve ter percebido que as linhas 30 e 60 são idênticas. Pode-se nesse caso, utilizar uma única linha e enviar a execução a essa linha usando a instrução GOSUB cada vez que seja necessário.

O programa ficaria da seguinte forma:

```
10 INPUTA:INPUTB
20 C=A*100/B:PRINTA;
30 GOSUB90
40 PRINTB
50 C=B*100/A:PRINTB;
60 GOSUB90
70 PRINTA
80 END
90 PRINT"E 0";C;"% DE";
100 RETURN
```

Ao chegar na linha 30, o BASIC continuará o programa pulando para a linha 90. Quando ele acha a instrução "RETURN", volta para a linha 40. O mesmo acontece na linha 60.

As instruções contidas entre as linhas 9Ø e 1ØØ são chamadas de "sub-rotina". Quando o BASIC chega na linha 9Ø como resultado de um "GOSUB", lembra a linha da qual saiu, e ao achar o "RETURN" volta para a linha seguinte àquela da qual saiu.

Neste programa, a linha 8Ø é particularmente importante. Caso ela não existisse, e a linha 9Ø seguisse à 7Ø, o BASIC mandaria uma mensagem de erro "RETURN" SEM "GOSUB" ao chegar na linha 1ØØ.

NOTA: Ver mais informações sobre sub-rotinas no item 2.10 deste capítulo, referente a interrupções.

2.2.14. FUNÇÕES

Além dos comandos e das instruções, o BASIC dispõe também de um grande número de funções, que fornecem o resultado de um cálculo com uma constante ou uma variável.

Essas funções só podem ser utilizadas em combinação com uma instrução.

Exemplo:

```
NEW
1Ø  PRINT SIN (5)
RUN
-.9589242746631
```

ou então

```
NEW
1Ø  X = 5
2Ø  PRINT SIN (X)
RUN
-.9589242746631
```

Damos a seguir um resumo completo de todas as funções disponíveis no Manual de BASIC. As funções podem ser divididas nos seguintes grupos:

1. FUNÇÕES MATEMÁTICAS:
   ABS, ATN, COS, EXP, INT, LOG, SGN, SIN, SQR, TAN

2. FUNÇÕES ALFANUMÉRICAS:
   ASC, LEFT$, LEN, MID$, RIGHT$, STR$, STRING$, VAL, BIN$, HEX$, OCT$, INSTR$

3. FUNÇÕES DE CONVERSÃO:
   CDBL, CINT, CSNG, FIX

4. FUNÇÕES DE ENTRADA:
   INKEY$, INPUT$

5. FUNÇÕES DE SAÍDA:
   CHR$, POS, LPOS, SPC, TAB

6. FUNÇÕES DIVERSAS:
   CSRLIN, ERL, ERR, FRE, PEEK, RND, USR, VARPTR, POINT, VPEEK, STICK, STRIG, PDL, PAD, PLAY, EOF

O usuário poderá também definir as suas proprias funções usando a instrução DEF FN. (ver Manual de BASIC)

2.3. BITS E BYTES

Um bit é uma informação (elementar) representada por uma cifra binária. Em termos elementares, a presença ou a ausência, o + ou o — , o alto ou o baixo, podem ser representados por uma informação de um único bit. Estudando em detalhe a lógica de trabalho do computador, verificaremos que ela também se origina na informação proporcionada por um único bit, cujo valor está representado pela presença ou ausência de sinal elétrico.

Um grupo de 8 bits é chamado de byte. Enquanto um bit pode somente assumir os valores Ø ou 1, representando conseqüentemente dois estados, um byte pode variar entre ØØØØØØØØ e 11111111 e conter $2^8 = 256$ informações. (Fig. 32)

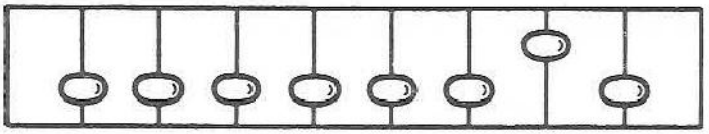
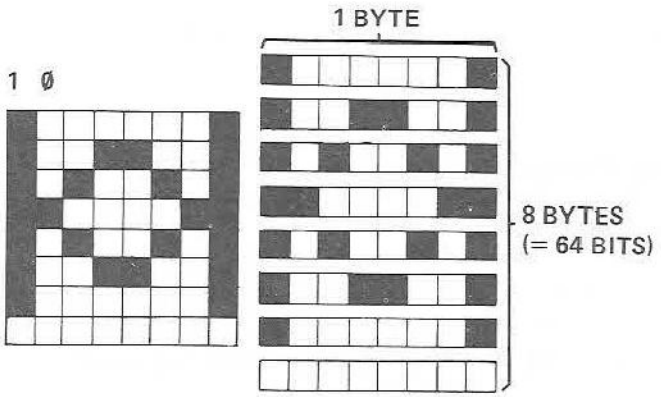


Fig. 32: Bits ou Bytes

Por ser o nosso um computador de 8 bits, ele pode manipular dados de 1 byte de cada vez. Por esse motivo, os caracteres e os códigos de controle poderão ter valores compreendidos entre Ø e 255, e os conteúdos da memória são lidos e escritos de um byte por vez.

Por esse mesmo motivo o conceito de bits e bytes aplica-se também a parte gráfica (por exemplo, no SPRITE$).

Para definir dados a serem mandados para o vídeo com a função SPRITE como no exemplo a seguir, são necessários oito bytes de dados:



EM BINÁRIO	EM DECIMAL
10000001	129
10011001	153
10100101	165
11000011	195
10100101	165
10011001	153
10000001	129
00000000	0

Falemos novamente sobre os dígitos binários.

Um número decimal está composto por cifras entre 0 e 9, e todos os números em notação decimal são ordenados da seguinte forma:

0, 1, 2, 3 ... 9, 10, 11 ...

Após o 9 vem o 10, após o 99 vem o 100 e assim por diante.

O número 1983 pode ser decomposto da seguinte maneira:

$$1983 \rightarrow 1 \times 10^3 \quad 9 \times 10^2 \quad 8 \times 10^1 \quad 3 \times 10^0$$

Um número binário está formado somente pelas cifras 0 e 1, e a numeração é feita da seguinte forma:

0, 1, 10, 11, 100, ...

O número binário 10101 poderá ser decomposto por sua vez assim:

$$10101 \rightarrow 1 \times 2^4 \quad 0 \times 2^3 \quad 1 \times 2^2 \quad 0 \times 2^1 \quad 1 \times 2^0$$

Por ser o byte um número binário de 8 cifras ele poderá representar qualquer número entre 0 e 255.

$$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 = 255$$

$$\times 2^7 \qquad \times 2^0$$

O bit da extrema direita, o correpondente a 2^0 é chamado de bit menos significativo (LSB). O bit da extrema esquerda correspondeente a 2^7, é chamado de bit mais significativo (MSB).

O usuário poderá, achar alguma dificuldade na conversão de números binários para decimais e vice-versa. Por outro lado, pelo fato de possuírem tantas cifras, as operações com números binários são demoradas e complexas.

Para resolver esses problemas provocados pelos números binários, tem sido introduzido o conceito de número hexadecimal, cuja conversão para o binário (e vice-versa) é muito simples e requer o uso de menos número de cifras.

Para representar um número hexadecimal, usam-se 16 cifras:

0, 1, 2 ... 9, A, B, C, D, E, F

O A representa o 10 decimal, o B representa o 11 decimal e assim por diante; os valores de 0 a 15 são representados por um único caracter (ou cifra) hexadecimal. Os números de duas cifras, isto é, os superiores a 15 são ordenados da seguinte forma:

10, 11, ... , 1A, 1B, ... , 1F, 20, ... , 9F, A0, ... , FF, 100, ...

O valor decimal de 3F é:

$$3F = 3 \times 16^1 + 15 \times 16^0 = 63$$

x 16^1 x 16^0

De maneira análoga, a terceira coluna vale $16^2 = 256$, a quarta vale $16^3 = 4096$, etc.

Por ser $16 = 2^4$, quatro cifras binárias podem ser representadas por uma única cifra hexadecimal.

BINÁRIO		DECIMAL		HEXADECIMAL
0000	=	0	=	0
0001	=	1	=	1
0010	=	2	=	2
1001	=	9	=	9
1010	=	10	=	A
1011	=	11	=	B
1100	=	12	=	C
1101	=	13	=	D
1110	=	14	=	E
1111	=	15	=	F
1 0000	=	16	=	10

Nessa forma, um número binário pode facilmente ser convertido em um número hexadecimal.

11011010 = DA

D A

11110011 = F3

F 3

Um número binário de 8 cifras, ou seja 1 byte, ou seja um número decimal compreendido entre 0 e 255, pode ser escrito com duas cifras hexadecimais.

O HOT-BASIC pode trabalhar também com notação octal, que usa as cifras Ø a 8.

1 2 = 10

x 8^1 x 8^0

Vejamos por exemplo, como seria escrito o número decimal 175 nas outras notações.

	Binário	Octal	Decimal	Hexadecimal
Representação	&B10101111	&Ø257	175	&HAF
Para obter o string	BIN$ (175)	OCT$ (175)	STR$ (175)	HEX$ (175)

As cifras que seguem a &B, &0 e &H (esses símbolos não são cifras) são convertidos em strings de caracteres usando respectivamente as funções BIN$, OCT$ e HEX$.

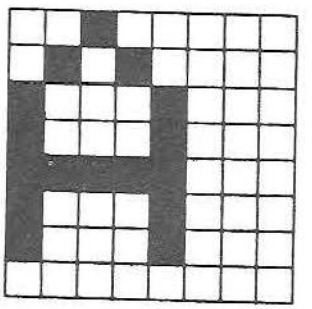
Vejamos os seguintes exemplos de conversão:

```
N  = &H AF
N  = VAL ("&H" + "AF")        N = 175
N$ = HEX$ (AS)                N$ = "AF"
```

- **GERAÇÃO DE CARACTERES**

A geração de caracteres é realizada através de uma estrutura de dados, usada pelo computador para a visualização desses caracteres.

Um caracter está composto por um conjunto de 8 x 8 pontos no vídeo. Por exemplo, a letra A maiúscula, é formada por um grupo de pontos, dispostos da seguinte forma:



Ao ligar o computador, durante a fase de iniciação, os dados originadores dos caracteres de (CHR$ (Ø) até CHR$ (255) são carregados numa parte *da RAM destinada* a esse uso.

Os dados correspondentes a qualquer um dos caracteres, ocupam 8 bytes, cada um dos quais contém 8 pontos da tela.

Um caracter terá, conseqüentemente, uma representação similar àquela de um sprite. Para representar 256 caracteres, ocuparemos então 256 x 8 = 2048 bytes.

Mudando os dados dos caracteres, é possível atribuir a um caracter uma forma diferente no vídeo.

Como esses dados estão memorizados numa área particular da RAM de vídeo, deverão ser utilizadas as instruções VPEEK e VPOKE para modificá-los.

Vejamos como criar a letra A:

Os 255 caracteres estão dispostos em unidades de 8 bytes, começando pelo endereço 2048, isto é, o CHR$ (0) está colocado entre os endereços 2048 ∿ 2071.

O código correspondente à letra A é CHR$ (65), sendo o endereço do seu primeiro byte : 2048 + 65 x 8 = 2568 e o endereço do último byte 2575.

Vejamos o seguinte programa:

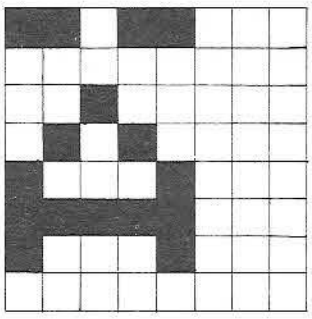
```
10  FOR I = 2568 TO 2575
20  D = VPEEK (I)
30  PRINT RIGHT$ ("00000000" + BIN$ (D), 8)
40  NEXT
RUN
00100000
01010000
10001000
10001000
11111000
10001000
10001000
00000000
```

Dessa forma, os dados geradores de caracteres são representados na mesma forma que os dados dos SPRITE.

Mudemos agora os dados da letra A com o seguinte programa.

```
10  FOR I = 2568 TO 2575
20  READ B$
30  VPOKE I, VAL ("&B" + B$)
40  NEXT I
50  END
60  DATA 11011000
70  DATA 00000000
80  DATA 00100000
90  DATA 01010000
100 DATA 10001000
110 DATA 11111000
120 DATA 10001000
130 DATA 00000000
```

Executando este programa veremos que no vídeo o caracter A é visualizado como A.



Se os dados que originam os caracteres A — Z e a — z forem mudados, será difícil ler a listagem do programa. Por esse motivo é conveniente colocar os novos caracteres após CHR$ (123).

A explicação precedente supõe que o computador está trabalhando no modo Ø (modo "default" do computador).

Trabalhando no modo 1, os endereços de localização dos caracteres começam a partir de Ø (e não do 2Ø48 como no modo Ø).

O modo Ø usa uma estrutura de caracteres de 6 pontos de largura por 8 de altura, sendo assim é possível visualizar no vídeo um grande número de caracteres. Note-se que os dois últimos bits são ignorados.

2.4. INTERFACES

Como já foi explicado no capítulo referente a "noções gerais", a CPU pode, além de fazer cálculos, receber ou transmitir dados (sinais elétricos) de/e para o mundo externo.

Esses sinais elétricos diferem nas suas características e na velocidade de transmissão. Usando por exemplo o gravador e cassete, depois que os sinais forem convertidos em sinais de som, eles serão mandados muito lentamente ao gravador, já que a sua velocidade de registro é relativamente baixa. Pelo contrário, usando um disquete, os sinais poderão ser mandados a uma velocidade muito maior e não há necessidade de convertê-los em sinais de som. Vemos então que é necessário utilizar uma interface para modificar os sinais transmitidos a aparelhos com estruturas diversas. (Fig. 33)

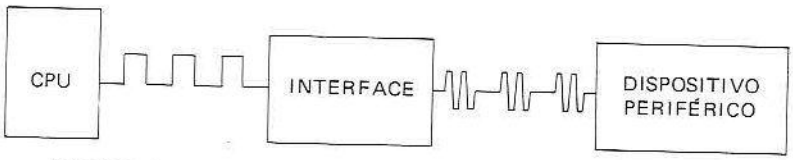


Fig. 33: Sinais transmitidos utilizando interface

A função da interface é a de intérprete entre a CPU e os dispositivos periféricos.

Como dispositivos standard, a nossa CPU possui interface para teclado, para gravador e cassete, para monitor e TV, para joysticks e para impressora. Todos esses dispositivos podem ser usados livremente.

Adicionalmente, prévia ligação das interfaces correspondentes nos slots de extensão, poderão ser usados "floppy disks" e o circuito RS232C.

A palavra "interface" pode ser usada num contexto mais amplo. Podemos por exemplo dizer, que o teclado e o monitor ou TV são interfaces entre o computador e o homem. (Fig. 34)

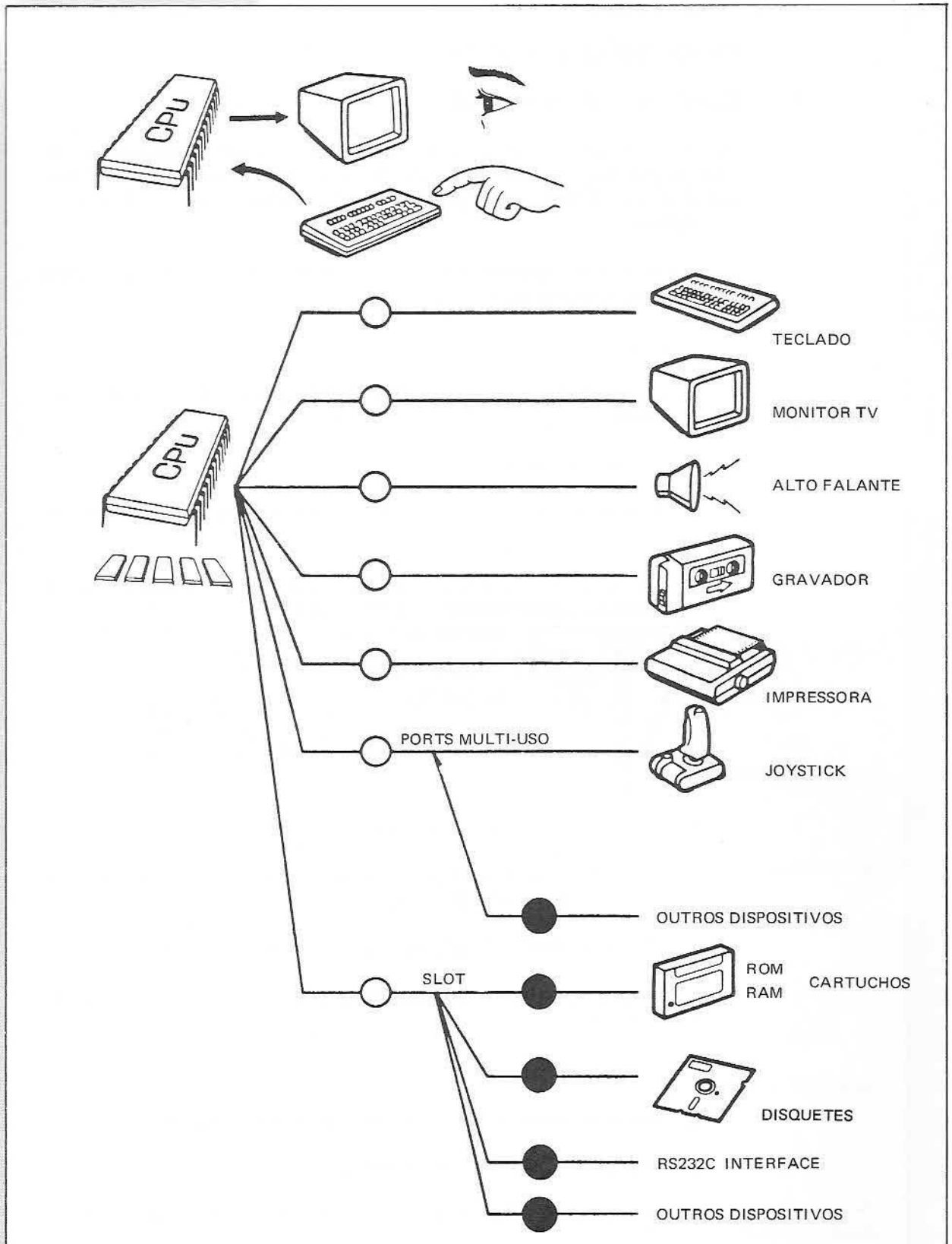


Fig. 34: Interfaces

2.5. REGISTRO DE DADOS COM CASSETE

2.5.1. COMO USAR O GRAVADOR

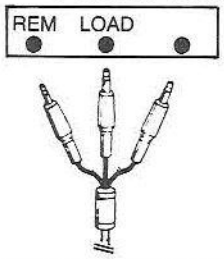
Ao ser desligado o computador, ou no caso de falta momentânea de energia elétrica, o programa existente na memória é perdido. Querendo salvar um programa para a sua posterior utilizaçãc pode-se gravá-lo em fita cassete.

Toda vez que for necessário acessar esse programa na memória, bastará recarregá-lo. Vejamos qual é o procedimento a seguir:

- O GRAVADOR

  Para obter bons resultados, o gravador deverá ter as seguintes características:

  - Possuir um contador de giros.
  - Possuir um terminal "REMOTE" para controlar o acionamento do motor.
  - Ser "mono", ou então "estereo" mas com a possibilidade de trabalhar como"mono", unindo as duas pistas.



PLUG VERMELHO (grande) – Conecte à tomada SAVE ("MIC") do gravador
PLUG PRETO (grande) – Conecte à tomada LOAD ("EAR") do gravador
PLUG PRETO (pequeno) – Conecte à tomada REM ("REMOTE") do gravador
NOTA: Em alguns gravadores não existe a tomada REMOTE. Nesse caso não é necessário conectar o plugue.

Fig. 35: Cabo de ligação para o gravador

O computador vem acompanhado de um cabo de ligação para o gravador, do tipo mostrado na figura 35.

- COMO "SALVAR" UM PROGRAMA

  Um programa pode ser transferido a uma fita através do comando CSAVE.

  O procedimento a seguir é:

  a) Inserir uma fita no gravador e rebobiná-la totalmente.

  b) Avançar a fita até superar o trecho de guia inicial.

  c) Pressionar simultâneamente as teclas PLAY e REC do gravador.

  d) Introduzir o comando CSAVE seguido no nome do programa, sob o qual ficará registrado na fita. O nome deverá ser escrito entre aspas (CSAVE "nome").

  e) Pressionar a tecla "RETURN".

  O programa será transferido à fita cassete.

- **COMO "LER" UM PROGRAMA GRAVADO EM FITA**

Um programa pode ser lido na memória do computador usando o comando CLOAD.

a) Inserir a fita que contém o programa desejado, no gravador, e rebobiná-la completamente.

b) Pressionar a tecla PLAY do gravador.

c) Introduzir o comando CLOAD seguido do nome do programa entre aspas (CLOAD "nome").

d) Pressionar a tecla RETURN.

Se introduziu o comando CLOAD sem nome do programa, o primeiro programa gravado na fita será carregado na memória do computador.

Quando o BASIC encontra um nome de programa diferente daquele solicitado, aparece na tela a mensagem:

PULEI (nome do programa)

Nesses casos, o BASIC continua com a procura do programa. Quando ele é achado, aparece na tela a mensagem:

ACHEI (nome do programa)

No fim da recuperação, o BASIC volta ao nível de comandos.

- **VERIFICAÇÃO DE UM PROGRAMA GRAVADO**

Após ter gravado um programa na fita, é conveniente verificar se a gravação foi realizada corretamente.

Para isso, o BASIC dispõe do comando CLOAD ? "nome".

a) Rebobina-se a fita até a posição inicial de gravação.

b) Digita-se CLOAD ? "nome" e pressiona-se a tecla RETURN.

A fita começa a rodar, e se o computador achar o programa previamente gravado, mandará ao vídeo a mensagem:

ACHEI "nome"

Se após alguns instantes essa mensagem não aparecer no vídeo, quer dizer que o programa não foi gravado corretamente, devido a algum problema de ligação ou de procedimento.

Se o computador verificar que o programa foi "salvo" corretamente, aparece na tela a mensagem OK, e a fita se detém.

Se o computador encontra alguma diferença entre o programa que ainda tem na memória, e aquilo que foi gravado, aparecerá a mensagem:

```
ERRO / VERIF
OK
```

ou seja, erro na fase de verificação. Nesse caso, será necessário voltar a gravar o programa.

- **MOTOR ON, MOTOR OFF**

Alguns gravadores possuem o dispositivo REMOTE que habilita ou inabilita o avanço da fita.

Nesses casos, o comando MOTOR habilita o motor do gravador.

Ao introduzir o comando MOTOR ON, o modo REMOTE é cancelado. Dando o comando MOTOR OFF, ele é reativado.

NOTAS SOBRE O USO DO GRAVADOR:

1. Fixar o nível de gravação num valor ligeiramente superior ao normal. Se o nível for muito baixo, o sinal do computador poderá ser perturbado por ruído externo, se for muito alto poderá haver distorção.

2. Fixar o controle de tom no setor de tons agudos. Nas altas freqüências é mais fácil gravar melhor as informações.

3. Dentro do possível, é conveniente gravar e carregar os programas com o mesmo gravador. A velocidade de gravação e o alinhamento dos cabeçotes de gravação pode diferir de um gravador para outro.

4. Manter sempre limpos os cabeçotes do gravador.

Para escrever um programa na fita, o computador transforma as linhas e as instruções em sinais elétricos, e os envia ao gravador, o qual procede a memorização em fita.

2.5.2. REGISTRO DE DADOS EM FITA

As informações gravadas na memória podem ser por sua vez gravadas e conservadas em fita cassete.

As informações são gravadas em seqüência. Esse grupo de informações é chamado de "arquivo". Cada arquivo recebe um nome, e deve ser "aberto" para com ele poder trabalhar. O número de arquivos com os quais o programa trabalha, deve ser especificado no começo do mesmo, com a instrução "MAXFILES".

Vejamos um exemplo:

```
1Ø MAXFILES = 1
2Ø OPEN "CAS : ENDER" FOR OUT PUT AS # 1
3Ø A = 15 : B = 23 : C = 25 : D = 11,5
4Ø PRINT # 1, A : B
5Ø PRINT # 1, C : D
6Ø CLOSE # 1
```

Nesse exemplo, na linha 2Ø é aberto o arquivo chamado "ENDER".

A palavra "CAS" indica que o arquivo é aberto em fita cassete. As informações são efetivamente registradas na fita com a instrução "PRINT #". Após cada instrução "PRINT #", é inserido na fita um código "CR" e um "LF". Após as variáveis numéricas é inserida automaticamente uma vírgula. No exemplo a seguir, uma "imagem" da fita:

15	,	23	CR	LF	25	,	11.5	CR	LF

Introduzindo variáveis alfanuméricas, a vírgula deve ser colocada manualmente, com a instrução "PRINT #".

Exemplo:

```
1Ø MAXFILES = 1
2Ø OPEN "CAS : NOME" FOR OUTPUT AS # 1
3Ø A$ = "MARIA" : B$ = "JOSÉ"
4Ø PRINT # 1 , A$; " , " ; B$
5Ø CLOSE # 1
```

Esse programa originará a seguinte gravação:

MARIA	,	JOSÉ	CR	LF

2.5.3. LEITURA DE DADOS DE FITA

Os dados registrados numa fita cassete podem ser recuperados para a memória do computador usando a instrução "INPUT #".

Para ler as variáveis numéricas do exemplo precedente, usamos as seguintes instruções:

```
1Ø MAXFILES = 1
2Ø OPEN "CAS : NOME" FOR INPUT AS # 1
3Ø INPUT # 1 , W, X
4Ø INPUT # 1 , Y, Z
5Ø CLOSE # 1
```

Para ler as variáveis alfanuméricas, o procedimento é o seguinte:

```
1Ø MAXFILES = 1
2Ø OPEN "CAS : NOME" FOR INPUT AS # 1
3Ø INPUT # 1 , Y$ , Z$
4Ø CLOSE # 1
```

Uma instrução "INPUT #" preenche as variáveis com as informações encontradas até a primeira vírgula ou até o código "CR" e "LF".

Uma instrução "LINE INPUT #" preenche as variáveis indicadas, com as informações encontradas até o próximo "CR" e "LF", como no exemplo seguinte:

```
NEW
10 MAXFILES = 1
20 OPEN "CAS : NOME" FOR INPUT AS # 1
30 LINE INPUT # , X$
40 CLOSE # 1
```

Nesse exemplo, a variável X$ conterá depois da leitura da fita : X$ = MARIA, JOSÉ.

Usando a instrução "EOF" poderemos verificar se foi atingido o fim do arquivo.

2.6. ARQUIVOS

Entende-se por "arquivo" o agrupamento de informações significativas. Um livro, um disco ou uma fita poderão ser considerados arquivos, podendo dessa forma agrupar ou armazenar símbolos e sons.

As pessoas usam esses arquivos para conservar certas informações numa forma ordenada, mas o computador só poderá ler essas informações se elas estivessem na forma de sinais elétricos. Por esse motivo, os arquivos deverão ser lidos e escritos por um aparelho elétrico ou eletrônico.

Esses aparelhos são basicamente: os gravadores, a fita cassete ou os acionadores de discos magnéticos.

Os arquivos, para um computador podem ser divididos em duas categorias:

Na primeira categoria estão os arquivos que contém programas, e para gerar esse tipo de arquivos o computador conta com os seguintes comandos:

SAVE e CSAVE

Para ler ou manipular esses arquivos, usa-se os comandos:

LOAD, CLOAD, MERGE

Na outra categoria de arquivos, estão os que contém dados, ou seja, informações do tipo seqüencial, como por exemplo, uma lista telefônica.

Para operar com esse tipo de arquivos usam-se os comandos:

OPEN, PRINT # , INPUT # , LINE INPUT # , CLOSE, MAXFILES

O uso desses comandos já foi explicado no parágrafo 2.5, referente a gravação de arquivos em cassete.

Quando o dispositivo de gravação do arquivo não for fita cassete, a instrução OPEN deverá conter a sigla correspondente.

A instrução OPEN serve para preparar o arquivo, seja para leitura ou para escrita, porém alguns arquivos podem ser utilizados somente para emissão de dados (output), isto é, só podem ser escritos e não lidos:

```
OPEN "GRP :" FOR OUTPUT AS # 1
OPEN "CRT :" FOR OUTPUT AS # 2
OPEN "LPT :" FOR OUTPUT AS # 3
```

O vídeo e a impressora são os dispositivos que só aceitam esses tipos de arquivos. Neles poderemos executar somente operações de emissão de dados (OUTPUT) e nunca de leitura (INPUT). Para esses arquivos não será necessário definir um nome, já que eles não podem ser chamados para leitura.

Especificando #1, #2 e #3, como no exemplo anterior, faremos com que a execução de um PRINT # 1 escreva dados no SCREEN 2 (devido a GRP) e um PRINT # 3 que escreva por sua vez os dados na impressora.

Se no seu programa houver necessidade de especificar um número de arquivo diferente de 1, deve-se indicar o número máximo utilizado com o comando MAXFILES.

Exemplo:

```
MAXFILES = 3 :
```

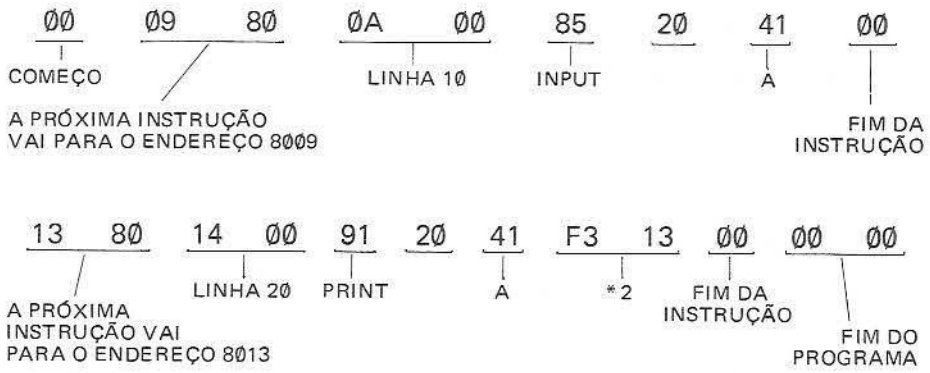
2.7. FORMATO ASCII E FORMATO BINÁRIO

O capítulo referente a CRIAÇÃO E EXECUÇÃO DE UM PROGRAMA, descreve em detalhes o formato no qual o programa em BASIC é carregado na memória.

Vejamos por exemplo o seguinte programa:

```
10 INPUT A
20 PRINT A * 2
```

Ele é memorizado entre os endereços &H 8000 e &H 8014 na seguinte forma:



Um programa escrito nesse formato aumenta a velocidade de execução e ocupa um espaço menor. Esse formato é chamado de "formato binário".

Embora o interpretador BASIC execute somente programas escritos no formato binário, será muito complicado para uma pessoa escrever e ler programas nesse formato. Um programa memorizado como sucessão de caracteres do tipo [1], [Ø], [I], [N], [P], [U], [T], é chamado de programa em formato ASCII.

Pode-se concluir então, que as pessoas usam basicamente o formato ASCII, e o computador executa o programa em formato binário. O comando LIST converte o programa existente na área texto para o formato ASCII e o apresenta no vídeo.

Segue uma explicação sobre qual é o melhor formato para memorizar os programas nos dispositivos externos, tais como fitas cassetes ou disquetes.

Para memorizar um programa em formato binário, deve-se simplesmente ler e escrever os dados da memória, exatamente como eles estão colocados. Pelo contrário, usando o formato ASCII analogamente ao que acontece com o comando LIST, o programa, antes de ser gravado na fita cassete ou em outro dispositivo, deve ser decodificado e reestruturado em strings de caracteres linha por linha. Cada caracter de cada linha deve ser convertido ao código ASCII (no fim deste manual temos a tabela de conversão). O carregamento de programas da fita para a memória implica no procedimento inverso: os caracteres lidos da fita formam uma linha de instrução como aquelas que introduzimos através do teclado antes de dar o RETURN.

Depois, cada linha é convertida em binário e transferida para a memória. A gravação e o carregamento de um programa em formato binário é muito mais veloz, já que não realiza aquela dupla conversão. Por outro lado, o lugar ocupado também é menor. Por esse motivo os programas são geralmente salvos em formato binário.

O formato ASCII é vantajoso quando se deseja combinar dois programas.

Suponhamos ter na área texto o seguinte programa:

```
1Ø INPUT A
2Ø PRINT A * 2
```

Para adicionar as duas linhas do programa seguinte,

```
 5 REM CALCULO
3Ø END
```

basta simplesmente introduzí-las pelo teclado. O editor BASIC é muito poderoso, por este motivo a linha 5 entrará em memória antes da linha 16. Em outras palavras, o conteudo da área texto é reescrito cada vez que é introduzida uma nova linha. Dessa forma, esta área estará sempre bem organizada.

Todavia, essas novas instruções, poderão ser introduzidas por um meio diferente do teclado, por exemplo, tirando-as de um programa arquivado em formato ASCII. Os códigos ASCII sobre a fita cassete ou sobre o disco, são lidos um de cada vez com o comando MERGE, e quando se pressiona a tecla RETURN , o string de caracteres é anexado ao programa na área texto. Se o programa arquivo mencionado estiver em formato binário, o MERGE não será possível.

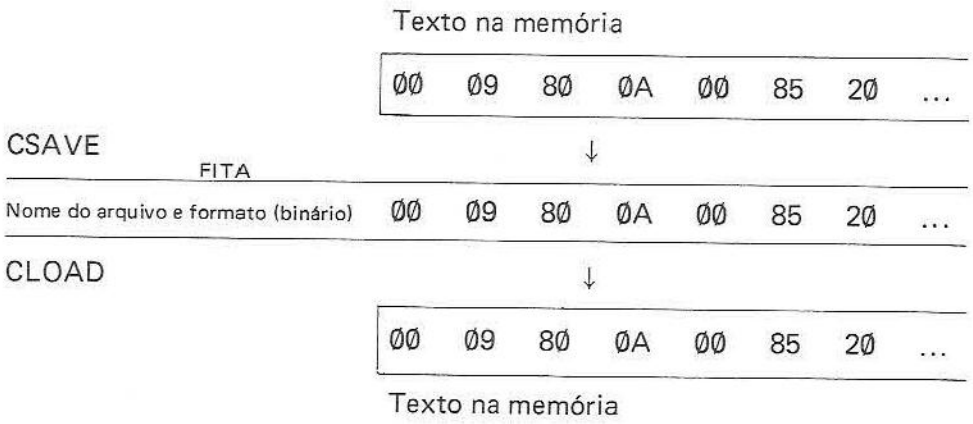
A função de MERGE é muito útil para anexar em vários programas um mesmo bloco de intruções, sem necessidade de introduzí-lo a cada vez pelo teclado.

Uma outra vantagem de salvar um programa em formato ASCII, é a de poder transferir esse programa de/e para outro computador que não utilize o mesmo BASIC. Em geral será suficiente modificar só algumas instruções.

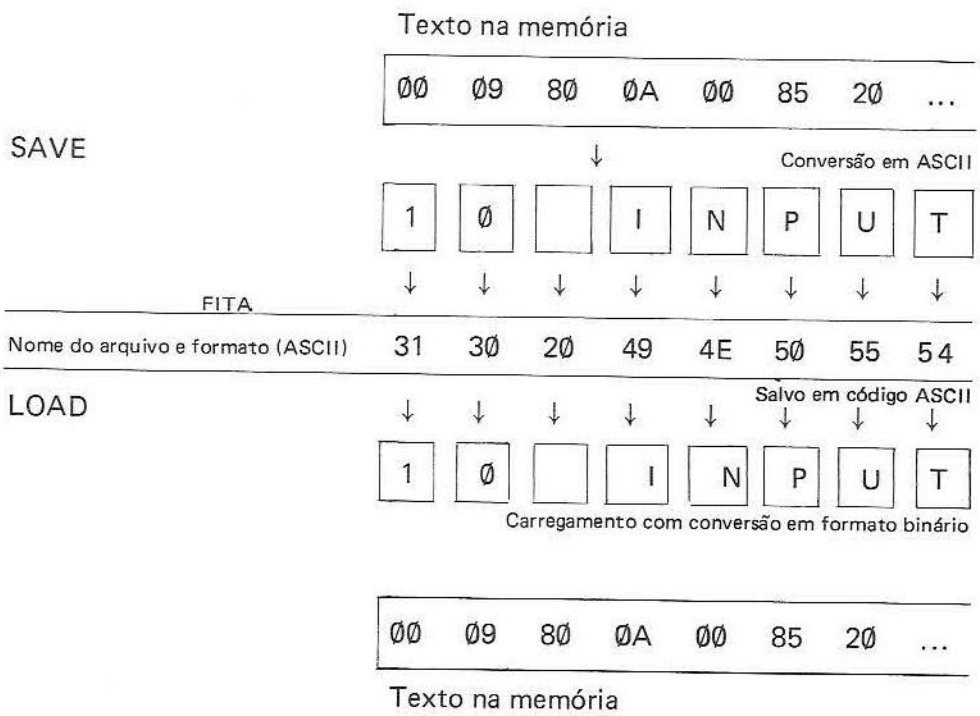
O nosso BASIC com o de outros computadores, são muito similares, o que possibilita o intercâmbio de programas. Os códigos intermediários poderão, em compensação, ser diferentes, motivo pelo qual a transferência de programas de um computador para outro, em formato binário, é quase sempre impossível.

O exemplo abaixo mostra como é processada a gravação e o carregamento de um programa sobre a fita, usando formato ASCII e formato binário.

- FORMATO BINÁRIO



- FORMATO ASCII



A sigla ASCII significa:
"AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE".

2.8. A MEMÓRIA

Conforme apresentado no parágrafo "CONCEITOS GERAIS SOBRE O COMPUTADOR"; o coração de um computador é a sua CPU, que controla todos os cálculos e todas as operações de entrada e saída (I/O).

Todavia, a capacidade de memória da CPU é muito restrita. O Z-80 pode memorizar temporariamente 26 grupos de bytes. Com isso, a CPU não pode manipular valores numéricos muito elevados ou de muita cifras. Obviamente, não servirá para conter um programa.

As posições de memória da CPU são chamadas de REGISTROS, identificados pelas letras A, B, C ... etc. Vejamos como funcionam e para que servem.

Pode-se pensar na CPU como tendo internamente 26 divisões, e que toda a memória da máquina esteja formada por unidades consecutivas, numeradas a partir de Ø. Cada uma dessas unidades é chamada de endereço de memória, o conteúdo de cada endereço é um byte. Podemos por exemplo ter, no endereço Ø o número 243 e no endereço 1 o 195, etc. (Fig. 36)

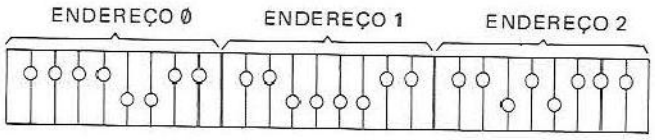


Fig. 36: Divisões da memória

Para a CPU poder ler o conteúdo da memória, deverá estar ligada a ela por meio de fios elétricos (address bus), que servem também para fornecer a CPU o endereço de memória no qual deve operar.

Como o Z-80 possui 16 terminais de endereço, ele pode endereçar 2^{16} locações de memória, ou seja, do endereço Ø até o 65535 em notação decimal ou FFFF em hexadecimal.

2.8.1. ROM E RAM

Assim, já sabemos o que é memória, porém, o que o conteúdo da memória representa e como foi escrito?

Como já foi explicado, ao ser ligado o computador, a CPU vê o conteúdo da memória na ordem, a partir do endereço Ø. Usando a função PEEK, podemos ver o conteúdo dos endereços Ø a 3:

Endereço:	0	1	2	3
Conteúdo em hexadecimal:	F3	C3	D7	Ø2

A CPU interpreta esses dados da seguinte forma:

F3 ... Inibe as interrupções da máquina (ver parágrafo 2.10). Em seguida vai para o endereço Ø1.

C3 ... Vê o conteúdo dos dois bytes seguintes (D7 e Ø2), inverte-os, e compõe o endereço da próxima instrução.

Em seguida o conteúdo é passado à locação de memória Ø2D7.

Assim que o computador é ligado, esse programa é executado, após o que, aparece na tela a mensagem OK, que indica estar pronto o computador para aceitar comandos e instruções BASIC.

Os comandos e instruções explicados acima, estão escritos na linguagem de máquina.

O que acontece se as instruções em linguagem de máquina escritas nos endereços inferiores da memória (como por exemplo F3, C3) fossem modificadas?

Pode por exemplo, acontecer que o computador não reconheça mais o BASIC, ou que não aceite o teclado. O computador torna-se então apenas uma caixa de componentes eletrônicos.

Para evitar esse tipo de situações, essa importantíssima parte da memória que não deve ser alterada de maneira nenhuma, é constituída por uma memória ROM (read-only-memory). O conteúdo da ROM é definido no momento de fabricação da memória, e nunca mais poderá ser alterada.

Como exemplo façamos a seguinte experiência:

```
? PEEK (Ø)
243
OK
POKE Ø, Ø
OK
? PEEK (Ø)
243
```

POKE é um comando BASIC para introduzir um conteúdo determinado na memória, porém, caso tentar colocar alguma coisa no endereço Ø (ROM) não conseguirá.

No nosso computador, os endereços compreendidos entre Ø e &H7FFF estão em memória ROM, e é onde está localizado o interpretador BASIC.

Para poder programar em BASIC será necessário ter uma área de memória para guardar o texto e as variações utilizadas no programa. Tanto o programa quanto as variáveis, são modificadas pelo usuário durante a execução do mesmo.

Para memorizar essa parte de maneira que possa ser modificada, usamos uma memória RAM (random-access-memory).

Os endereços de &HF38Ø até &HFFFF são utilizados como área de trabalho da CPU para executar corretamente o BASIC. Se através de uma instrução POKE os valores dessa área forem modificados de maneira inadequada, algumas funções vitais do computador poderão ser inibidas, ou condições anormais serem verificadas.

Desligando o computador, o conteúdo da RAM é apagado. Por isso, os programas devem ser salvos (gravados) em fita ou em disco antes de se desligar o computador.

Também a área de trabalho (work-area) da CPU, é limpa ao ser desligado o computador, mas ela é recriada automaticamente pelo programa existente na ROM.

Analisando finalmente o conteúdo da RAM, ao ligar o computador, vejamos o que tem no endereço &H8ØØØ:

```
? PEEK (&H8ØØØ)
Ø
OK
```

Escrevamos agora alguma coisa através da instrução POKE:

```
POKE &H8ØØØ , 15
OK
? PEEK (&H8ØØØ)
15
OK
```

Vemos que a modificação foi aceita.

Desligando e ligando o computador novamente veremos:

```
? PEEK (&H8ØØØ)
Ø
OK
```

Vemos que o valor escrito precedentemente foi perdido e substituído pelo valor Ø.

2.9. LINGUAGEM DE MÁQUINA

É pré-requisito para entender este parágrafo, ter lido os parágrafos referentes a "CONCEITOS GERAIS", "MEMÓRIA" e "BITS E BYTES".

Para maiores detalhes sobre linguagem de máquina, aconselhamos consultar um manual específico do Z-80.

Por ora, pretende-se somente explicar como chamar, em um programa BASIC, uma sub-rotina em linguagem de máquina, usando a instrução USR.

A função USR é utilizada da seguinte forma:

1. É alocada uma parte da memória na RAM através da instrução CLEAR no início do programa. (Fig. 37)

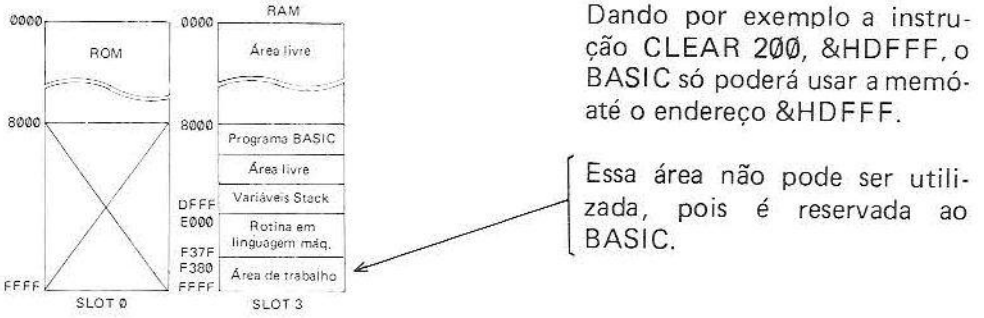


Fig. 37: Utilização da memória

2. O início da rotina é definido com DEFUSR = &HEØØØ (no nosso exemplo).

3. Escreve-se o programa em linguagem de máquina, na área previamente definida, usando a instrução POKE.

4. A rotina USR é chamada onde for necessário, usando o formato A = USR (B).

   O valor de B (argumento) é transferido para uma área de trabalho chamada FAC (floating accumulator counter), e os valores dos registros A e HL mudam dependendo do tipo do argumento conforme Fig. 38.

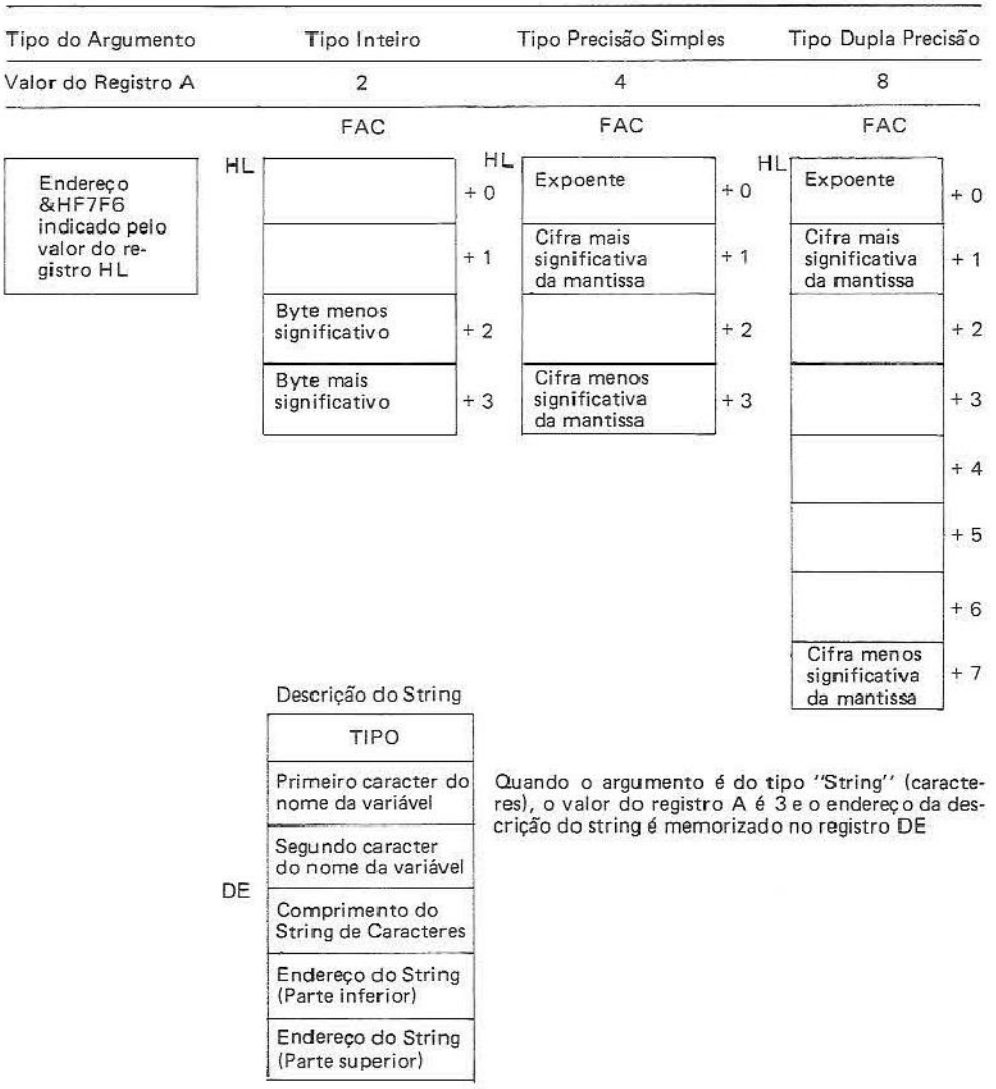


Fig. 38: Formato A = USR (B)

Em seguida, o controle passa ao endereço definido na instrução DEFUSR, e volta ao BASIC através do comando RET (&HC9). Nesse ponto da operação, atribuindo à FAC o resultado da função USR, o valor retorna à variável.

A = USR (B) → A = f (B)

Ilustramos a seguir o procedimento acima descrito, com um exemplo não muito complicado. Será somado 1 a variável cujo valor pode estar compreendido entre 0 e 254, obtendo-se a seguir, como resultado, o novo valor da variável.

```
10 CLEAR200,&HDFFF
20 DEFINTA-Z
30 AD=&HE000:DEFUSR=AD
40 FORI=0TO3
50 READA$
60 POKEAD+I,VAL("&H"+A$)
70 NEXT
80 DATA23,23,34,C9
90 REM***INICIO DO PROGRAMA***
100 INPUT"B=(0-254)";B
110 A=USR(B)
120 PRINT"B+1=";A
130 END
```

As instruções dadas em linguagem de máquina são:

Endereço	Código	Mnemônico
E000	23	INC HL
E001	23	INC HL
E002	34	INC (HL)
E003	C9	RET

Esta é a operação aritmética de tipo inteiro mais simples. Os valores em simples e dupla precisão são também armazenados na FAC; o endereço do expoente é indicado pelo valor do registro HL e o endereço dos primeiros dois dígitos da mantissa está indicado pelo valor de HL + 1. A parte da mantissa é representada em codificação BCD (Binary Code Digits).

Deve-se prestar especial atenção no caso em que o argumento da função USR é uma variável de tipo alfanumérico.

Enquanto não é executada nenhuma operação, o conteúdo de uma variável string encontra-se na área de texto, com o "descritor do string" como ponteiro. (Fig. 39)

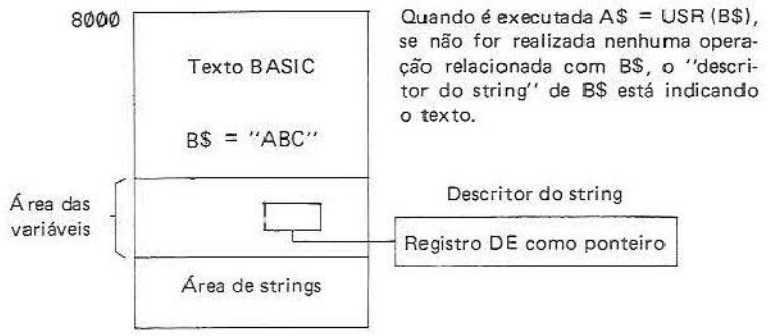


Fig. 39

Como resultado de uma operação sobre um string de caracteres no texto de um programa, se o comprimento do string for modificado, o texto BASIC será destruído e o programa danificado. Para evitar isso, basta escrever A$ = USR (B$ + " ").

Fazendo assim, B$ é copiado na área de strings, sempre com o "descritor do string" apontando para ele.

Mesmo quando o string de caracteres se encontra na área de strings, corremos o risco de danificar o conteúdo se não operarmos corretamente. Para evitar isto, convém alocar uma área suficientemente larga para o string e especificar um valor grande como comprimento da variável caracter.

Lembremos o explicado acima, cada vez que for modificado o comprimento de um string. Deve-se também substituir o valor do comprimento do string dentro do "descritor".

Ignorando o FAC, é possível usar a função USR simplesmente como um comando, para chamar uma rotina em linguagem de máquina, mas, como USR é chamada como uma função, convém usar uma variável e um argumento fictício, por exemplo: A = USR (Ø). Para utilizar mais de um argumento e de um resultado, o usuário deverá alocar uma área de trabalho na área de linguagem de máquina e forçar os argumentos com POKE. Após ter chamado a função USR, o resultado é obtido com a instrução PEEK.

Para concluir, daremos como exemplo prático, uma rotina de ordenamento (SORT), de alta velocidade, que usa matrizes e a função VARPTR.

O algoritmo usa o método simples de inserção. N valores são atribuídos aleatoriamente à matriz inteira D e depois reordenados em ordem crescente mediante a rotina USR. O mesmo trabalho em BASIC requer um tempo de execução cem vezes maior.

```
10 CLEAR200,&HF32F
20 DEFINTA-Z
30 AD=&HF330:DEFUSR=AD
40 FORI=ØTO71
50 READA$:V=VAL("&H"+A$)
60 POKEAD+I,V
70 NEXT
80 DATA23,23,5E,23,56,D5,DD,E1
90 DATADD,5E,FE,DD,56,FF,1B,DD
100 DATA4E,00,DD,46,01,DD,6E,02
110 DATADD,66,03,DD,23,DD,23,DD
120 DATAE5,AF,E5,ED,42,E1,30,18
130 DATADD,70,01,DD,71,00,DD,74
140 DATAFF,DD,75,FE,DD,2B,DD,2B
150 DATADD,46,FF,DD,4E,FE,18,E1
160 DATADD,E1,1B,7A,B3,20,C8,C9
170 INPUT"Nº DE DADOS";N
180 DIMD(N):D(Ø)=Ø:I=RND(-TIME)
190 FORI=1TON
200 D(I)=RND(1)*N
210 NEXT
220 PRINT"COMEÇOU!"
230 TIME=Ø
240 V=USR(VARPTR(D(Ø)))
250 V=TIME
260 FORI=1TON
270 PRINTUSING"#####";D(I)
280 NEXT:PRINT
290 PRINTV;"/60 SEG."
300 END
```

A listagem da rotina em linguagem de máquina é:

```
                                ORG     OF330H

F330    23
F331    23                      INC     HL
F332    5E                      INC     HL
F333    23                      LD      E, (HL)
F334    56                      INC     HL
F335    D5                      LD      D, (HL)
F336    DDE1                    PUSH    DE
F338    DD5EFE                  POP     IX
F33B    DD56FF                  LD      E, (IX – 2)
F33E    1B                      LD      D, (IX – 1)
F33F    DD4E00          STRT:   DEC     DE
F342    DD4601                  LD      C, (IX + 0)
F345    DD6E02                  LD      B, (IX + 1)
F348    DD6603                  LD      L, (IX + 2)
F34B    DD23                    LD      H, (IX + 3)
F34D    DD23                    INC     IX
F34F    DDE5                    INC     IX
F351    AF              COMP:   PUSH    IX
F352    E5                      XOR     A
F353    ED42                    PUSH    HL
F355    E1                      SBC     HL, BC
F356    3018                    POP     HL
F358    DD7001                  JR      NC, OK
F35B    DD7100                  LD      (IX + 1), B
F35E    DD74FF                  LD      (IX + 0), C
F361    DD75FE                  LD      (IX – 1), H
F364    DD2B                    LD      (IX – 2), L
F366    DD2B                    DEC     IX
F368    DD46FF                  DEC     IX
F36B    DD4EFE                  LD      B, (IX – 1)
F36E    18E1                    LD      C, (IX – 2)
F370    DDE1            OK:     JR      COMP
F372    1B                      POP     IX
F373    7A                      DEC     DE
F374    B3                      LD      A, D
F375    20C8                    OR      E
F377    C9                      JR      NZ, STRT
                                RET
```

O seguinte programa serve para criar programas em linguagem assembler na área de RAM compreendido entre os endereços &HØØØØ e &H7FFF, no slot 3.

```
10 DEFUSR=&HE000
20 AB=&HE000
30 FOR I=AB TO AB+20
40 READ A$:A=VAL("&h"+A$)
50 POKE I,A
60 NEXT I
70 INPUT "endereço inicial";C$
80 A$=LEFT$(C$,2)
90 B$=RIGHT$(C$,2)
100 AD =&HE100
110 A=VAL("&h"+A$)
120 B=VAL("&h"+B$)
130 POKE AD,B
140 POKE AD+1,A
150 INPUT "dado hexadecimal";D$
160 D%=VAL("&h"+D$)
170 R%=USR(D%)
180 INPUT "continua";I$
190 IF I$="S"GOTO 150
200 END
210 DATA f3,3e,ff,d3,a8,2a,00,e1,3a,f8,f7,
    77,23,22,00,e1,3e,f0,d3,a8,c9
```

A seguinte sub-rotina executa programas em linguagem assembler localizados entre &HØØØØ e &H7FFF.

Em E% deverá ser introduzido o endereço inicial do programa em linguagem assembler.

```
1000 DEFUSR=&HE100
1010 AB=&HE100
1020 FORI=ABTOAB+20
1030 READA$:A=VAL("&H"+A$)
1040 POKEI,A
1050 NEXTI
1060 R%=USR(E%)
1070 RETURN
1080 DATAF3,3E,FF,D3,08,21,10,E1,E5,2A,F8.F7,E9
1090 DATA00,00,00,3E,F0,D3,08,C9
```

2.10. INTERRUPÇÕES

As interrupções, usadas para suspender o fluxo dos programas em execução, são provocadas ao se verificar certas condições externas específicas. Caso a interrupção se verifique, o programa é desviado para executar uma "rotina de elaboração" da interrupção.

As interrupções baseiam-se num conceito similar ao das sub-rotinas, com a diferença que, estas são executadas através de uma instrução GOSUB. Em outras palavras, uma sub-rotina é uma possibilidade determinada pelas condições internas do programa, enquanto que uma interrupção é processada devido a algum fator externo.

Assim que a rotina de elaboração da interrupção é concluída, a execução do programa principal é retomado, exatamente da mesma forma que acontece quando uma sub-rotina chega ao fim da sua execução.

Ao se verificar uma interrupção no programa, o BASIC possui numerosos comandos de transferência de controle do programa principal, a uma rotina de elaboração da interrupção.

Para habilitar uma interrupção, deve-se primeiro declarar a sua utilização através de um comando, especificando também o número de linha inicial da sub-rotina de elaboração da interrupção.

FATOR DETERMINANTE DE INTERRUPÇÃO	COMANDO DE DECLARAÇÃO DA INTERRUPÇÃO
É pressionado uma tecla de função	ON KEY GO SUB número de linha
É pressionada a barra espaçadora ou o botão disparador do joystick	ON STRIG GO SUB número de linha
[CTRL] + [STOP] são pressionados simultâneamente	ON STOP GO SUB número de linha
Superposição de sprites	ON SPRITE GO SUB número de linha
Transcurso de um determinado período de tempo	ON INTERVAL = intervalo GO SUB número de linha

Por exemplo, a instrução

```
ON KEY GO SUB 1000
```

declara que ao ser pressionada uma tecla de função, a execução do programa será transferida à rotina que começa na linha 1000.

2.10.1. UTILIZAÇÃO DAS INTERRUPÇÕES

Para executar uma interrupção não basta simplesmente fazer a declaração ON – GO SUB. Essas instruções devem ser precedidas por um comando que habilite a detecção das interrupções verificadas.

Por exemplo, no caso de uma interrupção que se verifica pressionando a tecla [F1] , o comando deve ser:

```
KEY (1) ON
```

O BASIC possui 5 comandos que habilitam as interrupções, e que funcionam da seguinte forma:

COMANDO	INTERRUPÇÃO VÁLIDA
KEY (número de tecla de função) ON	Interrupção devida a uma tecla de função
STRIG (número de joystick) ON	Interrupção executada com a barra espaçadora ou joystick
STOP ON	Interrupção executada com as teclas [CTRL] + [STOP]
SPRITE ON	Interrupção executada por uma superposição de SPRITES
INTERVAL ON	Interrupção provocada por um determinado intervalo de tempo

Exemplo de programa usando interrupção:

```
1Ø  ON KEY GO SUB 1ØØ
2Ø  KEY (1) ON
3Ø  SCREEN 2                          Programa principal
4Ø  LINE (5Ø, 5Ø) – (2ØØ, 15Ø), , B
5Ø  GO TO 4Ø
1ØØ ' SUB-ROTINA
11Ø BEEP : CLS
12Ø FOR I = 1Ø TO 9Ø STEP 1Ø
13Ø CIRCLE (12Ø, 1ØØ), I              Rotina de elaboração
14Ø NEXT I                            da interrupção
15Ø CLS
16Ø RETURN 4Ø
```

Nesse programa, as linhas 1Ø e 2Ø fazem com que pressionando a tecla [F1] , a execução do programa seja transferido à sub-rotina que começa na linha 1ØØ. As linhas 4Ø e 5Ø do programa principal são as responsáveis pela visualização contínua de um retângulo na tela. Todavia, se a tecla de função [F1] for pressionada, verificar-se-á uma interrupção na visualização do retângulo. Simultâneamente se ouvirá um som de BEEP e começará o traçado de 9 círculos concêntricos. Assim que o 9º círculo for traçado a tecla será liberada e o programa voltará à linha 4Ø, mostrando novamente o retângulo original.

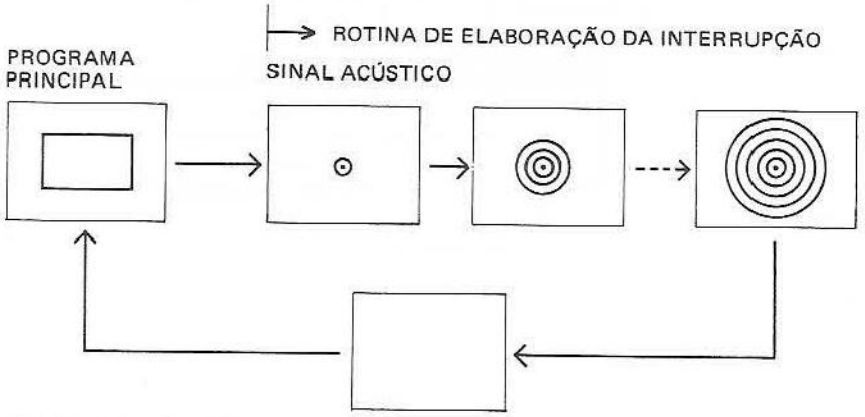


Fig. 40: Rotina de elaboração da interrupção

2.10.2. INVALIDAÇÃO DE UMA INTERRUPÇÃO

Anexando ao programa acima a seguinte linha:

```
105 KEY (1) OFF
```

veremos que, ao ser pressionada a tecla [F1] pela primeira vez, o programa executa a sub-rotina de elaboração da interrupção da mesma forma que antes.

Uma vez que o programa volta à linha 40, a pressão da tecla de função [F1] não terá mais efeito. Isto se deve à linha 105 que invalida a interrupção provocada pela tecla [F1] .

2.10.3. SUSPENSÃO DA INTERRUPÇÃO DENTRO DA ROTINA DE ELABORAÇÃO

Quando uma interrupção transfere a execução do programa para uma sub-rotina de elaboração da interrupção, verifica-se um estado de "suspensão" da interrupção. Esse estado impede a detecção de novas interrupções, até que através de um RETURN a execução volta ao programa principal. Nesse momento, processa-se automaticamente uma instrução — ON que volta a habilitar a detecção da interrupção.

Em outras palavras, quando uma interrupção é intentada durante o estado de "suspensão", a execução não retorna à linha inicial da sub-rotina. Todavia, a interrupção verificada durante esse período será memorizada, e logo após a sub-rotina ser completada, ela será processada novamente.

Voltando ao programa exemplo, tínhamos visto que pressionando a tecla [F1] verificava-se uma interrupção durante a qual eram desenhados 9 círculos concêntricos. Se a tecla [F1] for pressionada enquanto os círculos estão sendo traçados, não notaremos interrupção nenhuma até o último círculo ser traçado. Logo após a execução voltar ao programa principal, a interrupção que estava "suspensa" será processada e novamente serão visualizados os círculos na tela. (Fig. 41)

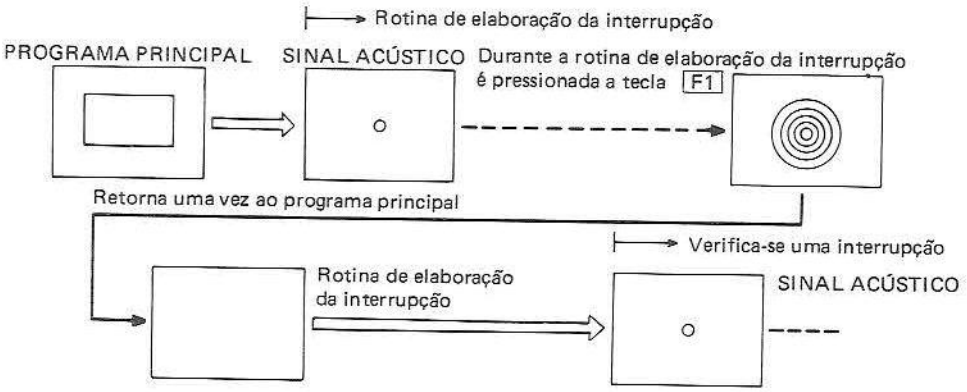


Fig. 41: Suspensão da interrupção dentro da rotina de elaboração

2.10.4. HABILITAÇÃO DE UMA INTERRUPÇÃO DURANTE UMA ROTINA DE ELABORAÇÃO DA INTERRUPÇÃO

Introduzindo um comando KEY (1) ON, pode-se habilitar uma interrupção durante a execução de uma rotina de elaboração. Desse modo, a rotina de elaboração poderá ser executada novamente desde o início, caso se verifique uma interrupção durante a execução da mesma.

PROGRAMA EXEMPLO:

```
10  ON KEY  GO SUB  100
20  KEY (1)  ON
30  SCREEN 2
40  LINE (50, 50) – (200, 150), , B
50  GO TO 40
100 'SUB-ROTINA
105 KEY (1) ON
110 BEEP ( CLS
120 FOR I = 10 TO 90 STEP 10
130 CIRCLE (120, 100) , I
140 NEXT I
150 CLS
160 RETURN 40
```

Esse programa é igual ao precedente, só que foi introduzido a linha 105, contendo a instrução KEY (1) ON.

Nesse modo, se a tecla [F1] for pressionada durante a visualização dos círculos, verifica-se imediatamente uma interrupção, na qual a sub-rotina é executada novamente desde o início. (Fig. 42)

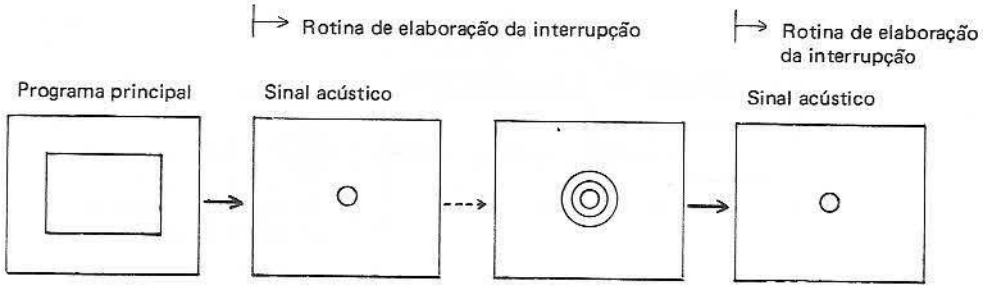


Fig. 42: Habilitação de uma interrupção

2.10.5. SUSPENSÃO DA INTERRUPÇÃO EM UM PROGRAMA

Para entrar novamente no estado de interrupção, após ter habilitado uma interrupção mediante – ON durante a rotina de elaboração, usa-se – STOP.

Exemplo:

```
10 ON KEY GOSUB 100
20 KEY(1) ON
30 SCREEN 2
40 LINE (50,50)-(200,150),,B
50 GOTO 40
100 'SUBROUTINE
105 KEY(1) ON
110 BEEP:CLS
120 FOR I=10 TO 90 STEP 10
130 CIRCLE (120,100),I
135 IF I=50 THEN KEY(1) STOP
140 NEXT I
150 CLS
160 RETURN 40
```

Este programa é igual ao anterior, só que foi introduzida a linha 135 contendo o comando KEY (1) STOP.

Agora, uma vez que a rotina de elaboração começa a ser executada, toda interrupção que acontecer antes do 5º círculo ser traçado (I = 50) fará com que a rotina seja recomeçada. Depois de I = 50, a interrupção não se verificará imediatamente. (Fig. 43)

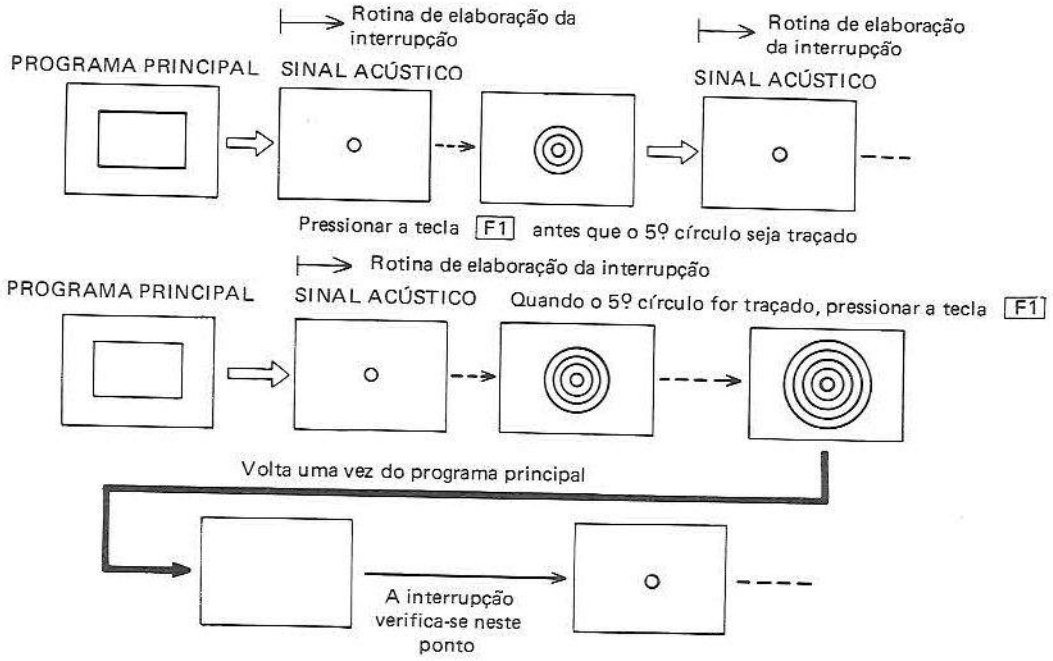


Fig. 43: Suspensão da interrupção em um programa

2.10.6. EXEMPLO DE INTERRUPÇÃO PROVOCADA POR SUPERPOSIÇÃO DE SPRITES

Com uma instrução ON SPRITE GO SUB e SPRITE ON pode-se gerar uma interrupção que se verifica quando duas ou mais formas de sprite sobrepõem-se em um ponto.

Vejamos o seguinte exemplo:

```
10 SCREEN2
20 B$=""
30 FORI=1TO8:READA$
40 B$=B$+CHR$(VAL("&h"+A$))
50 NEXT
60 SPRITE$(0)=B$
70 ONSPRITEGOSUB140
80 SPRITEON
90 FORX=0TO255
100 PUTSPRITE0,(X,100),15,0
110 PUTSPRITE1,(255-X,100),10,0
120 NEXTX
130 END
140 SPRITEOFF
150 FORI=0TO500:NEXT
160 BEEP
170 SPRITEON
180 RETURN
190 DATA3c,7e,81,81
200 DATAff,7e,24,42
```

2.11. UTILIZAÇÃO DO VÍDEO – DESENHO DE FIGURAS

2.11.1. COMO UTILIZAR O VÍDEO

Para operar com o vídeo, existem dois modos, o modo gráfico e o modo texto.

O modo texto é utilizado para representar caracteres, números, expressões; o modo gráfico serve por sua vez para imprimir na tela pontos, linhas, desenhos e símbolos gráficos.

O modo texto inclue 2 sub-modos: um que permite apresentar uma tela de 24 linhas x 40 colunas e um outro de 24 linhas x 32 colunas.

Também o modo gráfico inclue 2 sub-modos: um de alta resolução e um de baixa resolução.

MODO TEXTO

24 linhas x 40 colunas
24 linhas x 32 colunas

MODO GRÁFICO

Alta resolução
Baixa resolução

Esses modos de operação são definidos mediante a instrução SCREEN. O formato dessa instrução é:

SCREEN ⟨ modo ⟩ onde ⟨ modo ⟩ pode valer:

Modo Ø : modo texto 24 x 40
Modo 1 : modo texto 24 x 32
Modo 2 : modo gráfico de alta resolução
Modo 3 : modo gráfico de baixa resolução

Quando se trabalha no modo texto, não utilizamos nenhuma das instruções gráficas (exceto PUT SPRITE).

O modo gráfico por sua vez, só pode ser chamado por uma instrução de programa (e não no modo direto, através do teclado), e quando o programa termina, o vídeo retorna automaticamente ao modo texto.

A execução de uma instrução INPUT dentro de um programa também fará o vídeo voltar ao modo texto.

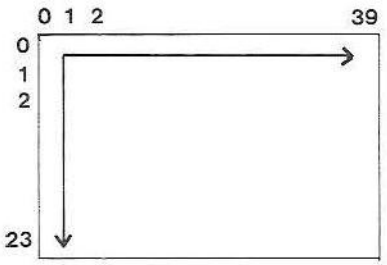
NOTA: O número de caracteres por linha inicialmente utilizado é:

No SCREEN Ø → 37
No SCREEN 1 → 29

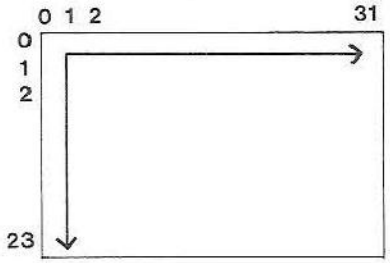
- **MODO TEXTO**

Como foi explicado anteriormente, o modo texto inclue dois sub-modos: o modo 24 x 40 e o 24 x 32.

As coordenadas de cada um deles são definidas da seguinte forma:



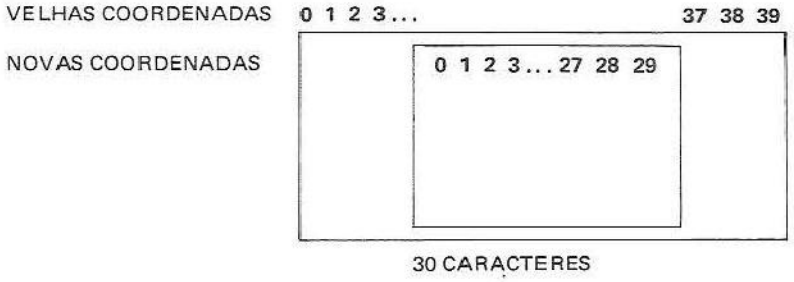
Modo 40 x 24



Modo 32 x 24

A largura da tela pode ser reduzida com o comando WIDTH.

Por exemplo, com o comando WIDTH 3Ø, e trabalhando no modo 24 x 4Ø, o sistema de coordenadas é modificado da seguinte forma:

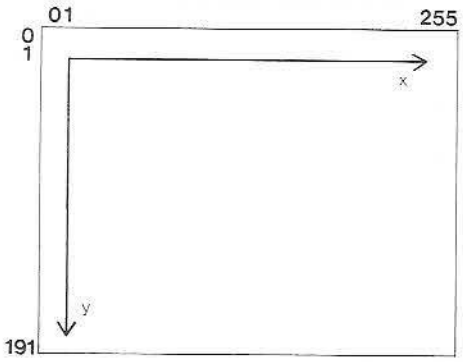


Trabalhando no modo texto podem-se utilizar as seguintes instruções para impressões no vídeo.

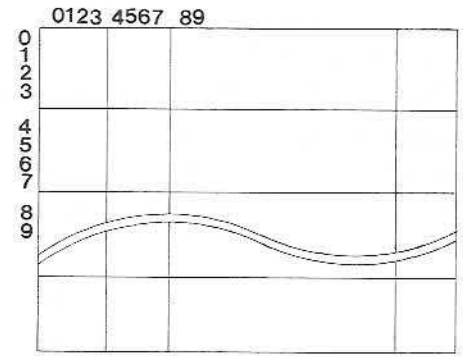
PRINT	Impressão só de caracteres (letras, números, ...)
LOCATE	Para o posicionamento do cursor
CLS	Para apagar toda tela
COLOR	Para especificar as cores dos caracteres e do fundo (vide parágrafo relativo ao modo gráfico)

- MODO GRÁFICO

A tela gráfica tem uma resolução de 256 pontos na horizontal por 192 na vertical, e as coordenadas são definidas como segue:



Trabalhando em alta resolução, cada ponto no vídeo corresponde a um ponto das coordenadas. Em baixa resolução, por sua vez, um grupo de 4 x 4 pontos no vídeo é associado a um ponto das coordenadas, por isso nesse caso, a resolução fica sendo de 64 x 48 pontos.



2.11.2. DESENHO DE FIGURAS

Trabalhando no modo gráfico (SCREEN 2 ou 3) existem numerosos comandos para desenhar figuras na tela de vídeo.

Desenhar figuras quer dizer traçar pontos, linhas e círculos com uma cor definida.

Vejamos os detalhes da operação:

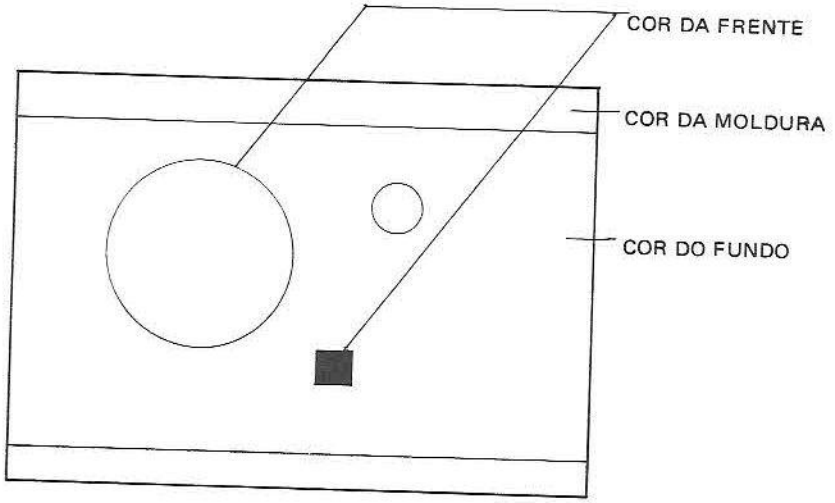
- AS CORES

Para desenhar uma figura, dispomos de 16 cores.

A cada cor é associado um número, seguindo a seguinte tabela:

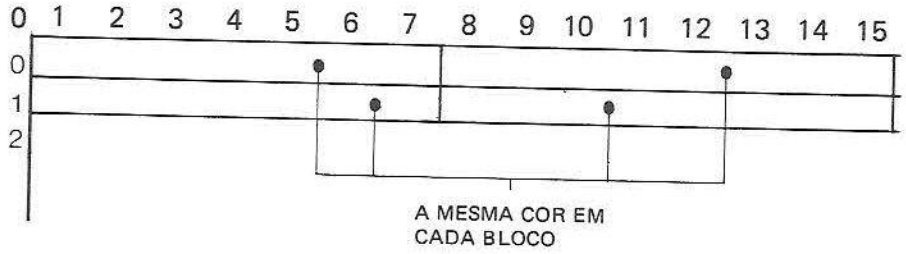
Ø : transparente	8 : vermelho médio
1 : preto	9 : vermelho claro
2 : verde médio	10 : amarelo escuro
3 : verde claro	11 : amarelo claro
4 : azul escuro	12 : verde escuro
5 : azul claro	13 : magenta
6 : vermelho escuro	14 : cinza
7 : ciano	15 : branco

A instrução COLOR define as cores que serão utilizadas no vídeo; mediante os códigos acima especificados.



No modo de alta resolução (SCREEN 2) em cada bloco (composto por 8 pontos horizontais do vídeo), só pode ser especificada uma cor.

Por exemplo, desenhando um ponto azul na coordenada (3, Ø) e um ponto vermelho no ponto (6, Ø), o ponto azul também ficará vermelho. Nesse modo, a última cor especificada será a cor do bloco inteiro.



- **COMO DESENHAR UM PONTO**

Para desenhar um ponto, usa-se a instrução PSET. Por exemplo, para desenhar um ponto nas coordenadas (5, 1Ø), usa-se a instrução PSET (5, 1Ø).

Se quisermos pintar o ponto de vermelho, daremos PSET (5, 1Ø), 8.

O seguinte programa desenha pontos aleatoriamente colocados na tela, pintando-os também aleatoriamente.

```
1Ø  SCREEN 2
2Ø  X = RND (1) * 255 : Y = RND (1) * 191 :
    C = INT (RND (1) * 16)
3Ø  PSET (X, Y) , C
4Ø  GO TO 2Ø
```

Mudando a linha 1Ø para SCREEN 3, o programa resultante desenhará os pontos em baixa resolução. Deve-se lembrar que todas as coordenadas que cabem em um bloco de 4 x 4 geram um mesmo ponto no vídeo. Por exemplo, as duas instruções seguintes produzem o mesmo resultado, que é o de desenhar um ponto no vídeo de 4 x 4 no ângulo superior esquerdo.

```
PSET (Ø, 3) , 15
PSET (3, 2) , 15
```

Para apagar um ponto previamente desenhado, pode-se usar a instrução PRESET.

Por exemplo, o ponto desenhado nas coordenadas (5, 1Ø) é apagado com a instrução:

```
PRESET (5, 1Ø)
```

Quando se omite a indicação da cor, a instrução PSET assume que foi especificada a cor do primeiro plano. A instrução PRESET por sua vez, assume que foi especificada a cor do fundo.

Em outras palavras, se especificamos COLOR 15, 4, 7, uma posterior instrução PSET (X, Y) sem especificação de cor, assumirá o valor 15 enquanto que uma PRESET (X, Y) sem especificação de cor, assumirá o valor 4.

• COMO DESENHAR UM QUADRADO

Para desenhar uma linha, usa-se a instrução LINE. Por exemplo, para desenhar um segmento entre as coordenadas (1Ø, 1Ø) e as (2Ø, 2Ø), dar-se-á a seguinte instrução:

```
LINE (1Ø, 1Ø) – (2Ø, 2Ø)
```

Para colorir a linha de vermelho, deve-se especificar a cor na instrução:

```
LINE (1Ø, 1Ø) – (2Ø, 2Ø) , 8
```

A instrução LINE serve também para traçar um quadrado, devendo para isso usar a letra B no fim da instrução.

As coordenadas indicam os pontos extremos de uma das duas diagonais.

Exemplo:

```
LINE (1Ø, 1Ø) – (2Ø, 2Ø) , 8 , B
```

Para colorir a parte interna do quadrado, usa-se BF.

```
LINE (1Ø, 1Ø) – (2Ø, 2Ø) , 8 , BF
```

O seguinte programa desenha linhas em forma de grelha, com as cores das linhas variando entre uma e outra.

```
10 COLOR15,0,7:SCREEN2
20 FORX=0TO255STEP10
30 C=C+1
40 IFC=>15THENC=1
50 LINE(X,1)-(X,191),C
60 NEXTX
70 GOTO70
```

NOTA: Para voltar as cores iniciais, basta pressionar as teclas SHIFT e F1 simultaneamente.

- COMO DESENHAR UM CÍRCULO

Para traçar círculos, usa-se a instrução CIRCLE, especificando as coordenadas do centro e o raio. Por exemplo, querendo traçar um círculo com o seu centro localizado nas coordenadas (6Ø, 5Ø) e de raio igual a 1Ø, usa-se a instrução:

```
CIRCLE (6Ø,5Ø) ,1Ø
```

O círculo poderá também ser colorido especificando como último parâmetro o código da cor.

Com a seguinte instrução desenha-se um círculo igual ao anterior mas pintado de vermelho.

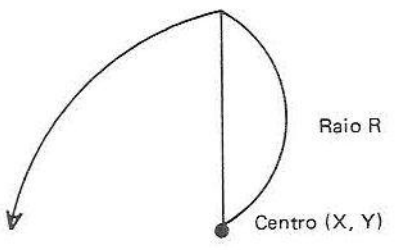
```
CIRCLE (6Ø,5Ø)
```

O programa seguinte desenha círculos de dimensões e cores variadas em forma aleatoria:

```
10 SCREEN2
20 X=RND(1)*255:Y=RND(1)*151
25 R=RND(1)*50:C=INT((RND(1)*15)+1)
30 CIRCLE(X,Y),R,C
40 GOTO20
```

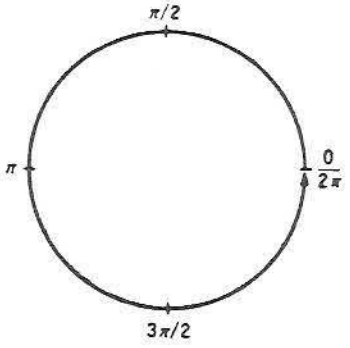
Com a instrução CIRCLE também é possível desenhar um arco, ou seja, uma porção de círculos. Nesse caso, deve-se também especificar o ângulo de início e o ângulo final. Esses ângulos vem expressos em radianos (e não em graus!).

Para desenhar um arco como o da figura:



usam-se as seguintes instruções:

```
10 SCREEN2
20 X=128:Y=96:R=80:PI=3.14
30 CIRCLE(X,Y),R,,PI/2,PI
40 GOTO40
```



Lembramos que a indicação dos ângulos é feita em radianos, isto é: 360° = 2π (= 6.2831853071796).

Isto é, se especificarmos como ângulo de saída Ø e como ângulo final 2π , obteremos uma circunferência.

Quando os valores dos ângulos são omitidos na instrução CIRCLE, o computador assumirá respectivamente Ø e 2π .

Existe um outro parâmetro para a instrução CIRCLE. O uso desse parâmetro permitirá o traçado de elipses, das quais o círculo é um caso particular. O valor assumido nesses casos é 1; embora o valor 1 não produzirá um círculo perfeitamente redondo. Isto se deve as características do monitor de TV, e à relação de densidade de pontos entre a horizontal e a vertical.

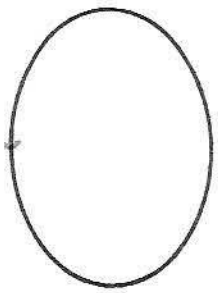
O círculo resultante é assim um pouco mais largo do que alto.

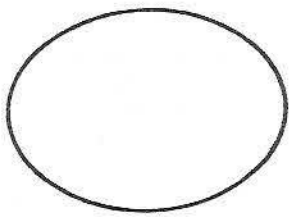
As instruções:

```
CIRCLE (128,96) , 8Ø , , , 1
CIRCLE (128,96) , 8Ø
```

dão o mesmo resultado.

Aplicando outros valores





```
10 SCREEN2
20 CIRCLE(120,100),50,8
30 PAINT(100,100),8
40 GOTO40
```

no modo de baixa resolução (SCREEN 3), a circunferência e o seu interior podem ser coloridos com cores diferentes.

```
10 SCREEN3
20 CIRCLE(128,100),50,8
30 PAINT(100,100),5,8
40 GOTO40
```

- **COMO DESENHAR OUTROS GRÁFICOS**

O uso da instrução permite desenhar um quadrado, um triângulo, e assim por diante.

A instrução DRAW usa as seguintes instruções-macro-gráficas, para desenhar as linhas:

Un :	Traça uma linha para acima da posição fixada para o cursor
Dn :	Movimenta para baixo
Ln :	Movimenta para esquerda
Rn :	Movimenta para direita
En :	Movimenta diagonalmente para cima e direita
Fn :	Movimenta diagonalmente para baixo e direita
Gn :	Movimenta diagonalmente para baixo e esquerda
Hn :	Movimenta diagonalmente para cima e esquerda
M x , y :	Provoca movimentos absolutos ou relativos
An :	Especifica o ângulo de um gráfico (n = Ø, 1, 2, 3)
Cn :	Especifica a cor (Ø ⩽ n ⩽ 15)
Sn :	Especifica a escala (Ø ⩽ n ⩽ 255)

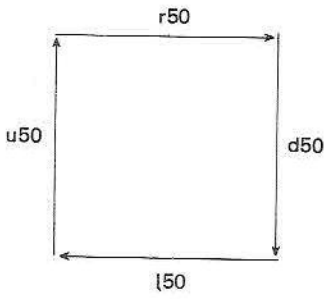
Para maiores detalhes, ver Manual de BASIC (ilustração DRAW).

Por exemplo, a seguinte instrução desenhará um quadrado sendo o lado de 50 pontos do vídeo.

DRAW " R50 D50 L50 U50 "

Mediante o seguinte programa, são traçados quadrados em posições e cores cambiantes.

```
10 SCREEN2
20 FORX=0TO150STEP10
30 DRAW"bm=x;,=x;"
40 C=X/10
50 DRAW"c=c;"
60 DRAW"r50d50l50u50"
70 NEXTX
80 GOTO80
```



Observamos a linha 3Ø do programa acima:

```
3Ø  DRAW " bm = x ; , = x ; "
```

Nessa instrução, o caracter x compreendido entre o "=" e o ";" representa uma variável. Dentro da instrução DRAW, sempre que se quiser colocar uma variável, ela deverá estar entre os sinais "=" e ";". A variável C é por sua vez utilizada na linha 5Ø.

O programa a seguir desenha quadrados, mudando agora a dimensão e a cor.

```
10 SCREEN2
20 DRAW"bm0,0"
30 FOR S=1TO15
40 DRAW"s=s;c=s;"
50 DRAW"r50d50l50u50"
60 NEXTS
70 GOTO70
```

2.11.3. VISUALIZAÇÃO DE CARACTERES

No modo gráfico é impossível visualizar os caracteres no vídeo através da instrução PRINT.

Para conseguí-lo, deveremos usar as instruções:

```
OPEN " GRP : " FOR OUTPUT AS # ⟨ Nº do arquivo ⟩
PRINT # ⟨ Nº do arquivo ⟩ , " string "
```

Para posicionar os "strings" no vídeo, usa-se PSET ou PRESET (e não LOCATE).

Vejamos o seguinte exemplo:

```
10 SCREEN2
20 OPEN"grp:"FOROUTPUTAS#1
30 PRESET(20,100)
40 PRINT#1,"Nosso computador"
50 GOTO50
```

2.12. MOVIMENTAÇÃO DE UM DESENHO (SPRITES)

O HOT- BASIC permite desenhar na tela objetos em movimento, ou seja, criar efeitos de animação. No manual, chamaremos a esses objetos passíveis de movimentação, de "sprites". Uma das propriedades mais marcante dos "sprites", é a de poder aparecer no vídeo em planos diferentes. A ilustração do fundo, por exemplo, definida usando as instruções gráficas do capítulo precedente, permanece intacta.

Para utilizar os sprites, deve-se satisfazer duas condições:

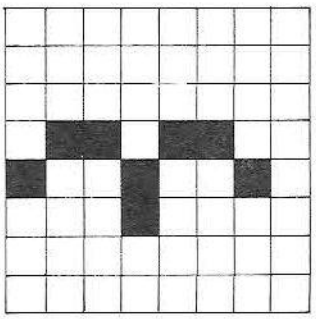
1. O sprite deve ser definido, ou seja, deve-se determinar a forma do objeto que representará.

2. A posição da tela na qual o sprite aparecerá, deverá também ser definida.

- DEFINIÇÃO DOS SPRITES

Os sprites podem ser definidos em formato grande ou pequeno. Um formato pequeno significa que os sprites serão compostos de 8 x 8 pontos de imagem. Usando sprites de formato pequeno, é possível definir um máximo de 256. A definição de um sprite é feito com a variável "SPRITE $".

Suponhamos querer definir o seguinte sprite:



Cada grupo de 8 pontos horizontais de imagem, formam um byte que é dividido em 2 grupos de 4 bits cada.

Eles são definidos em forma hexadecimal tal como está indicado à direita da figura. A dimensão do sprite é definida na instrução "SCREEN", que determina o modo de trabalho da tela.

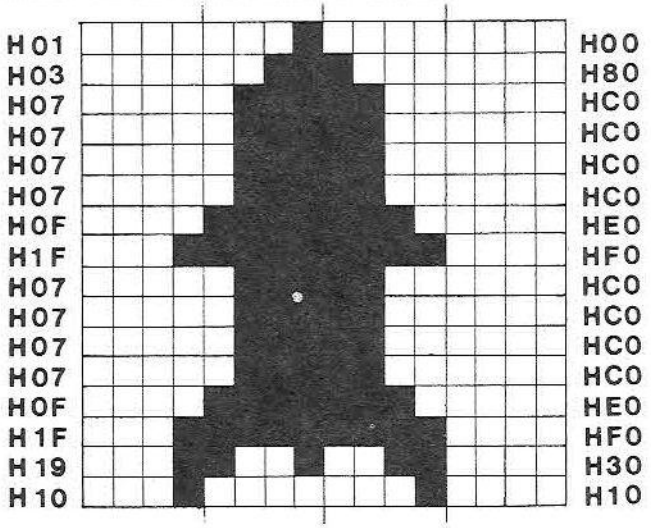
O sprite acima desenhado pode ser codificado da seguinte forma:

```
10 SCREEN 2,0
20 A$=CHR$(&H00)+CHR$(&H00)+CHR$(&H00)+CHR$(&H6C)+
CHR$(&H92)+CHR$(&H10)+CHR$(&H00) +CHR$(&H00)
30 SPRITE$(1)=A$
```

Os sprites também podem ser definidos em formato grande. Nesse caso cada um dos sprites será composto por 16 x 16 pontos de imagem.

Usando sprites de formato grande, é possível definir um máximo de 64.

Suponhamos querer criar o seguinte sprite:



Cada 16 pontos horizontais da imagem, formam 2 bytes, cada um dos quais é subdividido em 2 grupos de 4 bits cada.

Esses subgrupos são codificados em forma hexadecimal. No nosso BASIC, os bytes estão em seqüência vertical.

O sprite do desenho acima pode ser codificado como segue:

```
10 SCREEN 2,2
20 A$=CHR$(&H01)+CHR$(&H03)+CHR$(&H07)+CHR$(&H07)+
CHR$(&H07)+CHR$(&H07)+CHR$(&H0F)+CHR$(&H1F)
30 B$=CHR$(&H07)+CHR$(&H07)+CHR$(&H07)+CHR$(&H07)+
CHR$(&H0F)+CHR$(&H1F)+CHR$(&H19)+CHR$(&H10)
40 C$=CHR$(&H00)+CHR$(&H80)+CHR$(&HC0)+CHR$(&HC0)+
CHR$(&HC0)+CHR$(&HC0)+CHR$(&HE0)+CHR$(&HF0)
50 D$=CHR$(&HC0)+CHR$(&HC0)+CHR$(&HC0)+CHR$(&HC0)+
CHR$(&HE0)+CHR$(&HF0)+CHR$(&H30)+CHR$(&H10)
60 SPRITE$(1)=A$+B$+C$+D$
```

Os sprites podem também ser codificados de maneiras diversas.

Vejamos o exemplo do sprite em formato pequeno:

```
10 SCREEN 2,0
20 DATA 0,0,0,108,146,16,0,0
30 A$=""
40 FOR I=1 TO 8
50 READ A:A$=A$+CHR$(A)
60 NEXT I
70 SPRITE$(1)=A$
```

Nesse exemplo, o valor decimal de 8 bytes é memorizado em uma instrução "DATA". Esses valores são lidos na linha 50 e memorizados na variável "A$" como valores alfanuméricos.

O sprite de formato grande poderá ser codificado também assim:

```
10 SCREEN2,2
20 DATA1,3,7,7,7,7,15,31
30 DATA7,7,7,7,15,31,25,16
40 DATA0,128,192,192,193,192,224,240
50 DATA192,192,192,192,224,240,48,16
60 A$=""
70 FORI=1TO32
80 READA:A$=A$+CHR$(A)
90 NEXTI
100 SPRITE$(1)=A$
```

Nesse exemplo, o valor decimal de todos os 32 bytes é memorizado em 4 instruções "DATA". Esses valores são lidos na linha 80 e memorizados na variável "A$" como valores alfanuméricos.

2.12.1. INSERÇÃO DOS SPRITES NA TELA

Na instrução "SCREEN" vem inserido um código imediatamente após o modo de trabalho da tela, que indica o tipo dos sprites que serão utilizados. O código utilizado tem o seguinte significado:

Ø = sprites pequenos, formato 8 x 8
1 = sprites pequenos, formato aumentado 16 x 16
2 = sprites grandes, formato 16 x 16
3 = sprites grandes, formato aumentado 32 x 32

NOTA: Nos códigos Ø e 2, cada bit de definição de sprite se corresponde com um ponto de imagem (pixel). Nos códigos 1 e 3, cada bit se corresponde com 2 x 2 pontos de imagem. Nos casos Ø e 1 o sprite será formado por 8 bytes. Nos casos 2 e 3 o sprite será formado por 32 bytes. (Para maiores detalhes, vide Processador de Vídeo)

Na seguinte linha de programa, indica-se em primeiro lugar o modo da tela (2), e em seguida a informação relativa ao formato dos sprites, nesse caso 32 x 32.

```
1Ø SCREEN 2,3
```

Existem algumas limitações quanto ao uso dos sprites:

a) Os sprites não podem ser utilizados no modo texto 1.

b) Não é possível usar uma combinação de dimensões para os diversos sprites.

c) É possível inserir simultaneamente numa linha horizontal da tela um máximo de 4 sprites.

Um sprite é colocado na tela mediante a instrução "PUT SPRITE", seguida das seguintes informações:

a) Número de prioridade
b) Posição na tela
c) Cor do sprite
d) Número do sprite

O número de prioridade deve estar compreendido entre Ø e 31, e tem o seguinte significado:

Quando 2 sprites são inseridos na mesma posição na tela, o sprite de prioridade menor precederá aquele de prioridade maior.

A posição na tela é indicada mediante as coordenadas X e Y. A coordenada X pode ser um número compreendido entre −32 e 255 e a coordenada Y um número entre −32 e 191.

Isto é, será possível colocar um sprite fora da tela.

As coordenadas X = Ø e Y = Ø indicam o ponto mais alto à esquerda da tela.

A cor do sprite é indicada pelo número do código de cor (vide apêndice C).

O número do sprite determina qual será o sprite colocado na tela.

Vejamos o seguinte exemplo:

```
NEW
10 SCREEN 2,0
20 DATA 0,3,31,63,255,255,63,7
30 DATA 0,128,248,255,255,254,252,192
40 DATA 0,0,0,108,146,16,0,0
50 FOR I=1 TO 3
60 A$=""
70 FOR J=1 TO 8
80 READ A:A$=A$+CHR$(A)
90 NEXT J
100 SPRITE$(I)=A$
110 NEXT I
120 PUT SPRITE 4,(180,50),15,1
130 PUT SPRITE 5,(188,50),15,2
140 FOR I=-32 TO 212
150 PUT SPRITE 3,(I,50),1,3
160 NEXT I
170 PUT SPRITE 3,(I,209)
180 FOR I=212 TO -32 STEP -1
190 PUT SPRITE 3,(I,50),1,3
200 NEXT I
210 GOTO 140
RUN
```

Se o programa for introduzido no computador corretamente, veremos aparecer na tela um pássaro preto que voa por cima de uma nuvem branca.

A nuvem branca é definida por dois sprites (sprite número 1 e o número 2) enquanto que o pássaro está definido pelo sprite número 3.

A nuvem aparece na tela graças às instruções das linhas 12Ø e 13Ø.

O pássaro preto se movimenta da esquerda para direita, comandado pela interação FOR ... NEXT das linhas 14Ø a 16Ø.

A interação"FOR ... NEXT"das linhas 18Ø a 2ØØ faz com que o passarinho voe da direita à esquerda.

Pressione agora as teclas [CTRL] + [STOP] para deter o programa e modifique a linha Ø da seguinte forma:

```
1Ø SCREEN 2, 1
```

Executando agora o programa veremos que a dimensão dos sprites foi duplicada.

Interrompa novamente o programa e modifique a linha 19Ø.

```
19Ø PUT SPRITE 6 , (I , 15Ø) , 1 , 3
```

Executando agora o programa novamente, veremos que o pássaro passa por trás da nuvem na sua viagem de volta, devido à mudança do número de prioridade.

- **COLISÃO DE SPRITES**

O BASIC tem a possibilidade de executar uma dada sub-rotina quando dois sprites colidem. Usa-se para isso a instrução:

"ON SPRITE GO SUB"

Vamos interromper o programa (mediante CTRL + STOP) e modificá-lo como segue:

```
5 SCREEN 2,0
10 ON SPRITE GOSUB 300
135 SPRITE ON
210 GOTO 135
300 PUT SPRITE 3,(I,209)
310 FOR J=50 TO 191
320 PUT SPRITE 3,(I,J),1,3
325 NEXT J
330 I=0:SPRITE OFF
340 RETURN 140
```

Execute o novo programa. A linha 10 indica agora o número de início da sub-rotina à qual o BASIC deve saltar. Na linha 135 é ativada a função de colisão.

Na sub-rotina das linhas 300 a 340, o pássaro é movimentado para baixo através de um FOR ... NEXT.

Na linha 330 a função de colisão é desativada e a linha 340 instrue o computador a interromper a sub-rotina voltando ao programa principal.

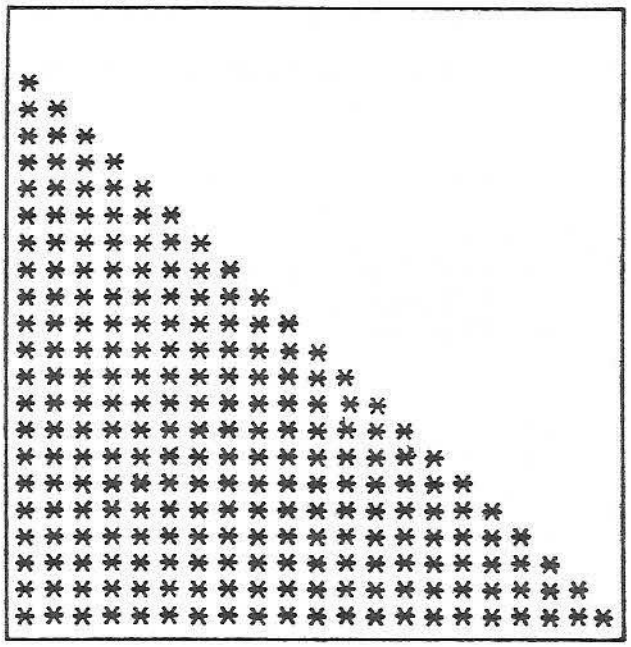
2.13. CRIAÇÃO DE FIGURAS USANDO CARACTERES

É possível no modo texto, criar figuras usando caracteres.

Vejamos o seguinte exemplo:

```
10 N=1
20 FOR S=1 TO N
30 PRINT "*";
40 NEXT S
50 PRINT
60 N=N+1
70 IF N>21 THEN END
80 GOTO 20
```

Executando esse programa veremos no vídeo:



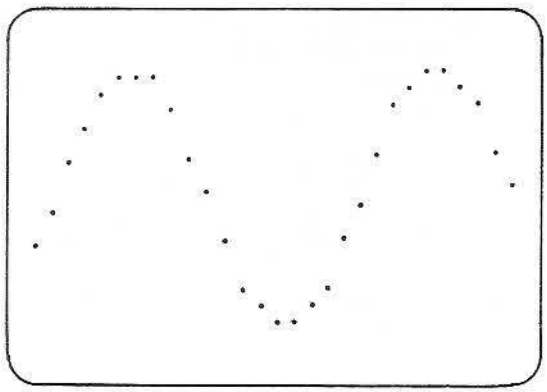
Pode-se simplificar o programa anterior usando STRING$.

```
10 FOR I=1 TO 21
20 PRINT STRING$ (I, "*")
30 NEXT I
```

O programinha acima desenha o mesmo triângulo de asteriscos do programa precedente.

Vejamos agora um exemplo de utilização da instrução LOCATE. O seguinte programa desenha a curva da função trigonométrica SIN (seno) usando pontos.

```
10 CLS
20 FOR X=0 TO 28
30 Y=SIN (T) * 8+12
40 LOCATE X, Y : PRINT ".";
50 T=T+.35
60 NEXT X
70 LOCATE 0, 0
```



Pode-se desenhar também um diagrama de barras (histograma).

O seguinte programa lê os dados necessários com a instrução DATA e os apresenta no vídeo com um histograma.

```
10 CLS
20 FORK=1TO9
30 READN$,D
40 PRINTN$;TAB(10);
50 FORI=1TOD
60 PRINT"*";
70 NEXTI
80 PRINT
90 NEXTK
100 DATAMARIO,5
110 DATASERGIO,10
120 DATALUIS,8
130 DATAMARIA,7
140 DATAINES,10
150 DATAJORGE,9
160 DATACELIA,8
170 DATAMARCELA,3
180 DATAPAULO,1
```

Suponhamos que 10 seja a nota máxima. Executando o programa veremos o seguinte:

```
MARIO       *****
SERGIO      **********
LUIS        ********
MARIA       *******
INES        **********
JORGE       *********
CELIA       ********
MARCELA     ***
PAULO       *
```

O programa pode ser simplificado da seguinte forma:

```
10 CLS
20 D$=STRING$ (10, "#")
30 FOR K=1 TO 9
40 READ N$, D
50 PRINT N$ ; TAB (10) ;
60 PRINT LEFT$ (D$, D)
70 NEXT K
100 DATA Mario, 5
      :                    Seguem as outras instruções DATA.
```

O programa também podería ser escrito da seguinte forma:

```
10 CLS
20 FOR K=1 TO 9
30 READ N$, D
40 PRINT N$ ; TAB (10) ;
50 PRINT STRING$ (D, "#")
60 NEXT K
100 DATA Mario, 5
       ⋮
```

Seguem as outras instruções DATA.

2.14. MÚSICA

Para a geração de sons, e eventualmente de música, usa-se a instrução PLAY. Deve-se conhecer para isso, as instruções "macro" correspondentes.

PLAY "instruções macro musicais"

NOTA:	A , B , ... G	indicam a nota musical
	On	indica a nésima oitava musical
	Nn	indica nésima nota (numeração absoluta)

Duração do som:

Ln	indica o comprimento do som

Pausa:

Rn	indica o comprimento da pausa (ausência de som)

Tempo:

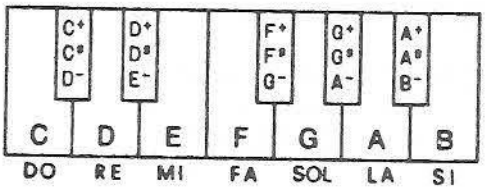
Tn	indica o tempo da música.

2.14.1. AS NOTAS MUSICAIS

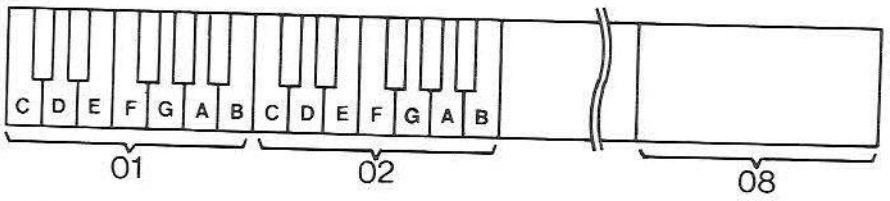
c, d, e, f, g, a, b indicam segundo a notação musical americana, respectivamente as notas do, ré, mi, fá, sol, lá, si.

Especificando PLAY "c d e f g a b ", obteremos uma escala do tipo do ré mi fá sol lá si. Os caracteres alfabéticos podem ser maiúsculos ou minúsculos.

Os semi-tons musicais são especificados escrevendo os caracteres # ou + (sustenido) ou – (para bemol).



"On" indica à oitava da nota, onde n pode variar entre 1 e 8. Na ausência dessa indicação o computador assume o valor n = 4 (quarta oitava).

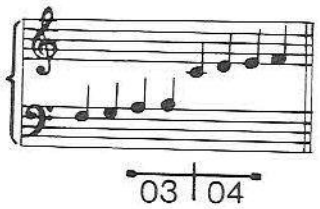


As letras A até G indicam a nota dentro da oitava.

A notação "Nn" permite usar uma numeração absoluta referente ao teclado musical completo, n pode variar entre 0 e 96.

A correspondência fica assim estabelecida:

Exemplo: O DO da quarta oitava corresponde a N36.

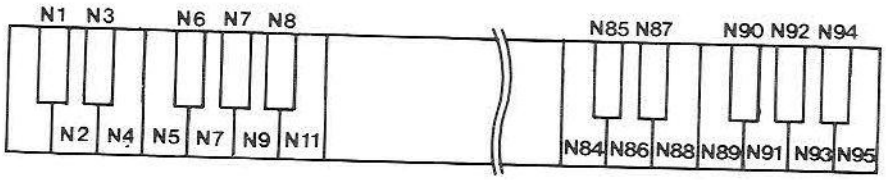


- DURAÇÃO DO SOM

"Ln" especifica a duração de uma nota, onde n é um número inteiro compreendido entre 1 e 64.

1 indica uma nota semibreve (nota de maior duração)
2 indica uma nota mínima (duas mínimas = 1 semibreve)
4 indica uma nota semínima (4 semínimas = 1 semibreve)
8 indica uma colcheia (8 colcheias = 1 semibreve)

e assim por diante até 64 que indica uma semifusa.



Como nas partituras musicais, para indicar uma duração de uma vez e meia a duração original da nota, basta escrever a letra que indica a nota seguida de um ponto (.).

Poderá haver inclusive mais do que um ponto. No caso por exemplo de 3 pontos sucessivos, o comprimento final é igual ao original multiplicado por 3.375 (3.375 = 1.5 * 1.5 * 1.5).

(semibreve)	= L1	(mínima)	= L2
(semínima)	= L4	(colcheia)	= L8
(semicolcheia)	= L16	(fusa)	= L32
(semifusa)	= L64		

- **AS NOTAS**

As notas são indicadas pela respectiva sigla e pelo comprimento do som.

PLAY "L 4 O 4 G." equivale a PLAY "L 4 N 4 3."

PLAY "L 8 O 4 D+" equivale a PLAY "L 8 N 3 9"

É possível também omitir a letra L e escrever a duração da nota (n) após o nome da mesma.

PLAY "L 4 0 4 G" equivale a PLAY "0 4 G 4"

A cada caracter de A a G (ou Nn como nota absoluta) deve ser especificado uma correspondência com cada nota do pentagrama.

O comprimento ou a oitava podem ser omitidos, em cujo caso será repetido o último valor especificado para L e O.

- **PAUSA**

Rn indica uma pausa musical, onde n é um inteiro que vai de 1 a 64.

O n indica a duração da pausa, com o mesmo critério que é usado para a duração da nota.

R1 indica uma pausa equivalente à duração de uma semibreve, e assim por diante até R64.

	=R1		=R2
	=R4		=R8
	=R16		=R32
	=R64		

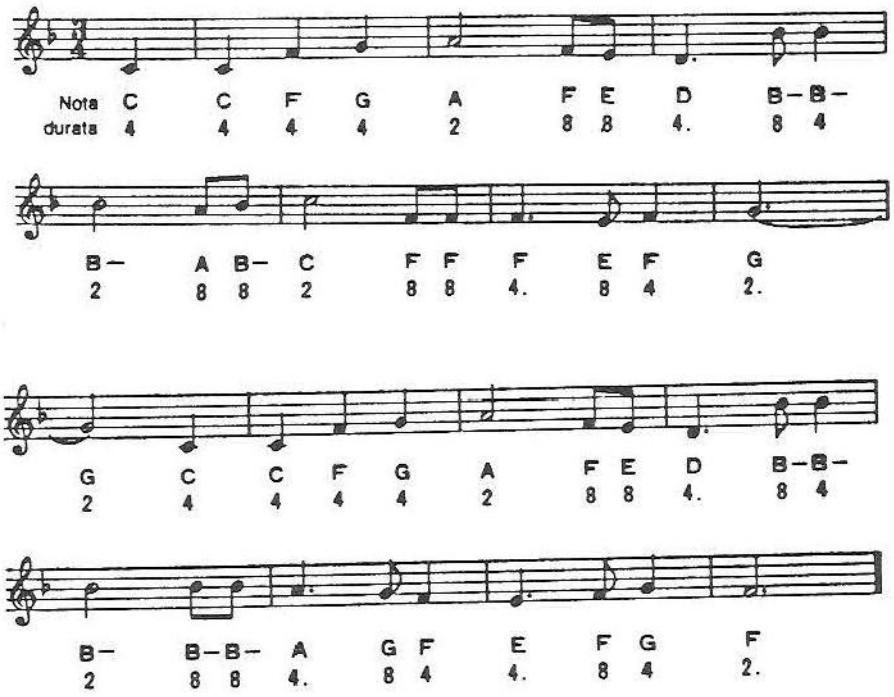
Como no caso das notas, o ponto aumenta a duração da pausa em um fator de 1.5 .

- **TEMPO**

"Tn" especifica a velocidade da música. O valor de n pode variar entre 32 e 255. "Tn" indica a velocidade na qual será executada a música.

Na ausência de indicação de tempo, o valor assumido será 120.

Podemos finalmente programar a seguinte música:




```
10 PLAY "T128"
20 PLAY "O4L4CCFGL2AL8FED4. B-B-4"
30 PLAY "B-2AB-O5C2O4FFF4. EF4G2.G2"
40 PLAY "L4CCFGA2L8FED4. B-8B-4"
50 PLAY "B-2L8B-B-A4. GF4E4. FG4F2."
60 END
```

- ACORDES MUSICAIS

O comando PLAY permite também a programação de acordes musicais. Para gerar o seguinte acorde, programamos:

PLAY "C" , "E" , "G"

- VARIÁVEIS

Na linguagem macro musical, podem-se usar também variáveis para conter os dados com as notas, as durações, as pausas, etc.

Nesse caso deve-se escrever a variável entre os sinais de "=" e ";" (ponto e vírgula).

Por exemplo:

```
PLAY "N10" pode também ser programado como: X = 10
PLAY "N = X ;"
```

Executando o programa a seguir, ouviremos todas as notas possíveis.

```
1Ø FOR I = 1 TO 96
2Ø PLAY "N = I ;"
3Ø NEXT
```

- **VOLUME**

Na linguagem macro musical são previstos também outros parâmetros:

Vn	especifica o volume do som
Sn	modifica a forma da envoltória
Mn	muda a freqüência da envoltória

(Para maiores detalhes sobre envoltória, vide apêndice J)

S e M são utilizados juntos. Por outro lado, não devem ser usados com o parâmetro V. Executando o seguinte programa, o som da nota "do" será escutado continuamente, mas com variações na freqüência e na forma da envoltória.

```
1Ø FOR S = 1 TO 15
2Ø FOR M = 5ØØ TO 15ØØ STEP 1ØØ
3Ø PLAY "S = S ; M = M ; C"
4Ø NEXT M
5Ø NEXT S
```

2.15. JOYSTICKS

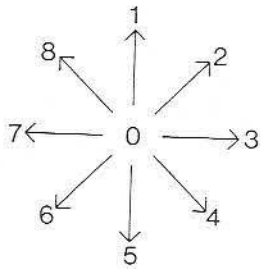
É possível ligar um ou dois joysticks no micro computador. O mesmo efeito é conseguido com as teclas de comando do cursor.

A direção do joystick pode ser controlada com a função "STICK (n)" ; "n" é um número compreendido entre Ø e 2 e tem o seguinte significado:

Ø = As teclas de comando do cursor são usadas como joystick
1 = Joystick ligado no conector 1
2 = Joystick ligado no conector 2

O valor obtido com a função STICK (n) estará compreendido entre Ø e 8.

Ø = Neutro
1 = Norte
2 = Nordeste
3 = Leste
4 = Sudeste
5 = Sul
6 = Sudoeste
7 = Oeste
8 = Noroeste



Vejamos um exemplo:

```
NEW
10 A = STICK (Ø)
20 IF A = STICK (Ø) THEN 20
30 PRINT STICK (Ø)
40 GOTO 10
```

Nesse exemplo, a variável "A" na linha 10 assume o valor das teclas de comando do cursor. Até não ser pressionada uma das teclas de controle, o valor de "A" será Ø.

Ao ser pressionada uma das teclas de comando do cursor, será executada a linha 30.

Experimente pressionar cada uma das teclas de controle. (Para interromper o programa basta pressionar as teclas [CTRL] + [STOP]).

Para o BASIC saber se algum dos botões disparadores do joystick foi pressionado (o mesmo efeito obter-se-á pressionando a barra espaçadora), usa-se a função "STRIG (n)" onde "n" é um número compreendido entre Ø e 4, e assume o seguinte significado:

Ø	=	Para a barra espaçadora
1 e 3	=	Botão disparador do joystick 1
2 e 4	=	Botão disparador do joystick 2

O valor obtido com STRIG (n) é Ø quando não é pressionado o botão disparador e — 1 quando é pressionado.

Exemplo:

```
   NEW
10 IF STRIG (Ø) = — 1 THEN PRINT STRIG (Ø)
20 GO TO 10
```

Na linha 10, o BASIC controla se foi pressionada a barra de espaços, escrevendo na tela o valor da função STRIG (n).

- **LANÇAMENTO DE UM FOGUETE**

  No seguinte exemplo é lançado um foguete pressionando a barra de espaços. A movimentação na horizontal é feita com as teclas de comando do cursor.

  Tente introduzir o programa com cuidado para não cometer erros.

```
NEW
10 SCREEN 2,3:COLOR 1,4,1
20 DATA 0,0,0,0,3,7,15,63
30 DATA 63,127,127,63,31,1,0,0
40 DATA 0,0,0,0,192,252,255,255
50 DATA 255,255,255,255,255,255,28,0
60 DATA 0,0,0,1,15,31,255,255
70 DATA 255,255,255,255,255,255,30,0
80 DATA 0,0,0,128,224,248,248,252
90 DATA 252,248,248 240,240,224,0,0
100 DATA 0,0,0,28,126,127,255,127
```

```
110 DATA 31,15,7,3,0,0,0,0
120 DATA 0,0,0,0,96,240,248,252
130 DATA 254,254,252,252,248,0,0,0
140 DATA 0,0,0,0,0,0,3,15
150 DATA 63,255,255,127,63,31,0,0
160 DATA 0,0,0,0,0,248,252,254
170 DATA 254,252,252,252,252,248,248,0
180 DATA 0,0,0,0,0,0,0,0
190 DATA 0,0,0,0,0,0,0,0
200 DATA 3,3,7,7,7,7,7,15
210 DATA 15,15,15,31,31,31,31,31
220 DATA 0,0,0,0,0,0,0,0
230 DATA 0,0,0,0,0,0,0,0
240 DATA 255,255,255,16,40,16,0,0
250 DATA 0,0,0,0 0,0,0,0
260 DATA 255,255,255,0,1,0,0,0
270 DATA 0,0,0,0,0,0,0,0
280 DATA 240,240,240,128,64,128,0,0

290 DATA 0,0,0,0,0,0,0,0
300 DATA 8,28,28,62,62,62,127,255
310 DATA 62,62,62,62,127,255,201,128
320 DATA 0,0,0,0,0,0,0,128
330 DATA 0,0,0,0,0,128,128,128
340 DATA 8,28,28,62,62,62,62,28
350 DATA 28,20,0,0,0,0,0,0
360 DATA 0,0,0,0,0,0,0,0
370 DATA 0,0,0,0,0,0,0,0
380 FOR J=1 TO 9:A$=""
390 FOR I=1 TO 32:READ A:A$=A$+CHR$(A):NEXT I
400 SPRITE$(J)=A$:NEXT J
410 CIRCLE (88,120),16,6,-0.0001,-3.415,0.5
420 PAINT (88,118),6
430 LINE (0,192)-(255,120),12,BF
440 PUT SPRITE 1,(80,16),15,1
450 PUT SPRITE 2,(112,16),15,2
460 PUT SPRITE 10,(160,80),15,3
470 PUT SPRITE 11,(16,48),15,4
480 I=120:J=144:GOSUB 1000
490 IF STRIG(0)=-1 GOTO 2000
500 IF STICK(0)=3 THEN GOSUB 3000
510 IF STICK(0)=7 THEN GOSUB 4000
520 GOTO 490
1000 PUT SPRITE 5,(I,J),1,5:J=J+32
1010 PUT SPRITE 6,(I,J),1,6:I=I+32
1020 PUT SPRITE 7,(I,J),1,7:J=J-32
1030 PUT SPRITE 8,(I,J),7,8:I=I-32
1040 RETURN
2000 I=I+32:FOR K=144 TO -33 STEP -1
2010 PUT SPRITE 8,(I,K),7,8:M=K+32
2020 PUT SPRITE 9,(I,M),9,9
2025 FOR T=1 TO 1:NEXT T
2030 PUT SPRITE 9,(I,209)
2040 NEXT K:FOR K=1 TO 1000:NEXT K:COLOR 15,4,7:END
3000 I=I+1:IF I>200 THEN I=200
3010 GOSUB 1000:RETURN
4000 I=I-1:IF I<0 THEN I=0
4010 GOSUB 1000:RETURN
```

3. Apêndices

A. MENSAGENS DE ERRO

TABELA DE MENSAGENS DE ERRO		
CÓDIGO	MENSAGEM	DESCRIÇÃO
1	"NEXT" SEM "FOR"	Em uma instrução "NEXT" uma variável não se corresponde com nenhuma outra previamente executada, ou então uma instrução "NEXT" que não é precedida pela FOR correspondente.
2	ERRO SINTAXE	Foi encontrada uma linha contendo uma seqüência incorreta de caracteres (por exemplo: falta fechar parênteses, comando com erro de sintaxe, sinais de pontuação errados, etc.).
3	"RETURN" SEM "GOSUB"	Foi encontrada uma instrução RETURN sem a sua correspondente GOSUB prévia.
4	SEM "DATA"	Foi executada uma instrução READ quando já não existe instrução DATA com dados ainda não lidos durante o programa.
5	FUNÇÃO ILEGAL	Um parâmetro que está fora da faixa foi passado para uma função matemática ou para uma função string. Este erro também pode acontecer como resultado de: 1. Um índice negativo ou extremamente grande. 2. Um argumento negativo ou zero usado para LOG. 3. Um argumento negativo usado para SQR. 4. Um argumento impróprio usado em MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, SPACE$, STRING$, INSTR$ ou ON ... GOTO.
6	OVERFLOW	O resultado de um cálculo é grande demais para ser representado no formato numérico do BASIC.
7	FALTA MEMÓRIA	O programa é muito longo, tem excesso de arquivos, tem excesso de FOR ou GOSUB, tem excesso de variáveis ou expressões muito complexas.

TABELA DE MENSAGENS DE ERRO		
CÓDIGO	MENSAGEM	DESCRIÇÃO
8	Nº LINHA INEXISTENTE	Faz-se referência a uma linha inexistente em uma instrução GOTO, GOSUB, IF ... THEN ... ELSE.
9	ÍNDICE FORA DO LIMITE	Um elemento de arranjo foi definido com um índice que está fora das dimensões pré-fixadas para esse arranjo ou então com um número errado de índices.
10	"DIM" REDEFINIDO	Duas instruções "DIM" foram dadas para o mesmo arranjo, ou uma instrução "DIM" foi dada para um arranjo depois que a dimensão "default" de 1Ø foi fixada para esse mesmo arranjo.
11	DIVISÃO POR ZERO	Foi encontrada uma divisão por zero em uma expressão, então uma situação na qual Ø é elevado a uma potência negativa.
12	DIRETO / ILEGAL	Uma instrução que é ilegal no modo direto foi introduzida como comando no modo direto.
13	TIPO DESIGUAL	Foi atribuído um valor numérico a uma variável string ou vice-versa, ou então, uma função que espera um argumento numérico recebe um argumento string ou vice-versa.
14	FALTA ÁREA "STRING"	As variáveis string fizeram com que o BASIC supere o espaço de memória disponível. O BASIC posiciona os strings dinamicamente conforme a necessidade, até ficar sem memória disponível. Pode-se usar a instrução CLEAR para criar maior espaço livre.
15	"STRING" LONGA	Tentou-se criar um "string" com mais de 255 caracteres.
16	"STRING" COMPLEXA	Foi criada uma expressão "string" excessivamente longa ou complexa. A expressão deverá ser partida em expressões menores.

TABELA DE MENSAGENS DE ERRO		
CÓDIGO	MENSAGEM	DESCRIÇÃO
17	NÃO CONTINUO !	Tentou-se continuar um programa que: 1. Tinha sido detido devido a um erro. 2. Foi modificado durante um "BREAK" na execução. ou 3. Não existe.
18	FUNÇÃO NÃO DEFINIDA	Foi chamada uma função definida pelo usuário (FN) sem antes tê-la definido através de uma instrução DEF FN.
19	ERRO / PERIFÉRICO	Aconteceu um erro de I/O em uma operação de disco, cassete, impressora ou TRC. Este é um erro fatal, isto é, o BASIC não consegue se recuperar deste tipo de erro.
20	ERRO / VERIF.	O programa na memória é diferente do programa gravado na fita cassete.
21	"RESUME"	O BASIC entrou numa rotina de tratamento de erros que não contém a instrução RESUME.
22	"RESUME" SEM "ERROR"	Foi encontrada uma instrução "RESUME" antes de ter entrado numa rotina de tratamento de erros.
23	ERRO INDEFINIDO	Não existe mensagem de erro disponível para a condição de erro existente. Isto é causado normalmente por um ERROR com um código de erro indefinido.
24	FALTA OPERANDO	Foi encontrada uma expressão que contém um operador sem o operando correspondente a seguir.
25	LINHA MUITO LONGA	Foi introduzida uma linha com excessivo número de caracteres.
26 ⋮ 49	ERROS INDEFINIDOS Esses códigos não têm definições. Foram reservados para futuras expansões do BASIC.	

TABELA DE MENSAGENS DE ERRO		
CÓDIGO	MENSAGEM	DESCRIÇÃO
50	CAMPO MAIOR	Uma instrução FIELD tentou posicionar maior número de bytes dos que foram especificados para o registro de um arquivo aleatório (255 bytes). Ou então, o fim de um buffer "FIELD" foi encontrado enquanto estava se fazendo I/O sequencial (PRINT #, INPUT #) em um arquivo aleatório.
51	ERRO INTERNO	Erro provocado por alguma situação não prevista pelo interpretador BASIC.
52	Nº DO ARQUIVO	Algum comando ou instrução faz referência a um arquivo para o qual não foi dado previamente o OPEN ou então o Nº está fora da faixa fixada para o Nº de arquivos através do MAXFILES.
53	ARQ. NÃO EXISTE	Um comando LOAD, KILL ou OPEN faz referência a um arquivo não existente no disco.
54	ARQUIVO ABERTO	Tenta-se abrir, através de uma instrução "OPEN" um arquivo que já tinha sido aberto; ou então dá-se um KILL para um arquivo aberto.
55	FIM DO ARQUIVO	Foi executada uma instrução INPUT após todos os dados já terem sido introduzidos, ou para um arquivo nulo (vazio). Para evitar este erro, usa-se a função EOF afim de detectar o fim do arquivo.
56	NOME ARQUIVO	Foi utilizada uma forma ilegal na definição do nome do arquivo usada em LOAD, SAVE, KILL, NAME, etc.

TABELA DE MENSAGENS DE ERRO		
CÓDIGO	MENSAGEM	DESCRIÇÃO
57	COMANDO DIRETO / ARQUIVO	Foi encontrada uma instrução direta enquanto estava sendo carregado (LOAD) um arquivo em formato ASCII. O carregamento do arquivo é encerrado.
58	ARQUIVO SEQÜENCIAL	Uma instrução própria para o acesso aleatório é dada para um arquivo seqüêncial.
59	FALTA "OPEN"	O arquivo especificado em um PRINT # , INPUT # , etc., não foi aberto através do OPEN correspondente.
60 : : : 255	ERROS INDEFINIDOS Esses erros não estão definidos. O usuário poderá fazer a sua própria definição de códigos nessa área. Essa área poderá ser utilizada por algum dos periféricos. Consulte manual do periférico.	

B. FUNÇÕES MATEMÁTICAS

As funções matemáticas derivadas que não foram inclusas diretamente no BASIC, podem ser definidas da seguinte forma:

FUNÇÕES DERIVADAS	DEFINIÇÃO PARA O BASIC
SECANTE	= 1/COS(X)
COSECANTE	= 1/SIN(X)
COTANGENTE	= 1/TAN(X)
ARCO SENO	= ATN(X/SQR(–X * X + 1))
ARCO COSENO	= –ATN(X/SQR(–X * X + 1)) + 1 . 5708
ARCO SECANTE	= ATN(X/SQR(X * X – 1)) + SGN (SGN(X)–1) * 1 . 5708
ARCO COSECANTE	= ATN(X/SQR(X * X – 1)) + (SGN(X) – 1) * 1 . 5708
ARCO COTANGENTE	= ATN(X) + 1 . 5708
SENO HIPERBÓLICO	= (EXP(X) – EXP(–X)) / 2
COSENO HIPERBÓLICO	= (EXP(X) + EXP(–X)) / 2
TANGENTE HIPERBÓLICA	= EXP(–X) / EXP(X) + EXP (–X) * 2 + 1
SECANTE HIPERBÓLICA	= 2/(EXP(X) + EXP (–X))
COSECANTE HIPERBÓLICA	= 2/(EXP(X) – EXP(–X))
COTANGENTE HIPERBÓLICA	= EXP(–X) / (EXP(X) – EXP (–X)) * 2 + 1
ARCO SENO HIPERBÓLICO	= LOG(X + SQR (X * X + 1))
ARCO COSENO HIPERBÓLICO	= LOG(X + SQR (X * X – 1))
ARCO TANGENTE HIPERBÓLICA	= LOG((1 + X) / (1 – X)) / 2
ARCO SECANTE HIPERBÓLICA	= LOG((SQR (–X * X + 1) + 1) / X)
ARCO COSECANTE HIPERBÓLICA	= LOG((SGN(X) * SQR (X * X + 1) + 1) / X)
ARCO COTANGENTE HIPERBÓLICA	= LOG((X + 1) / (X – 1)) / 2

C. TABELA DE CÓDIGOS DE CORES

CÓDIGO	COR
Ø	Transparente
1	Preto
2	Verde médio
3	Verde claro
4	Azul escuro
5	Azul claro
6	Vermelho escuro
7	Ciano
8	Vermelho médio
9	Vermelho claro
1Ø	Amarelo escuro
11	Amarelo claro
12	Verde escuro
13	Magenta
14	Cinza
15	Branco

D. TABELA DE CARACTERES DE CONTROLE NO HOT-BASIC

.TEREMOS DE 00H A 1FH OS SEGUINTES CARACTERES:

TECLADO	D	HEX	FUNCAO
CTRL+A	01	&H01	DETERMINA CARACTER GRAFICO
CTRL+B	02	&H02	DESVIA O CURSOR PARA O INICIO DA PALAVRA ANTERIOR
CTRL+C	03	&H03	ENCERRA A CONDICAO ENTRADA
	04	&H04	
CTRL+E	05	&H05	CANCELA CARACTERES DO CURSOR ATE O FIM DA LINHA
CTRL+F	06	&H06	DESVIA O CURSOR PARA O INICIO DA PALAVRA SEGUINTE
CTRL+G	07	&H07	ACIONA BEEP
CTRL+H	08	&H08	APAGA A LETRA ANTERIOR AO CURSOR(BS)
CTRL+I	09	&H09	MOVE O CURSOR PARA A POS. DA TABULACAO SEGUINTE (TAB)
CTRL+J	10	&H0A	MUDA DE LINHA(LINE FEED)
CTRL+K	11	&H0B	VOLTA O CURSOR PARA APOSICAO (1,1) (HOME)
CTRL+L	12	&H0C	LIMPA A TELA E VOLTA O CURSOR PARA POS. (1,1)
CTRL+M	13	&H0D	RETORNO DO CARRO (RETURN)
CTRL+N	14	&H0E	MOVE O CURSOR PARA O FIM DA LINHA
	15	&H0F	
	16	&H10	
	17	&H11	
CTRL+R	18	&H12	LIGA E DESLIGA O MODO DE INSERCAO (INS)
	19	&H13	
	20	&H14	
CTRL+U	21	&H15	APAGA TODA A LINHA EM QUE ESTA O CURSOR
	22	&H16	
	23	&H17	
CTRL+X	24	&H18	(SELECT)
	25	&H19	
	26	&H1A	
CTRL+[	27	&H1B	(ESC)
CTRL+\	28	&H1C	MOVE O CURSOR PARA A DIREITA
CTRL+]	29	&H1D	MOVE O CURSOR PARA A ESQUERDA
CTRL+^	30	&H1E	MOVE O CURSOR PARA CIMA
CTRL+_	31	&H1F	MOVE O CURSOR PARA BAIXO
	127	&H7F	APAGA A LETRA QUE ESTA SOBRE O CURSOR(DEL)

E. TABELAS DE CARACTERES

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
0	00	BLANK (NULL)
1	01	☺
2	02	☻
3	03	♥
4	04	♦
5	05	♣
6	06	♠
7	07	•
8	08	◘
9	09	○
10	0A	◙
11	0B	♂
12	0C	♀
13	0D	♪
14	0E	♫
15	0F	☼
16	10	►
17	11	◄
18	12	↕
19	13	‼
20	14	¶
21	15	§
22	16	▬
23	17	↨
24	18	↑
25	19	↓
26	1A	→
27	1B	←
28	1C	∟
29	1D	↔
30	1E	▲
31	1F	▼
32	20	BLANK (SPACE)
33	21	!
34	22	"
35	23	#

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	—
96	60	‘
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	78	y
122	7A	z
123	7B	{
124	7C	¦
125	7D	}
126	7E	~
127	7F	DEL
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	Á
133	85	à
134	86	¨
135	87	ç
136	88	è
137	89	Í
138	8A	Ó
139	8B	Ú
140	8C	Â
141	8D	Ê
142	8E	Ô
143	8F	À

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	ç
156	9C	£
157	9D	¥
158	9E	Pt
159	9F	ƒ
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	Ã
177	B1	ã
178	B2	Ĩ
179	B3	ĩ

CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL	CARACTER
180	B4	Õ
181	B5	õ
182	B6	Ũ
183	B7	ũ
184	B8	Ŋ
185	B9	ŋ
186	BA	¾
187	BB	ſ
188	BC	◊
189	BD	‰
190	BE	¶
191	BF	§
192	C0	□
193	C1	▚
194	C2	▬
195	C3	▭
196	C4	▪
197	C5	▬
198	C6	▯
199	C7	▚
200	C8	▮
201	C9	▯
202	CA	▮
203	CB	▨
204	CC	▨
205	CD	▼
206	CE	▲
207	CF	▶
208	D0	◀
209	D1	▣
210	D2	▣
211	D3	▛
212	D4	▜
213	D5	▙
214	D6	▟
215	D7	▩

216	D8	Δ
217	D9	‡
218	DA	ω
219	DB	▮
220	DC	▬
221	DD	▯
222	DE	▯
223	DF	▬
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	μ
231	E7	τ
232	E8	Φ
233	E9	θ
234	EA	Ω
235	EB	δ

236	EC	∞
237	ED	∅
238	EE	∈
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	CURSOR

Se um dos caracteres gráficos da tabela precedente, for introduzido pelo teclado, o que na realidade está sendo mandado são 2 bytes : de CRH$ (1) + CHR$ (código do caracter gráfico + 64).

Analogamente acontece quando se emite um caracter gráfico com a instrução PRINT ou LPRINT, na realidade são emitidos 2 bytes : CHR$ (1) + CHR$ (código do caracter gráfico + 64).

O primeiro caracter CHR$ (1) é um código de controle que indica que o caracter sucessivo é gráfico.

Para emitir um caracter gráfico usando CHR$, deve-se colocar CHR$ (1) + CHR$ (código do caracter gráfico + 64).

TABELA DE CARACTERES PADRÃO HB-8000 (IMPRESSORA)

OBS.: CÓDIGO 01H:
Código especial para as impressoras padrão HB-8000, especifica impressão de caracteres gráficos especiais (00H A 1FH).

CARACTERES DE CONTROLE PARA IMPRESSORA		
COMANDO	CÓDIGO	DESCRIÇÃO
Bell	07H	Ativa alarme sonoro
LF	0AH	Avança uma linha
FF	0CH	Salto de página
CR	0DH	Finalização de comandos e retorno do carro
ESC	1BH	
SO	0EH	Entra no modo caracter expandido
S1	0FH	Sai do caracter expandido

TABELA DE CARACTERES PADRÃO ABICOMP

CARACTERES DE CONTROLE PARA IMPRESSORA		
COMANDO	CÓDIGO	DESCRIÇÃO
Bell	07H	Ativa alarme sonoro
LF	0AH	Avança uma linha
FF	0CH	Salto de página
CR	0DH	Finalização de comandos e retorno do carro
ESC	1BH	
SO	0EH	Entra no modo caracter expandido
S1	10H	Sai do caracter expandido

TABELA DE CARACTERES PADRÃO ABICOMP

CÓDIGO (HEX)	CARACTER
20	ESP
21	!
22	"
23	#
24	$
25	%
26	&
27	'
28	(
29	)
2A	*
2B	+
2C	,
2D	—
2E	.
2F	/
30	0
31	1
32	2
33	3
34	4
35	5
36	6
37	7
38	8
39	9
3A	:
3B	;
3C	<
3D	=
3E	>
3F	?

CÓDIGO (HEX)	CARACTER
40	@
41	A
42	B
43	C
44	D
45	E
46	F
47	G
48	H
49	I
4A	J
4B	K
4C	L
4D	M
4E	N
4F	O
50	P
51	Q
52	R
53	S
54	T
55	U
56	V
57	W
58	X
59	Y
5A	Z
5B	[
5C	\
5D	]
5E	^
5F	_

CÓDIGO (HEX)	CARACTER
60	`
61	a
62	b
63	c
64	d
65	e
66	f
67	g
68	h
69	i
6A	j
6B	k
6C	l
6D	m
6E	n
6F	o
70	p
71	q
72	r
73	s
74	t
75	u
76	v
77	w
78	x
79	y
7A	z
7B	{
7C	¦
7D	}
7E	~
7F	

TABELA DE CARACTERES PADRÃO ABICOMP

CÓDIGO (HEX)	CARACTER
AØ	
A1	À
A2	Á
A3	Â
A4	Ã
A5	
A6	Ç
A7	
A8	É
A9	Ê
AA	
AB	
AC	Í
AD	
AE	
AF	Ñ
BØ	
B1	Ó
B2	Ô
B3	Õ
B4	
B5	
B6	
B7	Ú
B8	
B9	Ü
BA	
BB	
BC	
BD	
BE	
BF	°

CÓDIGO (HEX)	CARACTER
CØ	
C1	à
C2	á
C3	â
C4	ã
C5	
C6	ç
C7	
C8	é
C9	ê
CA	
CB	
CC	í
CD	
CE	
CF	ñ
DØ	
D1	ó
D2	ô
D3	õ
D4	
D5	
D6	
D7	ú
D8	
D9	ü
DA	
DB	
DC	a
DD	o
DE	
DF	±

OPÇÕES DE PADRÕES DE CARACTERES PARA A IMPRESSORA: HB8000 OU ABICOMP

A seleção da tabela de caracteres padrão HB8000 ou ABICOMP poderá ser feita alterando-se o conteúdo da posição de memória F417H, se este conteúdo for zero (0) a tabela selecionada será padrão ABICOMP, se for um (1) será padrão HB8000.

Inicialmente ao ligar o seu micro a tabela que estará vigorando será o padrão ABICOMP, então se quiser passar para o padrão HB8000 faça a seguinte operação:

```
POKE &HF417 , 01
```

Para retornar ao padrão ABICOMP utilize:

```
POKE &HF417 , 00
```

F. PALAVRAS RESERVADAS

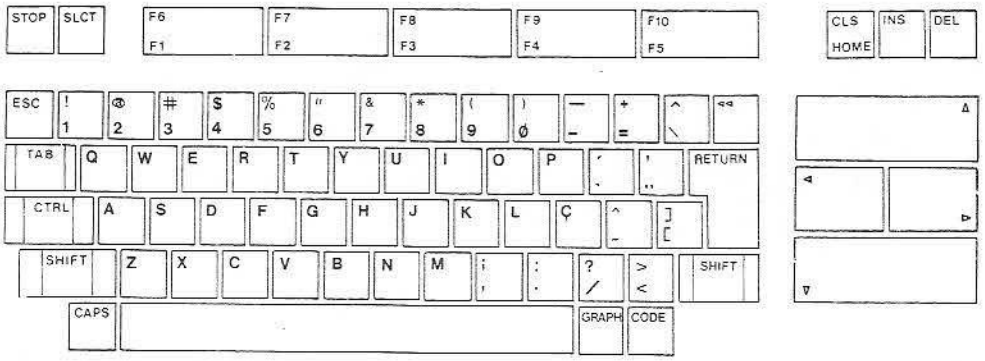
As seguintes palavras não podem ser utilizadas como nomes de variáveis.

ABS
AND
AS
ASC
ATN
AUTO
BASE
BEEP
BIN$
BLOAD
BSAVE
CALL
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOSE
CLS
COLOR
CONT
COPY
COS
CSAVE
CSNG
CSRLIN
CVD
CVI
CVS
DATA
DEF
DEFINT
DEFDBL
DEFSNG
DEFSTR
DEFUSR
DELETE
DIM
DRAW
DSKF
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FN
FOR
FRE
GET
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INPUT$
INSTR
INT
INTERVAL
KEY
KILL
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET
MAXFILES
MERGE
MID$
MKD$
MKI$
MOD
MOTOR
MKS$
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OR
OUT
PAD
PAINT
PDL
PEEK
PLAY
POINT
POKE
POS
PRINT
PSET
PRESET
PUT
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SCREEN
SGN
SIN
SOUND
SPACE$
SPC
SPRITE
SPRITE$
SQR
STEP
STICK
STOP
STR$
STRIG
STRING$
SWAP
TAB
TAN
THEN
TIME
TROFF
TRON
USING
USR
VAL
VARPTR
VDP
VPEEK
VPOKE
WAIT
WIDTH
XOR

G. TECLADOS

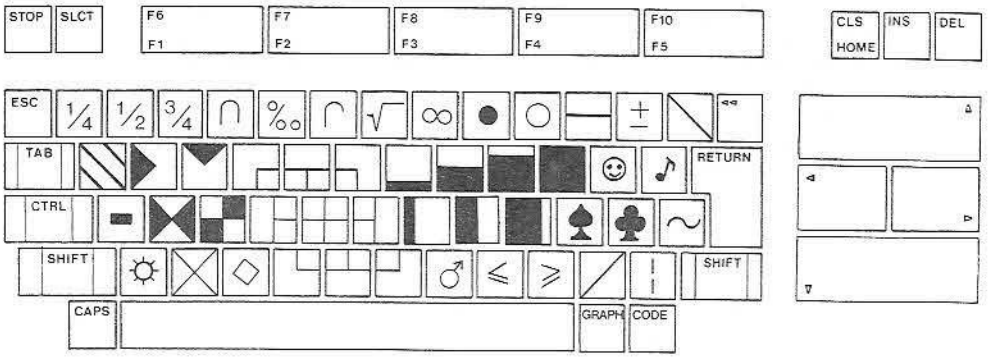
Nesse apêndice temos todos os caracteres disponíveis através do teclado.

- **CARACTERES BÁSICOS**



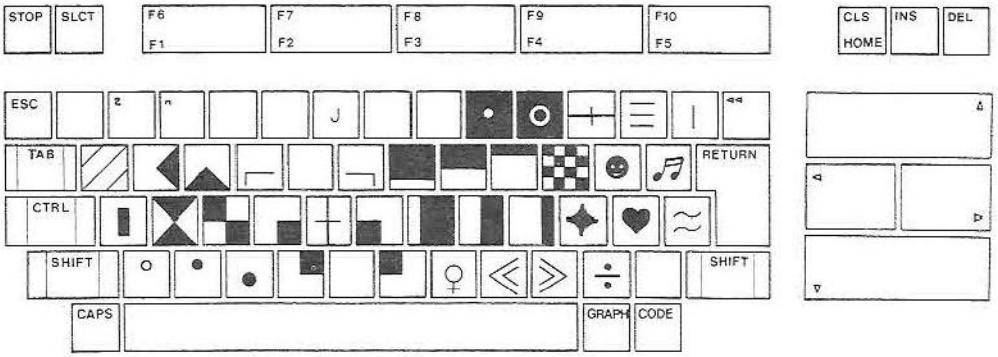
- Normal → Letras minúsculas e símbolos inferiores
- Com SHIFT → Letras maiúsculas e símbolos superiores
- CAPS → Trava em maiúsculo só as letras
- As teclas de acentuação só serão executadas após a pressão da vogal.

- **CARACTERES OBTIDOS PRESSIONANDO A TECLA GRAPH**



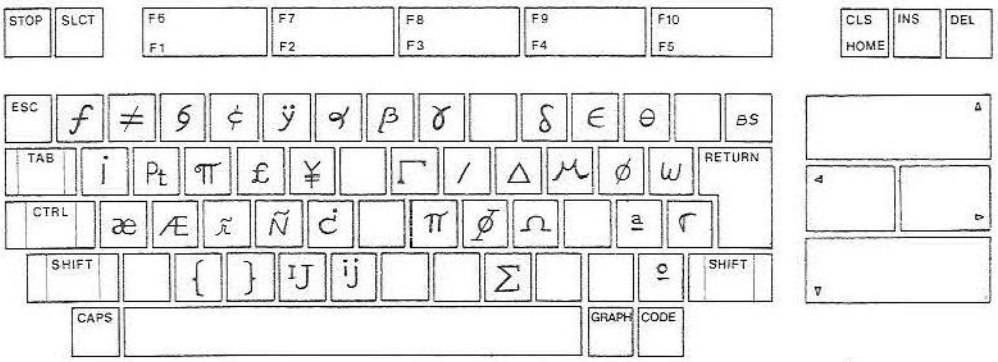
- Com GRAPH pressionado.

- CARACTERES OBTIDOS PRESSIONANDO "SHIFT + GRAPH"



  - Com SHIFT e GRAPH pressionados.

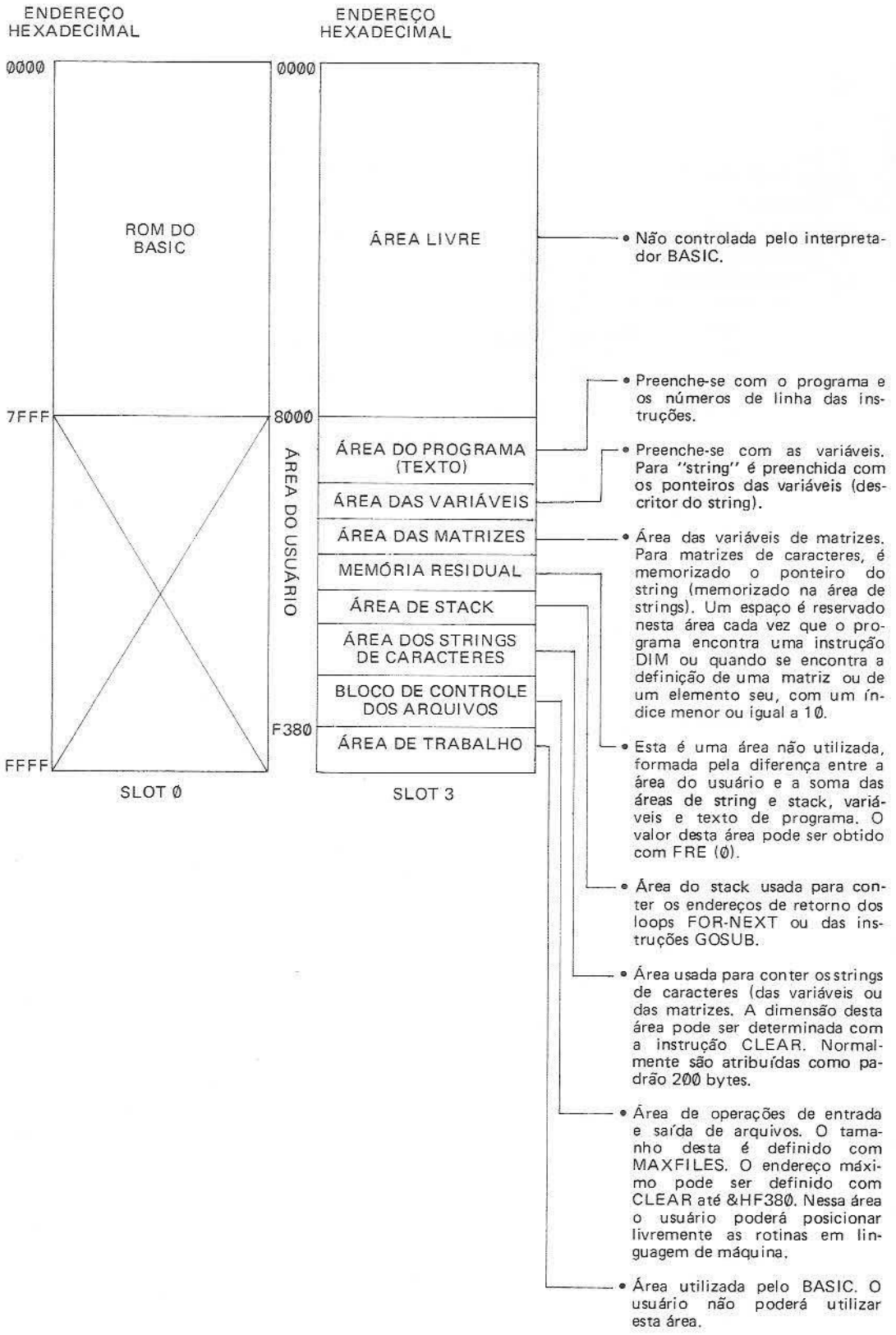
- CARACTERES OBTIDOS PRESSIONANDO "CODE + SHIFT"



  - CODE pressionado
  - SHIFT CODE pressionado.

H. ESPECIFICAÇÕES TÉCNICAS

MAPA DE MEMÓRIA



MAPAS DE I/O

78	8Ø colunas
8Ø	* RS-232C
9Ø	IMPRESSORA
98	VDP
AØ	PSG
A8	PPI
DØ	* FDC

RW		DESCRIÇÃO	NOTAS
W	&H98	Escreve os dados na V-RAM	9128
R		Lê dados da V-RAM	
W	&H99	Comando, coloca endereço	
R		Leitura do estado da V-RAM	
W	&HAØ	Seleciona registros	AY-3-8910A
W	&HA1	Escreve os dados	
R	&HA2	Lê os dados	
W	&HA8	Escreve dados port A	8255
R		Lê dados port A	
W	&HA9	Escreve dados port B	
R		Lê dados port B	
W	&HAA	Escreve dados port C	
R		Lê dados port C	
W	&HAB	Programa o modo	
W	&H90	bØ = strobe	Saída paralela
R		b1 = busy	
W	&H91	Impressão de dados	

Os endereços de I/O de 90 a FF são usados para as funções acima descritas. Os endereços em branco são reservados para o sistema.

Os endereços marcados com asterisco (*) servem para alguns dispositivos opcionais.

R = Read (Leitura)

W = Write (Escrita)

ATRIBUIÇÃO DE BITS DO PIO

PORT	BIT	I/O	NOME DO SINAL	DESCRIÇÃO
A	Ø 1	↑ O ↓	CSØL CSØH	Número de atribuição do slot, do endereço ØØØØ até o 3FFF
	2 3		CS1L CS1H	Número de atribuição do slot, do endereço 4ØØØ até o 7FFF
	4 5		CS2L CS2H	Número de atribuição do slot, do endereço 8ØØØ até o BFFF
	6 7		CS3L CS3H	Número de atribuição do slot, do endereço CØØØ até FFFF
B	Ø 1 7	↑ I ↓		Sinais de retorno do teclado
C	Ø 1 2 3	↑ O ↓	KBØ KB1 KB2 KB3	Sinais de varredura do teclado
	4		CASON	Controle de cassete (LOW-ON)
	5		CASW	Escreve sinais na cassete
	6		CAPS	Acende lâmpada de maiúsculas
	7		SOUND	Emite sons via programa

ATRIBUIÇÃO DE BITS DO PSG

PORT	BIT	I/O	NÚMERO DO PINO DO CONECTOR	SINAL DO JOYSTICK
A	Ø	↑ I ↓	J4–1 pino *1	FWD1
			J4–1 pino *2	FWD2
	1		J3–2 pino *1	BACK1
			J4–2 pino *2	BACK2
	2		J3–3 pino *1	LEFT1
			J4–3 pino *2	LEFT 2
	3		J3–4 pino *1	RIGHT1
			J4–4 pino *2	RIGHT2
	4		J3–6 pino *2	TRGA1
			J4–6 pino *2	TRGA2
	5		J3–7 pino *1	TRGB1
			J4–7 pino *2	TRGB2
	6			"H"/"L" Nível alto/baixo
	7		CSAR (leitura fita cassete)	
B	Ø	↑ O ↓	J3–6 pino *3	
	1		J3–7 pino *3	"H" Nível alto
	2		J4–6 pino *3	
	3		J4–7 pino *3	
	4		J3–8 pino	
	5		J4–8 pino	
	6		Seleção input port A	
	7		KLAMP (Não utilizado)	

*1 Para o joystick 1, usado quando o bit 6 do port B é baixo (LOW)
*2 Para o joystick 2, usado quando o bit 6 do port B é alto (HIGHT)

CONECTORES PARA OS PERIFÉRICOS

DISPOSITIVO	ESPECIFICAÇÕES STANDARD N-45326
Gravador	Conector DIN de 8 pinos (45326)
Joystick	Conector tipo AMP de 9 pinos
Bus do carregador e cartucho	Conector de 50 pinos com passo de 2,54 mm
Impressora	Conector de 14 pinos

INTERFACE PARA GRAVADOR CASSETE

ENTRADA: Ligado ao conector de áudio do gravador cassete.

SAÍDA: Ligado ao conector de microfone do gravador cassete.

FORMA DE SINCRONISMO: Assíncrono.

VELOCIDADE DE TRANSMISSÃO: 12ØØ Bd (Ø para um freqüência de 12ØØ Hz e 1 para uma freqüência de 24ØØ Hz) Standard.

24ØØ Bd (Ø para uma freqüência de 24ØØ Hz e 1 para uma freqüência de 48ØØ Hz) Switch Software.

(A velocidade de 24ØØ Bd é usada só para registrar dados).

MODULAÇÃO: Do tipo FSK.

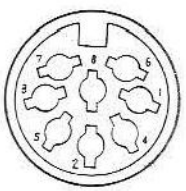
FUNÇÃO REMOTE: Inclusa.

CONECTOR: Tipo DIN com 8 pinos (o oitavo pino pode ser omitido).

SINAIS

PINO	SINAL
1	GND
2	GND
3	GND
4	CMTOUT
5	CMTIN
6	REM +
7	REM –
8	GND

POSIÇÃO FÍSICA DOS PINOS



PORT DE ENTRADA / SAÍDA DO JOYSTICK

CIRCUITO UTILIZADO: AY–3–8910

ENTRADA/SAÍDA: 4 bits de entrada, 16 bits de saída e 2 bits bidirecionais (por port).

LÓGICA: Positiva.

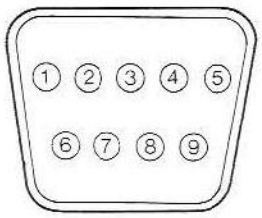
NÍVEL: TTL

CONECTOR: De 9 pinos.

SINAIS

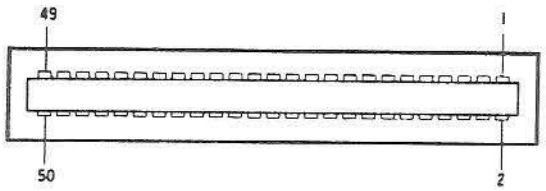
PINO	SINAL
1	FWD
2	BACK
3	LEFT
4	RIGHT
5	+ 5V
6	TRG 1
7	TRG 2
8	OUT
9	GND

POSIÇÃO FÍSICA DOS PINOS



LINHA DE SINAL DO BUS DO CARTUCHO

PINO	DESCRIÇÃO	I/O	PINO	DESCRIÇÃO	I/O
1	CS1	O	2	CS2	O
3	CS12	O	4	SLTSL	O
5	Não utilizado	—	6	RFSH	O
7	WAIT	I	8	INT	I
9	M1	O	10	BUSDIR	I
11	IORQ	O	12	MERQ	O
13	WR	O	14	RD	O
15	RESET	O	16	Não utilizado	—
17	A9	O	18	A15	O
19	A11	O	20	A10	O
21	A7	O	22	A6	O
23	A12	O	24	A8	O
25	A14	O	26	A13	O
27	A1	O	28	A0	O
29	A3	O	30	A2	O
31	A5	O	32	A4	O
33	D1	I/O	34	D0	I/O
35	D3	I/O	36	D2	I/O
37	D5	I/O	38	D4	I/O
39	D7	I/O	40	D6	I/O
41	GND	—	42	CLOCK	O
43	GND	—	44	SW1	—
45	+ 5V	—	46	SW2	—
47	+ 5V	—	48	+ 12V	—
49	SUNDIN	I	50	— 12V	—



LINHA DE SINAL DO BUS DO CARTUCHO

PINO	SINAL	CONTEÚDO
1	CS1	Seleciona o sinal da ROM do endereço 4ØØØ ao 7FFF
2	CS2	Seleciona o sinal da ROM do endereço 8ØØØ ao BFFF
3	CS12	Seleciona o sinal da ROM do endereço 4ØØØ ao BFFF para uma ROM de 256 K bytes
4	SLTSL	Sinal de seleção do slot. Um sinal específico de seleção é adicionado por slot
5	Reservado	Sinal reservado para usos futuros. Não utilizar
6	RFSH	Sinal do ciclo de refresh
7	WAIT	Esperar sinal de chamada da CPU
8	INT	Sinal de chamada de interrupt para a CPU
9	M1	Sinal do ciclo da CPU
10	BUSDIR	Sinal de controle da direção para o buffer do externos São selecionados os carregadores, depois é emitido o nível L dos carregadores exceto a memória no momento de mandar os dados
11	IORQ	Chamada de I/O
12	MERQ	Chamada de memória
13	WR	Tempo de escrita
14	RD	Tempo de leitura
15	RESET	Reset do sistema
16	Reservado	Sinal reservado para usos futuros. Não utilizar
17 ∿ 32	AO ∿ A15	Sinal do bus de endereços
33 ∿ 40	DO ∿ D7	Sinal dos dados
41	GND	Terra
42	CLOCK	Clock da CPU, de 3,579545 MHz
43	GND	Terra
44, 46	SW1, SW2	Usado para proteção na inserção
45,47	+ 5V	+ 5V cc
48	+ 12V	+ 12V cc
49	SUNDIN	Entrada do som (— 5 bdm)
50	— 12V	— 12V cc

INTERFACE DA IMPRESSORA

PADRÃO: Paralela de 9 bits, com sinais BUSY e STROBE

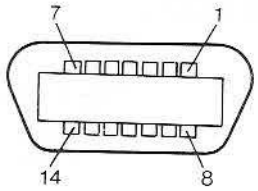
NIVEL: TTL

CONECTOR: De 14 pinos

SINAIS

PINO	SINAL	PINO	SINAL
1	PSTB	8	PDB6
2	PDB0	9	PDB7
3	PDB1	10	NC
4	PDB2	11	BUSY
5	PDB3	12	NC
6	PDB4	13	NC
7	PDB5	14	GND

POSIÇÃO FÍSICA DOS PINOS



I. PROCESSADOR DE VÍDEO (VDP)

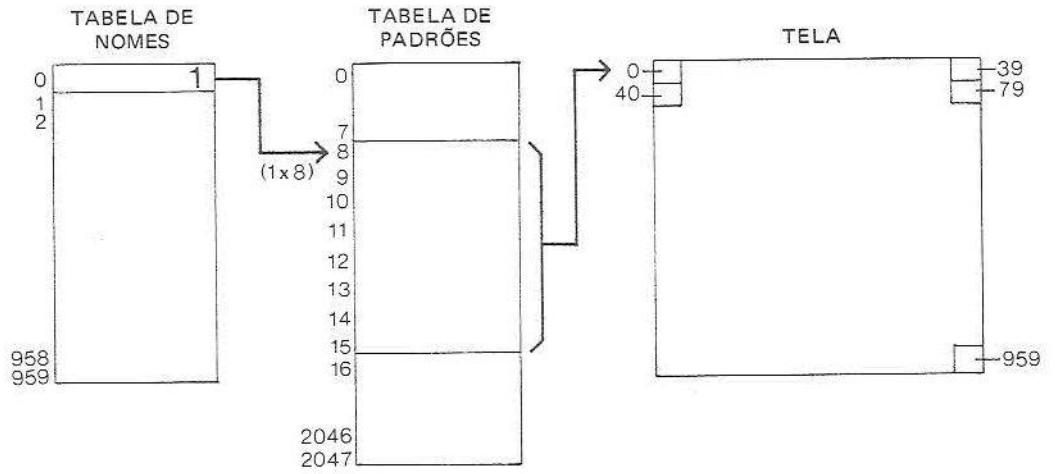
O nosso microcomputador possui um processador de vídeo (VDP) que fornece todas as informações necessárias para produzir uma imagem na tela. Para possibilitar isso, o VDP tem uma memória RAM de 16K.

As informações da RAM de vídeo podem ser chamadas usando a função "VPEEK", e podem ser memorizadas com a instrução "VPOKE".

Todas as instruções de impressão, por exemplo "PRINT", e todas as instruções gráficas, por exemplo "DRAW", são elaboradas automaticamente pelo BASIC e inseridas na memória RAM de vídeo. Tudo aquilo que aparece na tela é registrado na memória RAM de vídeo em tabelas que são utilizadas em relação com o modo da tela.

MODO TEXTO 1 (SCREENØ)

No modo texto 1 a tela é dividida em 24 linhas com 4Ø caracteres cada. Nesse modo existem duas tabelas na memória RAM : uma tabela de nomes e uma tabela de padrões. A tabela de nomes contém todos os códigos dos caracteres e todos os símbolos que comparecem na tela. Na tabela de padrões são reservados 8 bytes por nome para definir como será apresentado o caracter. A posição na tabela de nomes indica também a posição na tela.



A posição na qual se encontram as tabelas na memória RAM de vídeo é indicada pela variável "BASE (n)".

"BASE (Ø)" dá o primeiro endereço da tabela de nomes, enquanto que "BASE (2)" dá o primeiro endereço da tabela de padrões.

Exemplo:

```
1Ø  SCREENØ : WIDTH 4Ø
2Ø  PRINT " BRASIL "
3Ø  PRINT BASE (Ø) , BASE (2)
4Ø  A = VPEEK (BASE (Ø) ) : PRINT A
5Ø  B = A * 8 : FOR I = B TO B + 7
6Ø  C$ = "ØØØØØØØØ" + BIN$ (VPEEK (BASE (2) + I) )
65  PRINT RIGHT$ (C$, 8)
7Ø  NEXT I
RUN
```

No exemplo acima, o primeiro endereço da tabela de nomes e da tabela de padrões, está escrito na linha 3Ø.

Na linha 4Ø está escrito o código do caracter (nome) da letra "B". A seguir é reproduzido em forma binária, o perfil da letra "B".

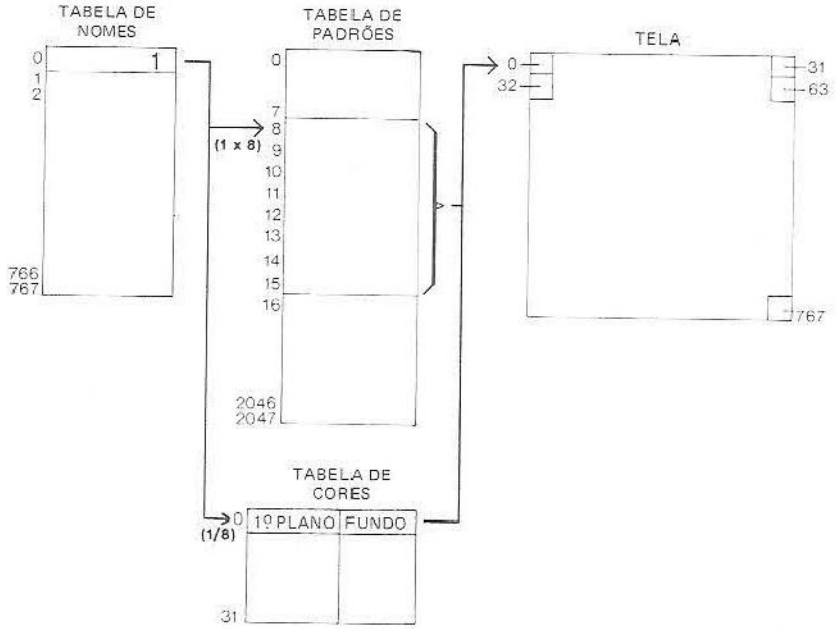
MODO TEXTO 2 (SCREEN 1)

Nesse modo a tela é dividida em 24 linhas de 32 caracteres cada. Para isso, existem 3 tabelas na memória RAM de vídeo: uma tabela de nomes, uma tabela de padrões e uma tabela de cores. As tabelas de nomes e de padrões funcionam da mesma forma que no modo texto 1. A tabela de cores contém 32 combinações das cores de primeiro plano e de fundo.

Os códigos dos caracteres de Ø a 7 na tabela dos nomes usam a primeira combinação da tabela de cores. As seguintes oito entradas usam uma segunda combinação da tabela de cores, e assim por diante.

Uma combinação da tabela de cores é memorizada em um byte subdividido em 2 grupos de 4 bits. Os primeiros 4 bits indicam a cor do primeiro plano, os segundos 4 bits indicam a cor do fundo.

A posição da tabela de nomes na memória RAM de vídeo é registrada na variável BASE (6), a posição da tabela de padrões na variável BASE (7), e a posição da tabela de cores na variável BASE (6).



Tente modificar o programa anterior da seguinte forma:

```
10 SCREEN1,WIDTH32
20 PRINT"brasil"
30 PRINTBASE(5),BASE(6),BASE(7)
40 A=VPEEK(BASE(5)):PRINTA
50 B=A*8:FORI=BTOB+7
60 C$="00000000"+BIN$(VPEEK(BASE(7)+I))
65 PRINTRIGHT$(C$,8)
70 NEXTI
80 D=VPEEK(BASE(6))
90 PRINTUSING"##";D/16
100 PRINTUSING"##";DMOD16
```

Nesse exemplo são escritos os primeiros endereços da tabela de nomes, da tabela de padrões e da tabela de cores, seguidos da forma do caracter e do nome do caracter.

MODO GRÁFICO 1 (SCREEN 2)

Nesse modo a tela é dividida em 24 linhas x 32 caracteres por linha. Esse modo da tela compreende também uma tabela de nomes que por sua vez indica também a posição na tela. O número na tabela de nomes indica onde é possível achar o padrão (desenho) e a cor do caracter que deve ser escrito. Para cada nome na tabela de nomes é possível definir um desenho ou padrão na tabela de padrões.

Para isso, a tabela de padrões tem um comprimento de 6144 bytes (768 nomes x 8 bytes). Para cada byte da tabela de padrões pode ser definido uma cor de fundo e uma de primeiro plano na tabela de cores.

Por esse motivo a tabela de cores tem o mesmo tamanho que a tabela de padrões. A posição da tabela de nomes na memória RAM de vídeo é registrada na variável BASE (1Ø), e a posição da tabela de padrões na "BASE (2).

A posição da tabela de cores é registrada na variável BASE (11).

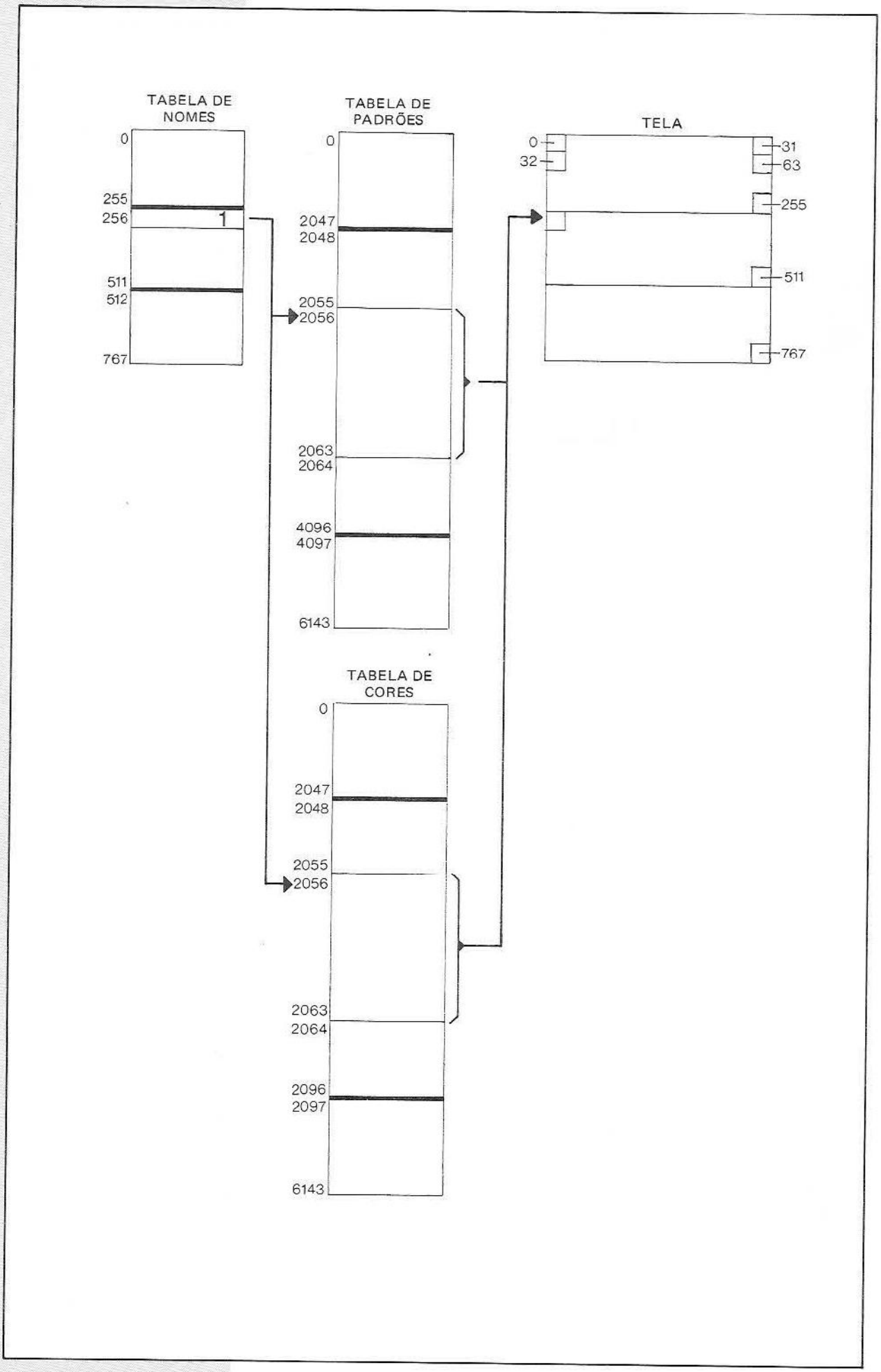


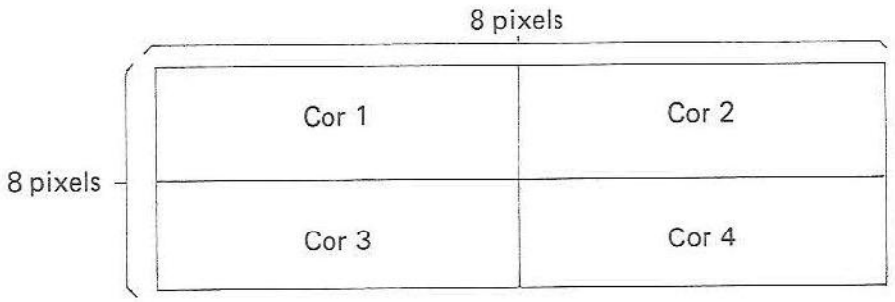
TABELA DE NOMES
0
255
256
1
511
512
767
TABELA DE PADRÕES
0
2047
2048
2055
2056
2063
2064
4096
4097
6143
TELA
0
32
31
63
255
511
767
TABELA DE CORES
0
2047
2048
2055
2056
2063
2064
2096
2097
6143

MODO GRÁFICO 2 (SCREEN 3)

Nesse modo a tela é dividida em 24 linhas de 32 caracteres cada. Cada posição é composta de 4 quadradinhos, com uma dimensão de 4 x 4 pontos de imagem cada.

Cada quadrado contém uma cor. Nesse modo da tela usa-se uma tabela de nomes que indica também a posição na tela.

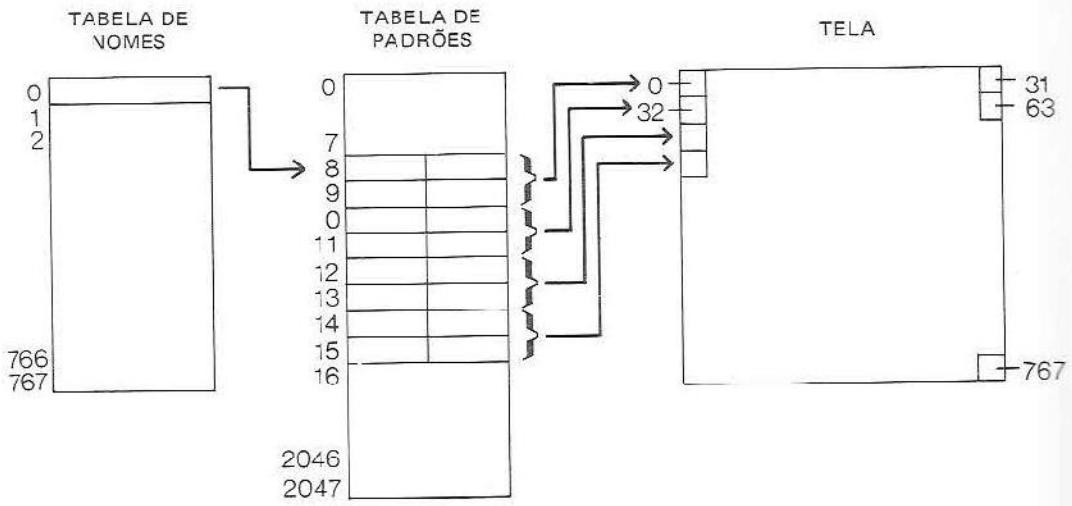
O número na tabela de nomes indica onde encontrar na tabela de padrões, as cores das quatro posições.



Cada byte na tabela de cores é dividido em 2 grupos de 4 bits e cada combinação de 4 bits indica um número de cor.

Isso significa que cada nome na tabela de nomes chama 8 bytes na tabela de padrões.

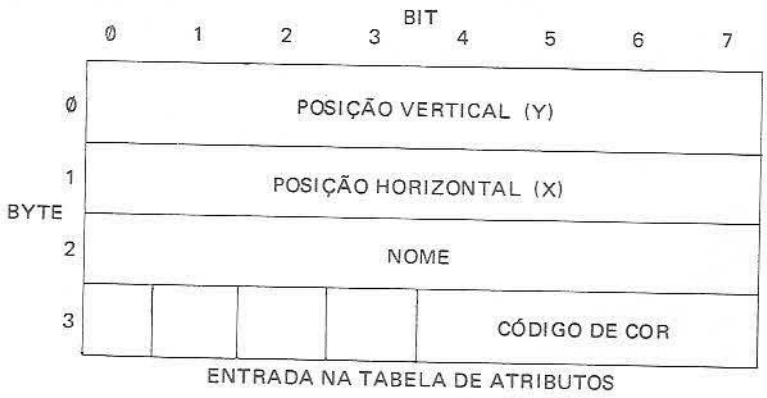
A posição da tabela de nomes é registrada na variável BASE (15). A posição da tabela de padrões é registrada na variável BASE (17).



- **SPRITES**

Os sub-blocos da RAM de vídeo que definem os sprites são a tabela de atributos dos sprites e a tabela geradora dos padrões de sprites.

A tabela de atributos dos sprites indica onde o sprite é localizado na tela enquanto que a tabela geradora dos padrões indica a forma do sprite.



Na tabela de atributos é possível indicar um máximo de 32 posições.

Cada entrada na tabela compõe-se de 4 bytes (vide desenho acima).

No byte da posição vertical é possível inserir um valor de Ø a 255. Esse número determina a linha na tela fornecendo as seguintes informações:

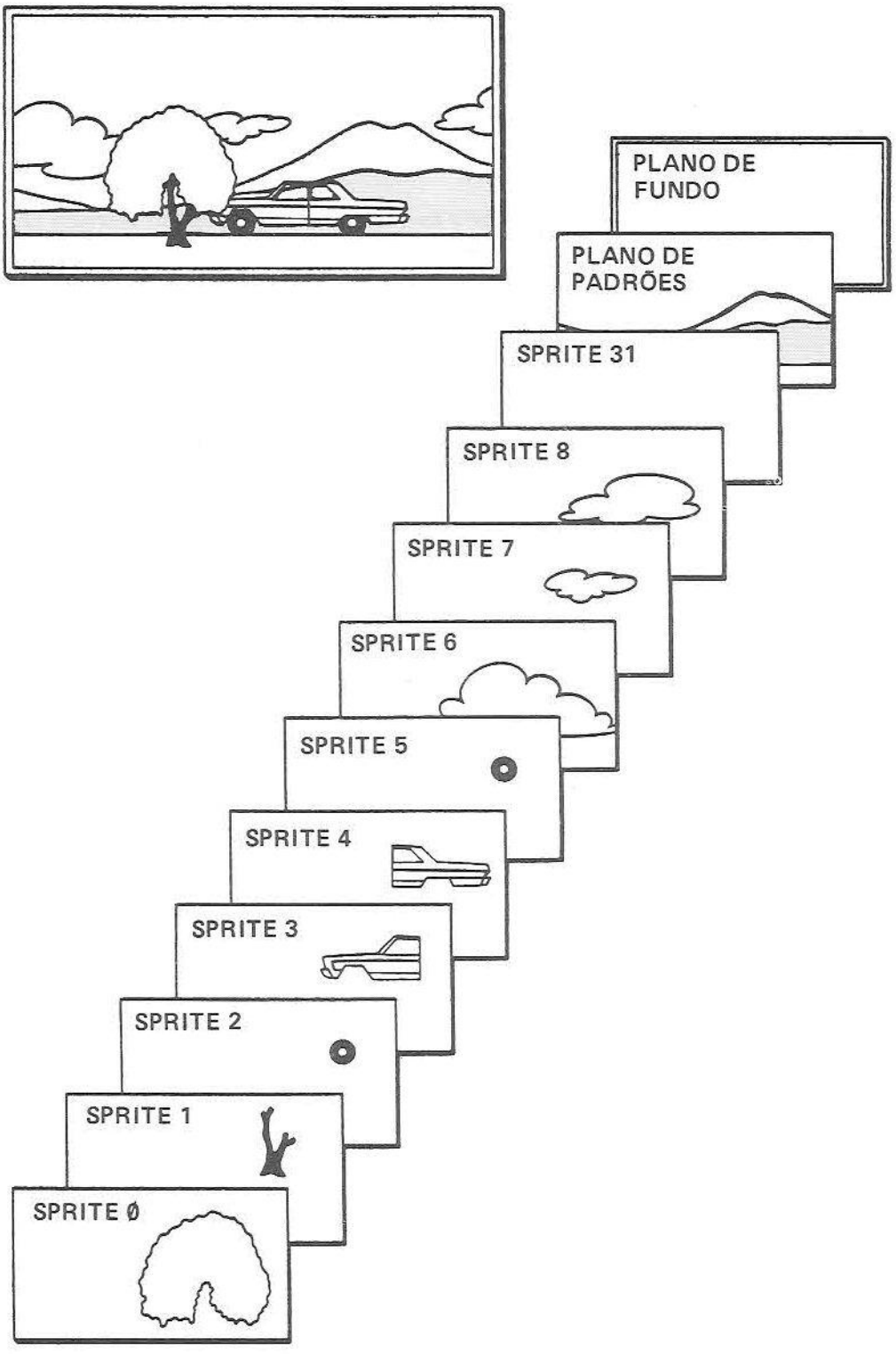
0 – 191	é a linha da tela
208	significa que o sprite não aparece mais na tela
224 – 255	representa as linhas da tela compreendidas entre –31 e –1

No byte da posição horizontal pode ser inserido um valor entre Ø e 255. Esse número determina a coluna na tela. Se o bit 7 do byte de cor é igual a 1, a posição da coluna do sprite equivale à posição X menos 32.

Se a posição X é igual a Ø e o bit 7 do byte de cor é igual a 1, o sprite será inserido na coluna –32. Assim é possível inserir um sprite antes da primeira coluna.

As posições X e Y indicadas aplicam-se ao ponto de imagem correspondente ao extremo superior esquerdo do sprite.

O sprite que foi inserido em primeiro lugar na tabela de atributos de sprites, é o que tem a prioridade mais elevada (sprite Ø).


PLANO DE FUNDO
PLANO DE PADRÕES
SPRITE 31
SPRITE 8
SPRITE 7
SPRITE 6
SPRITE 5
SPRITE 4
SPRITE 3
SPRITE 2
SPRITE 1
SPRITE Ø

Na ilustração da página precedente vemos um carro localizado numa paisagem.

A árvore está formada pelos sprites Ø e 1.

O carro está formado pelos sprites 2 a 5.

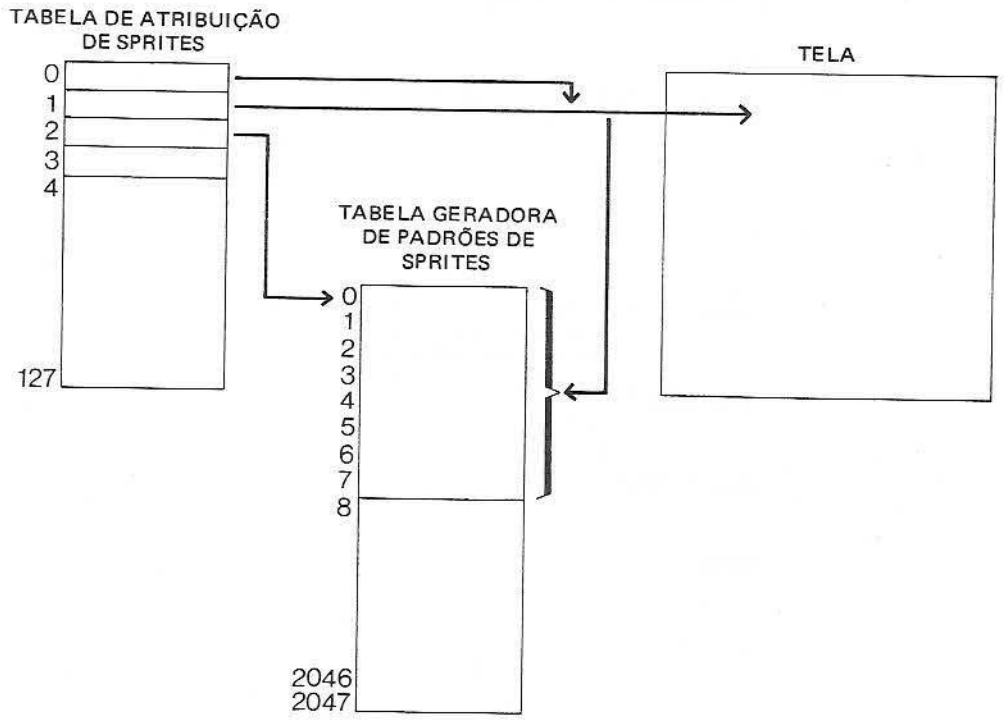
As nuvens estão formadas pelos sprites 6, 7 e 8.

Dado que os sprites Ø e 1 tem prioridade em relação aos sprites 2 e 5, a árvore aparece na frente do carro.

O nome na entrada do sprite na tabela de atributos, é um número que indica a localização do padrão na tabela geradora de padrões.

O primeiro endereço da tabela de atributos na memória RAM de vídeo é registrado na variável BASE (8) no modo texto 2 na BASE (13) no modo gráfico 1 e na BASE (18) no modo gráfico 2.

O primeiro endereço da tabela geradora de padrões é registrado na variável BASE (9) no modo texto 2, na BASE (14) no modo gráfico 1 e na BASE (19) no modo gráfico 2.



Tente introduzir o seguinte exemplo:

```
10 SCREEN1,WIDTH29
20 DATA16,56,68,254,16,16,16,16
30 FORI=0TO7
40 READA:VPOKE(BASE(9)+I),A
50 NEXTI
60 PRINT:PRINT:PRINT
70 PRINT"UM SPRITE NO MODO TEXTO 2"
80 VPOKE(BASE(8)+0),40
90 VPOKE(BASE(8)+2),0
100 VPOKE(BASE(8)+3),1
110 FORI=0TO255
120 VPOKE(BASE(8)+1),I
130 NEXTI
140 GOTO110
```

Esse exemplo demonstra como pode ser colocada a informação referente aos sprites diretamente na RAM de vídeo através do comando VPOKE.

- **REGISTROS**

O processador de vídeo possui 9 registros: do número Ø até o 8.

Através do BASIC o conteúdo dos registros é carregado na variável "VDP (n)", onde n é um número compreendido entre Ø e 8. Com exceção do registro 8, essas variáveis podem ser modificadas. Os significados dos registros são os seguintes:

Bit	0	1	2	3	4	5	6	7
Registro 0	0	0	0	0	0	0	A	D
Registro 1	1	0	E	B	C	R	S	M
Registro 2	0	0	0	0	TABELA DE NOMES			
Registro 3	TABELA DE CORES							
Registro 4	0	0	0	0	0	TABELA DE PADRÕES		
Registro 5	0	TABELA DE ATRIBUTOS DOS SPRITES						
Registro 6	0	0	0	0	0	TABELA DE PADRÕES DOS SPRITES		
Registro 7	COR TEXTO 1				COR TEXTO 2			
Registro 8	T	N	V	5º SPRITE				

A B C

Ø Ø Ø = Modo texto 2
1 Ø Ø = Modo gráfico 1
Ø Ø 1 = Modo gráfico 2
Ø 1 Ø = Modo texto 1

D = Ø desativa input externo para o VDP
D = 1 habilita input externo para o VDP

E = Ø desativa interrupção para o VDP
E = 1 ativa interrupção para o VDP

R = estado reservado

S = Ø formato de sprites igual a 8 x 8
S = 1 formato de sprites igual a 16 x 16

M = Ø sprites não aumentados
M = 1 sprites aumentados

TABELA DE NOMES = primeiro endereço da tabela de nomes na RAM de vídeo.
TABELA DE CORES = primeiro endereço da tabela de cores na RAM de vídeo.
TABELA DE PADRÕES = primeiro endereço da tabela de padrões na RAM de vídeo.
TABELA DE ATRIBUTOS DE SPRITES = primeiro endereço da tabela de atributos de sprites.
TABELA DE PADRÕES = primeiro endereço na tabela de padrões dos sprites na RAM de vídeo.
COR TEXTO 1 = cor do primeiro plano no modo texto 1.
COR TEXTO 2 = cor do fundo no modo texto 1.

T O flag de estado T no registro 8 é colocado em 1 no fim da varredura da imagem até a última linha do vídeo.

É colocado a Ø após a leitura do estado Ø quando o processador de vídeo recebeu um "reset" externo.

N O flag de estado N no registro 8 é colocado em 1 se dois ou mais sprites entram em colisão. A colisão verifica-se quando dois sprites quaisquer na tela tem pelo menos 1 pixel sobreposto. O flag N é gerado logo após um dos registros Ø a 8 seja lido ou quando o processador de vídeo recebe um "reset" externo.

O registro 8 deverá ser lido logo após ligar o micro, para ter certeza que o flag de colisões é gerado. O processador controla a coincidência entre pixels dos sprites, durante a geração dos mesmos, independentemente da sua posição na tela. Essa verificação realiza-se cada 1/50 seg. com TMS9128A.

Conseqüentemente, sobrepondo-se os sprites em mais de uma posição de pixel durante esse intervalo, é possível que os sprites tenham maior número de pixel coincidentes, ou estejam totalmente sobrepostos no momento em que o processador de vídeo controla a coincidência.

V e 5º sprite O flag de estado V no registro 8 é colocado em 1 cada vez que 5 ou mais sprites coexistem numa linha horizontal (linhas entre Ø e 192) e o flag do bloco é igual a Ø. O flag de estado V e zerado após o registro 8 ter sido lido ou após o processador de vídeo ter recebido um "reset" externo.

O número do quinto sprite é inserido no 5º bit inferior do registro 8 quando o flag V é colocado no valor lógico 1 e é válido cada vez que o flag V é igual a 1.

As tabelas na memória RAM de vídeo podem ser inseridas em uma posição diferente do programa do usuário, daquela atribuída pelo BASIC, respeitando-se as seguintes regras:

1. A tabela de nomes deve começar no endereço Ø ou em um endereço correspondente a um múltiplo de 1Ø24.

2. A tabela de cores deve começar no endereço Ø ou em um endereço correspondente a um múltiplo de 64.

3. A tabela de padrões deve começar no endereço Ø ou em um endereço correspondente a um múltiplo de 2Ø48.

4. A tabela de atributos dos sprites deve começar no endereço Ø ou em um endereço correspondente a um múltiplo de 128.

5. A tabela de padrões de sprites deve começar no endereço Ø ou em um endereço correspondente a um múltiplo de 2Ø48.

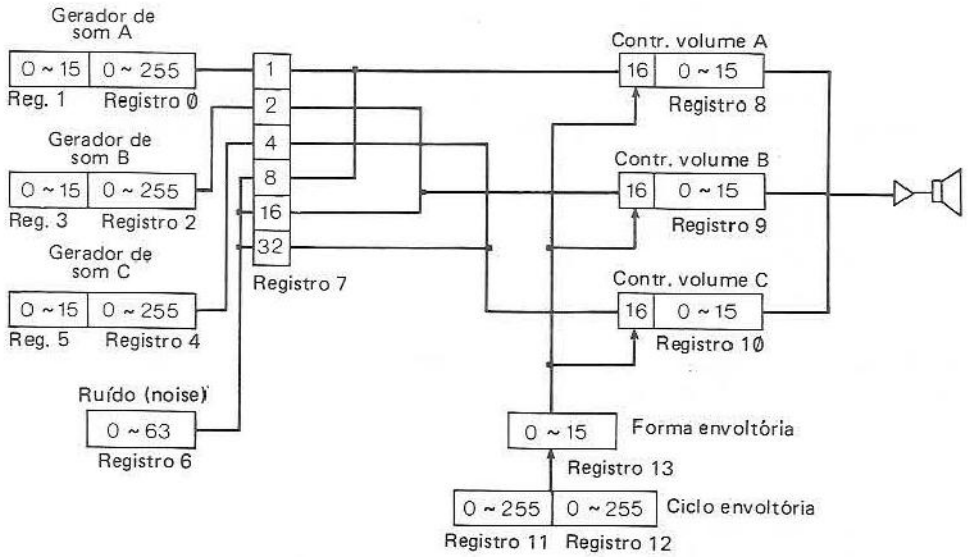
J. GERADOR DE SOM PROGRAMÁVEL

O computador possui um gerador de som programável (GSP) que produz o som. Para isso, o GSP possui 16 registros aos que pode-se atribuir diretamente um valor, usando a instrução "SOUND" do BASIC.

Contrariamente à instrução "PLAY" o som continua até que o registro interessado é voltado a Ø através de uma outra instrução "SOUND".

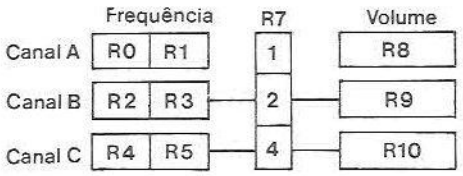
As funções dos registros estão resumidas na seguinte tabela:

Registro \ Bit		B7	B6	B5	B4	B3	B2	B1	BØ
R0	Canal A	8BIT					Tono FTA		
R1	Período do tono						4BIT CTA		
R2	Canal B		8BIT				Tono FTB		
R3	Período do tono						4BIT CTB		
R4	Canal C	8BIT					Tono FTC		
R5	Período do tono						4BIT CTC		
R6	Período do rumor					5BIT NP			
R7	Posição do canal			NOISE			TONE		
				C	B	A	C	B	A
R8	Volume Canal A				M	L3	L2	L1	L0
R9	Volume canal B				M	L3	L2	L1	L0
R10	Volume canal C				M	L3	L2	L1	L0
R11	Período de envoltória	8BIT Tono							
R12		8BIT Tono							
R13	Ciclo/forma envoltória					E3	E2	E1	E0



• GERAÇÃO DE SOM

Como o gerador de som e o controlador de volume são separados para cada um dos três canais, eles podem ser manipulados independentemente um do outro.



Cada canal recebe um input quando o seu bit de pertence é igual a Ø, e ignora-o quando esse bit é igual a 1.

O volume é anulado quando o valor é Ø, o volume é máximo quando o valor é 15.

O valor FT (FINE TONE) e CT (COARSE TONE) introduzidos nos registros RØ e R1 (para o canal A), podem ser calculados usando as seguintes fórmulas:

$$TP = \frac{f\ clock}{16 \times ft} \qquad CT + \frac{FT}{256} = \frac{TP}{256}$$

f clock = 1,78977 MHz
ft = freqüência de saída
FT = valor de FTA (RØ) (de Ø a 255 em decimal)
CT = valor de CTA (R1) (de Ø a 15 em decimal)

Executando o seguinte programa; podemos gerar através do canal A todos os tipos de sons:

```
1Ø SOUND 7, 56
2Ø FOR I = Ø TO 15
3Ø FOR J = Ø TO 255
4Ø SOUND Ø, I : SOUND 1, J
5Ø FOR K = 1 TO 15
6Ø SOUND 8, K
7Ø NEXT K
8Ø NEXT J
9Ø NEXT I
```

Mudando no exemplo anterior as linhas 4Ø e 6Ø, se introduzirmos os dados nos outros registros, os sons serão gerados através do canal B e do canal C.

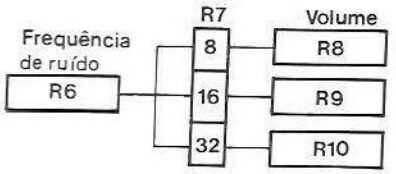
```
4Ø SOUND 1, I : SOUND 3, J
6Ø SOUND 9, K                    Canal B

4Ø SOUND 4, I : SOUND 5, J
6Ø SOUND 1Ø, K                   Canal C
```

Executando o seguinte programa, os sons serão gerados simultaneamente através dos canais A, B e C.

```
1Ø SOUND Ø, Ø : SOUND 1, 1
2Ø SOUND 2, 128 : SOUND 3, 1
3Ø SOUND 4, 128 : SOUND 5, 0
4Ø SOUND 8, 6 : SOUND 9, 9 : SOUND 10, 12
5Ø SOUND 7, 56
```

- GERAÇÃO DE RUÍDOS (NOISE)

Como só existe um gerador de ruído, terá quer ser usado em conjunto com os canais.



O valor NP que será introduzido no registro de indicação de freqüência de ruído R6, é calculado através da fórmula:

$$NP = \frac{f\ clock}{16 \times fN}$$

onde,

f clock = 1,78977 MHZ
fN = freqüência do rumor (saída) (Hz)
NP = valor que será introduzido no NP (R6) (Ø ∿ 31 no 1Ø° passo)

O seguinte programa também gera todos os ruídos através do canal A.

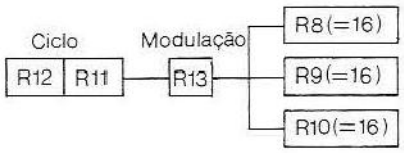
```
1Ø SOUND 7,7
2Ø FOR I = Ø TO 31
3Ø SOUND 6,I
4Ø FOR J = 1 TO 15
5Ø SOUND 8,J
6Ø NEXT J
7Ø NEXT I
```

Se mudarmos a linha 5Ø, o ruído será gerado pelo canal B e pelo canal C. O som ouvido será o mesmo.

```
5Ø SOUND 9,J ... gera pelo canal B
5Ø SOUND 10,J ... gera pelo canal C
```

- GERAÇÃO DE UMA ENVOLTÓRIA

Colocando o valor 16 nos registros 8, 9 ou 1Ø é ativado o modo envoltória, no qual os volumes são variados com o transcorrer do tempo. Nesse momento, posiciona-se no registro 13, o formato da envoltória e nos registros 11 e 12 o período da envoltória.



O ciclo da envoltória pode ser calculado como segue:

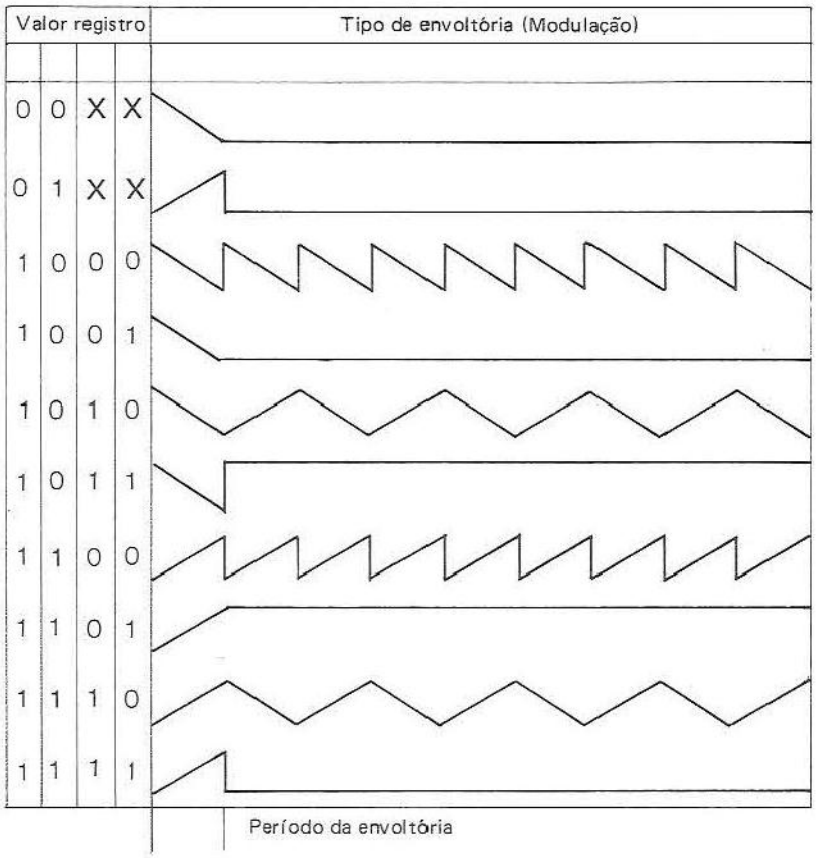
$$EP = \frac{f\ clock}{256\ fE} \qquad CT + \frac{FT}{256} = \frac{EP}{256}$$

onde

f clock	=	1,78977 MHz
EP	=	freqüência (Hz) com a qual os sons são amplificados ou atenuados
FT	=	valor a colocar em FT (R11) (Ø ∿ 255 decimal)
CT	=	valor a colocar em CT (R12) (Ø ∿ 255 decimal)

O formato da envoltória é indicado com o valor de EØ e E3 do registro 13. Indicamos na ilustração, como a forma da onda seria com os valores de EØ à E3. O X dentro da ilustração pode ser Ø ou 1.

SAÍDA DO GERADOR DE ENVOLTÓRIA



Vejamos um programa que serve para gerar um som de explosão, do tipo produzido por um helicóptero, usando o registro 8.

```
1Ø  FOR I = Ø TO 13
2Ø  READ A : SOUND I , A
3Ø  NEXT
4Ø  DATA 2Ø , Ø , 3Ø , Ø , Ø , 9 , Ø
5Ø  DATA 48 , 16 , 4 , 6 , 1ØØ , 2 , 12
```

CERTIFICADO DE GARANTIA

A EPCOM Equipamentos Eletrônicos da Amazônia Ltda., nos limites fixados neste certificado, assegura, como fabricante, ao comprador-consumidor o equipamento aqui identificado, garantia contra qualquer defeito do material ou de fabricação apresentado no prazo de 180 dias, contado a partir da data da emissão da nota fiscal de venda.

Limita-se à responsabilidade da EPCOM o reparo ou substituição, a seu critério, de peças e componentes de sua fabricação, desde que seus técnicos credenciados constatem falhas em condições normais de uso.

A presente garantia ficará sem efeito, se o equipamento sofrer qualquer dano provocado por acidente, agentes da natureza, uso em desacordo com o manual de instruções, ter sido ligado à rede elétrica imprópria ou sujeita a flutuação excessivas; ou ainda no caso de apresentar sinal de violação de seu selo de garantia, ajuste ou conserto por pessoa não autorizada, bem como por defeito oriundo de caso fortuito ou força maior.

EPCOM Equipamentos Eletrônicos da Amazônia Ltda. obriga-se a prestar os serviços, tanto gratuitos como remunerados, exclusivamente nas localidades que mantenha oficinas de serviços próprias ou devidamente autorizadas.

O comprador-consumidor residente em outra localidade será responsável pelas despesas e riscos de transportes, de ida e volta, do equipamento à oficina mais próxima da EPCOM Equipamentos Eletrônicos da Amazônia Ltda. Quando o equipamento for transferido no período de garantia, esta ficará cedida de pleno direito, continuando em vigor até a expiração de seu prazo, contado da data da aquisição pelo primeiro comprador-consumidor.

A presente garantia somente será válida se devidamente preenchida, sem rasuras ou modificações, juntamente com a nota fiscal de compra.

CERTIFICADO DE GARANTIA

(Deve ser preenchido pelo revendedor no ato da venda).

EPCOM Equipamentos Eletrônicos da Amazônia Ltda., dá garantia a este equipamento.

Modelo e nº de Fabricação

Adquirido pelo Sr.: .....

Endereço: .....

Nº ..... Bairro: ..... CEP .....

Cidade: ..... Estado: .....

Nos termos contidos neste certificado de garantia.

Revendedor:

.....

Carimbo e Assinatura

PRODUZIDO PELA EPCOM
EQUIPAMENTOS ELETRÔNICOS DA AMAZÔNIA LTDA.
AV. BURITI, 3650 - BLOCO A
DISTRITO INDUSTRIAL - MANAUS/AM
C.G.C. 04.155.123/0001-52 - INDÚSTRIA BRASILEIRA