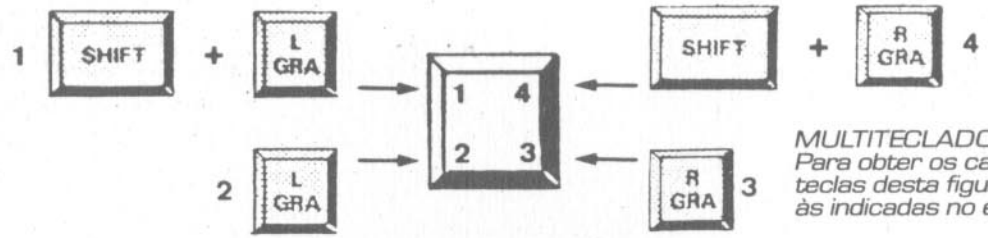
gradiente

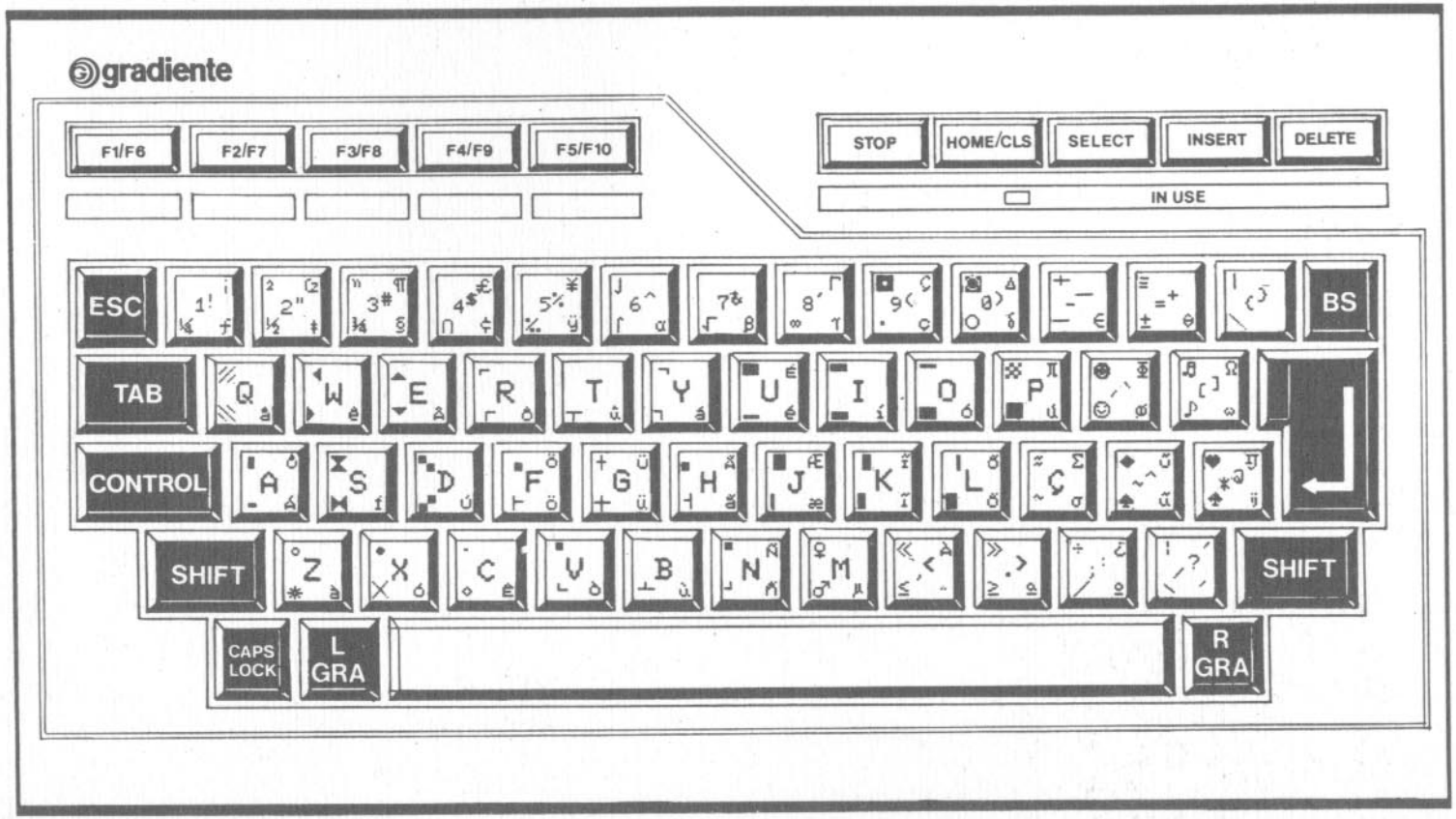
EXPERT Plus

MANUAL DE INSTRUÇÕES

gradiente



MULTITECLADO DO EXPERT DD PLUS
Para obter os caracteres especiais desenhados nos 4 cantos das teclas desta figura, pressione a tecla desejada simultaneamente às indicadas no esquema ao lado.



Na figura abaixo temos o "mapa" do teclado do Expert DD Plus para auxiliar o leitor na digitação dos caracteres especiais.

EXPERT Plus

Manual de Instruções

Editora Aleph

gradiente

EXPEDIENTE:

Coordenação Editorial: Pierluigi Piazzi

Coordenação Didática: Betty Fromer Piazzi

Redação didática: Pierluigi Piazzi
Mathias August Gruber

Editoração: Henrique de Figueredo Luz

Arte: Ana Lúcia Antico

Ilustrações: Nicoletti

ALEPH Publicações e Assessoria Pedagógica Ltda.
Av.Dr.Luis Migliano 1110 c/301
CEP 05711 São Paulo – SP
Tel (011) 843-3202
Caixa Postal 20.707 – CEP 01498

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

Piazzi, Pierluigi, 1943-
Expert PLUS : manual de instruções / Pierluigi Piazzi. -- São Paulo : Aleph, 1989.

Bibliografia.

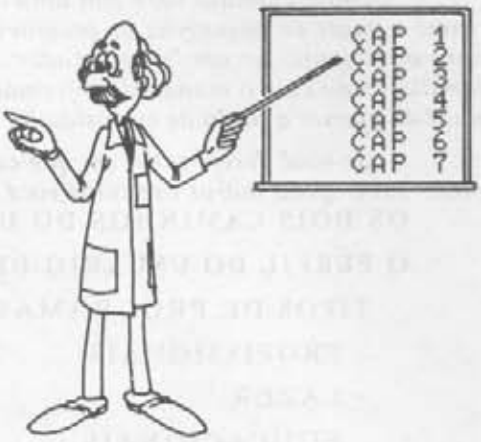
1. Expert PLUS (Computador) 2. Expert PLUS (Computador) - Programação I. Título.

CDD-001.64
89-1917 -001.642

Índices para catálogo sistemático:

1. Expert PLUS : Computadores : Processamento de dados 001.64
2. Expert PLUS : Computadores : Programação : Processamento de dados 001.642

SUMÁRIO



NÃO DEIXE DE LER ESSE TEXTO:

O Expert Plus é um produto de sofisticada tecnologia, fabricado segundo as normas de qualidade Gradiente, obedecendo às especificações do padrão MSX.

Para operá-lo não é necessário que o usuário conheça programação ou informática.

Existem, porém informações úteis contidas neste livro, cujo objetivo é permitir ao usuário do Expert Plus tirar o máximo proveito do seu equipamento.

Veja, a seguir, uma breve descrição desse livro para que você possa trilhar o caminho mais rápido para atender às suas necessidades.

É muito comum, por exemplo, se ver um usuário iniciante gastar horas e horas para aprender a programar seu micro e, ao cabo de muito esforço, decobrir que existem, no mercado, programas prontos que fazem exatamente aquilo que ele queria!

A situação simétrica também pode ocorrer: é a do usuário que quer usar seu micro numa tarefa muito específica e não encontra programas prontos que o atendam. Esse usuário, então, ou utiliza os serviços de terceiros ou aprende a programar para desenvolver seu próprio "software".

Vejamos, então, onde buscar as informações de que você precisa:

A ORGANIZAÇÃO DESSE LIVRO

Para esclarecer quaisquer dúvidas sobre o conteúdo desse livro e obter mais informações o padrão MSX, escreva para a Editora Aleph – Depto de Assistência ao leitor - C.P.-20707 –CEP-01498 São Paulo- SP.

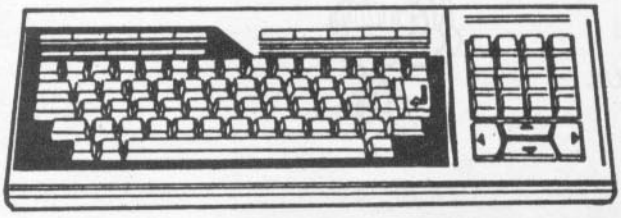
COMO INSTALAR SEU EXPERT PLUS

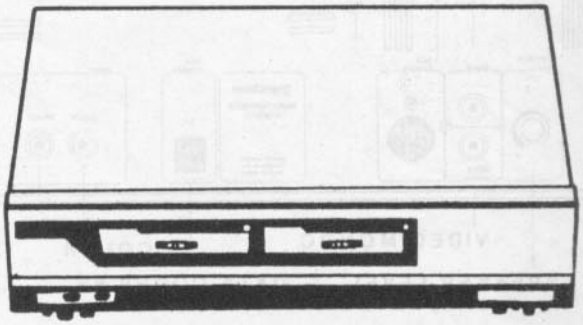
1

Antes de instalar o micro você deve conhecer cada componente e cada uma de suas partes. Vamos identificar a parte principal do seu sistema **EXPERT PLUS**: o *teclado* e o *console*.

fig 1.1 — O teclado e o console do EXPERT PLUS

TECLADO





CONSOLE

O teclado é o principal meio para que você dê ordens ao computador. Ele tem ligado a si um longo cabo que deverá ser conectado ao console. No console é que estão o "*coração*" e o "*cérebro*" do **EXPERT PLUS** e, se isso lhe ajudar, você pode considerar o teclado como sendo seus "*ouvidos*" ou seus "*olhos*". Levado essa analogia adiante, o vídeo (T.V. ou MONITOR) seria sua "*boca*".

Vamos reconhecer o console. Observe atentamente as figuras 1.2 e 1.3, acompanhando no texto a descrição de seus controles e encaixes.

fig 1.2 – Vista frontal do EXPERT PLUS

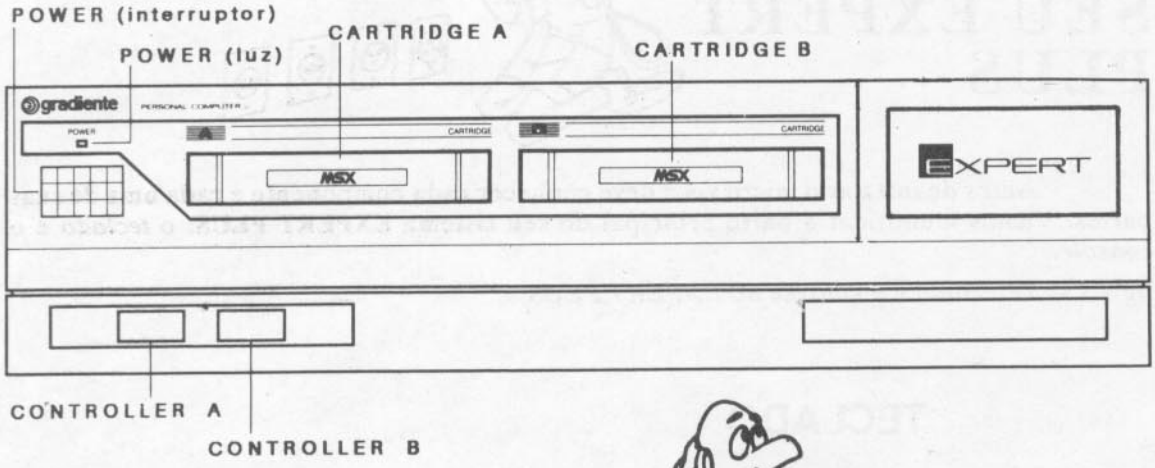
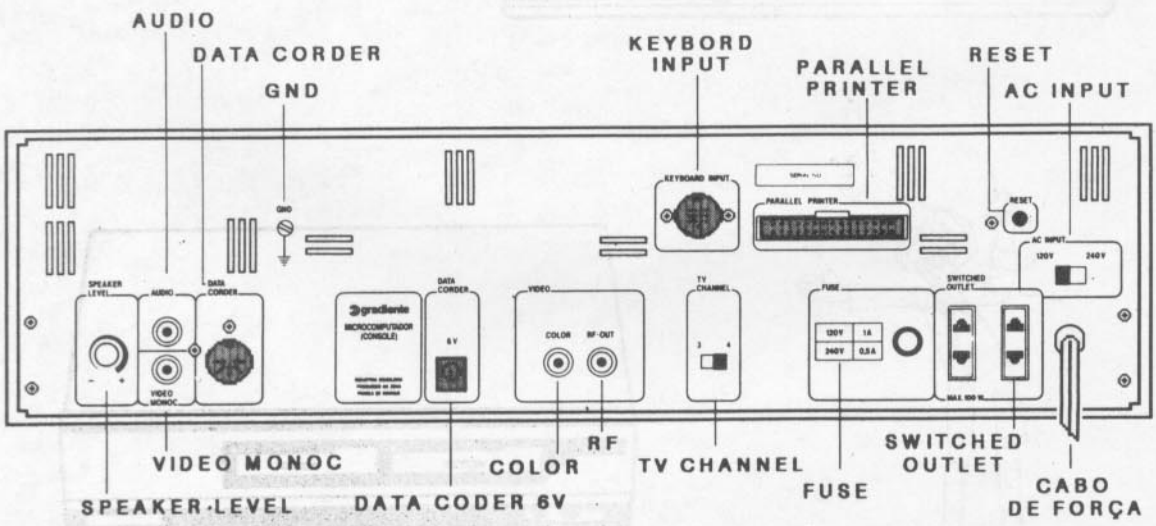


fig 1.3 – Vista traseira do EXPERT PLUS



PARTE FRONTAL

CARTRIDGE – Encaixes onde são conectados os cartuchos (memória pré-gravada), expansões ou interfaces de periféricos. O **A** está ligado no *SLOT 1* e o **B** ao *SLOT 2*..

POWER (interruptor) – Interruptor de pressão para ligar ou desligar o aparelho.

POWER (lâmpada) – Indicador luminoso vermelho que acende quando o **EXPERT PLUS** está ligado.

CONTROLLER – Encaixe de *JOYSTICKS* (alavancas de contrôle manual) A e B, ou *"mouse"*.

PARTE POSTERIOR

SPEAKER LEVEL – Contrôle de volume de som do altofalante interno do console.

AUDIO – Terminal de saída para audio (para ligação com amplificador de som ou AUDIO IN de TV ou vídeo-cassete).

VIDEO MONOC. – Terminal para ligação a monitor monocromático.

DATA CORDER – Terminal para ligação dos cabos que vão ao DATACORDER (ou gravador cassete).

GND – Ligação para fio-terra.

KEYBORD INPUT – Encaixe para conexão do cabo do teclado.

PARALLEL PRINTER – Encaixe para conexão de uma impressora paralela com conector padrão *CENTRONICS*.

DATA CODER 6V – Plug de alimentação elétrica DC 6V para o DATACORDER ou gravador que não tenha fonte de alimentação própria.

COLOR – Terminal para ligação a monitor de vídeo colorido PAL–M ou para conectar ao VIDEO IN de uma TV ou vídeo cassete.

RF – Terminal para ligação à entrada de antena (canal 3 ou 4) de uma TV P&B ou a cores padrão PAL–M (Brasil).

TV CHANNEL – Permite selecionar o canal 3 ou 4 de saída VHF. Utilize um canal que não tenha emissora em sua cidade, se possível.

RESET – Botão que permite "ressetar" o micro, ou seja, simular um desligamento e religamento sem que seja necessário pressionar POWER.

FUSE – Encaixe para fusível (veja texto a seguir)

SWITCHED OUTLET – Saída para ligação de cabo de força de outros equipamentos de baixa corrente. Ao desligar o micro a força nessas tomadas será automaticamente cortada. O consumo dos equipamentos pode ser, no máximo, de 100 W.

AC INPUT – Seletor de voltagem para rêde elétrica (VERIFIQUE A VOLTAGEM LOCAL!).

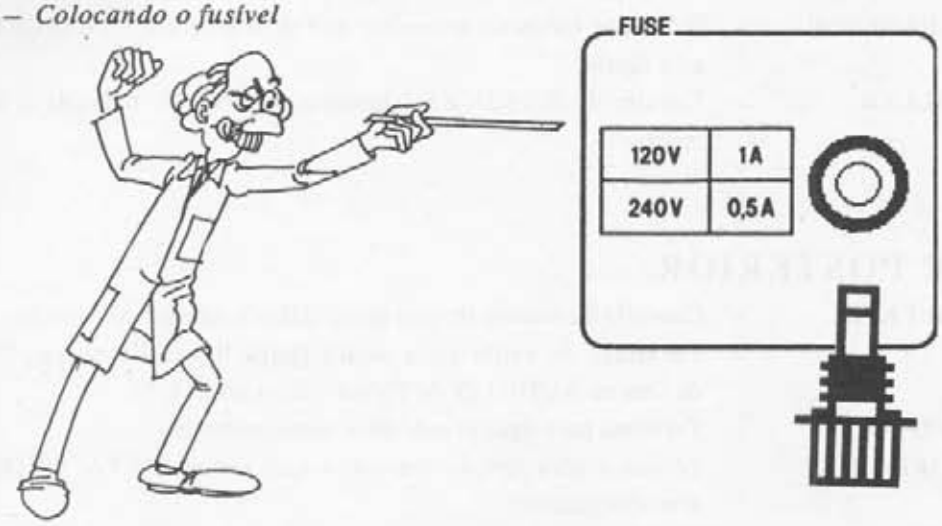
CABO DE FORÇA – Cabo para ligação à rede elétrica (NÃO LIGUE AINDA!)

ESCOLHENDO A TENSÃO DA REDE ELÉTRICA

1 – Verifique a tensão da tomada na qual você vai conectar seu micro. Se ela for de 110 a 127 V, coloque a chave de seleção do **AC INPUT** (veja fig 1.3) na posição 120 V. Ainda não ligue o micro na tomada!

Desparafuse o compartimento do fusível. Coloque, na tampa, um fusível de 1A (junto ao seu PLUS você recebeu um jogo de fusíveis). Rosqueie novamente a tampa no compartimento (fig 1.4).

fig 1.4 – Colocando o fusível



2 – Se a tensão na tomada for de 220 a 240 V, coloque um fusível de 0,5 A e selecione a chave do **AC INPUT** para 240 V. Ainda não ligue seu micro à rede!

3 – Se a rede elétrica à qual está ligada a tomada estiver sujeita a variações de tensão muito grandes, é conveniente usar um estabilizador de voltagem para proteger seu EXPERT PLUS.

4 – Se, no local da instalação, as interrupções de energia forem muito frequentes, isso pode causar sérios problemas ao seu equipamento e ao seu trabalho. Se faltar força no momento em que o micro está gravando alguma coisa no disquete, além de correr o risco de danificação do equipamento, todo o conteúdo da RAM do MSX (memória volátil) será perdido.

Se você está usando seu Expert Plus profissionalmente, é conveniente instalar um "No-Break", equipamento alimentado por bateria que se encarrega de fornecer a tensão adequada por alguns minutos enquanto você "salva" os arquivos que lhe interessam.

5 – Se de repente o micro parar de funcionar, apagando-se a luz vermelha do POWER e você verificar que não houve interrupção no fornecimento da energia elétrica, experimente trocar o fusível. Porém lembre-se

- 120 V – fusível 1 A
- 240 V – fusível 0.5 A

Se o fusível queimar novamente, não insista ou, pior ainda, não tente "*ligações diretas*" tipo papel alumínio.

Se o fusível queima (essa é sua função) significa que existe alguma anomalia interna, e antes que ocorram consequências graves (queima de componentes ou até princípios de incêndios), rompe-se o filamento do fusível, bloqueando assim a corrente elétrica. Ou seja, seu micro-computador está com um problema interno e deve, obrigatoriamente, ser enviado para uma assistência técnica autorizada (anexo ao microcomputador, vem uma lista das assistências técnicas autorizadas Gradiente).

CONECTANDO O TECLADO

1 – Com o micro ainda desligado, instale o console sobre a mesa onde ela vai ficar e ligue o cabo que sai do teclado no conector **KEYBOARD INPUT** que está na parte traseira do console (fig. 1.3). Na cinta metálica do plug deste cabo existe uma marca em baixo-relevo. Ela deve ficar na parte mais alta da conexão.

Não conecte ou desconecte o teclado com o micro ligado.

CONECTANDO O VÍDEO OU TELEVISÃO

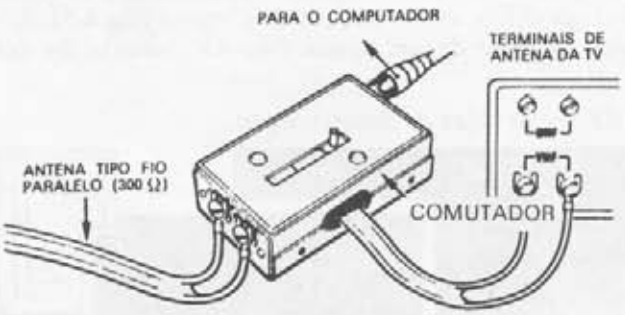
O periférico mais importante no micro-computador é o vídeo, pois é através deste que o usuário passa a receber as informações geradas pelo micro, inclusive, para conferir as mensagens digitadas pelo teclado.

Você pode ligar o seu EXPERT PLUS a uma TV P&B ou a cores (padrão PAL – M), a um monitor de vídeo monocromático, ou a um monitor de vídeo colorido.

1) A TV P&B ou a colorida devem ser conectados à saída **RF – OUT**, através do cabo "*RCA*" e o "comutador", fornecidos junto com o micro-computador.

Para isso desconecte o cabo da antena da entrada de VHF da TV, seguindo o manual de instruções de seu televisor. No lugar do cabo de antena coloque os terminais do comutador; conecte o cabo RCA do micro na entrada "COMPUTER" do "comutador", e por fim, conecte os terminais da antena na entrada "TV" do "comutador".

Para que se use o micro, lique a chave do "comutador" na posição COMPUTER, caso contrário, ligue na posição TV.



2) O monitor monocromático apresenta um qualidade de imagem superior à das televisões (mesmo sendo monocromático), possuindo um imagem mais estável e definida.

Para se usar o monitor monocromático em seu EXPERT PLUS, conecte o cabo tipo RCA, na entrada VÍDEO IN do monitor, e a outra extremidade na saida "VIDEO MONOC." do console de seu micro.

3) O monitor colorido apresenta a melhor imagem colorida do micro. Para usá-lo deve-se conectar o cabo tipo RCA, na entrada VIDEO IN do monitor colorido, e a outra extremidade na saída "COLOR" do console do micro.

A saída COLOR do micro, também pode ser usada conectada na entrada "VIDEO IN" do seu vídeo-cassete, ou Camcorder para gravação das imagens geradas pelo EXPERT PLUS.

Na parte traseira do console, o micro possue 2 tomadas de rede (SWITCHED OUTLET), independentes do fusível do micro, mas ligadas à chave POWER. Essas tomadas podem ser usadas para ligar a sua TV, ou monitor e eventuais periféricos, de maneira que pode-se, ao acionar uma única chave (POWER), ligar todo o sistema.

Observe que cada uma dessas tomadas suporta até 100 W de potência, e se colocarmos aparelhos de maior consumo (algumas televisões coloridas), pode-se comprometer a chave do POWER do micro.

CONECTANDO O ÁUDIO

O seu EXPERT PLUS possui um alto-falante interno que, para aplicações normais, é o suficiente. Para usuários que estejam interessados em qualidade de áudio, pode-se usar o alto-falante da TV, ou então ligar o áudio a um amplificador (pode ser um VIDEO CASSETE ou CAMCORDER), através de um cabo do tipo RCA, conectando em uma extremidade na entrada "**AUDIO IN**" do amplificador, e a outra extremidade na saída "**AUDIO**" do Expert.

LIGANDO O EXPERT PLUS

Depois de todas as instalações mostradas e conferidas nos ítens anteriores pode-se, então, ligar o Expert.

A seguir damos os passos finais que devem ser executados pelo usuário. Caso não se verifique o acionamento do computador confira novamente todos os ítens desse capítulo.

Proceda da seguinte forma:

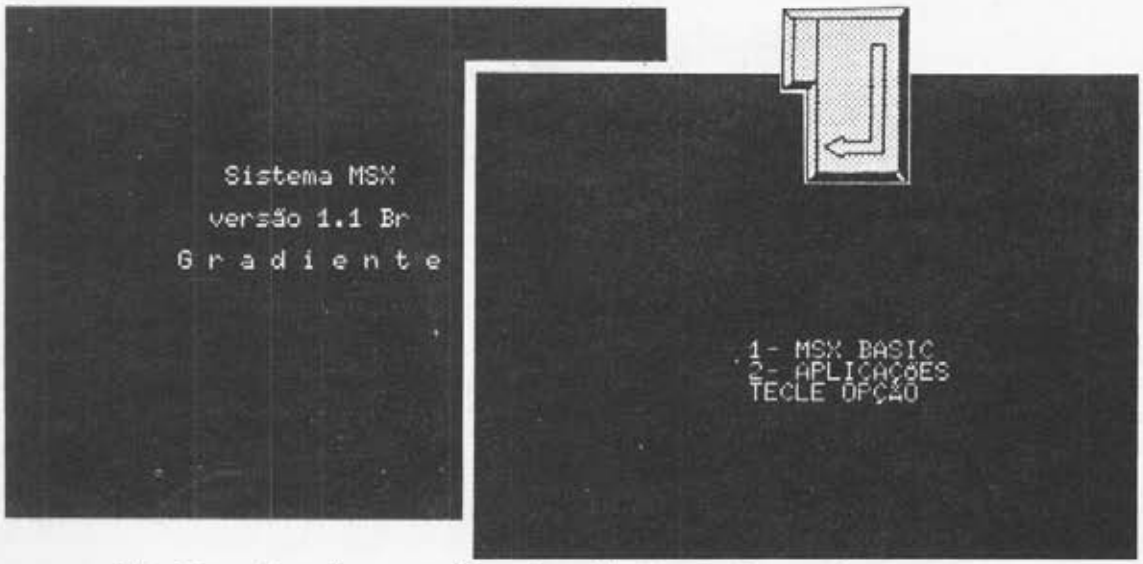
1) Coloque o "plug" do seu micro na tomada, verificando se a tensão da rede é correta .

2) Pressione a tecla POWER, e verifique se a luz de POWER também está acesa.

3) Verifique, no teclado, se a luz "IN USE" está acesa. Se não estiver, proceda novamente a instalação do teclado, mas faça isso somente com o aparelho desligado.

4) Você deverá obter as duas telas de apresentação do BASIC. Uma delas fica apenas alguns segundos no vídeo, sendo substituída outra (fig 1.5). Se você estiver usando uma TV talvez haja necessidade de um ajuste fino na sintonia do canal 3 ou 4 que você escolheu.

fig 1.5 — A tecla RETURN e as telas de apresentação .



Se todas os ítens forem confirmados, a instalação do seu micro teve sucesso, senão, releia este capítulo, até obter o resultado.Se você quiser ver seu Expert já funcionando com um programa residente digite a opção 2 (Aplicações).

Agora que está tudo bem, desligue o micro e, caso você não tenha noção nenhuma sobre computadores, leia o capítulo 2. Se você não conhece nada sobre a linguagem BASIC, leia também os capítulos 3, 4, 5, 6 e 7 . Se você é um "Expert" no assunto, basta apenas o dicionário do BASIC MSX do capítulos 8.

Boa Sorte e Bom Trabalho!

O QUE O EXPERT PLUS PODE FAZER POR VOCÊ

2

OS DOIS CAMINHOS DO USUÁRIO

Pronto! Agora seu micro está instalado e pronto para funcionar! Um universo virtualmente infinito de maravilhas se abre à sua frente, porém você precisa de algumas informações adicionais para penetrar neste novo mundo.

Você está um pouco na situação de alguém que comprou um excelente arco mas ainda está sem flechas! Existem, neste caso, duas possibilidades: Você pode comprar flechas prontas ou aprender a fabricá-las.

Se você decidir por fabricá-las, precisa saber quais as técnicas adequadas, que materiais usar e quanto tempo vai gastar nesta tarefa.

Se você for comprar flechas prontas, precisa conhecer qual o melhor modelo para cada finalidade, quem as fabrica e para que aplicação elas foram dimensionadas.

Da mesma forma, para operar seu micro, você precisa de uma "munição" chamada "Software" (programa de computador).

Você pode decidir se tornar um "programador", ou seja, "construir" seu próprio software, ou ser um "usuário de sofware pronto", utilizando programas elaborados por programadores experientes.

Cada um destes caminhos apresenta suas vantagens e desvantagens que enumeraremos na orientação que vem a seguir.

Vejamos qual dessas alternativas você deve seguir.

O PERFIL DO USUÁRIO DE SOFTWARE

Ao optar por este caminho, o fator que mais pesa é o tempo. Elaborar um software por conta própria exige, além de uma certa predisposição natural, uma quantidade de tempo muito grande, não só para incorporar os conhecimentos e as técnicas necessárias, mas também para o próprio desenvolvimento do programa.

Além disso, ao cabo de muito esforço, você pode descobrir que acabou de reinventar a roda! Você pode se sentir na situação de um alpinista que chega ao cume de uma montanha, após árdua escalada, e se defronta com uma espantada família fazendo piquenique dominical!

Existem muitos programas prontos que podem desempenhar tarefas interessantes sem necessidade de se perder tempo em elaborá-los!

Vejamos, a seguir, quais os principais tipos de programas que podem "municiar" seu Expert PLUS.

TIPOS DE PROGRAMAS

Dependendo da finalidade a que se destinam, os programas são classificados em:

PROFISSIONAIS

Classificamos como profissionais os programas que permitem usar o micro como elemento de automação de atividades empresariais.

São inúmeros mas, dentre eles, podemos destacar algumas grandes categorias:

1) EDITORES DE TEXTO

Fazem o micro substituir, com imensas vantagens, a mais sofisticada máquina de escrever eletrônica.

Permitem a digitação de textos e sua gravação em fita ou disco para uso posterior.

Só para se ter uma idéia, num disquete de seu MSX você pode armazenar algo em tôrno de 100.000 palavras!

Trabalhando na tela do micro, o texto pode ser alterado, trechos podem ser eliminados, outros acrescentados, de uma maneira extremamente simples e rápida.

Imagine, por exemplo, a seguinte situação (muito freqüênte na prática): num escritório estão sendo discutidos os termos de um contrato já digitado e as partes interessadas resolvem alterar uma cláusula.

Pelos processos tradicionais o contrato original deveria ser integralmente redatilografado (ou pelo menos a folha em que está a cláusula alterada).

Com um processador de texto basta alterar, na tela do Expert, o parágrafo em questão e, em questão de minutos, tirar outra cópia passada a limpo na impressora.

Imagine o que isto representa em termos de rapidez e eficiência!

Os processadores de texto tem um recurso especial: Busca e Substituição.

Digamos, por exemplo, que sua secretária tenha digitado um texto escrevendo casa com "z" (caza!). Ao perceber o erro, basta ordenar a procura de toda "caza" e sua substituição por "casa" e pronto! O texto ficará integralmente corrigido em segundos!

Além disso, pelo editor, você pode comandar a ativação de recursos especiais da impressora, como letras sublinhadas, negrito, fontes alternativas, etc.

Este livro, por exemplo, foi integralmente digitado no "Editor de Textos Gradiente", processador de texto para MSX.

Quando uma primeira versão do texto é tirada na impressora, ela passa por um processo de revisão. Os erros são corrigidos no texto gravado no Expert e a versão final é rapidamente obtida sem que haja necessidade de se redatilografar todo o texto.

Após cada revisão, os erros vão diminuindo. Antigamente o processo editorial exigia uma nova datilografia do texto corrigido: erros antigos eram eliminados mas novos erros eram introduzidos!

E tem mais: os disquetes contendo os textos digitados no seu Expert, podem ser lidos num IBM-PC, ou compatível, para serem processados, a nível de composição (Desk Top Publishing), de maneira a obter uma arte final.

Quem trabalha com editor de texto, normalmente se pergunta" como é que eu conseguia produzir textos, antes do advento do micro ?"

2) BANCOS DE DADOS

São programas que produzem um "fichário eletrônico inteligente".

Existem programas que fornecem as fichas pré-desenhadas de maneira a que o usuário tenha apenas o trabalho de preenchê-las.

Outros, permitem ao usuário desenhar as suas próprias fichas e determinar o formato dos relatórios impressos ou etiquetas de endereçamento. Outros ainda, como o dBASE, são verdadeiras linguagens dedicadas ao gerenciamento do banco de dados.

Em um banco de dados gerenciado pelo micro é possível realizar de maneira rápida e eficiente, tarefas que, num fichário tradicional, demandariam muito tempo.

Você pode, por exemplo, pedir a emissão das etiquetas de endereçamento de todos os clientes que façam aniversário na próxima semana. Ou então, levantar quantos fornecedores você tem no estado do Acre, ou ainda descobrir quantos ítens, no seu estoque, estão abaixo de um valor crítico, para desencadear um processo de compra.

Só para exemplificar a grande utilidade e versatilidade de um banco de dados no microcomputador, imagine os fichários de uma biblioteca.

Numa biblioteca não informatizada, cada livro novo exige a datilografia de pelo menos 3 fichas: uma para o arquivo alfabético dos AUTORES, outra para o de TÍTULOS e uma terceira para o de ASSUNTOS.

Usando um banco de dados, digitamos uma única ficha e pedimos ao próprio software para para criar 3 índices em ordem alfabética: AUTOR, TÍTULO e ASSUNTO.

Dependendo do tipo de consulta, seu Expert PLUS consultará o índice correspondente para saber em qual das 3 ordens estabelecidas o arquivo deve ser lido!

Tente imaginar estas tarefas sendo desempenhadas por um funcionário, consultando um fichário comum!

3) PLANILHAS ELETRÔNICAS

As planilhas têm a estrutura de uma gigantesca folha de papel dividida em colunas e linhas, com a tela do micro fazendo o papel de uma "janela" que pode correr horizontalmente ou verticalmente sobre esta folha, localizando a área de seu interesse.

Em cada "célula" da planilha (definida pela linha e coluna a que pertence) você pode colocar um texto, um número ou uma fórmula.

Digamos, por exemplo, que você tenha elaborado uma planilha de custos de um produto na qual alguns componentes têm seu preço em dólares.

Se a cotação do dólar mudar, não é preciso calcular de novo o valor destes ítens.

Basta colocar a nova cotação num lugar especial que você reservou para este fim e pedir à planilha para recalcular tudo. Num piscar de olhos você tem os novos valores (em moeda brasileira) atualizados!

Com estes recursos você pode, por exemplo, fazer simulações financeiras, prevendo o que vai acontecer com seus orçamentos se a taxa de inflação mudar, descobrindo dentro de que limites um projeto continua economicamente viável.

Da mesma forma, se você usa sua planilha para fazer uma folha de pagamento e houver alguma alteração na legislação fiscal ou trabalhista, em pouquíssimo tempo ela pode ser reconfigurada, permitindo uma atualização imediata.

Imagine, por exemplo, que você tenha uma confecção de roupas e receba um pedido. Se tiver uma planilha de produção preparada em seu Expert, onde está especificado, para cada modêlo e tamanho, a quantidade de tecido e o tempo de produção, sua tarefa será extremamente simplificada!

Basta introduzir os dados do pedido na planilha e, em questão de segundos, você terá a relação de matéria prima e o tempo necessário para atender o pedido. Pense no tempo e na dor de cabeça que você teria para fazer tudo isso na ponta do lápis!

Como você pode notar, as planilhas eletrônicas são instrumentos indispensáveis para organização de uma empresa e para tomadas de decisões.

E nem só em atividades empresariais as planilhas eletrônicas são úteis. Com todos os recursos que citamos, você ja deve ter deduzido como fica fácil organizar um orçamento doméstico ou até um receituário.

Imagine, por exemplo, que em seu receituário, um dado prato esteja dimensionado para 4 porções e você vai dar um jantar para 6 pessoas. Numa planilha eletrônica, todos os ingredientes serão recalculados num piscar de olhos!

4) PROGRAMAS DE COMUNICAÇÃO

São programas que permitem fazer seu Expert PLUS "conversar" com outros computadores, não importando sua marca ou tipo. Com seu Expert ligado a um MODEM, você pode acessar, através de uma linha telefônica comum, serviços como STM-400, VIDEOTEXTO, etc, nos quais você pode consultar os bancos de dados de outros computadores e obter as mais variadas informações.

Você pode obter, em segundos, as cotações da Bolsa, seu saldo bancário, a posição de suas aplicações no Open, a cotação do Dólar no oficial e no paralelo, etc.

Depois de tanto trabalho, você vai querer descansar um pouco: basta acessar, um serviço especial e você terá a crítica dos filmes que estão passando hoje!

5) APLICATIVOS

São programas profissionais dedicados a uma atividade particular.

Dentro desta classificação você pode encontrar "softwares" de controle de estoque, matemática financeira, estatística, agendas, editores gráficos, musicais, etc.

Já existem inúmeras versões destes programas no mercado e muitas em fase de desenvolvimento. Se você necessita de um aplicativo específico, pode optar por encomendar um programa especial a alguma "Software-house".

Se, em sua categoria profissional, existem outras empresas com a mesma necessidade, é usual a formação de um "POOL" de maneira a diluir os custos de desenvolvimento entre vários usuários.

Se você quer ter uma noção mais específica dos vários tipos de programas que existem para MSX, aconselhamos a leitura do "Guia do MSX" onde você achará, inclusive, os endereços e telefones dos produtores e distribuidores.

LAZER

Nesta área, além de uma quantidade enorme de jogos tipo vídeo-game (alguns dos quais sofisticadíssimos!), temos programas que não se destinam à recreação pura e simples, trazendo algum tipo de crescimento nas habilidades do usuário.

Nesta faixa existem vários jogos de inteligência, simuladores de pilotagem de aviões ou automóveis, xadrez e outros jogos de raciocínio.Existem também os chamados "adventures" nos quais o jogador, além de defrontar com situações de ação (tipo vídeo-game), deve responder perguntas, tomar decisões e elaborar um planejamento estratégico para prosseguir o jogo.

Note que muitos usuários consideram um "hobby" extremamente relaxante e divertido programar e criar seus próprios jogos!

EDUCACIONAIS

O microcomputador revelou ser poderosíssimo instrumento na área educacional. Com seus recursos de som e imagem, o MSX é insuperável nesta área.

Existem basicamente 3 tipos de programas educacionais:

1) DE MEMORIZAÇÃO

Nestes programas, o micro repete, com paciência infinita, testes de correlação do tipo *"elementos químicos - seus símbolos"*, *"países - capitais"*, taboada, acentuação da língua portuguesa, vocabulário de línguas estrangeiras, etc.

O micro se transforma num professor particular com uma paciência sobre-humana e disponível a qualquer hora.

2) DE SIMULAÇÃO

No microcomputador podemos simular experimentos quese impossíveis de serem realizados em laboratório ou que demandariam recursos absurdos em tempo e dinheiro.

3) DE RACIOCÍNIO

No ensino tradicional, dá-se pouca ênfase ao desenvolvimento do raciocínio, ficando esta tarefa normalmente relegada como sub-produto do ensino da Matemática.

O micro-computador e o MSX em particular, chegaram para revolucionar este conceito, sendo usados para desenvolver as habilidades de raciocínio das crianças, de maneira a produzir jovens mais inteligentes.

O próprio ato de programar o micro é um estímulo muito grande para o desenvolvimento intelectual da criatividade.

Neste contexto a linguagem "LOGO", desenvolvida no M.I.T. por Pappert com base nos estudos de Piaget, tem um papel relevante.

Note que existe uma região limítrofe entre os programas de lazer e os educacionais pois existem "softwares" que, disfarçados de "jogos", acabam gerando um efeito educacional importante.

Vejamos agora, sob que forma todos estes programas podem ser "transportados" para seu Expert PLUS:

OS VEÍCULOS DO SOFTWARE

Estes programas são comercializados e divulgados basicamente de 5 formas:

CARTUCHOS

São programas gravados no circuito eletrônico de um cartucho que será conectado num dos slots do micro (no painel frontal do console do seu Expert PLUS você tem dois slots: o "cartridge" A e o B). Os cartuchos devem sempre ser colocados ou retirados com o micro desligado.

Alguns, ao serem ligados, começam a executar o programa "tomando conta" do micro, não permitindo a execução de outra tarefa.

Outros, ficam "dormentes" enquanto o micro pode executar outra tarefa e só são ativados quando "chamados" (normalmente através de um comando denominado CALL...).

FITA CASSETE

São programas gravados em fita e precisam ser carregados no micro através de um gravador ou "DATACORDER". Como o micro deve "ler" os programas sequencialmente, o tempo de carga é muito mais demorado do que no caso dos cartuchos. Podem apresentar algum problema de carga quando o gravador não está devidamente ajustado ao seu micro.

Como as formas de gravação podem ser variadas, os comandos para ordenar o car-regamento podem ser de vários tipos, estando especificados no manual do fornecedor.

DISQUETES

Esta é a forma mais racional e versátil para se carregar um programa no micro. Ele vem gravado num disco que será lido num periférico chamado acionador de discos ou "disk-drive. Existem dois tipos de discos (e acionadores): de 5 1/4" e de 3 1/2" .

Alguns são carregados automaticamente, não havendo necessidades de comandos especiais.

Outros são carregados a partir de instruções especiais que devem estar especificadas no manual do programa.

A carga é normalmente mais lenta do que nos cartuchos mas é considerávelmente mais rápida que no caso da fita cassete.

É o sistema que fornece a melhor relação custo/benefício.

LISTAGENS

Quando o programa não é muito extenso, ele pode ser divulgado, em livros e revistas, na forma de listagens. Usando o teclado do micro, o leitor poderá digitá-lo de maneira a carregá-lo na memória do computador.

Posteriormente ele poderá gravar estes programas em fita ou disco.

Apesar de ser um procedimento trabalhoso e demorado tem a vantagem de induzir o aprendizado das técnicas de programação. Neste livro que você está lendo existem exemplos destas listagens nos capítulos subsequentes.

TELEFONE E RÁDIO

Através de uma linha telefônica comum, e com o uso de um periférico denominado MODEM, seu Expert PLUS pode receber e enviar dados, "conversando" com outro computador.

Note que o interlocutor de seu MSX pode ser qualquer tipo de computador.

Com este sistema você pode receber software transmitido por outro computador. Se você for radio-amador, existem modelos de modem que permitem esta "conversa" à distância sem necessidade de linhas telefônicas.

Um outro processo muito interessante é usado, por exemplo, por algumas emissorasde FM: a transmissão de "software radiofônico"!

Basta sintonizar na estação no horário do programa e, quando o locutor avisa, gravar os ruídos dos códigos enviados. Posteriormente o ouvinte pode mandar seu MSX "ler" esta fita, carregando o programa transmitido na memória do micro.

O PERFIL DO PROGRAMADOR

Existem dois tipos de programadores na área de microinformática: os amadores e os profissionais.

Os programadores amadores são "hobbistas" que extraem prazer do ato de criar coisas novas em seu micro. Acham, por exemplo, muito mais agradável criar um programa interessante do que passar horas jogando xadrez.

Apesar do carácter amadorístico desta atividade, ela pode gerar uma remuneração profissional quando estes trabalhos são aproveitados por alguma "Software-house" ou publicados em livros e revistas.

Além disso, a atividade de programar, faz "cócegas no cérebro", promovendo o desenvolvimento intelectual de quem a exerce e causando um grande prazer no momento em que se vê o trabalho realizado.

O programador profissional, por outro lado, é alguém que se dedica integralmente a esta atividade e dela retira a totalidade ou a parcela mais substancial de sua remuneração.

Como o ensino profissionalizante de computação, no Brasil, é ainda muito incipiente, o que se nota, é uma grande quantidade de amadores, em sua maioria auto-didatas, que migram para a área profissional, ao perceberem suas aptidões para esta tarefa.

Neste contexto, os micros pessoais e em particular o Expert, estão desempenhando um papel importantíssimo na formação destes profissionais.

Através de um MSX eles tem oportunidade de acessar, a baixo custo, as mais variadas e modernas linguagens de programação, num processo de atualização constante.

AS LINGUAGENS DE PROGRAMAÇÃO

A rigor, a única linguagem que seu micro entende é a Linguagem de Máquina, uma sequência de códigos binários que, ao serem introduzidos no microprocessador (Z80 no caso do MSX) o obrigam a executar diversas tarefas.

Esta linguagem, porém, é extremamente abstrata e exige um grande esforço (de raciocínio e conhecimentos) por parte do programador.

Consequentemente, para o programador iniciante, é interessante começar pelo aprendizado de linguagens de "alto-nível". Estas linguagens têm uma estrutura mais próxima da linguagem humana, sendo algumas praticamente coloquiais.

Existem muitas linguagens de alto nível desenvolvidas com esta finalidade, cada uma com particularidades que a indicam como mais adequada para a realização de uma determinada tarefa.

Dentre as linguagens disponíveis para MSX, podemos citar:

BASIC: especialmente desenvolvida para principiantes, é de fácil aprendizado e é extremamente versátil.

FORTRAN: uma das primeiras linguagens desenvolvidas para computadores, especialmente para finalidades científicas.

PASCAL: linguagem moderna, para programação estruturada, muito usada por profissionais em desenvolvimento de sistemas.

COBOL: concebida para aplicações comerciais. Apesar de obsoleta, é ainda muito utilizada para manutenção de sistemas desenvolvidos há mais tempo.

C: ferramenta muito útil no desenvolvimento de sistemas mais complexos. Permite a incorporação de rotinas desenvolvidas em Linguagem de Máquina.

LISP: muito usada em sistemas de "inteligência artificial".

LOGO: sub-conjunto da LISP, é muito usada em aplicações educacionais. Foi especialmente desenvolvida para introduzir crianças no mundo da informática. Apesar deste aspecto "introdutório" é uma linguagem extremamente poderosa e a versão do LOGO para MSX permite o desenvolvimento de programas até profissionais.

DBASE: é uma linguagem especialmente desenvolvida para criação e gerenciamento de Bancos de Dados. Pode ser usada tanto no modo interativo (comando a comando) quanto no modo programado.

Normalmente estas linguagens são carregadas no micro, como programas, podendo ser divididas em duas grandes categorias:

1) LINGUAGENS INTERPRETADAS

São linguagens em que cada instrução é lida no texto de um "Programa-Fonte", traduzida para o código de máquina e imediatamente executada.

Têm a vantagem de serem altamente interativas, ou seja, se durante a execução ocorrer algo de errado, o programa pode ser interrompido e o texto do "Programa- Fonte" pode ser alterado e corrigido.

A grande desvantagem está no tempo de execução pois, toda vez que o programa é executado, o tempo de tradução é sempre consumido.

2) LINGUAGENS COMPILADAS

Neste caso o texto do "programa-fonte" é integralmente traduzido para produzir um "programa-objeto" em Código de Máquina.

São muito menos interativas e a fase de depuração de erros é muito mais trabalhosa (a cada revisão é necessário recompilar tudo). Em compensação, uma vez depurado, o programa-objeto roda com alta velocidade, pois todo o tempo de "tradução" é excluído.

Note que, enquanto o Programa-Objeto é específico para cada microprocessador (um programa em L.M. para Z80 não roda, por exemplo num IBM-PC que usa outro microprocessador), o Programa-Fonte pode mais facilmente ser transportado de uma máquina para outra.

Você pode, por exemplo, escrever um Programa-Fonte no seu Expert, compilá-lo e checar se funciona. Posteriormente, você pode pegar o texto do Programa-Fonte, passá-lo para um IBM-PC (que consegue ler textos a partir dos disquetes de MSX) e recompilá-lo usando um compilador específico para o microprocessador desta máquina.

Se você não usar particularidades do MSX (como "sprites" ou rotinas de som sofisticadas), seu programa rodará tranquilamente na outra máquina!

Com estas possibilidades, ao invés de comprar, para uso em programação, 4 IBM-PC, o usuário pode adquirir apenas 1 e substituir os outros por MSX a um custo absurdamente menor!

A LINGUAGEM DO EXPERT PLUS

A rigor o Expert PLUS não tem uma linguagem própria (a não ser a L.M. do seu Z80) podendo ser configurado para usar qualquer uma das linguagens citadas.

Seus projetistas, porém, decidiram incorporar à máquina uma linguagem residente (uma espécie de cartucho já montado em seu interior) e escolheram o MSX-BASIC.

Desta forma, ao ser ligado, o Expert já se oferece para o uso do interpretador BASIC residente.

O MSX-BASIC do seu Expert PLUS é extremamente poderoso e cheio de recursos, permitindo a elaboração de programas sofisticados mesmo para usuários inexperientes.

Nos capítulos subsequentes você terá uma primeira noção de como utilizar esta linguagem.

Como a finalidade deste livro não é a de transformar seu leitor num "programador", o dicionário de BASIC contido no capítulo 8 apresenta-se de forma resumida, contendo apenas as informações essenciais.

Se você desejar se aprofundar mais nesta linguagem, aconselhamos a leitura do apendice E (Bibliografia Recomendada) para que você possa se orientar sobre a melhor sequência de aprendizado, tanto no BASIC, quanto em outras linguagens.

De qualquer forma, é bom frisar que, para utilizar seu Expert PLUS, não é indispensável aprender a programar. Existe uma quantidade enorme de softwares desenvolvidos para o padrão MSX e, sabendo escolher,você certamente encontrará os que atendem suas necessidades.

Os dois caminhos estão abertos à sua frente: ambos igualmente promissores.

Se você optar pela programação, leia atentamente os capítulos 3 a 7 deste livro e utilize o capítulo 8 para consultas. Não deixe de ler o apêndice E (bibliografia aconselhada) para saber como prosseguir.

Escolha o caminho que melhor se ajusta às suas necessidades e seja bem vindo ao maravilhoso mundo da informática!

DIGITANDOE EDITANDO

3

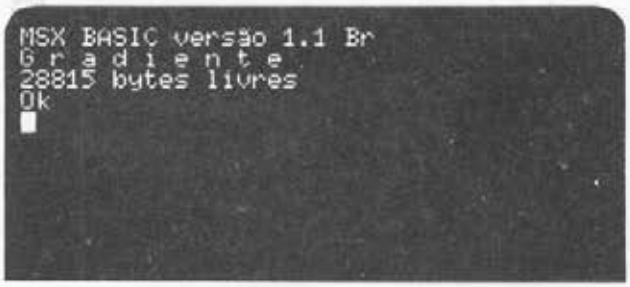
As memórias do Expert

Certifique-se que seu micro esteja convenientemente instalado (reveja o capítulo 1 em caso de dúvida) e ligue-o pressionando a tecla POWER. Ao receber o menu de opções, digite 1 (figura 3.1).

fig.3.1- Tela inicial.

Se não houver nenhum periférico ou cartucho conectado ao micro, o seu EXPERT PLUS irá operar em BASIC (em sua versão mais simples). Deverá aparecer a mensagem da figura 3.2.

fig.3.2- Tela de Apresentação



Uma das primeiras perguntas que surgem na maioria dos usuários é "Eu comprei um MSX com 80 Kbytes de memória e aqui estão aparecendo menos de 29! Será que meu micro está com defeito?".

Calma! Não existe defeito nenhum: A memória de 80 kbytes está dividida da seguinte maneira: 16 kbytes são para o vídeo e os outros 64 kbytes de memória RAM estão distribuidos em 4 páginas (de 0 a 3) de 16 kbytes cada uma.

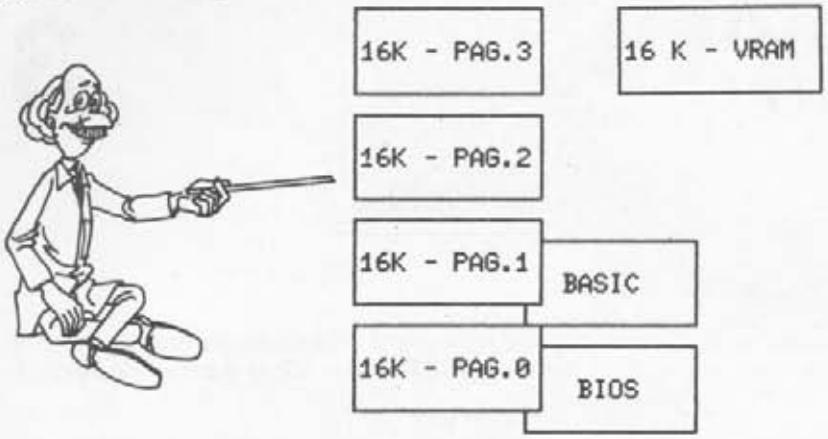
Para que o seu micro possa funcionar em BASIC, uma parte da memória RAM (páginas 0 e 1) não pode ser acessada; pois nesse lugar é colocada a ROM do micro, que nada mais é do que um programa muito grande e complexo que interpreta a linguagem BASIC.

"Sim - você dirá- mas isso me dá direito a 32 kbytes e não menos de 29!"

Você aprenderá mais tarde que todo programa, necessita de variáveis para poder armazenar dados; e, como o interpretador BASIC contido na ROM do micro é um programa, nada mais natural do que ele consumir um pouco da memória RAM para suas próprias variáveis.

De qualquer forma, não se preocupe: 28 Kbytes são mais do que suficientes para as finalidades a que nos propomos agora.

fig.3.3 - Os 80 Kbytes do BASIC



Familiarizando-se com o teclado

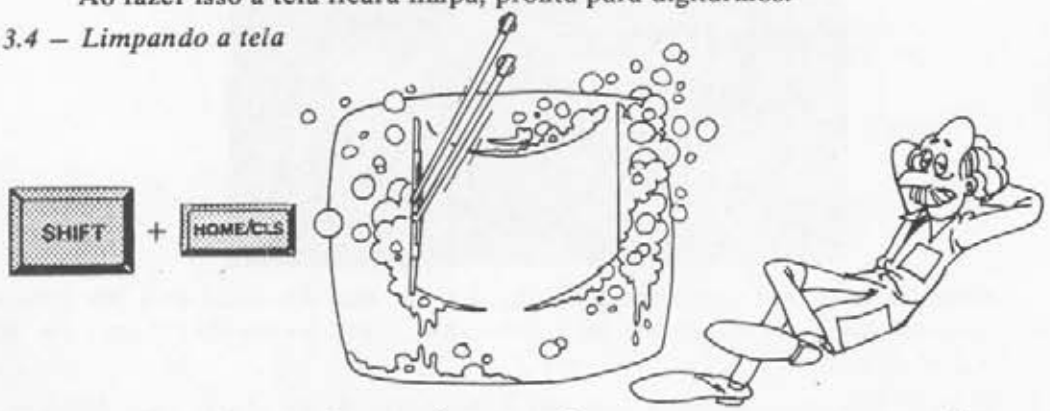
O teclado do MSX é muito parecido com o de uma máquina de escrever, acrescido de algumas teclas especiais com as quais vamos nos familiarizar agora.

A partir deste momento toda vez que escrevemos "aperte as teclas **[A] + [B]**" isto significará: pressione a primeira (no caso A) e SEM ABANDONÁ-LA, pressione também a segunda (no caso B).

Experimente digitar **[SHIFT] + [HOME/CLS]** (ou seja aperte com um dedo a tecla **[SHIFT]** e, mantendo-a pressionada, aperte **[HOME/CLS]**.

Ao fazer isso a tela ficará limpa, pronta para digitarmos.

fig.3.4 — Limpando a tela



Agora que já aprendemos a limpar a tela, vamos começar a escrever alguma coisa.

Pressione a tecla **[A]** uma única vez: você verá um "a" minúsculo surgir na tela e o quadradinho luminoso (cursor) se deslocará de uma posição para direita.

fig 3.5

a

Pressione a tecla **[A]** e mantenha-a pressionada algum tempo: isso ativará o repetidor automático de teclado e um monte de "aaa" surgirão na tela (fig. 3.6).

fig.3.6

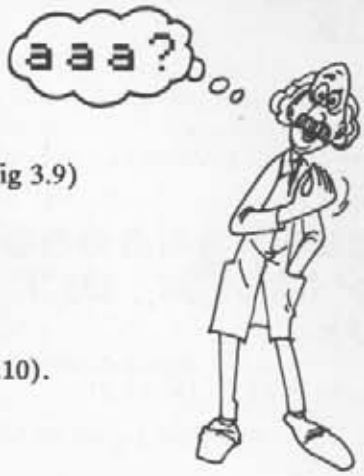
aaaaaaaaaaaaaaa

Pressione a tecla do **[RETURN]** (fig. 3.7)

fig.3.7

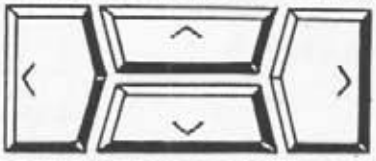
Na tela deverá aparecer uma mensagem de erro pois, para o BASIC, "aaaaaa..." Não significa nada !(fig. 3.8).

```
aaaaaaaaaaaaaaa
Syntax error
Ok
```



Vamos agora aprender a apagar: usando as setas (fig 3.9)

fig.3.9



leve o cursor até o "n" da mensagem "Syntax error" (figura 3.10).

fig.3.10

```
aaaaaaaaaaaaaaa
Syntax error
Ok
```

Pressione a tecla **[BS]** (Back Space) uma única vez: o "y" deve desaparecer (fig.3.11).

fig.3.11

```
aaaaaaaaaaaaaaa
Stax error
Ok
```

O efeito desta tecla é fazer o cursor retroceder de uma posição apagando o que encontra pelo caminho.

Se você apertasse novamente o **[BS]**, quem deveria sumir agora seria o "S".

Mas, ao invés disso, aperte uma vez **[DELETE]**.

Quem desapareceu foi o "n" (fig.3.12)

fig. 3.12

```
aaaaaaaaaaaaaaa
Sax error
Ok
```

O efeito do DELETE é o de manter o cursor no local, apagar o que está em baixo dele e "puxar" para esquerda o resto da linha.

Percebeu a diferença entre o **[BS]** e o **[DELETE]** ?

Vamos, agora, reestabelecer a mensagem "Syntax error". Pressione o **[Y]** uma vez. O "y" substitui o "n" e cursor foi parar em cima do "a" (fig. 3.13)

fig. 3.13

```
aaaaaaaaaaaaaaa
Syax error
Ok
```

Para não destruir o resto do texto, coloque o cursor no modo " inserção" pressionando uma única vez a tecla **[INSERT]**. O cursor fica pela metade (fig.3.14)

fig. 3.14

```
aaaaaaaaaaaaaa
Syax error
Ok
```

Digite agora **[N]** e **[T]** de maneira o reestabelecer a mensagem original. Note que agora, com o cursor pela metade (modo "inserção") as novas letras digitadas não apagam as seguintes, mas se "inserem" no texto, deslocando o resto para direita. Aperte mais uma vez a tecla **[INSERT]** e veja o cursor voltar ao normal (fig. 3.15)

fig. 3.15

```
aaaaaaaaaaaaaa
Syntax error
Ok
```

Leve agora o cursor até a primeira linha livre, usando as setas, e aperte uma vez a tecla **[CAPS LOCK]**.

A seguir digite, na sequencia **[C],[L],[S]** (fig.3.16).

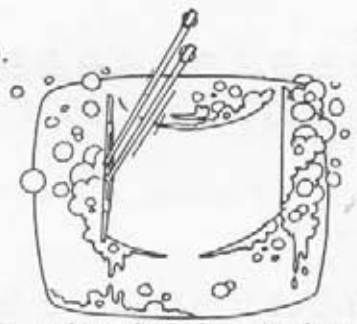


fig.3.16

```
aaaaaaaaaaaaaa
Syntax error
Ok
CLS■
```

A tecla **[CAPS LOCK]** funciona como a "trava de maiúsculas" da máquina de escrever. Note que ela age apenas sobre as letras (se você quiser o ponto de exclamação (!) terá que digitar **[SHIFT]** + **[1]**). Para "destravar", basta apertar **[CAPS LOCK]** mais uma vez.

Por enquanto, porém, deixe-o travado. Agora aperte o **[RETURN]** (a tecla da figura 3.7) .

Ao invés de aparecer "Syntax error" como aconteceu antes, a tela foi apagada e apareceu um "Ok". Isso porque a palavra CLS, para o BASIC, tem significado e ele conseguiu executar a ordem. Ela significa "LIMPE A TELA" (CLear Screen) e foi isso que ele fez.

Você deve ter notado, até agora, que aparecem dizeres no rodapé da tela: eles correspondem às teclas de função.

No teclado tem 5, mas na realidade, são 10. Se você apertar **[SHIFT]**, verá outras 5 mensagens: a tecla F1,por exemplo, pode funcionar como F6 se for pressionado em conjunto co **[SHIFT]** :

[F6] = [SHIFT] + [F1]

Se você digitar KEY OFF e **[RETURN]**, verá o rodapé desaparecer (fig.3.17).

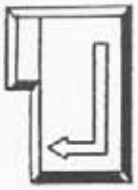
fig.3.17

```
Ok
KEY OFF■
```

Se quiser que o rodapé reapareça, digite KEY ON e **[RETURN]** (fig. 3.18).

fig. 3.18

```
Ok
KEY OFF
Ok
KEY ON
```



Se você quiser ver o conteúdo das teclas, basta digitar KEY LIST e **[RETURN]** (fig. 3.19)

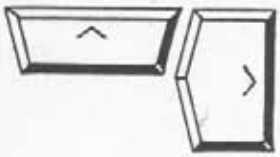
fig.3.19

```
KEY LIST
color
auto
goto
list
run
color 15,1,1
cload"
cont
list
run
```

Como você verá mais tarde, estas teclas podem ser reprogramadas com o conteúdo que você quiser. Mas, por enquanto, vamos continuar a exploração do teclado: leve o cursor até o "I" do KEY LIST, usando as setas (fig. 3.20)

fig. 3.20

```
KEY UN
Ok
KEY LIST
color
```



Pressione, agora, **[CONTROL]** + **[E]**. Como você pode observar na fig. 3.21 o efeito dessa combinação de teclas é o de apagar toda linha abaixo e à direita do cursor.

fig. 3.21

```
KEY UN
Ok
KEY L
color
```

Leve agora o cursor até a linha do KEY OFF, com as setas, colocando-o em cima do O, como indica a figura 3.22.

fig. 3.22

```
Ok
KEY OFF
Ok
KEY ON
```

Pressione as teclas **[CONTROL]** + **[U]**: a linha inteira desaparecerá (fig. 3.23).

fig. 3.23

```
Ok

Ok
KEY ON
```

Existem outras combinações da tecla **[CONTROL]** que veremos a seguir. Por enquanto vamos nos limitar em analisar o efeito de mais uma tecla. Coloque o cursor sobre o "Y" do KEY ON e pressione **[INSERT]** para colocar o cursor no modo "inserção" (pela metade) como mostrado na figura 3.24 .

fig. 3.24

```
Ok
KEY ON
Ok
KEY L
```

Pressione a tecla [TAB]: como você pode notar, ela funciona como um TABulador que desloca o cursor nas posições múltiplas de 8 na tela.(fig. 3.25).

fig. 3.25

```
Uk
KE      Y ON
Ok
KEY L
```

figura 3.26 - As combinações com a tecla [CONTROL]

COMBINAÇÕES DE TECLAS	FUNÇÃO	TECLA ESPECIAL EQUIVALENTE
CONTROL + B	Move o cursor até o começo da palavra anterior	
CONTROL + C	Dá um BREAK na espera do INPUT	
CONTROL + E	Apaga do cursor ao fim da linha	
CONTROL + F	Move o cursor até o começo da palavra seguinte	
CONTROL + G	Dá um BEEP sonoro	
CONTROL + H	Volta o cursor, apagando	[BS]
CONTROL + I	Move o cursor até a próxima tabulação	[TAB]
CONTROL + J	Coloca o cursor na próxima linha (line feed)	
CONTROL + K	Move o cursor até o canto superior esquerdo	[HOME]
CONTROL + L	Limpa a tela colocando o cursor em HOME	[CLS]
CONTROL + M	Insere a linha BASIC na memória	[RETURN]
CONTROL + N	Leva o cursor ao fim da linha	
CONTROL + R	Põe o cursor em modo inserção	[INSERT]
CONTROL + U	Apaga a linha inteira	
CONTROL + \	Move o cursor para a direita	[►]
CONTROL +]	Move o cursor para a esquerda	[◄]
CONTROL + ^	Move o cursor para cima	[▲]
CONTROL + –	Move o cursor para baixo	[▼]

ESCREVENDO E CALCULANDO

4

USANDO A SCREEN 0

O padrão MSX-1 impõe a existência de 4 tipos de tela no seu micro:

SCREEN 0 : para texto em 40 colunas, SCREEN 1: para texto em 32 colunas SCREEN 2: para gráficos de alta resolução, SCREEN 3: para gráficos de baixa resolução.

Quando o micro é ligado, ele assume a SCREEN 0, de 40 colunas. Limpe a tela do micro (Lembra ? Só digitar **[SHIFT] + [HOME/CLS]**). Agora digite **[F6]** (**[SHIFT]** + **[F1]**).

Na tela deve aparecer o comando mostrado na figura 4.1

figura 4.1

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

Em alguns televisores o tamanho da imagem "transborda" um pouco os limites da tela, fazendo com que as primeiras letras não apareçam.

Se este é seu caso, você pode re-ajustar a largura do SCREEN 0 usando um valor menor que o inicial de 40 colunas.

Digite, por exemplo:

WIDTH 4 **(e [RETURN])**

Obviamente estamos exagerando pois dificilmente uma tela de apenas 4 colunas terá alguma utilidade. Agora pressione novamente **[F6]** para ver como fica o comando que ela ativa (figura 4.2)

figura 4.2


```
Ok
colo
r 15
,1,1

Ok
```

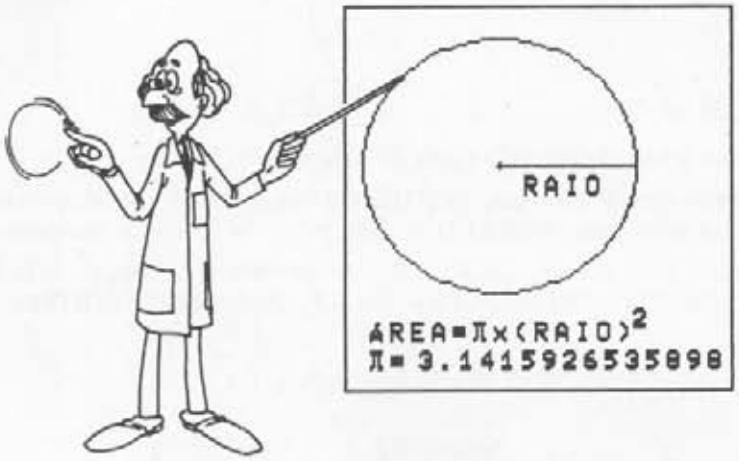
Agora comande WIDTH 40 para reestabelecer a largura original ou WIDTH 39 ou WIDTH 38 se sua TV está "transbordando" um pouco.

Pronto! Vamos agora aprender alguma coisa sobre outros recursos do BASIC para perceber a enorme utilidade desta linguagem.

Vamos começar com um exemplo bem simples:

digamos, por exemplo, que você queira calcular a área de um circulo, conhecendo seu raio R. A fórmula para fazer este cálculo é dada na figura 4.3 . A letra grega PI que aparece na fórmula é uma constante de geometria que, por enquanto, vamos considerar como aproximadamente igual a 3,14.

figura 4.3 Área de um circulo



ARMAZENAMENTO DE VARIÁVEIS NUMÉRICAS

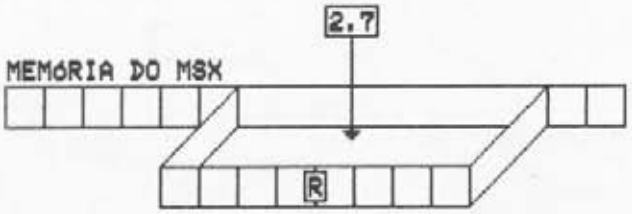
A primeira coisa que devemos fazer é informar ao micro qual o valor do raio. Digamos que o círculo tenha um raio de 2,7m. Limpe a tela e digite:

R= 2.7 (e [RETURN])

Neste momento o MSX abrirá uma gaveta em sua memória, colocará uma etiqueta "R" nela e armazenará lá dentro o valor 2.7 (note que o MSX usa a notação anglo-saxônica: para separar a parte inteira da decimal usa um ponto e não uma vírgula).

Após o pressionamento do RETURN, aparecerá a mensagem "Ok", indicando que a operação foi aceita.

figura 4.4.Armazenamento do valor de uma variável.



Se você quiser checar se o valor está corretamente armazenado, basta digitar:

PRINT R (e [RETURN])

Você deverá obter a tela da figura 4.5.

figura 4.5. Verificando o armazenamento.

```
R=2.7
Ok
PRINT R
 2.7
Ok
```

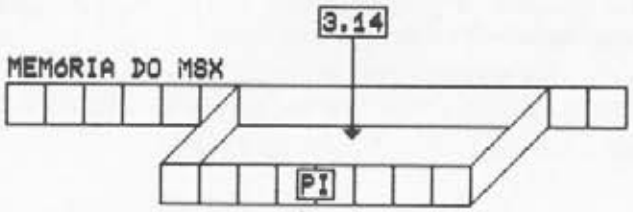
Como você deve ter percebido, o comando PRINT ("imprima") faz seu micro "imprimir" na tela o valor solicitado.

Agora que o valor do raio está armazenado na memória, vamos informar seu EXPERT qual o valor de PI. Basta então digitar:

PI=3.14 (e [RETURN])

Novamente você terá o armazenamento desta variável em outra posição da memória (figura 4.6)

figura 4.6. Armazenamento de outra variável.



Agora temos condições de calcular a área do círculo em questão.

NO BASIC MSX o sinal de multiplicação é o asterisco ("*")e o do exponenciação é o " ^ ". Não confunda este operador, que está na mesma tecla do 6, com o acento circunflexo que está na tecla do til!

Digite então:

A=PI*R^2 (e [RETURN])

e a seguir:

PRINT A (e [RETURN])

Você deverá obter uma tela parecida com a da figura 4.7

figura 4.7 -Calculando a área de um círculo de raio =2,7m

```
R=2.7
Ok
PRINT R
 2.7
Ok
PI=3.14
Ok
A=PI*R^2
Ok
PRINT A
 22.8906
Ok
```

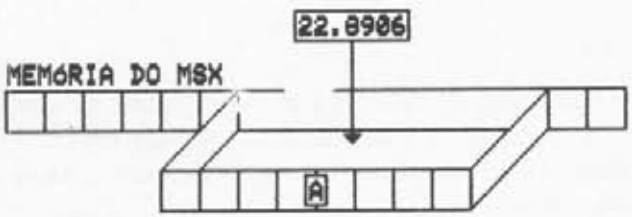
Vamos reconstruir o que seu Expert fez para chegar à conclusão de que a área do círculo éra de 22.8906 metros quadrados.

Inicialmente ele foi consultar a gaveta com etiqueta "R" e encontrou a valor 2.7. A seguir elevou este valor ao quadrado encontrou do 7.29.

Note que ele obedece à prioridade de operações que aprendemos na álgebra ginasial: primeiro a exp nenciação e depois o produto!

Em seguida ele abre a gaveta com etiqueta "PI" e encontra o valor 3.14, multiplica o valor do PI por 7.29 e armazena o resultado na gaveta "A" (figura 4.8)

figura 4.8. Armazenado o res ltado dos cálculos da área.



Antes de continuarmos no nosso cálculo da área, vamos aprender mais alguma coisa sobre variáveis. Limpe a tela e digite.

RAIO=2.7 (e [RETURN])

A seguir digite

? RAIO (e [RETURN])

Você obterá o valor da variável RAIO: o "?" é uma forma abreviadade de se escrever o "PRINT".

Experimente agora digitar:

?RAMBO **(e [RETURN])**

?RAFAEL **(e [RETURN])**

?RATO **(e [RETURN])**

?RAPADURA **(e [RETURN])**

Você deverá obter a tela da figura 4.9.

figura 4.9 -Pesquisando o "nome" de uma variável.

```
RAIO=2.7
Ok
? RAIO
 2.7
Ok
? RAMBO
 2.7
Ok
? RAFAEL
 2.7
Ok
? RATO
 2.7
Syntax error
Ok
? RAPADURA
 2.7
Syntax error
Ok
```

Vamos entender o que está acontecendo: no BASIC MSX a "etiqueta" que é colocada na "gaveta" de uma variável tem, no máximo, duas letras, sendo que a primeira delas deve, obrigatóriamente, estar entre A e Z. Assim sendo, "RAIO", "RAFAEL" ou "RAPADURA", serão considerados apenas como "RA"! Além disso, é importante notar que, neste caso, o micro não faz distinção entre maiúsculas e minúsculas: "RA","ra","Ra" e "rA" correspondem à mesma gaveta!

Existe uma outra particularidade que deve ser levada em conta na hora de se atribuir um nome a uma variável: ele não pode conter nenhuma "palavra reservada" do Linguagem BASIC.

Se você consultar o dicionário do capitúlo 8, verá que o "TO" de "RATO" e o "PAD" de "RAPADURA" são palavras reservadas do BASIC, o que explica a mensagem "Syntax error" que aparece nestes dois casos.

O que acontecerá, então, se você digitar:

? RAPRINT

Tente responder antes de fazer a experiência na prática!

Vamos aprender, agora, mais alguma coisa sobre as variáveis do BASIC MSX.

Limpe a tela e digite:

PI# = 4*ATN(1) (e [RETURN])

? PI# (e [RETURN])

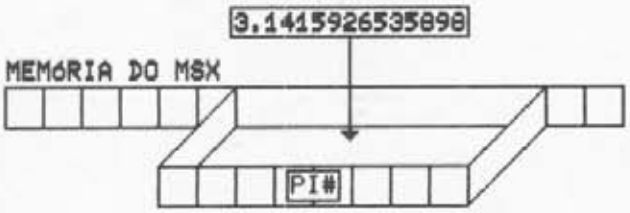
Você deverá obter a tela da figura 4.10.

figura 4.10 -Calculo do PI com precisão dupla.

```
PI#=4*ATN(1)
Ok
? PI#
 3.1415926535898
Ok
```

A "cerquinha" (#) que você colocam após o nome da variável, indica que você quer o cálculo da variável em precisão dupla, reservando 8 posições de memória (8 bytes) para a "gaveta" correspondente (figura 4.11)

figura 4.11 -"Gaveta" para precisão dupla.



Só como curiosidade, se você conhece um pouco de trigonometria, deve ter percebido que, como 1 é a tangente de P1/4, o quadruplo do arco-tangente (ATN) de 1 é o próprio PI!

Limpe a tela e digite agora:

PI! = 4*ATN(1) (e [RETURN])

? PI! (e [RETURN])

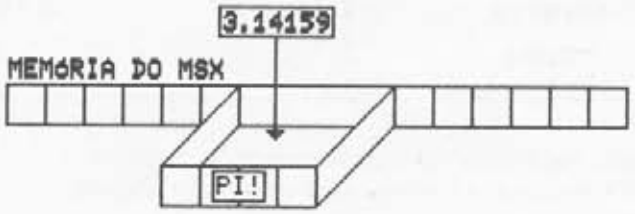
Vovê deverá obter a tela da figura 4.12.

figura 4.12 -Calculando o PI com precisão simples.

```
PI!=4*ATN(1)
Ok
? PI!
 3.14159
Ok
```

O ponto de exclamação que usamos como "sufixo" para o nome da variável, indica que ela pode ser armazenado em precisão simples, ocupando apenas 4 bytes na memória (figura 4.13)

figura 4.13 -"Gaveta" para precisão simples.



Obvialmente você obterá o valor de PI com uma precisão menor(menos casas).

Vamos finalizar nossa "pesquisa" sobre variáveis numéricas: limpe a tela e digite agora:

PI%=4*ATN(1) (e [RETURN])

?PI% (e [RETURN])

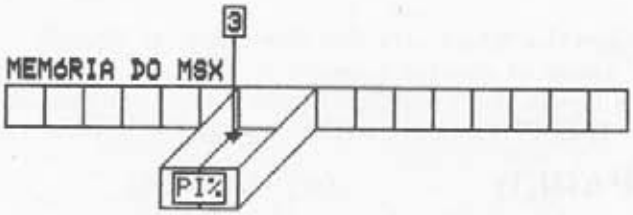
Você obterá a tela da figura 4.14.

figura 4.14 -Calculando o PI com precisão inteira.

```
PI%=4*ATN(1)
Ok
? PI%
 3
Ok
```

Como o sufixo "%" indica precisão inteira, a variável PI% ocupará apenas 2 posições de memória (2 bytes)e assim fazendo, perdemos todas as casas decimais(figura 4.15).

figura 4.15 -"Gaveta" para precisão inteira.



Resumindo, a precisão de uma variável numérica podem ser definida pelo sufixo que colocamos após seu nome, conforme a tabela da figura 4.16.

figura 4.16 -Os sufixos da precisão.

sufixo	precisão	bytes	exemplo: PI=4*ATN(1)
#	dupla	8	3.1415926535898
!	simples	4	3.14159
%	inteira	2	3

Obvialmente, você ja deve ter se perguntado: "e se eu não colocar sufixo nenhum?! Como você deve ser uma pessoa curiosa, com certeza ja digitou:

PI=4*ATN(1) **(e [RETURN])**

? PI **(e [RETURN])**

Obtendo a tela da figura 4.17.

figura 4.17 -A precisão "default" do MSX é dupla.

```
PI=4*ATN(1)
Ok
? PI
 3.1415926535898
Ok
```

Como você ja inferiu, se nenhum sufixo é colocado após o nome da variável, o seu EXPERT a assume como sendo de precisão dupla.

Esta é a situação "default", ou seja, é a situação que o micro assume na falta ("default") de indicações em contrário.

O MODO INTERATIVO E O MODO PROGRAMADO

Até agora usamos nosso Expert no chamado "MODO INTERATIVO", ou seja, demos ordens que foram imediatamente executadas. Até aqui você deve ter notado que há pouca diferença em se usar um micro ou uma simples máquina de calcular!

A grande diferença começa a ser sentida quando usamos o Expert no chamado "MODO PROGRAMADO".

Usa-lo neste modo, significa criar um programa, ou seja, uma sequência de ordens numeradas que serão executadas uma após a outra, obedecendo à sequencia de sua numeração.

Vamos então criar nosso primeiro programa em BASIC. Dê um RESET em seu Expert para "zerar" todas as gavetas e colocá-lo no mesmo estado em que ele estava no momento em que foi ligado. Para isso basta pressionar, simultaneamente, as teclas **[CONTROL] + [SHIFT] + [STOP]**. Chame o BASIC (opção 1), limpe a tela e digite:

10 PI! = 4*ATN(1) **(e [RETURN])**

Se você comandar, no modo interativo:

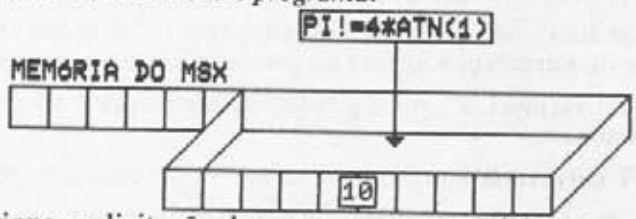
? PI! **(e [RETURN])**

Obterá "0", indicando que a gaveta do PI! está vazia!

Mas que confusão é essa!? Não acabamos de fazer o MSX calcular o valor de PI!?

Não! Ao colocarmos um número na frente do comando, ele não é executado: na realidade abrimos uma gaveta especial, cuja etiqueta é 10, na qual foi armazenada a ordem em questão (figura 4.18)

figura 4.18 -Armazenando "linhas" de um programa.



Vamos continuar a digitação do nosso programa. Digite Agora:

20 R=2.7 (e [RETURN])

Você acaba de abrir outra gaveta (com etiqueta "20") na qual foi armazenado a ordem seguinte. Para se certificar disso limpe a tela e comande.

LIST (e [RETURN])

Você deverá obter a tela da figura 4.19 mostrando a listagem do programa digitado até aqui.

figura 4.19 -Listando o programa

```
LIST
10 PI!=4*ATN(1)
20 R=2.7
Ok
```

Vamos completar nosso programa. Digite agora:

40 ? a (e [RETURN])

30 a=PI!*R^2 (e [RETURN])

Limpe a tela e aperte a tecla **[F9]** (**[SHIFT]** + **[F4]**). Você obterá a tela da figura 4.20.

figura 4.20 -Listagem do programa "completo".

```
list
10 PI!=4*ATN(1)
20 R=2.7
30 A=PI!*R^2
40 PRINT A
Ok
```

Bem, até agora podemos notar 4 coisas importantes:

1)A tecla F9 equivale a se digitar:

LIST (e [RETURN])

O uso das teclas de função pode economizar muito tempo de digitação!

2)O próprio micro foi suficientemente esperto (ou Expert!) para colocar as linhas em ordem crescente de numeração apesar de termos digitado a 40 e a 30 fora de ordem.

3)O nome da variável "a", foi digitado em minúscula e o Expert se encarregou de passá-lo para maiúscula.

4)O PRINT correspondente à abreviação "?", foi colocado por extenso.

Até agora, porém, não calculamos nada: simplesmente fizemos a "DIGITAÇÃO" de um programa. Para que ele seja executado devemos pedir ao micro para que "corra" (RUN) pelas linhas da listagem e as execute na sequência. Basta então digitar:

RUN (e [RETURN])

ou

[F5]

e obteremos o resultado, como mostra a figura 4.21.

figura 4.21 -Executando o programa.

```
list
10 PI!=4*ATN(1)
20 R=2.7
30 A=PI!*R^2
40 PRINT A
Ok
run
 22.9021911
Ok
```

Finalmente! Conseguimos a área do circulo de raio 2,7m!

EDITANDO UM PROGRAMA

Você há de convir que o programa digitado até agora é um pouco limitado: ele só calcula a área de círculos cujo raio mede 2.7m! E se o circulo tiver um raio de 9.2m, qual será sua área?

Limpe a tela, aperte **[F9]** para listar o programa e leve o cursor com as setas até a linha 20, na posição indicada na figura 4.22.

figura 4.22 -Alterando uma linha.

```
list
10 PI!=4*ATN(1)
20 R=2.7
30 A=PI!*R^2
40 PRINT A
Ok
```

Digite agora 9.2 para alterar o valor atribuido ao raio (R). Neste momento você terá a tela da figura 4.23.

figura 4.23 -O momento "crítico" dos distraídos .

```
list
10 PI!=4*ATN(1)
20 R=9.2
30 A=PI!*R^2
40 PRINT A
Ok
```

Agora...CUIDADO! Se você não apertar o **[RETURN]** após a alteração, ela só será efetuada na tela mas NÃO NA MEMÓRIA. Só para experimentar, não aperte o **[RETURN]**, leve o cursor até abaixo do Ok e aperte **[F9]**. Você obterá a tela da figura 4.24, mostrando que a listagem na memória não se alterou!

figura 4.24 -Após alterar uma linha, não esqueça o **[RETURN]***!*

```
list
10 PI!=4*ATN(1)
20 R=9.2
30 A=PI!*R^2
40 PRINT A
Ok
list
10 PI!=4*ATN(1)
20 R=2.7
30 A=PI!*R^2
40 PRINT A
Ok
```

Leve o cursor até a linha 20 da primeira listagem (em qualquer ponto dela) e aperte o **[RETURN]**.

Digitando [F9] e [F5] você verá a listagem alterada e o novo valor para a área (figura 4.25).

figura 4.25 -Calculando a àrea de outro círculo.

```
list
10 PI!=4*ATN(1)
20 R=9.2
30 A=PI!*R^2
40 PRINT A
Ok
run
 265.9041776
Ok
```

Uma outra maneira de se alterar uma linha de programa é simplesmente redigitá-la. Experimente digitar:

20 R=15457 (e [RETURN])

Aperte [F9] e [F5], você verá a listagem com a linha 20 alterada e terá o valor da área para o novo raio (figura 4.26)

figura 4.26 -Podemos alterar uma linha redigitando-a.

```
list
10 PI!=4*ATN(1)
20 R=9.2
30 A=PI!*R^2
40 PRINT A
Ok
run
 265.9041776
Ok
20 R=15457
list
10 PI!=4*ATN(1)
20 R=15457
30 A=PI!*R^2
40 PRINT A
Ok
run
 750585066.82991
Ok
```

O PROGRAMA "USER FRIENDLY"

Este processo de listar o programa e alterar uma linha do mesmo para cada novo valor de R é cansativo e não muito prático.

Vamos aprender mais um comando do BASIC que permite eliminar este inconveniente, digite:

20 INPUT R (e [RETURN])

Limpe a tela e aperte [F9] e [F5]. Ao invés de efetuar um cálculo de dar a resposta, o seu micro coloca um ponto de interrogação na tela e fica aguardando (figura 4.27)

figura 4.27 -O EXPERT está perguntando algo!

```
list
10 PI!=4*ATN(1)
20 INPUT R
30 A=PI!*R^2
40 PRINT A
Ok
run
?
```

Ao chegar na linha 20, o micro se defronta com a instrução INPUT R. Ela significa: micro, aguarde a digitação do valor de R! Só para experimentar, digite 4 (e **[RETURN]**).

Neste momento o micro assume 4 como sendo o valor de R e calcula a área do círculo correspondente (figura 4.28).

figura 4.28 -Calculando com o valor fornecido no INPUT.

```
list
10 PI!=4*ATN(1)
20 INPUT R
30 A=PI!*R^2
40 PRINT A
Ok
run
? 4
 50.26544
Ok
```

O programa ficou mais versátil, mas só é compreensível para você, que acompanhou sua construção. Se uma terceira pessoa rodar este programa, sem entender nada de BASIC, ficará olhando para o ponto de interrogação enquanto o micro ficará aguardando o valor de R. E o impasse continuará até que um dos dois se canse! Por isso, ao fazer um programa que eventualmente será usado por outra pessoa, devemos torná-lo um pouco mais "user friendly" ou seja, amistoso para o usuário.

Digite então as linhas da figura 4.29 (não esquecendo o **[RETURN]** no final de cada uma).

figura 4.29 -Tornando o programa "USER FRIENDLY".

```
12 CLS
14 PRINT"CÁLCULO DA ÁREA DE UM CÍRCULO"
16 PRINT"PELA FÓRMULA: A=πR² ":PRINT
20 INPUT"QUANTO VALE O RAIO";R
22 PRINT"π=";PI!
24 PRINT"R=";R
40 PRINT"ÁREA=";A
```

Atenção, na linha 16, a letra grega PI é obtida pressionando **[RGRA]** + **[SHIFT]** + **[P]** e o expoente 2 pressionando **[LGRA]** + **[SHIFT]** + **[2]**.

Limpe a tela e pressione **[F9]**: agora você entendeu porquê numeramos as linhas do programa original de 10 em 10 e não de 1 em 1. Foi para poder inserir linhas intermediárias nas futuras implementações! (figura 4.30)

figura 4.30 -A listagem com as novas linhas inseridas.

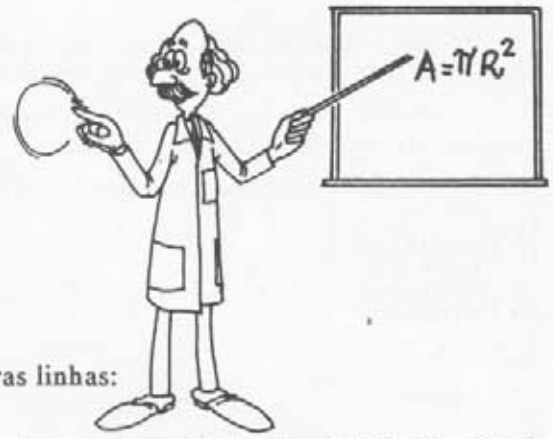
```
list
10 PI!=4*ATN(1)
12 CLS
14 PRINT"CÁLCULO DA ÁREA DE UM CÍRCULO"
16 PRINT"PELA FÓRMULA: A=πR² ":PRINT
20 INPUT"QUANTO VALE O RAIO";R
22 PRINT"π=";PI!
24 PRINT"R=";R
30 A=PI!*R^2
40 PRINT"ÁREA=";A
Ok
```

Agora rode o programa com **[F5]**. Ao ser questionado pelo micro com a mensagem: QUANTO VALE O RAIO? digite , por exemplo, 22.9 (e **[RETURN]**). Você deverá obter a tela da figura 4.31.

figura 4.31 -O novo programa "rodando"

```
CÁLCULO DA ÁREA DE UM CÍRCULO
PELA FÓRMULA: A=πR²

QUANTO VALE O RAIO? 22.9
π= 3.14159
R= 22.9
ÁREA= 1647.4812119
Ok
```



Vamos agora a uma explicação das novas linhas:

- 12 - Limpa a tela
- 14 - Imprime na tela a mensagem que está entre aspas, a partir da primeira linha livre. Toda vez que você mandar o micro imprimir alguma sequência de letras entre aspas, ele a imprimirá sem se preocupar com seu significado. Se você digitar:

PRINT R

ele imprimirá o valor da variável R.

Mas, se você digitar

PRINT "R"

ele imprimirá a letra R, sem se preocupar se ela é ou não o nome de uma variável. Imagine que, num teatro, um vizinho tagarela e incomoda falando alto. Se você lhe passar um bilhete com os dizeres:

FALE SUSSURRANDO

talvez ele passe a falar aos cochichos. Mas se o bilhete tiver a mensagem:

FALE "SUSSURRANDO"

você corre o risco de ouví-lo dizer, alto e bom tom:

SUSSURRANDO!

pois alguém, que fala alto na platéia de um teatro, costuma ter Q.I. de maçaneta!

- 16 - Imprime outra mensagem entre aspas. Aqui temos duas novidades: você pode dar mais de uma instrução por linha de programa, desde que as separe por "dois pontos"(:) e você pode comandar PRINT sem nada depois(em "informatiquês" dizemos "sem argumento"). O efeito deste PRINT sem nada é imprimir uma linha vazia.
- 20 - Mais uma novidade: o INPUT pode ser seguido de uma mensagem para indicar ao usuário do programa que variável está sendo esperada pelo micro. O nome da variável esperada pelo INPUT deve vir a seguir, separada por "ponto-e-vírgula"(;).
- 22 - Um mesmo PRINT pode ser usado para imprimir várias coisas, desde que separadas por "ponto-e-vírgula". Com esta separação elas são impressas uma após a outra.
- 24 - Como na 22. Note que, quando uma das coisas impressas é um número, o própio micro se encarrega de colocar um espaço vazio na frente.
- 40 - Veja comentário de 22 e 24.

Olhando a listagem do programa, a irregularidade na numeração das linhas pode ferir seu senso estético: do jeito que está dá a sensação de "programa remendado"! Não tem problema: seu Expert pode resolver este problema num piscar de olhos! Digite:

RENUM (e [RETURN])

limpe a tela e liste o programa: você obterá a listagem da figura 4.32.

figura 4.32 -O programa RENUMerado.

```
list
10 PI!=4*ATN(1)
20 CLS
30 PRINT"CÁLCULO DA ÁREA DE UM CÍRCULO"
40 PRINT"PELA FÓRMULA: A=πR²":PRINT
50 INPUT"QUANTO VALE O RAIO";R
60 PRINT"π=";PI!
70 PRINT"R=";R
80 A=PI!*R^2
90 PRINT"ÁREA=";A
Ok
```

Consulte o dicionário do capítulo 8 para aprender mais recursos do RENUM.

O PROGRAMA "IDIOT PROOF"

A rigor não existe nada, neste mundo, a prova de idiotas("idiot proof"): eles são muito mais criativos do que possamos imaginar ou prever!

O BASIC MSX usa, por conta própria, algumas técnicas que tentam impedir "abobrinhas". A própria mensagem:

Syntax error

que já vimos é uma tentativa neste sentido.

Imagine, agora, que você criou o programa de figura 4.32 para que ele seja usado pelo seu vizinho de teatro (lembra? Aquele que falava "sussurando").

Ao receber a mensagem:

QUANTO VALE O RAIO?

digamos que ele digite:

RAIO (e [RETURN])

Experimente fazer isso. Como o micro está esperando um número e não um nome, emite a mensagem:

? Redo From Start

e repete a instrução INPUT, tantas vezes quantas forem necessárias.

Imagine agora que seu vizinho de teatro, após algumas tentativas, resolva apertar simplesmente o **[RETURN]**.

Ele obterá a tela da figura 4.33.

figura 4.33 -A "criatividade" em ação.

```
CÁLCULO DA ÁREA DE UM CÍRCULO
PELA FÓRMULA: A=πR²

QUANTO VALE O RAIO? RAIO
?Redo from start
QUANTO VALE O RAIO? RAIO
?Redo from start
QUANTO VALE O RAIO? RAIO
?Redo from start
QUANTO VALE O RAIO?
π= 3.14159
R= 0
ÁREA= 0
Ok
```

Ao receber um **[RETURN]**, o micro assumiu o valor 0 para R e, obviamente, forneceu um resultado 0, pois, o seu vizinho está tentando calcular a área de um ponto!

Vamos ajudar o Expert em sua tarefa de prevenir bobagens.

Limpe a tela liste o programa, e acrescente as linhas adicionais da figura 4.34 (não esquecendo o **[RETURN]** no fim de cada uma!).

figura 4.34 -Implementando uma SUB-ROTINA.


```
55 IF R=0 THEN GOSUB 100 ELSE GOTO 60
95 END
100 REM ROTINA "IDIOT PROOF"
110 PRINT:PRINT"EU DISSE:"
120 RETURN 50
```

Limpe a tela e liste o programa, para acompanhar a análise das novas linhas:

- 55 -Nesta linha, aprendemos uma estrutura importantíssima em programação: a tomada de decisões por parte do micro.

  A tradução da linha em questão séria:

  "SE (IF) R for igual a zero (R = 0) ENTÃO (THEN) DESVIE a execução do programa para a SUB-ROTINA que começa na linha 100 (GOSUB 100), SE NÃO (ELSE) VÁ PARA a linha 60 (GOTO 60)."

  O efeito desta instrução é o seguinte: se o seu vizinho digitou 0 ou simplesmente **[RETURN]**, o raio será nulo e a execução desvia para 100. Se digitou um valor numérico, a execução prossegue normalmente (linha 60 em diante).
- 100 -Esta linha não faz nada: o REM (de REMark = nota, observação) serve para que o programador (e não o usuário), possa fazer anotações na listagem, para se lembrar do que faz cada trecho do programa.
- 110 -Pula uma linha e imprime a mensagem entre aspas.
- 120 -Faz a execução retornar (RETURN) à listagem principal, na linha indicada. Neste caso, como queremos a repetição do INPUT, retornamos à linha 50.
- 95 -Encerra o programa principal. Se não colocássemos esta linha, a execução invadiria a subrotina.

Rode agora o programa algumas vêzes tentamdo simular o comportamento do vizinho com QI de poça d'água.

Como a melhor forma de se lidar com alguém como seu vizinho é usar o senso de humor (e nunca se irritar!), acrescente as linhas da figura 4.35, sem esquecer o **[RETURN]** no fim de cada uma, e rode o programa.

Na hora de entrar com o raio, aperte **[RETURN]** em todas as tentativas para ver o que acontece.

figura 4.35 -Um pouco de "humor".

```
101 PRINT
102 IF F=3 THEN PRINT"DESISTO!":LIST
104 IF F=2 THEN PRINT"OH! MEU! VOCÊ BUZI
NA EM TÚNEL ?"
106 IF F=1 THEN PRINT"INSISTO!"
115 F=F+1
```

Note que uma linha de programa, pode ocupar mais que uma linha de tela, como no caso da linha 104.

Agora você já tem condições de entender todas as linhas novas menos, talvez, a 115: ela pega o valor de F (que inicialmente é 0 pois não foi definido anteriormente nenhuma variável com este nome) e lhe acrescenta 1 toda vez que a subrotina é usada. Isso permite graduar a ênfase das mensagens conforme o aumento de teimosia do vizinho de teatro.

Digite agora:

RENUM 100

e liste o programa (figura 4.36)

figura 4.36 -Programa renumerado com RENUM 100.

```
100 PI!=4*ATN(1)
110 CLS
120 PRINT"CÁLCULO DA ÁREA DE UM CÍRCULO"
130 PRINT"PELA FÓRMULA: A=πR²":PRINT
140 INPUT"QUANTO VALE O RAIO";R
150 IF R=0 THEN GOSUB 190 ELSE GOTO 170
160 PRINT"π=";PI!
170 PRINT"R=";R
180 A=PI!*R^2
190 PRINT"ÁREA=";A
200 END
210 REM ROTINA "IDIOT PROOF"
220 PRINT
230 IF F=3 THEN PRINT"DESISTO!":LIST
240 IF F=2 THEN PRINT"OH! MEU! VOCÊ BUZINA EM TÚNEL ?"
250 IF F=1 THEN PRINT"INSISTO!"
260 PRINT:PRINT"EU DISSE:"
270 F=F+1
280 RETURN 150
Ok
■
```

Note que a renumeração começou, agora, com o número 100 e que todos os números de linhas nos GOTO, GOSUB e RETURN foram automaticamente rearranjados! Cá entre nós: Seu Expert tem um QI bem maior que o do seu vizinho!

ARMAZENANDO O PROGRAMA

Agora você tem um programa suficientemente longo para ser guardado. Se você tem um DATA CORDER ou um gravador cassete, ele poderá ser gravado em fita para posterior recuperação . Neste caso vá até o capítulo 7 e leia atentamente o item GRAVANDO E LENDO PROGRAMAS EM BASIC.

Se você tem uma impressora, poderá imprimir a listagem de seu programa, para documentação, com o comando.

LLIST (e [RETURN])

Talvez você tenha algum problema com a impressão das letras acentuadas e caracteres especiais das mensagens entre aspas. Basta lembrar que o seu Expert envia caracteres à impressora segundo duas tabelas de códigos: ABNT ou MSX (consulte o apêndice B para maiores detalhes). Certifique-se qual tabela sua impressora está apta a receber e ative a tabela correspondente do seu Expert conforme as instruções do apêndice B.

Agora que seu programa está gravado e/ou listado, podemos desligar o micro para descansar um pouco.

Se você quiser continuar em frente, terá que limpar a memória do Expert para poder digitar novos programas que vêm a seguir. Isto pode ser feito de 5 maneiras:

1) Digitar um NEW (e **[RETURN]**). É o procedimento mais prático mas nem sempre funciona. Há certos programas que, ao rodar, não permitem interrupção, não devolvendo o cursor para que possamos dar o comando citado.

2) Digitar **[CONTROL]** + **[SHIFT]** + **[STOP]** simultaneamente.

O micro sofre um RESET (re-inicialização) e se coloca na situação em que estaria se acabando de ser ligado. Em alguns casos, porém, o programa presente na memória desliga a leitura do teclado e o micro não descobre que as teclas citadas foram pressionadas.

3) Apertar o botão RESET na traseira do micro. É um pouco incômodo, mas funciona se os dois modos anteriores foram ineficientes.

4) Empurrar a tampa protetora de um dos slots frontais. O micro desliga mas os periféricos ligados nas tomadas **SWITCHED OUTLET**, não. Desta forma você pode preservar dados armazenados, por exemplo, na impressora. Obviamente este procedimento não funciona se os dois SLOTS estiverem ocupados com cartuchos e/ou interfaces de periféricos.

5) Desligar tudo no botão POWER. Neste caso, estamos desligando o micro e tudo que estiver conectado nas tomadas do SWITCHED OUTLET. Não é um procedimento que deva ser usado frequentemente.

USANDO A SCREEN 1

Você poderia se perguntar: " Se tenho a SCREEN 0 que tem 40 colunas e eu posso até configurá-la para 32 usando WIDTH 32, para que preciso de uma SCREEN específica para 32 colunas ?".

A resposta é simples: ligue seu micro e limpe a tela.

Agora pressione a combinação de teclas **[LGRA]** + **[M]** para obter o símbolo de masculino da biologia. Afaste o cursor com a seta para direita e observe atentamente a tela. O símbolo aparece cortado!

Agora comande (no modo direto):

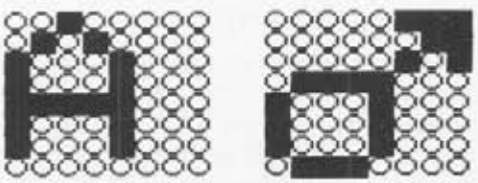
SCREEN 1 (e [RETURN])

Novamente digite **[LGRA]** + **[M]** : agora você obterá o símbolo completo.

Isso ocorre porque, apesar de todos os caracteres do MSX serem definidos uma matriz de 8x8 pontos (pixels, em "informatiquês"), a SCREEN 0 mostra apenas 6 na horizontal, enquanto que a SCREEN 1 mostra os 8.

Para as letras, a SCREEN0 não resulta em cortes porque todas elas tem uma largura máxima de 5 pixels (+1 para espaçamento) enquanto que outros caracteres podem ocupar todos os 8 pixels na horizontal (figura 4.37).

figura 4.37



OS CARACTERES DO MSX

No MSX temos 256 caracteres diferentes.

Para ver alguns deles, digite o programa da figura 4.38.

figura 4.38 - A função CHR$

```
100 SCREEN 1
110 FOR C=65 TO 77
120 PRINT"CóDIGO=";C,
130 PRINT"CARACTERE=";CHR$(C)
140 NEXT C
```

Rodando o programa, você obterá a tela da figura 4.39.

figura 4.39 - Listando alguns caracteres e seus códigos.

```
CóDIGO=  65      CARACTERE=A
CóDIGO=  66      CARACTERE=B
CóDIGO=  67      CARACTERE=C
CóDIGO=  68      CARACTERE=D
CóDIGO=  69      CARACTERE=E
CóDIGO=  70      CARACTERE=F
CóDIGO=  71      CARACTERE=G
CóDIGO=  72      CARACTERE=H
CóDIGO=  73      CARACTERE=I
CóDIGO=  74      CARACTERE=J
CóDIGO=  75      CARACTERE=K
CóDIGO=  76      CARACTERE=L
CóDIGO=  77      CARACTERE=M
```

Vamos à explicação:

- 100 Chama a SCREEN 1
- 110 Estabelece um laço(LOOP): uma outra estrutura importantíssima em programação.
  Esta linha dá, ao seu Expert, a seginte instrução: "Assuma inicialmente o valor 65 para a variável C(FOR C=65) e execute o programa que vem a seguir até encontrar um NEXT C(próximo C). Lá chegando, volte para cá para buscar o próximo valor de C(C+1). Repita a operação até o C valer 77 (TO 77).
- 120 Imprime uma mensagem e o valor de C. Note que, ao colocarmos uma vírgula(",") após o PRINT, estamos instruindo o Expert para que comece o próximo PRINT na metade da mesma linha.
- 130 Nesta linha pedimos para imprimir, logo após a mensagem (","), o caractere cujo código é C: CHR$(C)
- 140 Fecha o laço aberto na linha 110.
  Note que, se no programa há uma linha com FOR...TO..., OBRIGATÓRIAMENTE num outro ponto do programa deve haver o NEXT!

Se você quiser que a variável do laço não seja incrementada de 1 em 1, mas sim em "degraus" diferentes, basta especificar isso com STEP. Exemplificando, altere a linha 110 para:

110 FOR C=65 TO 77 STEP 4

e rode o programa para ver o que acontece. Você deverá obter a tela da figura 4.40.

figura 4.40 - Aumentando o "degrau" do laço

```
CóDIGO=  65      CARACTERE=A
CóDIGO=  69      CARACTERE=E
CóDIGO=  73      CARACTERE=I
CóDIGO=  77      CARACTERE=M
```

Mas, até agora, só descobrimos alguns caracteres do MSX.

Para vê-los todos,vamos digitar um curto programinha. Obviamente usaremos a SCREEN 1 para poder ver todos os caracteres integralmente. Digite o programa de figura 4.41

figura 4.41

```
100 N=0:SCREEN 1
110 FOR I=6212 TO 6692 STEP 32
120 FOR K=0 TO 15
130 VPOKE I+K,N
140 N=N+1
150 NEXT K,I
```

Agora rode o programa. Pronto! Na SCREEN 1 você tem em 16 linhas e 16 colunas seus 256 caracteres do MSX (figura 4.42)

figura 4.42

Antes de continuar, dê umas estudadinha em tudo que você tem a disposição: símbolos matemáticos, caracteres semi-gráficos, naipes do baralho, até um pouco do alfabeto grego!

Não se preocupe, por enquanto, em entender o programinha da figura 4.41. Basta saber que ele escreve diretamente na memória de vídeo (VPOKE) os códigos (N) dos caracteres de 0 a 255. O último, inclusive, é o caracter do cursor.

Passeie com o cursor (usando as setas) pela tabela e fique observando o último caractere!

Vamos agora aprender mais alguns truques: limpe a tela e digite

KEY 1, "SCREEN 0:LIST "+CHR$ (13) (e [RETURN])

Olhe o rodapé da tela: sabe o que aconteceu ?

Você reprogramou o conteúdo de tecla **[F1]**. O CHR$ (13) que aparece no fim, corresponde ao **[RETURN]**.

Aperte **[F1]**: o que aconteceu ? A sequência de ordens contidos na tecla **[F1]** foi executada, inclusive o **[RETURN]**!

Aperte **[F5]** para rodar o programa e, ao terminar, **[F1]**. cômodo, não ?

Para achar um caracter em nossa tabelinha, podemos usar a técnica da batalha naval, numerando colunas e linhas. Como temos 16 linhas x 16 colunas podemos usar numeração hexadecimal (não se preocupe se você não conhece esta numeração, o MSX a conhece muito bem!).

Acrescente algumas linhas ao programa de figura 4.41 de maneira a ficar como na figura 4.43

figura 4.43

```
100 N=0:SCREEN 1
110 FOR I=6212 TO 6692 STEP 32
120 FOR K=0 TO 15
130 VPOKE I+K,N
140 N=N+1
150 NEXT K,I
160 LOCATE 2,0
170 A$="0123456789ABCDEF"
180 PRINT A$
190 PRINT
200 FOR I=1 TO 16
210 PRINT MID$(A$,I,1)
220 NEXT I
230 PRINT
```

Antes de rodá-lo, vamos a uma rápida explicação:

O LOCATE 2,0 de linha 160, localiza o cursor na coluna 2, linha 0.

A$ é uma variável "string" ou alfa-numérica.

Quando o nome da variável é seguido pelo cifrão ($) a "gaveta" correspondente na memória não armazena dados númericos mas sim caracteres (no caso os caracteres do dígitos hexadecimais).

O laço de 200 a 220 vai "fatiando" o conteúdo de A$ com a função MID$ e imprimindo os caracteres nela contidos, 1 por 1. No próximo ítem veremos isso com mais deta-lhes.

Rode o programa e você obterá a tela da figura 4.44

figura 4.44

```
0123456789ABCDEF
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
Ok
```

Digamos agora, que você queira obter o caractere correspondente ao xadrezinho que está na linha D, coluna 7 da nossa tabelinha. Leve (com as setas) o cursor até o começo da linha do Ok e digite:

PRINT CHR$ (&HD7) **(e [RETURN])**

Você obterá o caracter desejado na linha de baixo. Você comandou: "escreva na tela (PRINT) o caractere (CHR$) cujo código, em hexadecimal (&H) é D7".

Vamos obter o ponto de interrogação de ponta cabeça? Este, por exemplo, é um caractere fundamental para escrever textos em espanhol.

Achou? ótimo! Agora é só levar o cursor até a linha que você acabou de digitar e substituir o D7 por A8 (e **[RETURN]**).

Agora vamos tentar descobrir como imprimir o símbolo de masculino que usamos no começo deste ítem: ele está na linha 0, coluna B da tabelinha. Leve o cursor até a linha do PRINT e substitua o A8 por 0B. Aperte o **[RETURN]**.

Epa! Que coisa estranha! Ao invés de produzir o caracter desejado, o cursor pulou lá para cima!

Isso acontece porque os 32 primeiros caracteres do MSX (linha 0 e 1 da tabelinha) são usados, também, como caracteres de controle. Você já viu, por exemplo, que o CHR$(13) corresponde ao RETURN.

No caso destes 32 caracteres, para avisar ao MSX que queremos um caracter e não uma ação de contrôle, devemos enviar antes o CHR$ (1) e a seguir o código que nos interessa acrescido de 64. Vamos tentar isso? Altere a linha do PRINT para

PRINT CHR$ (1) + CHR$ (64+&H0B)

e aperte o **[RETURN]**. Viu?

Mude o 0B por 0E: Que tal? Funcionou!

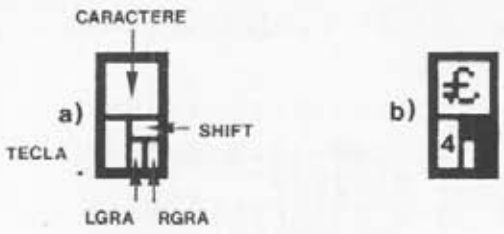
Você vai dizer "Isso é muito interessante mas não é um pouco complicado ?".

Na realidade todos estes caracteres podem ser obtidos através do teclado usando as teclas **[LGRA]**, **[RGRA]** e **[SHIFT]** .

Veja a tabela da figura 4.47, parecida com a que obtivemos na tela até agora.

Cada caractere está dentro de um retângulo com o seguinte código:

figura 4.45



Se você quiser, por exemplo obter o símbolo de libra no teclado, o CHR$ (&H9C), deverá pressionar as tecla **[SHIFT]** + **[RGRA]** + **[4]**.

Para consultas rápidas, esta tabela está também impressa na última contra-capa deste livro. Na primeira contra-capa está a reprodução do teclado com todos os símbolos possíveis em cada tecla, obedecendo ao código da figura 4.46 .

figura 4.46

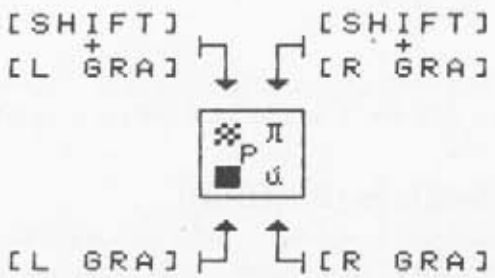
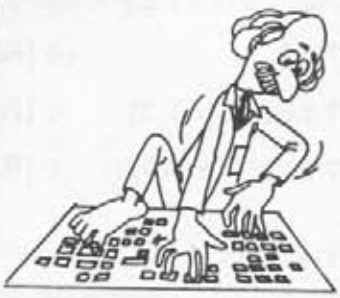


figura 4.47



Use aquela com a qual você se familiarizar melhor.

AS VARIÁVEIS TIPO STRING

Agora grave este programa se quiser usá-lo futuramente e limpe a memória do Expert com um NEW. Vamos aprender mais alguma coisa sobre as variáveis que usam o sufixo "cifrão" ($), chamadas de variáveis alfanuméricas ou variáveis "string".

Digite:

A$="ABCDEFG" **(e [RETURN])**

? A$ **(e [RETURN])**

? LEN(A$) **(e [RETURN])**

Você deve obter a tela da figura 4.48.

figura 4.48 - Variável string e seu comprimento

```
A$="ABCDEFG"
Ok
? A$
ABCDEFG
Ok
? LEN(A$)
 7
Ok
```

O Expert abriu, em sua memória, uma "gaveta" com a etiqueta A$ e armazenou nela a sequência de caracteres entre aspas. Como você deve ter notado a função LEN dá o "comprimento"(LENght) da string, ou seja o número de caracteres que ela tem.

Limpe a tela e digite agora

? A$ **(e [RETURN])**

? LEFT$(A$,3) **(e [RETURN])**

? RIGHT$(A$,2) **(e [RETURN])**

Você obterá a tela da figura 4.49.

figura 4.49 - Pegando "pedaços" de uma string

```
? A$
ABCDEFG
Ok
? LEFT$(A$,3)
ABC
Ok
? RIGHT$(A$,2)
FG
Ok
```

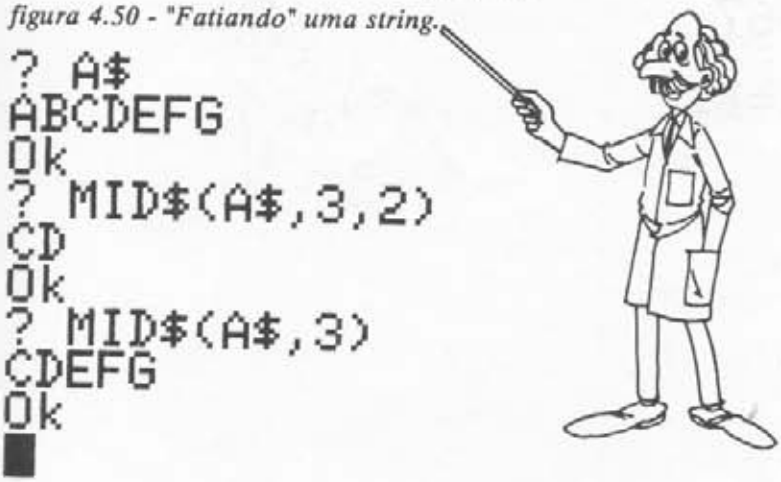
Como você deve ter notado LEFT$(A$,N) pega os primeiros N caracteres da esquerda(LEFT) e RIGHT$(A$,M) os últimos M caracteres da direita(RIGHT).

E se quisermos alguns caracteres no MEIO(MIDdle) da string? Limpe a tela e digite

```
? A$                    (e [RETURN])
? MID$(A$,3,2)          (e [RETURN])
? MID$(A$,3)            (e [RETURN])
```

Você obterá a tela da figura 4.50.

figura 4.50 - "Fatiando" uma string.



A função MID$(A$,N,X) pega X caracteres a partir do Nésimo. Se X for omitido, ele pegará todos do Nésimo ao último.

As string, apesar de não conterem números, podem ser "operadas". Limpe a tela e digite a seguinte sequência de comandos:

```
B$="1234567"            (e [RETURN])
C$=A$+B$                (e [RETURN])
?C$                     (e [RETURN])
MID$(A$,3,2)=B$         (e [RETURN])
?A$                     (e [RETURN])
?VAL(B$)*2              (e [RETURN])
?VAL(A$)                (e [RETURN])
```

Você obteve, certamente, a tela mostrada na figura 4.51 .

figura 4.51 - "Operações" com strings

```
B$="1234567"
Ok
C$=A$+B$
Ok
? C$
ABCDEFG1234567
Ok
MID$(A$,3,2)=B$
Ok
? A$
AB12EFG
Ok
? VAL(B$)*2
 2469134
Ok
? VAL(A$)
 0
Ok
```

Como podemos perceber, duas strings podem ser "emendadas" com o operador "+". Além disso podemos "enxertar" um pedaço de uma string em outra: quando fizemos MID$(A$,3,2) = B$ mandamos substituir 2 caracteres de A$ pelos 2 primeiros caracteres de B$, a partir da posição 3. Finalmente, podemos transformar uma string cheia de algarismos num número de verdade com a função VAL. Note que VAL(A$) deu zero porque ela não começa com algarismos. A recíproca do VAL é STR$ que transforma um número numa sequência de algarismos.

AS VARIÁVEIS TIPO MATRIZ

Para encerrar este capítulo, vamos aprender outro tipo de variável: as MATRIZES.

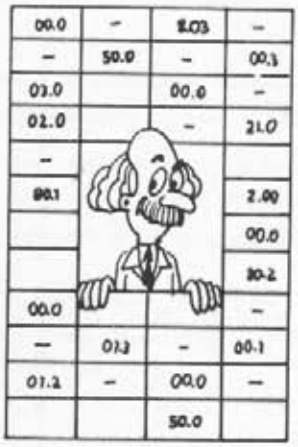
As matrizes nada mais são que variáveis com um mesmo nome que se distinguem uma da outra pelos índices. Elas são muito úteis para formação e manipulação de tabelas.

Digamos, por exemplo, que você queira montar um programa que dá o signo do zodíaco em função da data de nascimento.

Digite o programa da figura 4.52 e rode-o algumas vezes para checar se está tudo correto e perceber como ele funciona.

figura 4.52 - Expert em Astrologia!

```
100 REM ----------------------
110 REM LEITURA DAS LINHAS DATA
120 REM ----------------------
130 SCREEN 1:KEY OFF
140 DIM D$(13),S$(12),M$(12),C$(12)
150 FOR I=1 TO 12
160 READ D$(I),S$(I),M$(I),C$(I)
170 NEXT I
180 D$(13)="1232"
200 REM ----------------------
210 REM CONTEÚDO DE LINHAS DATA
220 REM ----------------------
230 DATA 0120,AQUÁRIO,JANEIRO,audaciosa
e sensitiva.
240 DATA 0220,PEIXES,FEVEREIRO,imaginati
va e romântica.
250 DATA 0321,ÁRIES,MARÇO,de espírito au
enturoso.
260 DATA 0421,TOURO,ABRIL,firme paciente
 e sensível.
270 DATA 0521,GÊMEOS,MAIO,curiosa e comu
nicativa.
280 DATA 0622,CÂNCER,JUNHO,quieta reserv
ada e sensível.
290 DATA 0723,LEÃO,JULHO,generosa e bond
osa. Liderança.
300 DATA 0823,VIRGEM,AGOSTO,prática e or
ganizada.
310 DATA 0923,LIBRA,SETEMBRO,agradável e
 amável.
320 DATA 1023,ESCORPIÃO,OUTUBRO,de forte
 espírito  de luta.
330 DATA 1122,SAGITÁRIO,NOVEMBRO,otimist
a e independente.
340 DATA 1221,CAPRICÓRNIO,DEZEMBRO,séria
  e disciplinada.
400 REM ----------------------
410 REM ROTINA DE ENTRADA DADOS
420 REM ----------------------
430 PRINT "* MICROR6SCOPO *":PRINT
440 PRINT "Em que dia você nasceu"
450 INPUT "(USE 2 DíGITOS)";DI$
460 IF LEN(DI$)<>2 OR VAL(DI$)<1 OR VAL(
DI$)>31 THEN 450
470 PRINT "Em que mês você nasceu"
480 INPUT "(USE 2 DíGITOS)";ME$
490 IF LEN(ME$)<>2 OR VAL(ME$)<1 OR VAL(
ME$)>12 THEN 480
500 REM ----------------------
510 REM ROTINA DE PROCESSAMENTO
520 REM ----------------------
530 U$=M$(VAL(ME$)):F$=ME$+DI$
540 FOR I=1 TO 12
550 IF F$<D$(I) THEN I=12:GOSUB 700
560 IF F$>=D$(I) AND F$<D$(I+1) THEN GOS
UB 700
570 NEXT I
580 END
700 REM ------------------
710 REM ROTINA DE IMPRESSÃO
720 REM ------------------
730 PRINT:PRINT"Se você nasceu no dia:"
740 PRINT DI$;" de ";U$;", então"
750 PRINT "seu signo é ";S$(I);", e"
760 PRINT "você deve ser uma pessoa"
770 PRINT C$(I):PRINT
800 REM ----------------------
810 REM ROTINA DE NOVA CONSULTA
820 REM ----------------------
830 PRINT "NOVA CONSULTA ? (S/N)
840 R$=INPUT$(1)
850 IF R$="S" OR R$="s" THEN RUN 100
860 PRINT:PRINT"CONSULTA ENCERRADA!"
```



Vamos às novidades:

- 140 Nesta linha "dimensionamos", ou seja, reservamos na memória espaço para 4 matrizes:
  D$(N) - Contém a data em que começa o signo N, na forma MES-DIA.
  S$(N) - Contém o nome dos signos N
  M$(N) - Contém o nome do mês N
  C$(N) - Contém as características do nativo do signo N
- 150-170 Neste laço são lidos os conteúdos das matrizes citadas, que estão armazenadas nas linhas DATA(de 240 a 340)
- 460 Após a entrada do dia(DI$) checa se foram usados dois dígitos e se o dia está entre 1 e 31 - A linha 490 faz o mesmo para o mês.
- 530 Seleciona o nome do mês e cria uma string(F$) de 4 dígitos(MMDD) para poder comparar com D$(I)
- 560 Compara F$ com D$(I) para determinar o signo. Uma vez achado, vai para a rotina 700. Note que se a data de nascimento for anterior a 20 de janeiro, cai em CAPRICóRNIO(12): esta é a razão da linha 550. Note também que, quando I = 12, I + 1 = 13: esta é a razão da linha 180.
- 840 Aguarde a digitação de uma string de 1 caractere. Note que as matrizes podem também ser numéricas e ter mais de um índice.

Grave seu programa de "astrologia", limpe a memória do micro com um NEW e digite o programa da figura 4.53, para encerrarmos este capítulo.

figura 4.53 - Montando uma matriz bidimensional

```
100 SCREEN 1:KEY OFF:WIDTH 32
110 DIM X(9,9):FOR I= 1 TO 9
130 PRINT CHR$(198);:FOR J=1 TO 9
140 X(I,J)=I*J
150 PRINT USING"##";X(I,J);
155 PRINTCHR$(1)+CHR$(198);
160 NEXT J
170 PRINT:PRINT STRING$(27,195)
180 NEXT I
```

Com isso você montou uma matriz de 9x9 posições e preencheu cada elemento com o produto dos índices. Rodando o programa você obterá a famosa "TABOADA"(lembra?)

figura 4.54 - A "TABOADA" no Expert.

```
| 1| 2| 3| 4| 5| 6| 7| 8| 9|
| 2| 4| 6| 8|10|12|14|16|18|
| 3| 6| 9|12|15|18|21|24|27|
| 4| 8|12|16|20|24|28|32|36|
| 5|10|15|20|25|30|35|40|45|
| 6|12|18|24|30|36|42|48|54|
| 7|14|21|28|35|42|49|56|63|
| 8|16|24|32|40|48|56|64|72|
| 9|18|27|36|45|54|63|72|81|
Ok
```

DESENHANDO E PINTANDO

5

USANDO A SCREEN 2

A tela de desenhos em alta resolução é a SCREEN 2. Ela permite desenhar 256 x 192 = 49152 pontos! Além disso o BASIC MSX tem recursos incríveis para se desenhar nela.

DESENHANDO COM RÉGUA E COMPASSO

Vamos começar a explora-lá um pouco. Ligue seu Expert e digite o programa da figura 5.1

figura 5.1

```
100 SCREEN 2:OPEN"GRP:" AS #1
110 PSET (10,10)
120 PRINT #1," (10,10)"
900 GOTO 900
Ok
```

A linha 100 chama a SCREEN 2 e abre (OPEN) um arquivo na tela gráfica (GRP:) como (AS) número(#) 1. Isso permite que,além desenhar na tela gráfica, possamos também escrever nela.

A linha 110 marca um ponto (Point SET) nas coordenandas X = 10, Y = 10. Note que enquanto o X cresce para direita (até 255) o y cresce para baixo (até 191), ao contrário do que acontece nos gráficos cartesianos normais.

A linha 120 escreve no arquivo número 1 que acabamos de abrir (PRINT#1) a mensagem entre aspas (que no nosso caso dá as coordenadas do ponto). Note que a mensagem é escrita a partir do último ponto setado na tela.

A linha 900 faz o papel de um cachorro tentando morder seu próprio rabo. Se ela não existisse, ao terminar a execução o MSX voltaria para a tela de texto, destruindo o desenho. Porisso, para obter sua listagem de volta aperte **[CONTROL]** + **[STOP]** e a seguir **[F4]** (e **[RETURN]**).

Você deve obter a tela da figura 5.2

figura 5.2

```
 (10,10)
```

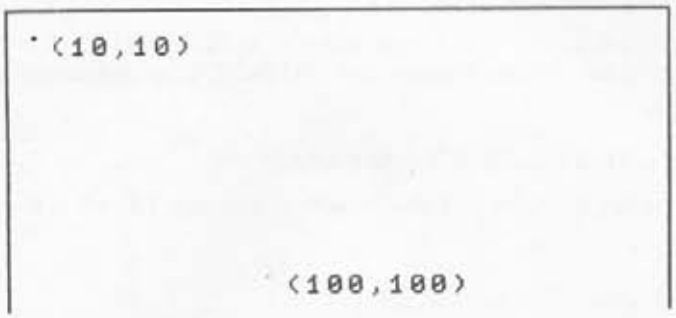
Agora acrescente algumas linhas ao seu programa, para que fique como a listagem da figura 5.3

figura 5.3

```
100 SCREEN 2:OPEN"GRP:" AS #1
110 PSET (10,10)
120 PRINT #1," (10,10)"
130 PSET (100,100)
140 PRINT #1," (100,100)"
900 GOTO 900
```

Rode o programa: Agora você "plotou" dois pontos (figura 5.4).

figura 5.4



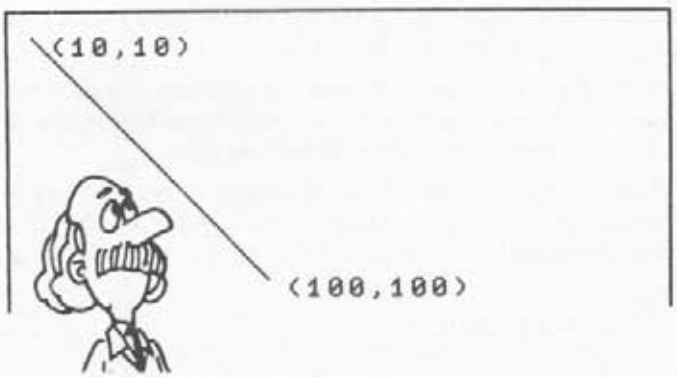
Experimente agora acrescentar a linha

150 LINE (10,10) – (100,100)

e rode o programa . Percebeu o que o LINE faz ? Desenha uma linha que vai do ponto (10,10) ao ponto (100,100).

Você deve obter uma tela como a da figura 5.5.

figura 5.5



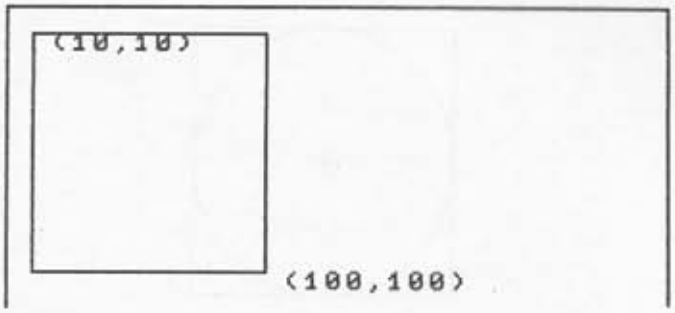
Mas não é só isso que o LINE faz!

Liste o programa e altere a linha 150 para

150 LINE (10,10) – (100,100),, **B**

Como você deve ter percebido, o B é de Box (caixa). Agora a tela ficou como na figura 5.6

figura 5.6



Novamente altere a linha 150 para

150 LINE (10,100)—(100,100),, BF

sendo BF de Box Full (caixa cheia). A tela agora ficará com o aspecto da figura 5.7

figura 5.7

Muito bem! Vamos aprender mais coisas sobre desenhos. Liste o programa , apagando as linhas de 100 a 150 comandando

DELETE 110-150 **(e [RETURN])**

Liste de novo para ver o efeito!

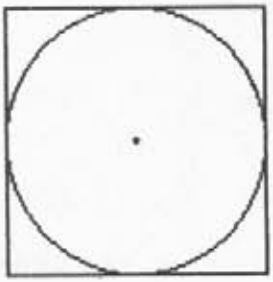
Agora digite as linhas adicionais indicadas na figura 5.8

figura 5.8

```
100 SCREEN 2:OPEN"GRP:" AS #1
110 PSET (100,100)
120 FOR T=0 TO 1000: NEXT T
130 LINE (50,50)-(150,150),,B
140 FOR T=0 TO 1000: NEXT T
150 CIRCLE (100,100),50
900 GOTO 900
```

As linhas 110 e 130 dispensam comentários. As linhas 120 e 140 são "laços temporizadores",servem apenas para dar um pouco de "suspense". A novidade está na linha 150. Ela diz ao micro: "Desenhe um círculo com centro em (100,100) e raio 50. A tela deve ficar como na figura 5.9

figura 5.9



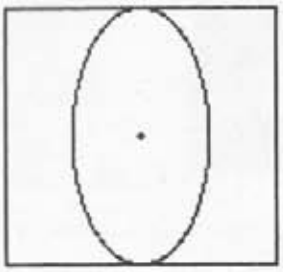
Agora altere a linha 150 para:

```
150 CIRCLE (100,100 ), 50,,,,2
```

(conte as vírgulas direitinho!)

Você desenha uma elípse cujo eixo vertical é 2 vezes maior que o horizontal (figura 5.10)

figura 5.10

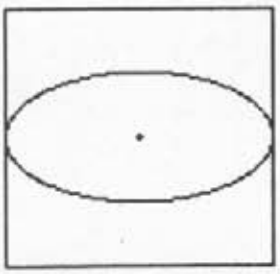


Se você tivesse digitado

```
150 CIRCLE (100,100), 50,,,,0.5
```

teria obtido a figura 5.11

figura 5.11



Vamos ver agora para que serve aquele monte de vírgulas. Como você deve lembrar, uma circuferência completa tem 360° ou 2 * PI radianos. Como o MSX não sabe o quanto vale PI, vamos contar para ele.

Digite a linha

```
145 PI = 4 * ATN (1)
```

Lembra? O arco tangente (ATN) de 1 é PI/4.

Agora altere a linha 150 para:

```
150 CIRCLE(100,100),50,,0,PI/2
```

Isso mesmo! O CIRCLE pode traçar arcos!

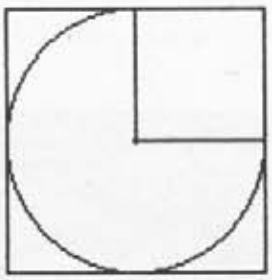
No nosso caso de 0 radianos (0) a PI/2 radianos (90).

Altere agora a linha 150 para

```
150 CIRCLE (100,100), 50 ,, -PI/2, -2* PI
```

Como você pode notar na figura 5.12, o CIRCLE também traça setores (se os arcos forem negativos). Você já deve estar pensando nos seus gráficos tipo "pizza"!

figura 5.12



Calma ! Vamos aprender um último recurso da tela gráfica.

Acrescente mais linhas ao seu programa de maneira a que ele fique como na figura 5.13

figura 5.13

```
100 SCREEN 2:OPEN"GRP:" AS #1
110 PSET (100,100)
120 FOR T=0 TO 1000: NEXT T
130 LINE (50,50)-(150,150),,B
140 FOR T=0 TO 1000: NEXT T
145 PI=4*ATN(1)
150 CIRCLE (100,100),50,,-PI/2,-2*PI
160 PAINT(52,52)
170 PAINT(148,148)
180 PAINT(52,148)
190 PAINT(148,52)
900 GOTO 900
```

Você deve obter uma tela como a da figura 5.14, pois o PAINT pinta toda região ao redor da coordenadas nele especificadas, até encontrar a fronteira. Cuidado! Se a fronteira for aberta, ele vai "vazar" e pintar a tela toda!

figura 5.14

Para se certificar disto, comande:

DELETE 110-190 **(e [RETURN])**

e digite as linhas

110 CIRCLE (127,85), 80,, 4.6,4.5 **(não confunda pontos e vírgulas!)**

120 PAINT (127.85)

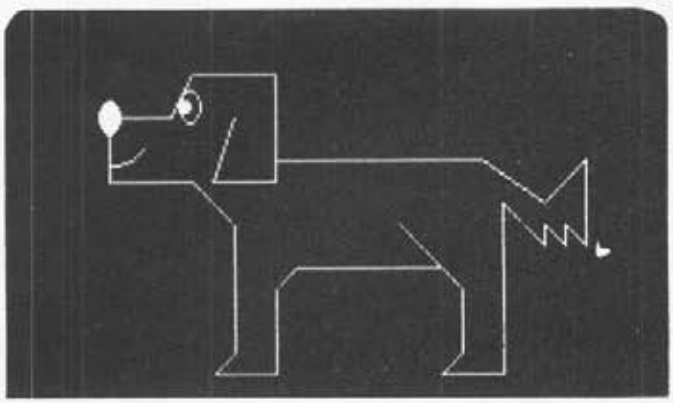
Bonito não? Dá até para perceber como o MSX "raciocina" para pintar uma região!

Vamos descansar um pouco? Não ? Ótimo então passe já para o item seguinte!

DESENHANDO A "MÃO LIVRE"

Vamos supor, por exemplo, que você queira desenhar o cachorro da figura 5.15 na SCREEN 2.

figura 5.15.



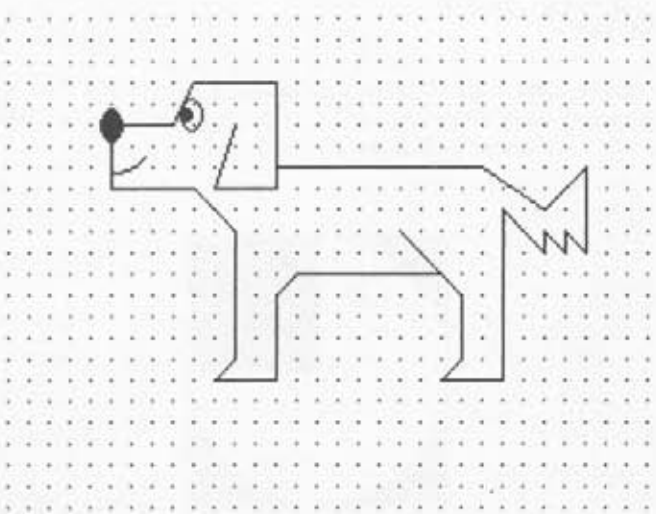
Tentar fazer o nariz, o olho e o sorriso com CIRCLE, funciona. Agora, tentar desenhar o restante do cachorro com LINE, seria uma loucura!

Mas não se preocupe: existe um comando do MSX-BASIC, o DRAW (Desenhe!), que é extremamente indicado para este tipo de tarefa.

Antes de começar, vamos reticular nossa figura, de 8 em 8 pontos, para podermos ter uma referência.

Para nos guiarmos, então, vamos usar o cachorro da figura 5.16.

figura 5.16



Vamos, inicialmente definir as coordenadas do ponto inicial do nosso desenho, bem no meio do nariz, ou seja no ponto (40,40) e definir a escala, no caso 8 (1 passo = 8 pixel). A seguir vamos "reticular" a tela. Para isso basta digitar as linhas da figura 5.17.

figura 5.17 - Reticulando a tela.

```
100 X=40
110 Y=40
120 E=8
130 SCREEN 2
140 FOR L=0 TO 191 STEP 8
150 FOR C=0 TO 255 STEP 8
160 PSET(C,L)
170 NEXT C,L
340 GOTO 340
```

Rode este programa, para ver seu efeito e breque-o com **[CONTROL] + [STOP]**.

Para desenhar e pintar o nariz, basta acrescentar as linhas da figura 5.18.

figura 5.18 - Desenhando o nariz.

```
180 CIRCLE (X,Y),.75*E,,,,1.5
190 PAINT (X,Y)
```

Rode o programa: não se preocupe se o nariz não foi pintado: enquanto houver retícula o PAINT cai em cima de um ponto já marcado. Quando, mais tarde, eliminarmos a retícula, tudo vai funcionar a contento.

Agora vamos à grande novidade; o comando DRAW A$ desenha em função dos comandos contidos em A$, seguindo uma certa escala.

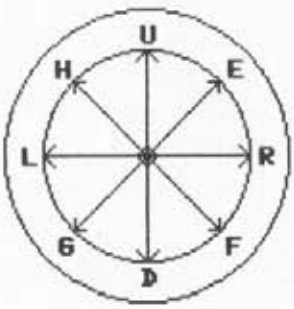
Se, por exemplo, comandarmos:

DRAW "S8R4"

ele desenhará 4 unidades para direita (Right 4) seguindo uma escala de 2 pontos para cada unidade. Isso ocorre porque o parâmetro que define o S(eScala) vale 4 para cada ponto na tela.

Os comandos que desenham nas 8 direções possíveis estão esquematizados na figura 5.19.

figura 5.19 - Desenhando nas 8 direções.



U = Up = para cima

D = Down = para baixo

L = Left = para esquerda

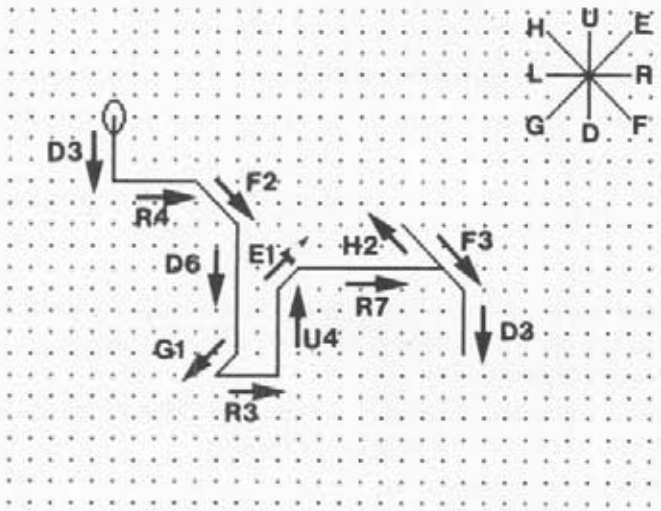
R = Right = para direita

Vamos então fixar o ponto inicial em (x,y), estabelecer o S da escala e já comandar o primeiro DRAW para definir a escala do desenho. Digite as linhas a seguir e rode o programa. Nada de novo vai acontecer pois ainda não começamos a desenhar.

Agora podemos começar o desenho do cão: na figura 5.20 acompanhe a sequência de comandos e veja as linhas correspondentes do programa.

figura 5.20. - Começando o desenho.

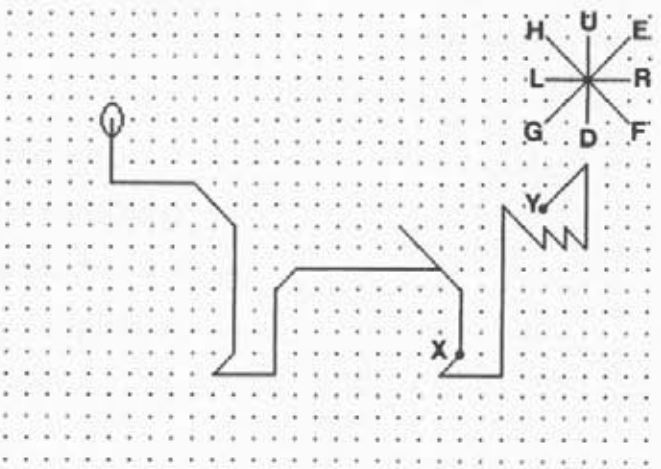
```
220 A$="D3R4F2D6G1R3U4E1R7H2F3D3"
230 DRAW A$
```



Continuando o desenho de X a Y, (consulte sempre a figura 5.16 para lembrar do nosso objetivo), temos a sequência indicada na figura 5.21.

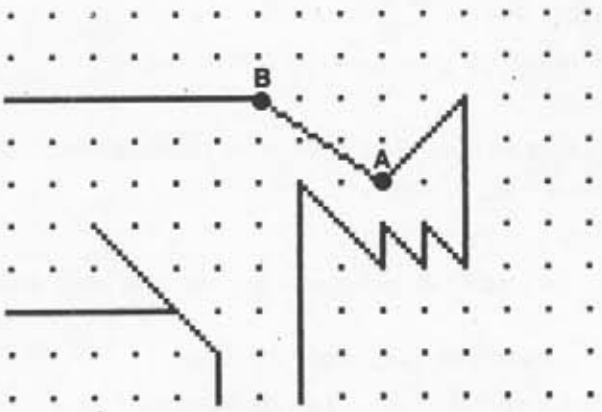
figura 5.21 - Continuando o desenho.

```
240 B$="G1R3U8F2U1F1U1F1U4G2
250 DRAW B$
```



Rode o programa para ver a formação da figura até aqui. Bem, agora temos um problema sério: para ir do ponto A ao ponto B da figura 5.22, nenhum dos comandos que aprendemos até agora serve, pois o movimento do nosso lápis não se dá seguindo a diagonal de um quadrado.

figura 5.22 - Um impasse!



A solução é simples: no DRAW existe o comando

M±x,±y

que movimenta o "lápis" somando ou subtraindo um certo valor das coordenadas de partida x e y.

No nosso caso devemos subtrair 3 do x(para ir para esquerda) e 2 do y(para ir para cima: lembre-se, o y cresce para baixo). Acrescente, então, ao seu programa, as linhas da figura 5.23 e rode-o para ver seu efeito.

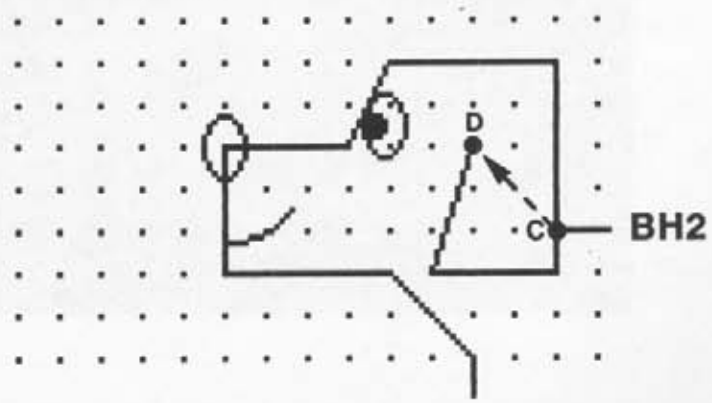
figura 5.23 - Usando o M±x,±y

```
260 C$="M-3,-2L10"
270 DRAW C$
```

Agora, outro problema: para continuar nossa figura, seria conveniente que pudéssemos pular do ponto C ao ponto D da figura 5.24 sem desenhar nada. Fácil, basta colocar à frente do comando um B. O "lápis" se moverá, mas é como se o levantássemos do papel, ou seja, não estamos "riscando".

figura 5.24 - Levantando o "lápis".

```
280 D$="BH2M-1,+3R3U5L4M-1,+2L3
290 DRAW D$
```



Finalmente, acrescente as linhas da figura 5.25 para completar o olho e a boca.Pelo que você já sabendo CIRCLE, eles dispensam comentários.

figura 5.25 - Completando o cachorro.

```
300 CIRCLE (X+E*3.9,Y-E/2),.75*E,,,,1.6
310 CIRCLE (X+E*3.7,Y-E/2),.37*E
320 PAINT  (X+E*3.7,Y-E/2)
330 CIRCLE (X,Y),2.25*E,,4.8,5.5
```

Rodando o programa completo, você deverá obtera tela já mostrada nafigura 5.16.

Se você alterar a linha 130 para:

130 SCREEN 2:GOTO 180

Você pulará toda a parte do programa que reticula atela e terá o cachorro como já mostrado'aa figura 5.15.

Para encerrar o uso deste programa, comande:

DELETE 100–170 (e [RETURN])

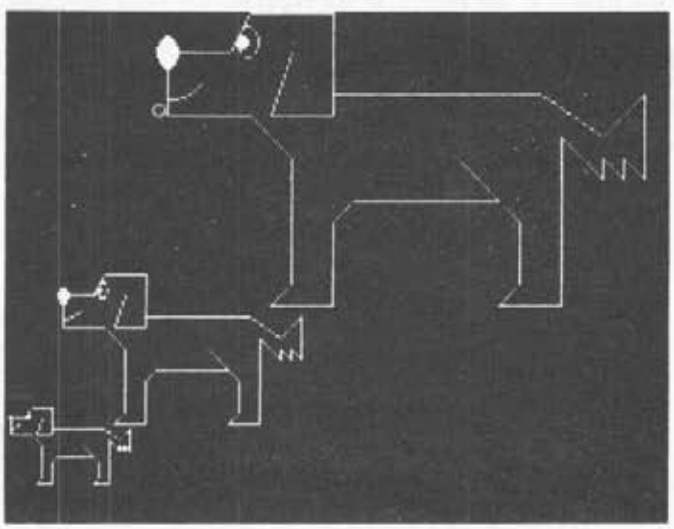
e digite as linhas adicionais da figura 5.26.

figura 5.26 -Alterando o programa

```
100 SCREEN 2
110 FOR K=1 TO 3
120 E=2^K
130 X=10*E-18
140 Y=199-23*E
150 GOSUB 180
160 NEXT K
170 GOTO 170
340 RETURN
```

Você deverá obter a tela da figura 5.27, o que dá uma boa demostração do efeito de escala sobre o DRAW.

figura5.27 -Gerações de cachorros.



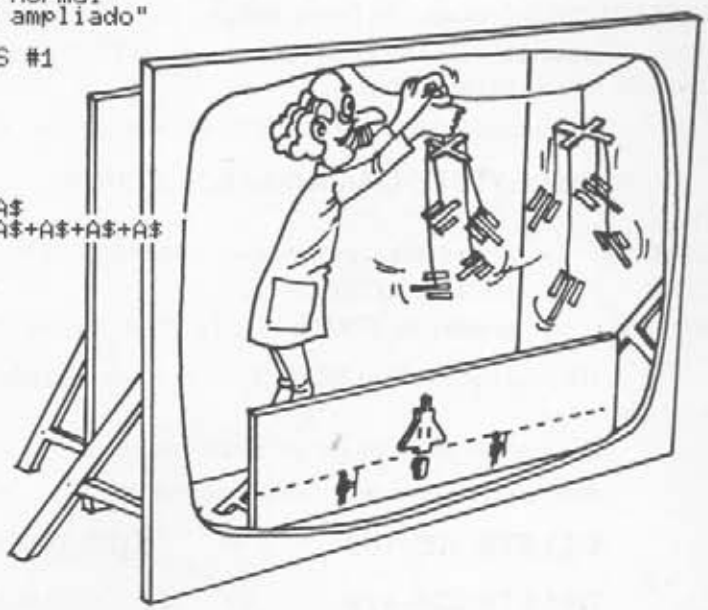
DESENHANDO "DUENDES"

Assim como podemos, com o DRAW, definir um desenho e localizá-lo em pontos diferentes da tela e em tamanhos diferentes, existe uma "entidade" no BASIC-MSX denominada SPRITE ("duende" ou "fantasminha"). Eles podem ser definidos como figuras de 8x8 ou 16x16 pontos e podem ser colocados na tela em ate 32"camadas" diferentes .

Para perceber melhor seu funcionamento, digite o programa da figura 5.28.

figura 5.28 -Definindo um SPRITE.

```
100 M$(0)="SCREEN 2,0 08x08 normal"
110 M$(1)="SCREEN 2,1 08x08 ampliado"
120 M$(2)="SCREEN 2,2 16x16 normal"
130 M$(3)="SCREEN 2,3 16x16 ampliado"
140 FOR S=0 TO 3
150 SCREEN 2,S:OPEN"GRP:" AS #1
160 LINE(0,0)-(255,191),,B
170 A$=""
180 FOR I=1 TO 8
190 READ B$
200 A$=A$+CHR$(VAL(B$))
210 NEXT I
220 IF S<2 THEN SPRITE$(1)=A$
230 IF S>1 THEN SPRITE$(1)=A$+A$+A$+A$
240 DATA &B11111111
250 DATA &B10000001
260 DATA &B10100101
270 DATA &B10000001
280 DATA &B10100101
290 DATA &B10011001
300 DATA &B10000001
310 DATA &B11111111
320 PRESET(20,20)
330 PRINT #1,M$(S)
340 FOR N=1 TO 4
350 FOR X=0 TO 191
360 PUT SPRITE 1,(X,X),15,1
370 NEXT X
380 BEEP:NEXT N
390 CLOSE:RESTORE 240
400 BEEP:BEEP:BEEP:NEXT S
410 RUN 100
```



O formato do SPRITE é definido nas linhas DATA(240-310) na forma de 8 números em notação binária, sendo que o "0" corresponde a pontos apagados e o "1" a pontos acesos. Esta formação de 8x8 pontos é lida nas linhas de 170 a 210, formando uma string(A$) de 8 caracteres.

```
DATA &B11111111
DATA &B10000001
DATA &B10100101
DATA &B10000001 →
DATA &B10100101
DATA &B10011001
DATA &B10000001
DATA &B11111111
```

Na linha 150 é dado o comando SCREEN com uma novidade: após a vírgula, colocamos um perâmetro (S), que define o tipo de SPRITE que queremos usar. A explicação deste S está nas linhas 100 a 130.

Quando o SPRITE é 8x8 (S=0 ou S=1) ele é definido por uma string de 8 caracteres: isso explica a linha 220.

Quando é 16x16, precisamos de 4 strings para definí-lo. Na linha 230 usamos 4 vezes a mesma string A$ por uma questão de brevidade.

Na realidade podemos definir 4 strings diferentes, formando figuras mais complexas, seguindo o esquema da figura 5.29.

figura 5.29 - O esquema de formação do SPRITE 16x16.

SPRITE(X)=A$+B$+C$+D$

Aqui definimos apenas um SPRITE, o SPRITE$(1). Na realidade podemos definir 256 SPRITE$ diferentes, se forem 8x8, ou até 64 se forem 16x16.

Nas linhas de 340 a 380 fazemos o SPRITE "passear" em diagonal pela tela, 4 vezes, antes de mudar parâmetro S.

O comando que coloca o SPRITE num ponto da tela é

PUT SPRITE CAMADA,(X,Y),COR,Nº

Onde:

CAMADA – nº da camada em que queremos colocá-lo. Veremos mais detalhes adiante.

COR – cor do SPRITE(de 0 a 15)

Nº – número do SPRITE – de 0 a 255 (se 8x8) ou de 0 a 63 (se 16x16)

Há uma novidade: o BEEP. Ele simplesmente faz o alto-falante do seu Expert emitir um som.

Do resto o programa é praticamente auto-explicativo: é só rodar e aprender.

Agora grave este programa e comande:

DELETE 100-160 **(e [RETURN])**

DELETE 320-410 **(e [RETURN])**

DELETE 220-230 **(e [RETURN])**

Complete agora a listagem conforme a figura 5.30 e rode o programa.

figura 5.30 - Sprite na SCREEN 1.

```
100 SCREEN 1,1
170 A$=""
180 FOR I=1 TO 8
190 READ B$
200 A$=A$+CHR$(VAL("&B"+B$))
210 NEXT I
220 SPRITE$(1)=A$
225 '--------------
230 'ESTE é O SPRITE--->
235 '--------------
240 DATA 11111111
250 DATA 10000001
260 DATA 10100101
270 DATA 10000001
280 DATA 10100101
290 DATA 10011001
300 DATA 10000001
310 DATA 11111111
400 X=210:Y=60
410 PUTSPRITE 1,(X,Y),15,1
420 LIST
```

Obs:
Assim como o "?" pode substituir o "PRINT", o "'" pode substituir o "REM".

A surpresa neste programa é que os SPRITES podem, também, ser colocados na SCREEN 1, coexistindo com textos e até, como neste caso, com uma listagem.

Rode o programa e você deve obter a tela da figura 5.31.

figura 5.31 - Visualizando pontos acesos e apagados.

```
100 SCREEN 1,1
170 A$=""
180 FOR I=1 TO 8
190 READ B$
200 A$=A$+CHR$(VAL("&B"+B$))
210 NEXT I
220 SPRITE$(1)=A$
225 '---------------
230 'ESTE É O SPRITE--->
235 '---------------
240 DATA 11111111
250 DATA 10000001
260 DATA 10100101
270 DATA 10000001
280 DATA 10100101
290 DATA 10011001
300 DATA 10000001
310 DATA 11111111
400 X=210:Y=60
410 PUTSPRITE 1,(X,Y),15,1
420 LIST
```

Leve o cursor até as linhas DATA e altere 0s e 1's até montar seu próprio desenho de SPRITE. Não esqueça do **[RETURN]** após a alteração de uma linha. Cada vez que você fizer uma alteração, rode o programa e você terá o padrão numérico nas linhas DATA e o efeito no SPRITE ao lado!

Na realidade podemos colocar até 32 SPRITES simultaneamente na tela, em 32 camadas diferentes(de 0 a 31).

A camada 0 é a que está mais próxima do observador e a 31 está mais "no fundo".

Isto significa que, se um SPRITE da camada 7, por exemplo, coincidir com um de camada 11, ele passará "por cima" dele pois, por estar uma camada de número menor, tem prioridade.

Rode o programa da figura 5.32 para perceber este efeito.

figura 5.32 - Sobreposição de SPRITES.

```
100 F=9:DIM X(F),Y(F),DX(F),DY(F)
110 SCREEN 2,1:BASE(2)=BASE(14)
120 SCREEN 0:SCREEN2:SPRITE ON
130 LINE(0,0)-(255,191),,B
140 ON SPRITE GOSUB 290
150 FOR M=0 TO F
160 X(M)=2*M^2:Y(M)=17*M+1
170 DX(M)=M/2+1:DY(M)=11-M
180 NEXT M
190 FOR M=0 TO F
200 IF X(M)<0   THEN DX(M)=-DX(M)
210 IF X(M)>244 THEN DX(M)=-DX(M)
220 IF Y(M)>175 THEN DY(M)=-DY(M)
230 IF Y(M)<1   THEN DY(M)=-DY(M)
240 X(M)=X(M)+DX(M):Y(M)=Y(M)+DY(M)
250 PUT SPRITE M,(X(M),Y(M)),M+2,M+48
260 SPRITE ON
270 NEXT M
280 GOTO 190
290 BEEP:COLOR ,,15:COLOR ,,1
300 SPRITE OFF
310 RETURN
```

Você percebe que, toda vez que dois SPRITES se encontram, toca um BEEP e aparecem uns "relâmpagos" na tela.

Isso ocorre devido à linha 140:

ON SPRITE GOSUB 290

que significa: "se dois SPRITES coincidirem, vá para a subrotina 290". Nesta, é dado um BEEP, é alterada a cor da borda e é desativado o "alarme"de coincidencia de SPRITE (SPRITE OFF).

Se não fizéssemos isso a execução ficaria "entalada" na sub-rotina. O alarme é reativado na linha 260.

Como você deve ter notado, 10 SPRITES (de 0 a 9) são colocados em 10 camadas (de 0 a 9) e começam a se mover (X + DX, Y + DY). Quando batem na borda, sua velocidade é invertida (DX = -DX; DY = -DY) para que permaneçam dentro dos limites da tela(linhas de 20 a 270).

Você notará, no começo de execução, que alguns sprites são apagados na hora em que se alinham horizontalmente. Isso ocorre porque o VDP (Vídeo Display Processor) do MSX só admite 4 sprites na mesma horizontal. Leve isso em conta nos seus futuros programas!

Nesta altura você já se tornou um programador com uma certa experiência e começou a procurar no programa onde foram definidos os 10 SPRITES com a forma dos algarismos de 0 a 9. Como não acha nenhum comando SPRITE$(X) = ..., deve ter ficado intrigado!

Vamos desfazer o mistério: o seu Expert, ao chamar a SCREEN 0, carrega na VRAM (RAM de vídeo) a tabela que dá a forma dos caracteres (pontos acesos e apagados) no endereço definido por BASE(?). Da mesma forma, ao definirmos SPRITE's, sua forma é colocada, na SCREEN 2, na região da VRAM definida por BASE(14).

Nas linhas 110 e 120, então, usamos o seguinte truque: chamamos a SCREEN 2, fazemos o BASE(2) igual ao BASE(14) e chamamos a SCREEN 0.

Neste momento o Expert carrega a tabela de formação dos caracteres na região que seria usada, pela SCREEN 2, para saber a forma dos SPRITEs.

Ao chamarmos novamente a SCREEN 2 ", temos 255 SPRITE's já definidos, cada um com a forma de um caracter do MSX! Veja também o Apêndice A.

Para perceber isso, acrescente ao seu programa a linha:

255 PUT SPRITE 11,(128,96),15,3

Na camada 11 e em cor branca(15), deverá aparecer o "coração" pois ele é o CHR$(3)!

Liste o programa e altere-o 3 para 224: agoradeverá aparecer o "Alfa" que é o CHR$(224), agora SPRITE$(224).

USANDO A SCREEN 3

As diferenças entre SCREEN 3 e SCREEN 2

Na SCREEN 3 temos também uma tela gráfica como na SCREEN 2, com uma desvantagem e uma vantagem.

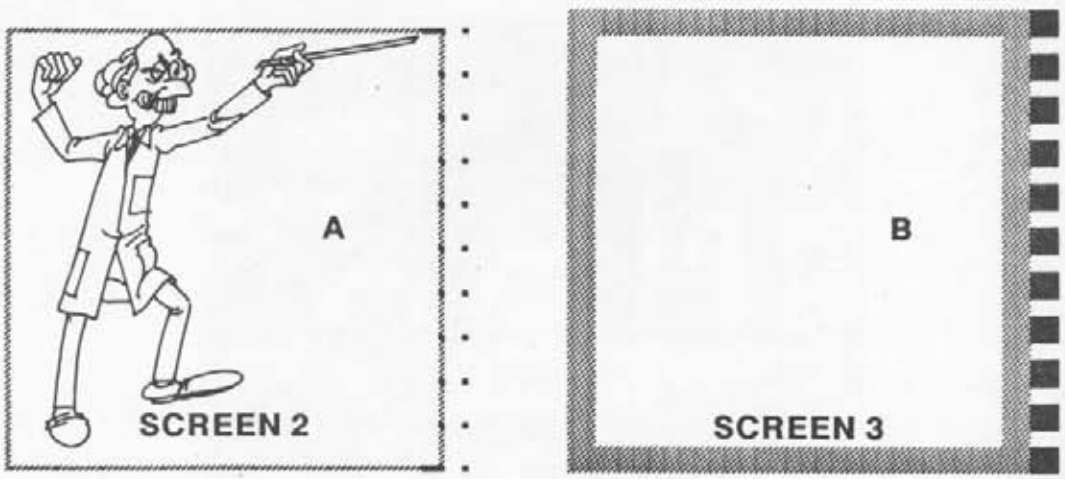
Quando usamos a SCREEN 2, podemos "plotar" na tela 256x192 pontos diferentes. Sofremos, porém, limitação nas cores pois podemos definir uma única conbinação de cor de frente e cor de fundo a cada 8 pontos horizontais, ou seja uma conbinação com x de 0 a 7, outra de 8 a 15, e assim por diante.Digite o programa de figura 5.33 para perceber isso.

figura 5.33 -Testando a SCREEN 2.

```
100 SCREEN 2
110 LINE(7,79)-(83,159),4,B
120 FOR Y=79 TO 159 STEP 8
130 PSET (87,Y),11
140 NEXT Y
150 GOTO 150
```

Com este programa, estamos desenhando um retângulo em AZUL ESCURO (cor 4) e uma fileira de pontos do seu lado direito em AMARELO (cor 11). Como você pode notar na tela (fig.5.34-A) a cor amarela "invade" alguns trechos do retângulo pois eles estão nos mesmos "grupos de 8" dos pontos que foram definidos como amarelos.

figura 5.34 -Desenhando na SCREEN 2.e 3.



Experimente mudar a linha 100 para: **100 SCREEN 3** e rode o programa. Você deverá obter uma tela como a da figura 5.34-B.

Como você pode notar, o desenho na SCREEN 2 é mais "grosseiro" pois ela usa um "pixelzão" 4 vezes mais largo e alto que o pixel da SCREEN 2. Em compensação cada pixelzão tem sua própia cor, não invadindo regiões vízinhas.

ESCREVENDO NA SCREEN 3

Vamos agora escrever alguma coisa na SCREEN 3 para perceber a outra grande diferença que existe com a SCREEN 2.

Digite o programa da figura 5.35 e rode-o.

figura 5.35 -Pintando e escrevendo na SCREEN 3.

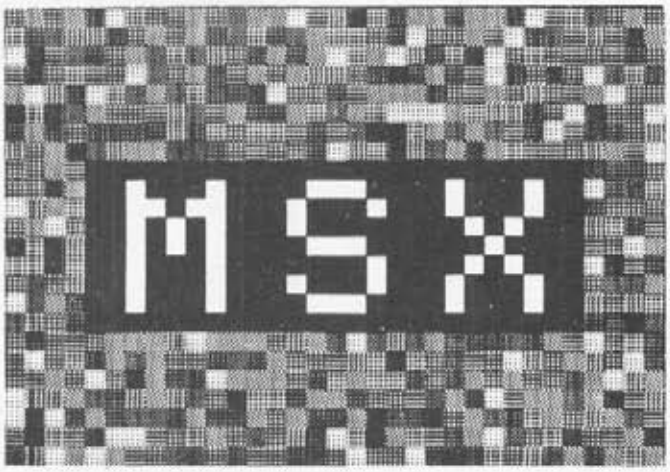
```
100 SCREEN 3:OPEN"GRP:" AS #1
110 U=RND(-TIME)
120 FOR X=16 TO 147 STEP 4
130 FOR Y=96 TO 191 STEP 4
140 C=INT(RND(4)*12)+2
150 PSET(X,Y),C
160 NEXT Y,X
170 LINE(32,128)-(32*4+3,32*5+3),1,BF
180 A$="MSX"
190 PRESET (40,132):PRINT#1,A$
200 GOTO 200
```

Nos laços das linhas 120 e 130, plotamos "pixelzões" em cores diferentes (linha 150), escolhidas na linha 140, que comentaremos depois. Note que usamos um STEP 4 pois na SCREEN 3 não adianta ir de 1 em 1.

Na linha 170 abrimos uma janela no "tapete" formado pelo PSET e nela escrevemos (190) a mensagem definida em 180.

Você deve obter uma tela parecida com a da figura 5.36.

figura 5.36 -"Tapete" com mensagem.



OS NÚMEROS ALEATÓRIOS

Vejamos, agora, como foi feita a escolha de cor de cada "pixelzão". Grave o programa que está na memória e comande um NEW.

Digite agora o curto programa da figura 5.37 e rode-o.

figura 5.37 -RND (8).

```
110 FOR I=1 TO 5
120 PRINT RND(8)
130 NEXT I
Ok
run
 .5952194399462
 .1065862805015
 .7659765177282
 .5775639293558
 .7347475950302
Ok
```

Como você pode notar, a função RND (X) gera números aleatórios (RaNDomics) entre 0 a 1. Se mandarmos o argumento da função para outro número positivo, teremos uma surpresa (fig.5.38)!

figura 5.38 -RND (678).

```
110 FOR I=1 TO 5
120 PRINT RND(678)
130 NEXT I
Ok
run
 .59521943994623
 .10658628050158
 .76527651772623
 .57756392935858
 .73474759503023
Ok
```

A função RND gera exatamente a mesma sequência!

Vamos experimentar um argumento negativo (fig.5.39).

figura 5.39 -RND (-8) e RND (-678).

```
110 FOR I=1 TO 5
120 PRINT RND(-8)
130 NEXT I
Ok
run
 .34389820420821
 .34389820420821
 .34389820420821
 .34389820420821
 .34389820420821
Ok
```

```
110 FOR I=1 TO 5
120 PRINT RND(-678)
130 NEXT I
Ok
run
 .02589820420821
 .02589820420821
 .02589820420821
 .02589820420821
 .02589820420821
Ok
```

Agora temos números diferentes para cada argumento, mas, na sequência é gerado sempre o mesmo!

O que podemos concluir?

Quando o argumento é negativo, o número gerado pelo RND é função do argumento. Portanto, fixando um argumento negativos geraremos sequencias sempre com o mesmo número.

Usando um argumento positivo, obtemos uma sequência de números diferentes, mas sempre a mesma sequência.

Para obter uma sequência diferente para argumentos positivos, o interessante é gerar antes um RND com argumentos negativo. Digite os dois programas da figura 5.40 para perceber isso.

figura 5.40.

```
100 U=RND(-123)
110 FOR I=1 TO 5
120 PRINT RND(8)
130 NEXT I
Ok
run
 .3992486816692
 .46769655852301
 .7095173630504
 .25475124336581
 .6039147084636
Ok
```

```
100 U=RND(-345)
110 FOR I=1 TO 5
120 PRINT RND(8)
130 NEXT I
Ok
run
 .5412486816692
 .16569655852301
 .9715173630504
 .23275124336581
 .1859147084636
Ok
```

O ideal seria se pudéssemos obter antes o RND de um número negativo ao acaso, e, depois gerar uma sequência com argumento positivo.

Isso é facil de se conseguir, pois existe no MSX uma variável reservada (o TIME) que é incrementada de uma unidade a cada 1/60 de segundo. Se antes de gerarmos a sequência de RND com argumento positivo, calcularmos o RND (-TIME), estaremos inicializando a sequência com um número que depende do tempo decorrido desde que o micro foi ligado. Isso faz com que a coisa ocorra bem ao acaso!

Rode várias vezes o programa da figura 5.41: a cada vez você obterá uma sequencia diferente.

figura 5.41.

```
100 U=RND(-TIME):PRINT TIME
110 FOR I=1 TO 5
120 PRINT RND(8)
130 NEXT I
Ok
```

É por esse motivo que foi colocado um RND(-TIME) na linha 110 do programa da figura 5.35.

Para encerrar a explicação sobre este programa e entender como gerar números inteiros entre I (de Inferior) e S (de Superior) rode o programa da figura 5.42.

figura 5.42 -Gerando números randômicos entre I E S.

```
100 A$="*"+STRING$(7,7)
110 INPUT"INFERIOR";I
120 INPUT"SUPERIOR";S
130 X=INT(RND(2)*(S-I+1))+I
140 PRINT USING "####";X;
150 IF X=I OR X=S THEN PRINT A$:PRINT
160 IF X<I OR X>S THEN BEEP:STOP
170 GOTO 130
Ok
```

Analise agora a linha 140 da figura 5.36, com base no que você aprendeu até aqui.

Com o que você aprendeu neste capítulo, já imaginou que telas de abertura você pode gerar para seus vídeos? Para gravá-las basta conectar a saída de vídeo do MSX com o "VIDEO-IN" do seu vídeo cassete. Se quiser gravar com som, basta fazer seu MSX tocar uma música (ou gerar um ruído) e conectar a saída de audio com o "AUDIO-IN" do seu vídeo-cassete ou camcorder.

Agora é só gravar!

Você dirá: "Calma! e a música ? Como é que eu faço?"

Fácil, é só ler o capítulo seguinte!

CONSTRUINDO SONS

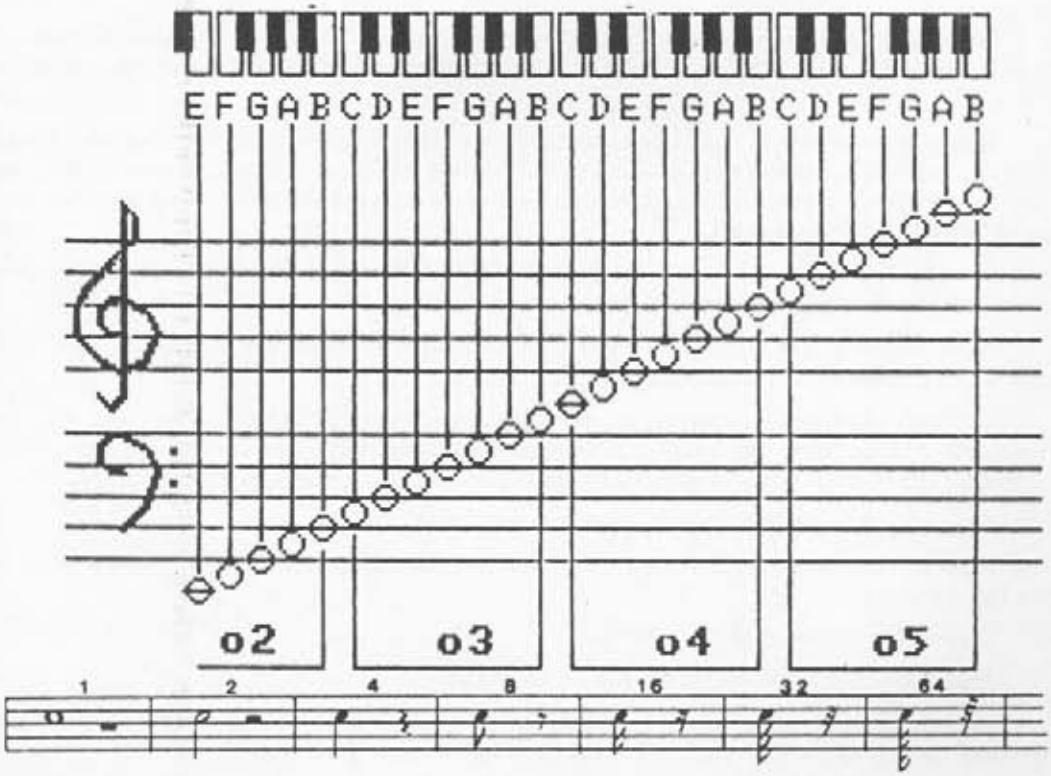
6

O COMANDO PLAY

Fazer música no MSX é bem fácil, mesmo que você não tenha noções muito avançadas de teoria musical. No BASIC MSX existe o comando PLAY que toca a que você quiser e como você quiser, em função dos códigos que ele usa como argumento.

As notas musicais são dadas pela notação Anglo-Germânica: o Do, Re, Mi, Fa, Sol, La, Si são representados por C, D, E, F, G, A, B respectivamente, as pausas por R, a duração por um número após a nota (se seguido por ponto " .", a duração é acrescida da metade). Veja na figura 6.1 a correspondência entre as notas na partitura, o teclados de um piano, as oitavas a que pertecem é os códigos das durações.

figura 6.1

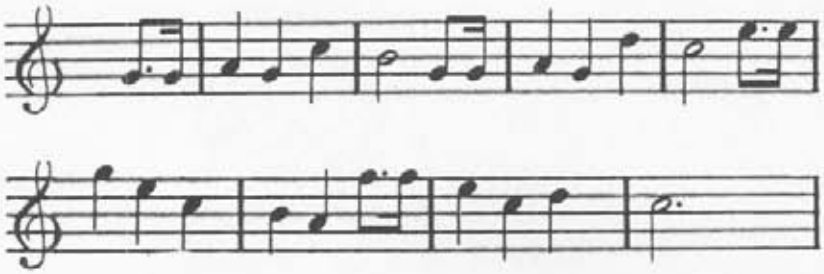


Digamos, por exemplo, que você queira gravar uma abertura para um vídeo de aniversário. Você já desenhou as velinhas na SCREEN 2, o nome dele, a data do aniver-

sariante, mas só falta a música de fundo.

Você encontrou a partitura do "parabéns a você" (figura 6.2).

figura 6.2



Olhando para as figuras 6.1 e 6.2, você pode digitar o programa listado na figura 6.3.

figura 6.3

```
5   PLAY"S0 M6000 O4 T120"
10  PLAY"G8.G16"
20  PLAY"A4G4O5C4"
30  PLAY"O4B2G8.G16"
40  PLAY"A4G4O5D4"
50  PLAY"C2E8.E16"
60  PLAY"G4E4C4"
70  PLAY"O4B4A4O5F8.F16"
80  PLAY"E4C4D4"
90  PLAY"C2."
```

Nesta digitação tome muito cuidado para não confundir o "O" (de oitava) com o "0" (zero) e B (nota SI) com o 8 (de duração). Por enquanto, não se preocupe em entender a linha 5.

Rode o programa e ouça (se necessário regule o volume do alto-falante na traseira da CPU). Se seu MSX estiver conectado a uma TV você pode ouvir também pelo alto-falante dela. Se quiser, pode ligar a saída de audio à entrada de uma amplificador: o efeito numa boa caixa acústica é fenomenal!

Vamos decifar agora a linha 5 : aquele T120 indica o "andamento" da música (tipo "*adagio*", "*allegro vivace*", "*maestoso*", "*prestissimo*", etc). É o " metrônomo" do MSX!

Leve o cursor até lá e altere para T 240 e toque de novo. Percebeu a efeito? Volte ao valor T120 e vamos mudar outra coisa.

O S define a forma do envelope, ou seja, como o volume emitido varia com o tempo, enquanto que o M define o quão rapidamente esas mudanças devem ocorrer. Altere o M6000 para M12000. Percebeu como o som ficou mais prolongado? Mude agora para M300.

O som fica tão curto que você simula um "*pizzicato*" !

Agora, vamos mudar a forma do envelope. Altere para S8 M500. Agora você tem um bandolim!

Deixe S8 e mude o M para M100:

Não parece o aniversário de um neandertal?

Agora elimine S8 e M100 e substitua por V6. Você tem um orgão tocando um "pianissimo". Mude para V15 e ouvirá o mesmo orgão num "*fortissimo*".

Se você quiser uma repetição continua da música, basta acrescentar a linha

100 GOTO 5

Agora ele só pára se você digitar **[CONTROL] + [STOP]**.

Para você perceber o alcance musical do seu MSX, apague esse programa (NEW) e digite o programinha listado na figura 6.4.

figura 6.4

```
100   PLAY"V15 T120"
110   A$="CDEFGAB"
120   FOR I=1 TO 8
130   PLAY"O"+STR$(I)
140   PLAY A$
150   NEXT I
```

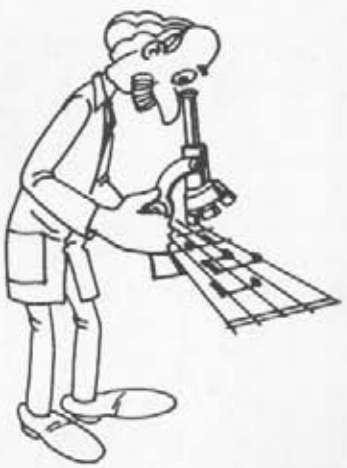
Mesmo com o som do MSX saindo por uma caixa acústica com um bom "Tweeter", existem agudos que quase fogem do alcance do ouvido humano!

A FUNÇÃO PLAY

Vamos, agora, digitar um programa um pouco mais ambicioso para mostrar outras particularidades do seu Expert. Tomando bastante cuidado na digitação, obedecendo rigorosamente à numeração das linhas indicada na figura 6.5.

figura 6.5 -Tocando em 3 vozes.

```
100 SCREEN 2:OPEN"GRP:" AS #1
110 PRESET(0,0)
120 PRINT #1,"MINUETO EM LA"
130 PRINT #1,"L.BOCCHERINI"
140 PRINT #1,"(1743-1805)"
150 FOR I=1 TO 10
160 READ A$,B$,C$
170 PLAY A$,B$,C$:GOSUB 210
180 NEXT I
190 CIRCLE(80+I,60+I),I*2 :I=I+1
200 IF PLAY(0) THEN 190 ELSE END
210 CIRCLE(80+I,60+I),I:RETURN
501 DATA "s0m6000T72","s0m6000T72","s0m6
000T72"
502 DATA "o5a16g#16a16b16","r4","r4"
503 DATA "a8o4a4o5c#4e8","o4c#8e8r8e8c#8
r8","o3a8r8e8r8r8a8"
504 DATA "e8d8d4d16c#16d16e16","o4d8e8r8
e8d8e8","o3g#8r8e8r8g#8r8"
505 DATA "d8o4e4b4o5d8","o4d8r8r8e8o3b8o
4e8","o3g#8r8e8r8g#8r8"
506 DATA "d8c#8c#4a8.f#16","o3a8o4e8c#8e
8c#8o3a8","o4r4r4a4"
507 DATA "e8d#8d#8d#8a8.f#16","o3f#8b8a8
b8o4d#8o3b8","o4a2a4"
508 DATA "e8d#8d#8d#8a8.f#16","o3f#8b8a8
b8o4d#8o3b8","o4a2b4"
509 DATA "g#8e8c#8a8g#32f#32g#32f#32g#32
f#32e32f#32","o3b8o4e8c#8e8d#4","o4b4o3a
8r8o4a4"
510 DATA "e4.r8","o4g#8o3b8e8r8","e8r8r8
r8"
```



O programa lê as variáveis A$, B$, C$ nas linhas DATA e as toca na linha 170.

Cuidado: se você cometer algum erro de digitação nas linhas DATA, ele será detectado na hora de tocar as strings, ou seja em 170. Nestas condições o seu Expert emitirá a mensagem:

Syntax error in 170

e você ficará procurando um erro em 170 quando ele foi cometido em outro lugar!

A novidade está na linha 200: experimente substituí-la por um simples END. Ao terminar a execução, o PSG ainda ficará tocando enquanto o programa já encerrou!

Recoloque a linha 200 como na listagem original (fig. 6.5): ela usa a função PLAY(0). Enquanto algum dos canais ainda estiver ativo, PLAY(0) será verdadeira e o BASIC continuará desenhando círculos. Quando nenhum dos 3 canais estiver ativo, PLAY(0) se torna falsa e passa a valer o ELSE.

Se quisermos testar apenas UM dos canais (1, 2 ou 3), deveremos usar o PLAY(1), PLAY(2) ou PLAY(3) respectivamente.

O COMANDO SOUND

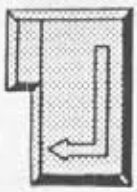
Imagine, agora, que no seu vídeo de viagem exista uma cena de despedida no aeroporto.

Você já desenhou um jato partindo na SCREEN 2 e falta a trilha sonora.

Experimente o programa da figura 6.6 (apagando, obviamente, o anterior).

Antes, porém um truque! Digite:

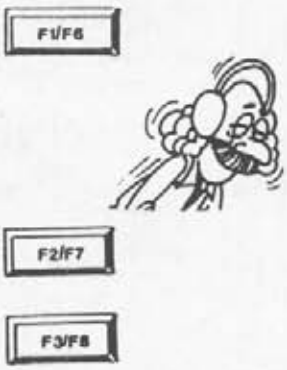
KEY1,"SOUND" **(e [RETURN])**
KEY2,"FOR L=" **(e [RETURN])**
KEY3,"NEXT L" **(e [RETURN])**
AUTO 100 **(e [RETURN])**



Agora basta apertar **[F1]**. completar com 7,28 (e **[RETURN]**). A linha 110 já aparece esperando um novo **[F1]** (SOUND), etc. nas linhas 190, 200, 220, 240 e 260 você vai usar **[F2]** (FOR L=) e **[F3]** NEXT L- e, ao chegar na linha 290 basta digitar **[CONTROL]** + **[STOP]** para interromper o processo de numeração automática. Limpe a tela, liste o programa, confira tudo e rode-o.

figura 6.6

```
100 SOUND    7,28
110 SOUND    1,0
120 SOUND    3,0
130 SOUND    6,0
140 SOUND    0,100
150 SOUND    8,13
160 SOUND    9,0
170 SOUND   10,15
180 SOUND   12,255
190 FOR L=1 TO 2000:NEXT L
200 FOR L=100 TO 30 STEP -.04
210 SOUND    0,L
220 NEXT L
230 SOUND    8,0
240 FOR L=2 TO 31 STEP .01
250 SOUND    6,L
260 NEXT L
270 SOUND   10,16
280 SOUND   13,0
```



Seu jato vai aquecer as turbinas, correr pela pista, decolar e sumir entre as nuvens!

E, por aqui, a gente também vai se despedindo deste capítulo. Como já foi dito, não é nossa intenção , neste livro, transformar o leitor num programador, mas, após ter trabalhado em conjunto com seu EXPERT PLUS ao longo desses páginas, você deve ter tido uma idéia do imenso potencial da Linguagem BASIC residente nesta máquina.

Se você quiser se aprofundar no MSX, leia a bibliografia recomendada no apêndice E, onde estão relacionados outros livros, com Cursos, Coleções de Programas, Dicas que podem levá-lo a se tornar um apaixonado por microcomputadores.

LENDO E GRAVANDO

7

AS DUAS MEMÓRIAS DO EXPERT PLUS

Nas memórias ROM foram gravados 48 kbytes de informações pela própria Gradiente, de forma permanente. Se você desligar o micro estas informações não são perdidas.

Nas memórias RAM você tem 80 Kbytes de posições livres que podem ser preenchidas pelo usuário enquanto o micro está ligado. Uma parte da RAM é ocupada pelo próprio Expert para poder gerenciar seus sistemas internos e o resto está disponível.

Quando você, por exemplo, digite um programa em BASIC, seu texto se localiza na RAM. Se você desligar o micro (intencionalmente ou não), as informações são perdidas.

Para evitar esta perda, você pode gravar trechos da memória RAM em cassete para posterior recuperação.

A LIGAÇÃO DO GRAVADOR

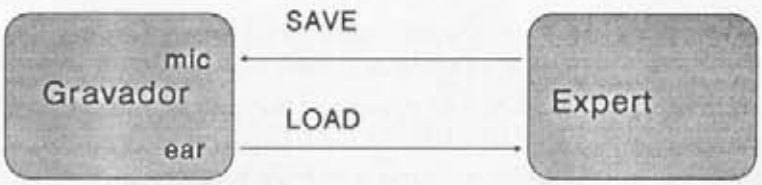
Ao comandar uma gravação (SAVE), seu Expert transforma os códigos contidos nas posições da RAM em um ruído especial.

Este ruído é gravado na fita cassete pela entrada do microfone do gravador.

Ao carregarmos informações (LOAD), os ruídos gravados na fita saem pelo conector do fone de ouvido ("ear phone") e entram no computador onde são traduzidos e transformados em códigos que se posicionam na memória.

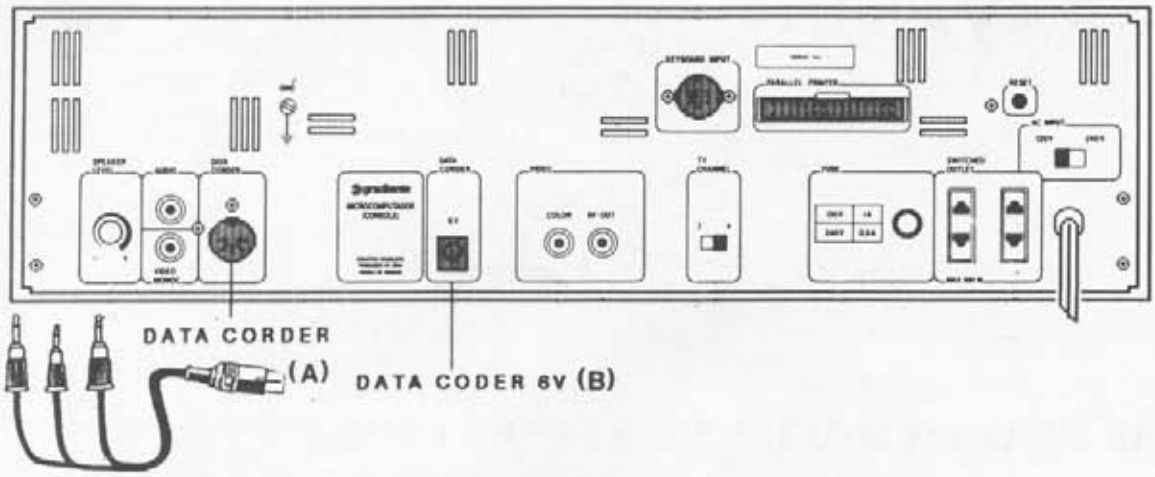
O fluxo de informações entre o Expert e o gravador cassete está esquematizado na figura 7.1.

figura 7.1 - Fluxo de informações Expert -Gravador.



Junto com seu Expert você recebeu um cabo que tem 3 conectores numa das extremidades. É ele que vai transportar este fluxo de informações entre o micro e o gravador. Sua outra extremidade deve ser encaixada na tomada DATA CORDER, no painel traseiro do Expert (figura 7.2-A).

figura 7.2

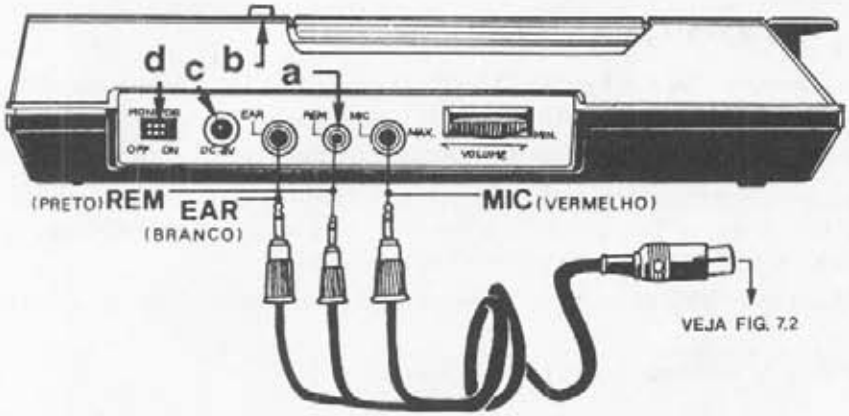


Agora vamos à ligação com o gravador, na outra extremidade do cabo.

O terminal vermelho deve ser ligado no MIC(levando informações do Expert para o gravador). O terninal branco é conectado no EAR (trazendo informações do gravador para o micro).

Existe um terceiro conector preto, de pino mais fino, que deve ser ligado no REM (controle REMoto). Se você tem um DACORDER da Gradiente ou um gravador que tenha este recurso, conecte-o também. Ele serve para que o próprio Expert se encarregue de ligar e desligar o motor do gravador (fig.7.3-a).

figura 7.3



É conveniente que seu gravador tenha um contador de voltas (fig.7.3b) para poder localizar mais repidamente a posição de uma certa gravação na fita.

Se você tem uma DATA CORDER, não precisa gastar pilhas: na traseira do Expert existe um outro conector: o 6V DATA CORDER (fig7.2-B).

Basta ligar uma das extremidades no cabo que você recebeu junto com o DATA CORDER neste conector e a outra extremidade no DC-6V do gravador (fig.7.3-c).

AS FITAS

A fita que você vai usar deve ser boa qualidade e, se possivel, curta.

Isso por dois motivos: inicialmente você deve lembrar que fitas curtas têm maior espessura, portanto maior resistência mecânica. Em segundo lugar devemos ter sempre em mente que a busca de um certo programa ou bloco de dados, na fita, é sequêncial.

Como a fita cassete grava por processo magnético, ela deve estar protegida de fortes campos que possam danificar as imformações nela contida. Além dos imãs [lembre-se, até tesouras podem ser imãs disfarçados] todos os enrolamentos e bobinas percorridos por corrente elétrica geram fortes campos magnéticos.

É o caso dos reatores de lâmpada fluorescentes, televisores, transformadores e motores e motores elétricos. Às vezes um passeio de ônibus elétrico ou metrô, com uma fita no bolso, pode ter tristes consequências!

GRAVANDO E LENDO PROGRAMAS EM BASIC

CSAVE, CLOAD? E CLOAD.

Vamos supor, agora, que na memória RAM do seu Expert esteja o programa de cálculo de áreas da figura 4.36.

Coloque uma fita virgem em seu gravador e avance-a um pouco para passar a parte inicial que não contém emulsão magnética.

Se você está com o REM conectado, mesmo apertando o PLAY do gravador (ou LOAD do DATA CORDER) a fita ficará parada. Basta então comandar:

MOTOR ON (e[RETURN])

Você ouvirá um "click" na traseira do seu Expert e a fita começará a rodar. Após passar o trecho inicial, pare o gravador comandando:

MOTOR OFF (e[RETURN])

Aperte as teclas de gravação do gravador (normalmente PLAY + REC) ou SAVE do DATA CORDER e digite:

CSAVE "ÁREA " (e[RETURN])

A fita começará a girar e a gravação se inicia. Se seu gravador tiver um monitor que permita ouvir a gravação, como no DATACORDER, deixe-o ligado(figura 7.3-d) pois é muito instrutivo "ouvir" o programa. Você notará que ele é gravado em 2 blocos: o primeiro contém o cabeçalho e o segundo o programa propriamente dito.

Ao terminar a gravação você terá na tela o "Ok" e o cursor. O REM desligará o gravador automaticamente. Se você não tem REM, desligue manualmente com STOP.

Agora, duas observações:

- 1)O CSAVE só serve para gravar programas em BASIC no cassete. Eles serão gravados de maneira compactada (mais adiante explicaremos o que vem a ser isso).
- 2)O nome do programa deve ter, no máximo 6 caracteres sendo que as letras maiúsculas são consideradas diferentes das minúsculas. Caracteres adicionais são ignorados e o "espaço" é considerado um caractere. Assim sendo:

"AREA"
" AREA"
"Area"
"aREa"

são considerados nomes diferentes. Por isso, ao gravar um programa, anote o nome!

Agora rebobine a fita, digite:

CLOAD?"AREA" (e[RETURN])

e aperte o PLAY do gravador (ou LOAD do DATA CORDER).

Este comando faz o micro ler o programa da fita e compará-lo com o que ainda está na RAM do computador.

Se houver alguma discrepância, o Expert emitirá a mensagem "Verify error", para que você faça novamente a gravação.

Digamos, agora, que você tenha desligado o micro.

Ao ligá-lo novamente, não haverá nenhum programa na RAM. Digamos que você tenha uma fita com 4 programas gravados com CSAVE na sequência:

Se você quiser carregar o programa AREA, deverá rebobinar a fita e comandar:

CLOAD"AREA" (e [RETURN]

ligando o gravador com PLAY (ou LOAD do DATA CORDER) a fita começará a "tocar" e aparecerá a tela da figura 7.4.

figura 7.4 - Carregando um programa.

```
CLOAD"AREA"
Skip  :BOLA
Skip  :artes
Found:AREA
Ok
■
```

O micro informou que "passou" pelo "BOLA" e "artes" e achou o "AREA".

Note a extrema utilidade do contador de voltas no gravador: se você anotar o nome do programa e o valor do contador, basta rodar a fita rapidamente até um pouco antes e depois ativar o PLAY. Isso evita a espera de se ler todos os programas sequencialmente.

Se você comandar simplesmente:

CLOAD (e[RETURN])

o micro carregará o primeiro programa que encontrar.

Se você testou tudo isso na prática e alguma coisa não deu certo, pule diretamente para o item PROBLEMAS NA LEITURA.

SAVE"CAS:",LOAD "CAS:" e MERGE

Quando um programa em BASIC está na memória, ele fica numa forma "compactada" para ocupar o menor espaço possível. O comando PRINT, por exemplo, é substituido pelo caractere "æ" de maneira a ocupar apenas um byte e não cinco!

Quando gravamos um programa em BASIC com CSAVE, ele é enviado para a fita de maneira compactada (em "informatiquês" dizemos que ele está "tokenizado"), para economizar tempo de gravação e leitura (pág 85).

Porém, em certos casos, é conveniente enviar o programa "por extenso", ou seja caracter-por-caracter.

Quando fazemos isso, dizemos que estamos enviando o texto do programa em ASCII, de American Standard Code for Interchange Information (leia "asqui-i" e não "asqui-dois"). Para fazer isso, usamos o comando: **SAVE "XXX:NOME"**
onde "XXX:" representa o dispositivo para o qual será enviado o texto intitulado "NOME".

Os vários dispositivos possíveis são:

CAS: gravador cassete (CASsete)

LPT: impressora (Line PrinTer)

CRT: SCREEN 0 ou 1 (Cathode Ray Tube)

GRP: SCREEN 2 ou 3 (GRaPhics)

Para você perceber a diferença entre um programa "tokenizado" e um programa em "ASCII", digite o programa da figura 7.5.

figura 7.5 - Simulando um CSAVE e um SAVE"CAS:na SCREEN 1

```
10 REM CSAVE E SAVE"CAS:
20 SCREEN 1:WIDTH 32:KEY OFF
30 FOR I=1 TO 700
40 X=PEEK(I+32768!)
50 Y=PEEK(I+32769!)
60 Z=PEEK(I+32770!)
70 VPOKE(6144+I),X
80 IF X=0 AND Y=0 AND Z=0 THEN GOTO 100
90 NEXT I
100 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
110 PRINT:PRINT:SAVE"CRT:BABA"
120 REM **FIM**
```

As linhas de 30 a 90 lêem (PEEK) o texto do programa que está na RAM a partir da posição 32769 e o enviam (VPOKE) tal e qual, para a região da VRAM (RAM de vídeo) onde se forma a SCREEN 1, ou seja, a partir da posição 6145 (veja o mapa da VRAM no Apêndice A). A leitura é interrompida ao encontrarmos 3 "nulos" consecutivos, indicativos de "fim de programa".

Na linha 110 enviamos o mesmo programa em "ASCII" para a tela 1(CRT:) com o nome "BABA". Note que, se no lugar de "CRT:"tivéssemos colocado "CAS:" ele seria enviado para o cassete. Da mesma forma, com "LPT:" ele iria para a impressora.

Como você deve ter percebido:

SAVE"LPT:" =LLIST **SAVE"CRT:" =LIST**

Ao rodar o programa, você deve ter obtido a tela da figura 7.6.

figura 7.6 - Diferenças entre um programa "tokenizado" e um programa em "ASCII".

```
 ]Ç■ A CSAVE E SAVE"CAS: *Ç├ ■ ┬
 ...
RT:BABA" ... *FIM**
10 REM CSAVE E SAVE"CAS:
20 SCREEN 1:WIDTH 32:KEY OFF
30 FOR I=1 TO 700
40 X=PEEK(I+32768!)
50 Y=PEEK(I+32769!)
60 Z=PEEK(I+32770!)
70 VPOKE(6144+I),X
80 IF X=0 AND Y=0 AND Z=0 THEN G
OTO 100
90 NEXT I
100 PRINT:PRINT:PRINT:PRINT:PRIN
T:PRINT
110 PRINT:PRINT:SAVE"CRT:BABA"
120 REM **FIM**
Ok
■
```

A única coisa reconhecível no programa "tokenizado" são as mensagens das linhas REM e o nome das variáveis! Se você tem propensão à criptografia, tente decifrar mais alguma coisa: é mais divertido que fazer palavras cruzadas!

De qualquer forma não se preocupe: seu Expert consegue decifrar tudo isso e,como você já percebeu, o espaço economizado vale o trabalho! Ainda mais porque é o micro que trabalha e não a gente!

Agora posicione a fita no seu gravador numa parte virgem (anote o valor no contador), coloque no modo gravação e comande:

SAVE"CAS:BABA" (e[RETURN])

Após gravar o programa em "ASCII", rebobine a fita até o ponto anterior ao começo da gravação, limpe a memória do Expert com NEW e digite o programa da figura 7.7

figura 7.7

```
25 REM Lê OS BYTES DA MEMÓRIA RAM
30 REM ABOBRINHA A SER ELIMINADA
105 REM ENVIA ASCII PARA CRT:
```

Apronte o gravador para leitura do programa "BABA" e comande:

MERGE "CAS:BABA" (e [RETURN])

Ao receber o "OK", comande LIST. Você obterá a tela da figura 7.8.

figura 7.8 - O efeito do MERGE

```
10 REM CSAVE E SAVE"CAS:
20 SCREEN 1:WIDTH 32:KEY OFF
25 REM Lê OS BYTES DA MEMÓRIA RAM
30 FOR I=1 TO 700
40 X=PEEK(I+32768!)
50 Y=PEEK(I+32769!)
60 Z=PEEK(I+32770!)
70 VPOKE(6144+I),X
80 IF X=0 AND Y=0 AND Z=0 THEN GOTO 100
90 NEXT I
100 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
105 REM ENVIA ASCII PARA CRT:
110 PRINT:PRINT:SAVE"CRT:BABA"
120 REM **FIM**
Ok
```

Tudo se passa como se alguém, depois de digitar as linhas da figura 7.7 (25,30 e 105) pedisse a outra pessoa para digitar as linhas do programa "BABA". Os dois programas são "emendados"!

Note que, o que é "digitado" pelo MERGE se sobrepõe ao que já estava no micro! Por isso a linha30 (REM ABOBRINHA A SER ELIMINADA) foi realmente suprimida.

Obviamente, como o MERGE simula uma digitação, só funcionará com programas gravados em "ASCII" com SAVE "CAS:". Se tentarmos fazer um MERGE com um programa "tokenizado", ou seja gravado com CSAVE, não vai dar certo!

Quando quisermos simplesmente carregar um programa gravado em ASCII para a memória RAM, substituindo o que lá se encontra, ou seja, sem fazer "emendas", basta comandar: **LOAD"CAS:BABA"**

ARQUIVOS BINÁRIOS

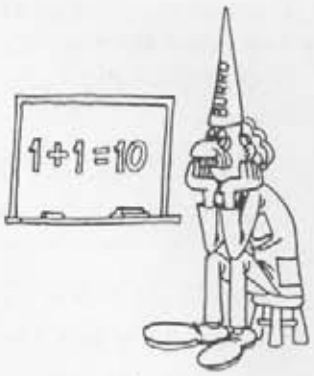
Às vezes é necessário enviar ao gravador um trecho da memória RAM que não contém um programa em BASIC, mas sim uma sucessão de códigos que podem representar, inclusive, um programa em Linguagem de Máquina (LM).

Para fazer uma experiência neste sentido, vamos carregar um programa em LM na memória do Expert e depois vamos ver como gravá-lo.

Digite, com muito cuidado nas linhas DATA, o programa (por enquanto em BASIC) da figura 7.9.

figura 7.9 - Preparando um programa em L.M.

```
1 DATA 21,00,00,22,3D,E0,06,18
2 DATA F3,C5,CD,1C,E0,C1,2A,3D
3 DATA E0,11,28,00,19,22,3D,E0
4 DATA 10,EF,FB,C9,E5,01,28,00
5 DATA C5,11,18,FC,D5,CD,59,00
6 DATA 21,19,FC,11,18,FC,01,27
7 DATA 00,ED,B0,3E,20,12,E1,C1
8 DATA D1,CD,5C,00,C9,00,00,E1
100 SCREEN 0
110 FOR F=&HE000 TO &HE03F STEP 8
115 N=0:PRINT (F-&HE000)/8+1;"->";
120 FOR K=0 TO 7
130 X=F+K:READ A$:Y=VAL("&H"+A$)
140 N=N+Y:REM
150 NEXT K:PRINT N
160 NEXT F
```



Rode-o: você deverá obter a tela da figura 7.10.

figura 7.10 - Checando os códigos das linhas DATA

```
 1 -> 382
 2 -> 1193
 3 -> 625
 4 -> 977
 5 -> 997
 6 -> 643
 7 -> 943
 8 -> 932
Ok
```

Se algum desses números não coincidir, você cometeu um erro de digitação na linha DATA correspondente. Digamos, por exemplo que você tenha digitado 1B no lugar de 18 no quinto código da linha 6. Na tela você obterá: **6 - >646** no lugar de : **6 - > 643** (figura 7.10). Comande então um: **LIST 6** e cheque esta linha com a mostrada na figura 7.9, corrigindo o erro. Repita a operação até ter certeza de que todos os códigos das linhas DATA coincidam com a listagem. Tendo tudo checado, substitua a linha 140 por:

140 N=N+Y: POKE X,Y

Rode o programa: agora os códigos de LM estão sendo colocados na memória a partir da posição &HE000 até &HE03F.

Dê um NEW! O programa em BASIC será apagado mas não o programa em LM!

Para perceber isso, digite o programa da figura 7.11.

figura 7.11-testando o programa em LM.

```
100 DEFUSR=&HE000
110 SCREEN 0:WIDTH 40:KEY OFF
120 FOR I=0 TO 16
130 PRINT"LINHA ";
140 PRINT USING"##";I;
150 PRINT " ";STRING$(30,224+I)
160 NEXT I
170 PRINT:PRINT"APERTE QUALQUER TECLA
180 A$=INPUT$(1):PLAY"S0M5000C2"
190 PRINT:PRINT"SAIDA PELA ESQUERDA!!!"
200 FOR T=0 TO 600: NEXT T
210 FOR I=1 TO 40
220 LOCATE 39,20:PRINT"<"
230 POKE0,USR(0)
240 NEXT I:LIST
```

Rode o programa várias vezes para estudar seu efeito.

A linha 100 avisa o Expert de que tem um programa em Linguagem de Máquina a partir da posição &HE000 da memória.

A linha 230 desvia a execução para este programa. O que ele faz? Toda vez que ele

é chamado, desloca os caracteres da tela de uma posição para esquerda. Como ele é chamado 40 vêzes(laço 210-240), a tela toda acaba sumindo para esquerda! Se você quiser gravar este programa em BASIC, basta preparar a fita e comandar: **CSAVE"DEMO"**

Mas, e se quisermos guardar na fita o programa em LM que está na RAM de &HE000 até &HE03F? Fácil: prepare a fita e o gravador e comande:

BSAVE"CAS:ESQUER",&HE000,&HE03F (e [RETURN])

Note que você deve informar o começo e o fim da região da memória a ser copiada. Agora você pode ressetar o micro, posicionar a fita no começo do "ESQUER" e comandar:

BLOAD"CAS:ESQUER" (e [RETURN])

e o programa em LM será carregado na memória na região própria.

Aliás, vamos aproveitar para fazer uma experiência: rebobine novamente a fita até o começo de "ESQUER" e comande:

BLOAD"CAS:ESQUER",R (e [RETURN])

Quando terminar a carga, toda a tela sofrerá um deslocamento de um caractere para esquerda! O ",R" que você colocou no fim do BLOAD significa: "carregue o programa em LM e EXECUTE-O. Foi o que aconteceu! Sem desligar ou ressetar o micro, rebobine a fita até o começo do "DEMO" e comande:

CLOAD"DEMO" (e [RETURN])

Rode-o: ele funcionará novamente em conjunto com seu programa em LM!

ARQUIVOS SEQUENCIAIS

Numa fita cassete você pode também arquivar dados que não sejam um programa em BASIC ou em LM. Digamos, por exemplo, que você queira criar uma pequena agenda telefônica. Digite o programa da figura 7.12 e grave-o numa fita com o comando:

CSAVE"ESCREV" (e [RETURN])

figura 7.12 - Criando um arquivo sequencial.

```
100 REM ENTRADA DOS NOMES E TELEFONES
110 SCREEN 0:KEY OFF
120 INPUT"QUANTOS NOMES";N
130 DIM N$(2,N)
140 FOR I=1 TO N
150 INPUT"NOME";A$(1,I)
160 INPUT"TELEFONE";A$(2,I)
170 NEXT I
180 REM ROTINA DE CONFERENCIA
190 CLS:FOR I=1 TO N
200 PRINT I;TAB(3);A$(1,I);TAB(30);A$(2,
I)
210 NEXT I:PRINT
220 PRINT "ALGUM ERRO (S/N)?"
230 R$=INPUT$(1)
240 IF R$="S" OR R$="s" THEN INPUT "QUAL
 NUMERO";X ELSE GOTO 280
250 PRINT X:INPUT"NOME";A$(1,X)
260 INPUT"TELEFONE";A$(2,X):GOTO 190
270 REM ROTINA DE GRAVACAO
280 INPUT "NOME DO ARQUIVO";C$
290 IF LEN(C$)>6 THEN 280
300 PRINT "PREPARE O GRAVADOR E APERTE U
MA TECLA":R$=INPUT$(1)
310 D$="CAS:"+C$
320 OPEN D$ FOR OUTPUT AS #1
330 FOR I=1 TO N
340 PRINT#1,A$(1,I):PRINT#1,A$(2,I)
350 NEXT I
360 CLOSE
370 PRINT "ARQUIVO GRAVADO COMO ";C$
```

Até a linha 260, não há novidades para você, a não ser o TAB, que funciona como tabulador do PRINT e é muito útil para formar tabelas. A novidade está a partir da linha 270. Você decide o nome do arquivo(280), com 6 letras no máximo(290), junta-o com o nome do dispositivo a quem vamos enviar dados(310) e, na linha 320, abre(OPEN) um arquivo para saída(OUTPUT) como número 1(AS#1).

A partir daí você lê a matriz A$ de nomes e telefones e os escreve no arquivo 1 (PRINT#1).Na linha 360 você fecha o arquivo (e o gravador, em seguida, vai parar).

Rode o programa e, quando ele solicitar o número de nomes, digite 3 para não tornar esse exemplo muito demorado. Digite, por exemplo, o que está indicado na figura 7.13.

figura 7.13 - Entrando com nomes e telefones.

```
QUANTOS NOMES? 3
NOME? JACQUELINE
TELEFONE? 298-9522
NOME? FLAVIA
TELEFONE? 240-4112
NOME? ADRIANA
TELEFONE? 530-6739■
```

Ao terminar a digitação dos 3 nomes, aparecerá a tela da figura 7.14 (mensagem da linha 220). Digite N (de Não) quando perguntar se há algum erro(ou eventualmente S se houve) e digite, para o nome do arquivo:

AGENDA (e [RETURN])

Prepare o gravador com a fita num trecho virgem e aperte uma tecla qualquer. Seu arquivo será gravado na fita com o nome "AGENDA".

Agora comande: **DELETE 100-270**, e altere as linhas 320, 330, 340, 345, 350 e 370 para obter o programa da figura 7.14.

figura 7.14 - lendo um arquivo sequencial.

```
280 INPUT "NOME DO ARQUIVO";C$
290 IF LEN(C$)>6 THEN 280
300 PRINT "PREPARE O GRAVADOR E APERTE U
MA TECLA":R$=INPUT$(1)
310 D$="CAS:"+C$
320 OPEN D$ FOR INPUT AS #1
330 I=1
340 INPUT#1,A$:INPUT#1,B$
345 PRINT I;TAB(3);A$;TAB(30);B$
350 I=I+1:IF EOF(1) THEN 360 ELSE 340
360 CLOSE
370 PRINT "CONTEUDO DO ARQUIVO ";CHR$(34
);C$;CHR$(34)
```

Limpe a tela e rode o programa. Digite "AGENDA" como nome do arquivo, prepare o gravador rodando a fita até o começo do arquivo "AGENDA" que você gravou anteriormente e pressione uma tecla qualquer. O seu micro lerá o arquivo e mostrará seu conteúdo no vídeo.

Uma novidade, nesse programa, é o EOF(End Of File = Fim de Arquivo). Se ele for verdadeiro encerra-se a leitura (360), se for falso, continuarmos a ler(340). A outra é o modo de abertura (INPUT) e a leitura dos dados(INPUT#1).

PROBLEMAS NA LEITURA

O ouvido humano é extremamente versátil e adaptável. Quando ouvimos um gravador cassete reproduzindo música, nosso cérebro "preenche" eventuais falhas e corrige distorções de maneira a ouvir a música que "quer ouvir" e não a que está sendo realmente tocada. Já o micro-computador não tem essa versatilidade: se faltar um único bit que seja na gravação, a leitura será inevitavelmente prejudicada.

Por isso, além de usarmos fitas de boa qualidade, devemos "afinar" nosso gravador de maneira a obter gravações e leituras confiáveis.

Em primeiro lugar o gravador deve ser de boa qualidade. O ideal é usar gravadores projetados especificamente para gravar programas de computador, como o DATACORDER, por exemplo.

Em segundo lugar ele deve estar limpo, principalmente no cabeçote de gravação. Use um solvente adequado(na falta de coisa melhor, álcool) e cotonete para eliminar resíduos de óxido de ferro que fitas de má qualidade deixam na cabeça de gravação.

Em terceiro lugar, a cabeça de gravação deve estar desmagnetizada. Use um desmagnetizador seguindo as instruções que o acompanham ou leve seu gravador a um técnico: em 5 minutos ele fará isso para você(se não fizer, troque de técnico!).

Em quarto lugar, verifique se os cabos não estão com defeito.

Em quinto lugar, repita os procedimentos de gravação e leitura e tente ler os dados mudando o ajuste de volume e tonalidade (se existir este controle no seu gravador). Tenha paciência e tente todas as combinações possíveis.

Assim que conseguir a leitura anote os valores ótimos ou, melhor ainda, faça uma marca nos controles.

Se tudo isto não funcionar,troque o gravador, pois existem alguns modelos que só são compatíveis com computadores após sérias modificações de circuito. Lembre-se que uma boa reprodução de música não tem nada a ver com gravação de bits.

Use, para leitura, o mesmo gravador usado para gravação. Ligeiras mudanças de rotação ou na regulagem do azimute (ângulo de alinhamento da cabeça de gravação) podem incompatibilizar a gravação com a leitura.

LEITURA DE FITAS TERCEIROS

Quando você tem uma fita que foi gravada num gravador diferente do seu, podem ocorrer problemas de leitura.

O mais fácil de ser resolvido é o que se refere ao volume: às vezes você deve procurar um ajuste diferente do usual.

O outro problema é do alinhamento do cabeçote.

Verifique o ajuste do azimute da cabeça que é feito por um pequeno parafuso do lado esquerdo (figura 7.15). Em alguns gravadores já existe um furo para alcança-lo. Ponha o programa para rodar e ouça-o! Girando o parafuso do azimute de um lado para o outro você deve achar um ponto de estridência máxima. Esse é o ponto ajuste.

figura 7.15 -O ajuste do azimute.

Lembre-se, porém, que ao regular o gravador para outras fitas, você o está desregulando para as suas.

Um procedimento inteligente seria de não mexer em seu gravador e fazer toda essa procura com outro. Após conseguir ler a fita e carregar o programa no seu computador, graveo-o novamente no SEU gravador de maneira a ter uma gravação compatível.

DICIONÁRIO DO BASIC MSX

8

Neste dicionário são listados, em ordem alfabética, os comandos, funções, operadores e variáveis do MSX-BASIC. Se você estiver buscando algum verbete já conhecido, para lembrar sua sintaxe ou particularidades, basta procurá-lo pela ordem alfabética.

Se você quiser executar alguma tarefa e quiser saber quais os recursos disponíveis, veja a relação a seguir:

RELAÇÃO DOS VERBETES POR TIPO DE USO:

DEFINIÇÃO DE TIPO OU MUDANÇA DE TIPO DE VARIÁVEL:
CDBL, CINT, CLEAR, CSNG, DEFINT, DEFSNG, DEFDBL, DEFSTR, STR$, VARPTR.

MANIPULAÇÃO DE VARIÁVEIS:
DATA, DEF FN, DIM, ERASE, INPUT, LET, READ, RESTORE, SWAP.

FUNÇÕES MATEMÁTICAS COM ARGUMENTO E RESULTADO NUMÉRICO:
ABS, ATN, COS, EXP, FIX, INT, LOG, RND, SGN, SIN, SQR, TAN.

FUNÇÕES COM ARGUMENTO E RESULTADO STRING:
INPUT$, INSTR, LEFT$, LINE INPUT, MID$, RIGHT$, SPACE$, STRING$.

FUNÇÕES COM ARGUMENTO NUMÉRICO E RESULTADO STRING:
BIN$, CHR$, HEX$, OCT$.

FUNÇÕES COM ARGUMENTO STRING E RESULTADO NUMÉRICO:
ASC, LEN, VAL.

FUNÇÕES COM RESULTADO CARACTERE:
INKEY$.

TELAS EM GERAL:
BASE, CLS, COLOR, SCREEN, VDP, VPEEK, VPOKE.

TELAS DE TEXTO:
CSRLIN, LOCATE, POS, PRINT, PRINT USING, SPC, TAB, WIDTH.

TELAS GRÁFICAS:
CIRCLE, DRAW, LINE, PAINT, POINT, PRESET, PSET.

MANIPULAÇÃO DE SPRITES:
PUT SPRITE, SPRITE$.

EDIÇÃO E EXCECUÇÃO DE PROGRAMAS:
AUTO, CONT, DELETE, END, LIST, NEW, REM, RENUM, RUN, STOP, TROFF, TRON.

ARQUIVOS:
BLOAD, BSAVE, CLOAD, CSAVE, CLOSE,EOF, INPUT#, LINE INPUT#, LOAD, MAXFILES, MERGE, MOTOR, OPEN, PRINT#, PRINT#USING SAVE.

IMPRESSORA:
LLIST, LPOS, LPRINT, LPRINT USING.

TECLAS DE FUNÇÃO:
KEY, KEY LIST, KEY OFF, KEY ON.

SETAS, JOYSTICK, PADDLE:
PAD, PDL, STICK, STRIG.

SONS:
BEEP, PLAY, SOUND.

LAÇOS, DESVIOS, DECISÕES E SUBROTINAS:
FOR, GOSUB, GOTO, IF, ON..GOTO, ON..GOSUB.

ERROS:
ERL, ERR, ERROR, ON ERROR GOTO, RESUME.

INTERRUPÇÕES:
INTERVAL ON (OFF, STOP), KEY ON (OFF, STOP), ON INTERVAL, ON KEY, ON SPRITE, ON STOP, ON STRIG, SPRITE ON (OFF, STOP), STOP ON (OFF, STOP), STRIG ON (OFF, STOP), WAIT.

RELACIONADOS COM A ARQUITETURA MSX:
BASE, CALL, DEFUSR, FRE, INP, OUT, PEEK, POKE, TIME, USR, VDP, VPEEK, VPOKE.

SIMBOLOGIA DO DICIONÁRIO:

Na sintaxe de cada verbete, usamos a seguinte convenção:

[ ] As construções entre colchetes, são opcionais.

| Apenas uma das duas construções separadas pela barra vertical pode ser utilizada.

{ } Pelo menos uma das costruções entre chaves deve ser utilizada.

Além disso, para orientação do leitor, colocamos em cada verbete 3 ícones com o seguinte significado:

–Elementar, pode ser usado por principiantes.

–Avançado. Para maiores detalhes consulte a bibliografia aconselhada.

–Comando.

–Função.

–Operador.

–Variável especial do BASIC MSX.

–Indicador de tipo de variável.

Os dispositivos a que se refere este dicionário são:

CAS: – Gravador cassete.

CRT: – Tela de texto.

GRP: – Tela gráfica.

LPT: – Impressora.

COM: – Interface RS-232C

Obs: Os dois pontos que seguem o nome do dispositivo, são parte integrante do mesmo.

ABS(expressão)

Fornece o valor absoluto de um número ou expressão.

Exemplo: A=–3:PRINT A,ABS(A)

Como resultado deverá aparecer o número –3 e ao lado seu valor absoluto.

AS

Veja **OPEN**

ASC(string)

Fornece o código **ASCII** do 1º caractere da string.

Exemplo: PRINT ASC("MSX")

A resposta deve ser 77 (código do caractere M)

AND

Operador que faz o "e" lógico *bit a bit* de dois números. Retorna um *bit* 1 somente se ambos os *bits* comparados forem iguais a 1.

Exemplo: PRINT BIN$(&B00101101 AND &B10110001)

Resulta 00100001 em binário.

X \ Y	1	0
1	1	0
0	0	0

ATN(expressão)

Fornece o valor do arco (em radianos) da expressão.

Exemplo: PI=4*ATN(1):PRINT PI

A resposta deve ser 3.1415926535898 pois **ATN(1)** é igual a **PI/4.**

ATTR$

Função não implementada.

AUTO[nº inicial da linha][,incremento]

Numera automaticamente as linhas do programa em edição. Para interrompê-lo, digitar **[CONTROL]** + **[STOP]**. Se aparecer um "*" ao lado do nº da linha isto significa que já existe uma linha com este número.

Exemplo: AUTO 100,5

Deve gerar linhas com numeração 100, 105, 110, 115, 120, etc.

BASE(expressão)[=expressão]

Permite ler ou redefinir o endereço inicial das tabelas do processador de vídeo.

Exemplo: FOR I=0 TO 19:PRINT BASE(I):NEXT I

Fornece os endereços (na *VRAM*) das tabelas utilizadas nas várias telas.

Consulte o Apêndice A

BEEP

Gera um sinal sonoro. Seu efeito é o mesmo de PRINT CHR$(7).

Exemplo: FOR I=1 TO 20:BEEP:FOR K=0 TO 300:NEXT K,I

Gera uma sucessão de 20 sinais sonoros.

BIN$(expressão)

Converte o valor da expressão numa string em notação binária.

Exemplo: FOR I=0 TO 22:PRINT I,BIN$(I):NEXT I

Imprime na tela os números de 0 a 22 em notação decimal e binária.

BLOAD"{[dispositivo][nome arq]}"[,R][,deslocamento]

Carrega arquivos em formato binário para uma região da memória.

dispositivo — Veja no início deste capítulo.

nome arq — Veja no início deste capítulo.

,R — Ativa a execução automática.

deslocamento — Transfere o bloco deslocando os endereços definidos no **BSAVE.**

Exemplo: BLOAD"CAS:JOGO",R,-300

Carrega, a partir do cassete o bloco "JOGO" na RAM subtraindo 300 dos endereços originalmente definidos no **BSAVE** e ativa a execução automática.

BSAVE"{[dispositivo][nome arq]}",inicio,fim [,execução]

Grava, num dispositivo, um arquivo de formato binário.

dispositivo – Veja início deste capítulo.

nomearq – Veja no início deste capítulo

inicio – Marca o endereço inicial do bloco binário

fim – Marca o endereco final do bloco binário

execução – Marca o endereço de execução para automátização na opção R do comando **BLOAD**. Caso não definido é assumido o início do bloco.

Exemplo: BSAVE"CAS:JOGO",20000,30000,21000

Grava no K-7 o bloco de memória, com o nome JOGO, e endereço 21000 para execução.

CALL nome do comando [,argumentos]

Executa um comando contido num cartucho. Pode ser usado o "_" como abreviação para "CALL".

Exemplo: CALL ASTEX

Chama o programa de cartucho "ASTEX" (Editor de textos disponível no cartucho CT-80E).

CDBL (expressão)

Converte a expressão em um dado numérico de precisão "dupla".

Exemplo: C=CDBL(12.34!):PRINT C

CHR$ (expressão)

Função inversa a "ASC". Fornece o caractere correspondente ao valor da expressão na faixa de 0 a 255.

Exemplo: FOR F=128 TO 255:PRINT F;CHR$(F),:FOR G=1 TO 200:NEXT G,F

Os códigos e caracteres de 128 a 255 serão impressos na tela.

CINT(expressão)

Converte a expressão em um número inteiro, desprezando as casas à direita do ponto decimal, na faixa de –32768 a +32767.

Exemplo: A=3.2:B=3.8:PRINT CINT(A);CINT(B);-CINT(-A);-CINT(-B)

Veja os resultados e compare com os do exemplo do **FIX**.

CIRCLE [STEP] (var1,var2),raio[,cor][,início][,fim][,proporção]

Traça na tela gráfica, círculos, elípses e setores.

var1 – Expressão que define o valor da abscissa do centro.

var2 – Expressão que define o valor da ordenada do centro.

raio – Expressão entre –32768 e 32767 que indica o raio.

cor – Indica a cor, entre 0 e 15, do arco traçado.

início – Início do arco, em *radianos*. Se negativo traça um setor.

fim – Final do arco, em *radianos*. Se negativo traça um setor.

proporção – Relação entre absissas e ordenadas; se diferente de um, forma elípses.

Exemplo:10 SCREEN 2:CIRCLE(128,96),20:CIRCLE(128,96),20,,,,.2
20 GOTO 20

Desenha um círculo e uma elípse no centro da tela.

CLEAR (área de string, HIMEM)

Inicializa todas variáveis, arquivos e define áreas de armazenamento de *strings* e topo de memória **RAM (HIMEM)**

Exemplo: CLEAR 600,50000

Inicializa variáveis e arquivos, define 600 bytes para string's e define 50000 como último endereço disponível para o **BASIC MSX**.

CLOAD[?] ["nome arq"]

Transfere dados em **BASIC MSX** gravados em cassete, para o computador. Se a opção "?" for usada, faz a verificação do arquivo proveniente da fita com o da memória. Se ocorrer uma falha, a mensagem "Verify error" será impressa. Se for definido o "nome arq" ele aguarda a ocorrência do arquivo na fita, e em seguida carrega-o para a memória.

Exemplo: CLOAD?

Compara a primeira ocorrência na fita com o programa da memória.

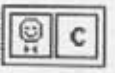
CLOSE [#][numarq1][,numarq2][,....

Fecha um ou mais arquivos abertos pelo "**OPEN**". Se for especificado o número do arquivo, então apenas o citado será fechado, senão, todos.

Exemplo: CLOSE #1

Fecha apenas o arquivo 1. Veja "**OPEN**".

CLS



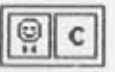
Limpa toda a tela, e coloca o cursor no canto esquerdo da tela.

Exemplo: CLS:FOR I=0 TO 21:PRINT"GRADIENTE":NEXT I:CLS

CMD

Comando não implementado. Pode ser implementado pelo usuário com bons conhecimentos em programação, para comandos próprios.

COLOR [frente],[fundo],[borda]



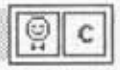
Especifica as cores da tela, onde "frente" define a cor dos caracteres, "fundo" a cor de fundo e, "borda", a cor da borda, segundo a tabela:

0 - Transparente	4 - Azul Escuro	8 - Vermelho	12 - Verde Escuro
1 - Preto	5 - Azul Claro	9 - Vermelho Claro	13 - Magente
2 - Verde	6 - Vermelho Escuro	10 - Ouro	14 - Cinza
3 - Verde Claro	7 - Ciano	11 - Amarelo	15 - Branco

Exemplo: COLOR 12,15,6

Define frente verde escuro, fundo branco, e borda vermelha escura.

CONT



Continua a execução de um programa, a partir da linha seguinte àquela onde o programa foi interrompido pelo [CONTROL] + [STOP].

CSAVE "(nom arq)" [,transmissão]

Transfere programas em **BASIC MSX** para o cassete com um nome de 6 letras (nom arq) e tipo de transmissão 1 ou 2 (1200/2400 bauds).

Exemplo: CSAVE "PROG1",2

Grava o programa da memória com o nome "PROG1" e velocidade de transmissão de 2400 bauds (bits p/ segundo).

OBS: No nome, as minúsculas são consideradas diferentes das maiúsculas.

CSNG (expressão)

Converte o valor da expressão num número em precisão simples.

Exemplo: A! = CSNG(1.234567891):PRINT A!

CSRLIN

Fornece o número da linha onde se encontra o cursor.

Exemplo: PRINT CSRLIN,CSRLIN

DATA dado [,dado2,...]

Define uma lista de dados numéricos ou alfanuméricos separados um a um por vírgula, para serem lidos posteriormente pelo "**READ**".

Exemplo:10 FOR F = 1 TO 10:READ A$,A:PRINT A$,A:NEXT F

20 DATA Fe,55.8,Cl,35.5,O,16.0,P,31.0

Lê e imprime os dados das linhas "**DATA's**". (Veja comando **READ** e **RESTORE**).

OBS: se um dos caracteres a ser lido for a própria vírgula, ela deve ser colocada entre aspas.

DEF

Veja **DEF FN, DEF INT, DEF SNG, DEF DBL, DEF USR.**

DEF FN nome [(parâmetros)] = expressão

Define uma função "nome" criada pelo usuário. Até 9 parâmetros podem ser passados, para serem usadas como variáveis para calcular a expressão. Não pode ser usado como comando direto, e ao chamar a função, o nome deverá ser precedido por **FN**.

Exemplo:10 DEF FN E(X,Y) = X + 2*Y

20 PRINT FN E(-3,2)

Obtém o resultado 1 ((-3) + 2*(2)).

DEF INT/SNG/DBL/STR caractere [-caracter] [,caracter] [,...]

Define o tipo de variável correspondente à primeira letra do nome. INT corresponde a inteiras, SNG a precisão simples, DBL precisão dupla e STR a alfanuméricas (string).

Exemplo: DEFSTR A-D:DEFDBL E,X-Z:DEFINT F-L:DEFSNG M

Variáveis com nome iniciando com letras entre A e D, são alfanuméricas; E e entre X e Z, precisão dupla; entre F e L, inteiras; e M do tipo precisão simples.

DEF USR[X] = endereço

Define o ponto de entrada de uma rotina em *linguagem de máquina*, onde X (entre 0 e 9) corresponde o número da entrada, e o endereço um inteiro. Para chamar a rotina em L.M. veja "USR".

Exemplo: DEFUSR0 = 0:DEFUSR8 = 40000

Define entrada 0 com endereço 0 e, entrada 8 com endereço 40000.

DELETE {[num1] [-num2]}[.]

Apaga linhas de um programa em BASIC: somente 1 linha (num1); bloco de linhas (num1-num2); do início até linha (-num2); ou linha corrente (opção ".").

DIM nome1 (indice1 [,indice2,...]) [,nome2....]

Define variável "nome" como tipo matriz, sendo que a sua dimensão depende do número de "índices" especificados. O valor do índice especifica o número de elementos relativos à matriz.

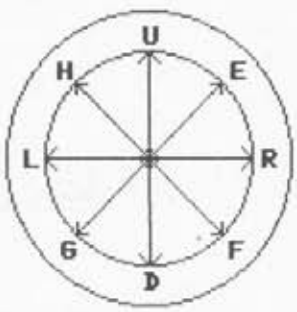
Exemplo: DIM A$(6),X(3,4,5)

Define uma matriz alfanumérica de 6 elementos e uma matriz numérica de dimensão 3 com 3x4x5 elementos.

DRAW "sub-comandos"

Desenha na tela gráfica figuras criadas pelo usuário, seguindo as ordens especificadas em "sub-comando", em ordem sequencial, de maneira que cada comando desenha a partir do último ponto marcado na tela.

O comando **DRAW** permite vários tipos de sub-comandos gráficos, listados a seguir, com suas respectivas definições:



Sx – x define a escala do desenho (default = 4);

Ax – x de 0 a 3: define a orientação dos eixos (default = 0);

Cx – x de 0 a 15: define a cor do traço (veja **COLOR**);

Ux – Move o ponto-cursor para cima, desenhando. Além de U existem outros 7 sub-comandos que desenham nas direções indicadas na figura:

B – Quando colocado antes de um sub-comando, desloca o ponto-cursor sem traçar nada;

N – Quando colocado antes de um sub-comando, não atualiza as coordenadas (mantem as anteriores);

Mx,y – Traça uma linha desde a posição atual até as coordenadas (x,y) da tela (não é afetado pelo "S");

M‡x,‡y – Traça uma linha desde a posição atual (X0,Y0) até X0‡x e Y0‡ y (é afetado pelo "S");

**X A$;** – Permite inserir o conteúdo de A$ na lista de sub-comandos.

=V; – Permite inserir o valor da variável V na string de sub-comandos.

Exemplo:
```
10 SCREEN 2:FOR I=0 TO 4:PSET(160+I,90+I)
20 FOR S=4 TO 28 STEP 4:A=(S/4)MOD 4
30 DRAW "S=S;A=A;U4F2E2D4BR3R3E1H1L2H1E1R3BR3F4BU4G4BE8":NEXT S,I
40 GOTO 40
```

Analise cada parte do programa acima, e observe a flexibilidade do comando **DRAW**.

Para interromper use **[CONTROL] + [STOP]**.

ELSE

Veja "**IF**".

END

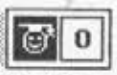
Termina a execução de um programa, fechando todos os arquivos e retorna ao modo de edição. Costuma ser colocado no final do programa, ou para separar a rotina principal das sub-rotinas.

EOF

Função que sinaliza o fim de um arquivo: caso este for detectado, a função retorna o valor – 1 (VERDADEIRO), senão retorna 0 (FALSO).

Veja o exemplo de **INPUT#**.

EQV



Operador lógico que testa dois argumentos bit a bit. Se os bits forem iguais gera um bit 1, caso contrário gera um bit 0.

Exemplo: PRINT RIGHT$("0000000" + BIN$(&B01010101 EQV &B11000011),8)

O resultado será 01101001. O exemplo acima ilustra ainda as técnicas obtidas pelo **RIGHT$**, formatando a saída para 8 dígitos.

X \ Y	1	0
1	1	0
0	0	1

ERASE nome1 [,nome2...]

Apaga da memória, a variável "nome" do tipo matriz.

Exemplo:
```
10 DIM A(6):FOR I=1 TO 6:A(I)=7-I:NEXT I
20 FOR I=1 TO 6:PRINT A(I):NEXT I:ERASE A
30 FOR I=1 TO 6:PRINT A(I):NEXT I:END
```

ERL / ERR

Variáveis que fornecem o número da linha em que ocorreu erro, e o número do erro respectivamente. **ERL** fornece 65535, se o erro ocorreu em um comando direto.

Exemplo: 10 LINHA ERRADA

Ao executar o programa (**RUN**), ocorrerá um erro. Digite então:

PRINT ERL;ERR

ERR

Veja **ERL**.

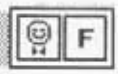
ERROR número

Permite que o usuário simule um erro, e defina o "número" deste.

Exemplo:
```
10 ON ERROR GOTO 1000:PRINT "DIGITE '1'"
20 A$=INPUT$:IF A$<>"1" THEN ERROR 230
30 PRINT "OBRIGADO":END
1000 PRINT "EU DISSE '1'":RESUME 10
```

OBS: Veja também RESUME

EXP (argumento)



Função que fornece o valor do exponencial natural do "argumento", ou seja, o *número de Euler* "**e**" (2.71828) elevado ao argumento.

Exemplo: PRINT EXP(1)

Que fornece como resultado o *número de Euler*.

FIX (argumento)

Função que fornece a parte inteira do argumento, desprezando os dígitos após o ponto.

Exemplo: PRINT FIX(3.8);FIX(3.2);FIX(-3.2);FIX(-3.8)

Veja os resultados e compare com os dos exemplos do "**INT**" e "**CINT**".

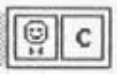
FN função

Executa uma função definida pelo usuário. Só pode ser usado no modo programado.

Veja também **DEF FN**.

Exemplo:
```
10 DEF FN COT(X) = 1/TAN(X)
20 PI = 4*ATN(1):X = PI/2
30 PRINT "COT(PI/3) = "; FN COT (X)
```

FOR var=ini TO fim [STEP passo] /NEXT [var [,var2]]



Este dois comandos formam uma das estruturas básicas da programação. "Var" é uma variável usada como contadora, que varia de "ini" até "fim" com um incremento de "passo" (não necessariamente inteiro), repetindo assim o bloco entre "**FOR**" e "**NEXT**".

O "**NEXT**" pode apontar mais de uma variável. O valor "*default*" de "**STEP**" é 1.

Exemplo:
```
10 FOR I = 1 TO 2 STEP .1
20 PRINT "EXP(";I;") = ";EXP(I)
30 FOR J = 0 TO 500:NEXT J,I:END
```

Ilustra a técnica do **FOR/NEXT**, expondo os valores da função no intervalo 1 e 2. Observe como facilita a criação de tabelas e gráficos.

FPOS

Função não implementada.

FRE (argumento)

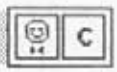
Função que retorna a área de memória livre (em bytes), se o argumento for um número. Se o argumento for do tipo "*string*", fornece a área de "*string*" livre, em bytes.

Exemplo: PRINT FRE(0);FRE("")

GO TO linha

Veja **GOTO**.

GOSUB linha1 / RETURN [linha2]



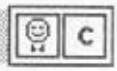
Transfere a execução do programa para "linha1", e executa então a "sub-rotina" até o ponto que encontrar o comando "**RETURN**", retornando assim de onde havia partido.

Se "linha2" for especificada, então a execução reinicia-se na linha indicada.

Exemplo:
```
10 PRINT "INICIO":GOSUB 500
20 PRINT "MEIO":GOSUB 500
30 PRINT "FIM":END
500 PRINT "*** SUBROTINA ***"
510 RETURN
```

O uso de sub-rotinas é adequada quando a mesma tarefa deve ser executada diversas vezes, evitando assim redigitar todo o trecho novamente.

GOTO linha



Comando que transfere a execução para a "linha" especificada.

Exemplo:
```
10 PRINT"GRADIENTE - ";
20 GOTO 10
```

Este programa só é interrompido com **[CONTROL] + [STOP]**.

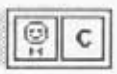
HEX$(argumento)

Converte o "argumento", em uma "*string*" contendo o valor deste, em notação hexadecimal (base 16). É útil quando se trabalha em "*Assembly*".

Exemplo: FOR F=0 TO 16:PRINT F;"decimal - ";HEX$(F);" hexadecimal":NEXT F

Mostra uma pequena tabela de conversão decimal-hexadecimal.

IF condição THEN comando1 [ELSE comando2]



Permite verificação de situações diversas. A "condição" é verificada e se verdadeira executa "comando1", caso falsa executa "comando2"; se omitido o **ELSE** executa linha seguinte.

Exemplo:
```
10 PRINT"Tecle '7'":A$=INPUT$(1)
20 IF A$="7" THEN PRINT "CORRETO" ELSE PRINT"ERRADO!":GOTO 10
30 END
```

Observe que esta estrutura é muito importante na tomada de decisões.

INKEY$

Função que retorna o caracter da tecla pressionada. Se não houver tecla pressionada retorna "*string*" nula.

Exemplo:
```
10 A$ = INKEY$
20 PRINT A$;:GOTO 10
```

Simula uma máquina de escrever. Pare com **[CONTROL] + [STOP]**.

IMP

Operador lógico de implicação. Comparando dois *bits*, retorna 0 apenas se somente o segundo *bit* for 0. Caso contrário retorna 1.

Exemplo:
```
PRINT RIGHT$("0000000" + BIN$((&B11110000)IMP(&B11000011)),8)
```

Deve retornar: 11001111

INP (argumento)

Lê a porta da via de acesso especificada no argumento. Este comando só tem utilidade para aqueles que estão bem familiarizados com o "*hardware*" do **MSX**.

Exemplo:
```
PRINT BIN$(INP(&HA8))
```

Imprime a atual configuração dos "*slots*".

INPUT

Veja as variações do **INPUT**: **INPUT, INPUT#** e **INPUT$**.

INPUT ["mensagem";] var [,var2...]

Imprime (na tela) a mensagem, se for especificada, e lê um valor dado pelo teclado e copia na variável "var". Se forem especificados mais variáveis, os seus respectivos valores devem ser separados por vírgula. **[RETURN]** termina a inserção.

Comando ideal para facilitar a entrada de dados pelo usuário, sem que esse, conheça necessariamente a estrutura do programa.

Exemplo:
```
10 INPUT "Entre nome e número ";A$,N
20 PRINT "Nome: ";A$,"Número:";N
30 PRINT:GOTO 10
```

Lê uma variável do tipo "string" e outra do tipo numérico e as imprime no vídeo.

INPUT #número,lista

Lê dados de um arquivo especificado por "número", associando cada dado às variáveis especificadas na "lista".

Exemplo:
```
10 OPEN "CAS:TESTE" FOR INPUT AS #1
20 IF EOF< >0 THEN 100
30 INPUT #1,A$,N
40 GOTO 20
100 CLOSE #1:END
```

Lê todos nomes (A$) e números (N) de um arquivo em cassete.

INPUT$(argumento) [,#arq]

Lê um número de caracteres pelo teclado, especificado pelo "argumento". Se for especificado o número do arquivo "arq", então a leitura é feita através do arquivo.

Exemplo: A$ = INPUT$(10):PRINT A$

INSTR([N,] string1,string2)

Localiza a posição (1º caracter) de "string2" em "string1"; se não for encontrado retorna 0. Se "N" especificado, então inicia a procura á partir do n-ésimo caracter da "string1".

Exemplo: PRINT INSTR("GRADIENTE & ALEPH","ENTE")

Retorna o valor 6.Experimente outras "strings".

INT (argumento)

Função que converte o argumento em um número inteiro, arredondando sempre para MENOS. Difere de **CINT** e **FIX**.

Exemplo: PRINT INT(3.8);INT(3.2);INT(-3.8);INT(-3.2)

Compare os resultados com os obtidos nos exemplos de **CINT** e **FIX**.

INTERVAL ON/OFF/STOP

Comando que habilita, desabilita ou adia (respectivamente) a interrupção feita pelo temporizador interno.

Veja o exemplo no comando "**ON INTERVAL GOSUB**"

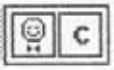
IPL

Comando não implementado no **BASIC MSX** padrão.

KEY

Veja as variações de **KEY**: **KEY, KEY LIST, KEY ON/OFF** e **KEY (n) ON/OFF/STOP**.

KEY tecla,string

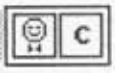


Associa a "string" à "tecla" de função (**F1** a **F10**).

Exemplo: KEY 1,"GRADIENTE":KEY 2,"ALEPH"

A seguir tecle **[F1]** e **[F2]** e observe o resultado.

KEY LIST



Lista o conteúdo associado à cada uma das 10 teclas de função.

Exemplo: KEY LIST

KEY ON/OFF

Ativa e desativa (respectivamente) a visualização na parte inferior do vídeo dos conteúdos das teclas de função.

Exemplo: KEY OFF

KEY (num) ON/OFF/STOP

Comando que ativa, desativa ou adia (respectivamente) a interrupção via teclas de função.

"num" indica a qual tecla de função deve-se atribuir a interrupção.

Veja o exemplo em **ON KEY GOSUB**.

LEFT$ (string,n)

Fornece como resultado os n primeiros caracteres da "*string*".

Exemplo: A$ = LEFT$("MICROCOMPUTADOR",5):PRINT A$

O resultado é: MICRO. Veja também **MID$** e **RIGHT$**.

LEN (string)

Função que retorna o comprimento da "string" em questão.

Exemplo: PRINT LEN("MICROCOMPUTADOR")

Retorna 15, pois MICROCOMPUTADOR tem 15 letras.

[LET] variável = expressão

Atribui o valor da expressão a uma variável. **LET** pode ser omitido no **BASIC MSX**. Deve ser usado quando quer se gerar um programa mais "universal", cujo texto possa ser lido por outros micros.

O **LET** pode ser considerado o comando básico para qualquer sistema. Através dele pode-se iniciar constantes; atualizar cálculos de fórmulas matemáticas; aceita funções matemáticas, lógicas e outras; trabalha com variáveis numéricas ou cadeias de caracteres (*string*), entre outros.

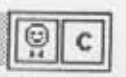
```
Exemplo:10 A = 3:LET B = A*A:PRINT A;" ^ 2 =";B
        20 PRINT "4 x";A;" = ";:A = A + A + A + A:PRINT A
        30 PRINT "4 x";A;" = ";:A = 4*A:PRINT A
        40 B$ = "EDI":C$ = "ALE":LET B$ = B$ + "TORA " + C$ + "PH":PRINT B$
```

Aqueles que se iniciam em programação, devem procurar conhecer bem as potencialidades deste comando. Observe que dentro do mesmo comando deve-se ter só um tipo de valor (numérico ou alfanumérico) – para eventuais conversões veja **STR$** e **VAL**.

LINE

Veja as variações em: **LINE, LINE INPUT** e **LINE INPUT#**.

LINE [[STEP] (X0,Y0)] – (X1,Y1) [,cor] [,B][,BF]



Traça no modo gráfico, uma linha entre os pontos de coordenadas (X0,Y0) e (X1,Y1), com cor especificada. Se **B** for especificado desenha um retângulo; **BF** desenha um retângulo com o interior pintado. A opção **STEP** faz o comando assumir o último ponto desenhado, como o primeiro par de coordenadas do comando.

```
Exemplo:10 SCREEN 2:LINE (10,10)-(245,182),15,BF
        20 LINE (50,50)-(205,142),,B:LINE STEP (0,0),6
        30 GOTO 30
```

Mostra os 3 tipos de opções do comando **LINE**.

LINE INPUT ["mensagem"] variável

Comando semelhante ao **INPUT** (veja **INPUT**), com a diferença de não imprimir o ponto de interrogação no ínicio da entrada e só aceitar variáveis alfanuméricas (string).

Exemplo: LINE INPUT A$:PRINT "COMPRIMENTO:";LEN(A$)

Lê uma "*string*" e imprime o seu comprimento.

LINE INPUT #num,variável

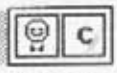
Lê uma "*string*" do arquivo numerado por "num", e atribui o conteúdo a uma variável de mesmo tipo.

Exemplo:
```
10 OPEN "CAS:TESTE" FOR INPUT AS #1
20 IF EOF THEN 100
30 LINE INPUT #1,A$:PRINT "NOME: ";A$:GOTO 20
100 CLOSE #1:END
```

Lê todos os nomes e imprime-os na tela.

É útil para ler strings com vírgulas que, de outra forma, seriam separadores de variáveis

LIST [{[num1] – [num2]}] / LLIST [{[num1] – [num2]}]



Lista o programa entre as linhas num1 e num2. Sem opção, lista tudo. As outras opções são:

– **num2** – Lista do início até num2;

num1 – – Lista à partir de num1;

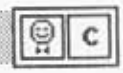
. – Lista a última linha inserida, alterada, ou executada.

LLIST é semelhante ao **LIST**, sendo que a listagem é enviada a impressora ao invés da tela.

Exemplo: LIST 10-30

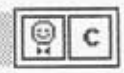
Lista as linhas entre 10 e 30 inclusive, se existirem.

LLIST



Veja **LIST**.

LOAD "disp nome" [,R] – (forma genérica)



Comando que carrega um programa em **BASIC MSX** do dispositivo (disp) para a memória do micro, que tenha o mesmo nome (nome), se especificado.

Para um sistema baseado em cassete ele deve ter o seguinte formato:

LOAD "CAS:[nome]" [,R]

"Nome" pode ser omitido, e neste caso carrega o primeiro arquivo encontrado; senão aguarda até a ocorrência do arquivo com o mesmo nome, e só então os dados são transferidos para a memória do micro. O nome deve ter no máximo 6 letras, sendo que maiúsculas e minúsculas são consideradas letras diferentes.

O formato de gravação na fita é o **ASCII**.

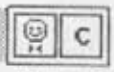
Exemplo: LOAD "CAS:TESTE",R

A opção "R", faz com que o programa seja executado automaticamente.

Exemplo: LOAD "B:TESTE.BAS"

Carrega o programa TESTE.BAS, se existir no diretório, na memória do micro.

LOCATE num1,num2,num3



Comando que posiciona o cursor na coluna num1 e na linha num2 da tela de texto. "Num1" deve estar entre 0 e 39 na **SCREEN 0** e entre 0 e 31 na **SCREEN 1**.

"Num3" controla a presença ou não do cursor durante a impressão. 1 ativa e 0 desativa. Normalmente é 0, mas alguns programas colocam-no em 1 (caso do **MSXDOS**).

Exemplo: LOCATE 20,10:PRINT "EXPERT PLUS"

Coloca o texto do PRINT a partir das coordenadas (20,10) da tela.

LOG (argumento)

Função que retorna o logaritmo natural (base e) ou *neperiano* do argumento.

Exemplo: X=3:PRINT LOG(X);LOG(X)/LOG(10)

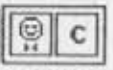
Fornece o logaritmo natural e decimal de X.

LPOS (argumento)

Função que, independente do valor do argumento, retorna a posição do cabeçote da impressora.

Exemplo:
```
10 A$="":FOR F= 0 TO 60
   20 LPRINT A$;LPOS(1)
   30 A$=A$+">":NEXT F: END
```

LPRINT



Funciona igualmente ao **PRINT**, com a diferença de direcionar a impressão para a impressora ao invés da tela. Veja **PRINT** para sintaxe. Consulte também o Apêndice B.

LPRINT USING

Funciona igualmente a **PRINT USING**, com a diferença de direcionar a saída para a impressora ao invés da tela. Veja **PRINT USING**.

MAX

Veja a variação **MAXFILES**.

MAXFILES=expressão

A expressão deve estar entre 0 e 15 e limita o número de arquivos que poderão ser abertos por **OPEN**. Se for especificado 0 então nenhum arquivo pode ser aberto, funcionando porém o **SAVE** e o **LOAD**.

Exemplo:
```
10 MAXFILES=1:OPEN"GRP:" AS #1
   20 SCREEN 2:PRINT#1,"SCREEN 2"
   30 GOTO 30
```

Este programa escreve um pequeno texto na tela gráfica. Para testar o uso de **MAXFILES** acrescente a linha:

```
15 OPEN "CRT:" AS #2
```

Execute o programa e verifique que ocorreu erro!!

MERGE "disp: nome"

Este comando soma um arquivo gravado no dispositivo (disp), no formato **ASCII**, com o programa da memória do micro. As linhas que tiverem números iguais serão substituídas pelas linhas do programa externo.

Para um sistema baseado em cassete, o programa externo deve ter sido gravado com **SAVE"cas:** (**CSAVE** ou **BSAVE** têm formato incompatível com **MERGE**).

Exemplo: MERGE "cas:TESTE"

MID$

MID$** possue duas variações: **MID$ e **MID$ =**. Veja a seguir.

MID$ (var$,num1 [,num2])

Seleciona um subconjunto de caracteres de uma string.

var$ — Variável do tipo string (ou cadeia de caracteres). Pode ser uma string constante.

num1 — Indica a posição inicial, a partir de onde será extraído o subconjunto.

num2 — Determina o comprimento da nova string. Se for omitido, a *string* resultante será formada por todos os caracteres à partir da posição "num1".

Exemplo: PRINT MID$("GRADIENTE",3,4)

O que resulta: ADIE. Para melhor manipulação de pedaços de string, veja ainda **LEFT$**, **RIGHT$**, **LEN**.

MID$(var$,num1 [,num2]) = var2$

Substitue em var$, à partir da posição num1, num2 caracteres de var2$.

Se num2 for omitido, então todos os caracteres de var2$ serão enxertados, até completar o comprimento de var$.

Exemplo:
```
10 A$ = "CARACTER":B$ = "VELAME"
20 MID$(A$,5) = B$:PRINT A$
30 END
```

MOD

Operador que retorna o resto inteiro da divisão de dois números, variáveis, constantes ou expressões.

Exemplo: PRINT 8 MOD 3;15 MOD 5

Que resulta 2 e 0.

MOTOR ON/OFF

Comando que controla o estado do relé para o motor do cassete, ou seja liga (ON) ou desliga (OFF) o motor do cassete.

Exemplo: MOTOR OFF

Paralisa o gravador, se este estiver ligado.

NEW

Reinicia a memória do micro, apagando o programa **BASIC**. Não costuma afetar programas em linguagem de máquina que estiverem na memória.

Exemplo: NEW

Se existir algum programa **BASIC** na memória, este será perdido. Observe que este comando não altera o conteúdo dos periféricos (cassete, Drive, etc.).

NEXT

Veja comando **FOR**.

NOT (expressão lógica)

Uma expressão lógica pode ser constituída por vários testes (=, >, <, **AND**, **OR**, etc.), e retorna apenas dois estados; são estes verdadeiro ou falso. O computador define 0 como falso e -1 como verdadeiro.

O operador lógico **NOT** simplesmente fornece o "*não*" lógico da expressão ou seja "não verdadeiro = falso" e "não falso = verdadeiro".

Isto significa que, se uma expressão lógica retorna um estado, **NOT** o inverte.

Exemplo: PRINT NOT(4>5);NOT(4<5)

Também inverte os "bits" de um número inteiro binário.

Exemplo: PRINT BIN$(NOT(&B01100000))

O que resulta 1111111110011111 (os 8 1's que apareceram na frente, se devem ao fato do **MSX-BASIC** manipular os números inteiros em 16 bits).

X	
1	0
0	1

OCT$(expressão)

Função que retorna uma cadeia de caracteres, contendo a notação octal (base 8) do valor da expressão inteira.

Exemplo: PRINT OCT$(16)

Converte o valor decimal 16 no valor da base octal (Veja **&O**).

OFF

OFF significa "*desligado*", e é encontrado apenas em conjunto com outros comandos. Veja: **KEY, INTERVAL, MOTOR, SPRITE, STOP** e **STRIG**.

ON

ON significa "*ligado*" *ou* "*em*", *e é encontrado como comando e em conjunto com outros comandos. Veja:* **KEY, INTERVAL, MOTOR, SPRITE, STOP, STRIG, ON ERROR, ON GOSUB, ON GOTO, ON INTERVAL, ON KEY, ON SPRITE, ON STOP** e **ON STRIG**.

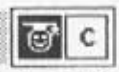
ON ERROR GOTO linha

Desvia o programa para a linha especificada, quando for detectado algum tipo de erro. **ERR** e **ERL** são atualizados com seus valores corretos. Após este desvio pode-se fazer qualquer tipo de manipulação dos erros, e para retornar usar **RESUME.**

Exemplo:
```
10 ON ERROR GOTO 100
20 INPUT A:PRINT "RAIZ DE";A;" =";SQR(A)
30 GOTO 20
100 PRINT "NAO EXISTE RAIZ REAL":RESUME
```

Digite e execute o programa acima, e entre um valor negativo.

ON var GOSUB linha1 [,linha2,....]



Salta para a linha indicada por "var". Se var=1 então salta para linha1; se var=2 então executa linha2, e assim por diante.

As sub-rotinas executadas devem terminar com **RETURN** (Consulte **GOSUB**).

Exemplo:
```
10 INPUT A:ON A GOSUB 30,40,50
20 GOTO 10
30 PRINT "SUB-ROTINA 1":RETURN
40 PRINT "SUB-ROTINA 2":RETURN
50 PRINT "SUB-ROTINA 3":RETURN
```

ON var GOTO linha1 [,linha2,...]

Salta para a linha1 se var=1, linha2 se var=2, e assim em diante.

Exemplo:
```
10 INPUT "Entre: 1-Criança 2-Jovem 3-Adulto";A
20 PRINT "Você é ";
30 ON A GOTO 100,200,300
40 END
100 PRINT "uma criança":END
200 PRINT "um jovem":END
300 PRINT "um adulto":END
```

Pede a entrada de dados de acordo uma tabela, e apresenta a mensagem correspondente.

ON INTERVAL=num GOSUB linha

Desvia periodicamente a execução do programa para uma sub-rotina em intervalos de tempo definido por num/60 segundos.

"Linha" indica o início da sub-rotina.

Exemplo:
```
10 TIME=0:ON INTERVAL=120 GOSUB 100
20 PI=4*ATN(1):INTERVAL ON
30 FOR X=0 TO 2*PI STEP .001
40 PRINT X;SIN(X):NEXT:END
100 PRINT TIME\60:RETURN
```

O programa acima imprime os vários arcos e seus respectivos senos e, periodicamente, é interrompido para imprimir o tempo de execução.

ON KEY GOSUB linha1 [,linha2...]

Desvia o programa para uma sub-rotina, dependendo da tecla de função pressionada. Para uma tecla ser ativada use **KEY (n) ON**.

Exemplo:
```
10 ON KEY GOSUB 100,200,300:KEY(1) ON:KEY(2) ON:KEY(3) ON
20 PRINT"PRESSIONE UMA TECLA DE FUNÇAO"
30 PI=4*ATN(1):FOR X=0 TO 2*PI STEP .001:PRINT X;COS(X):NEXT:END
100 PRINT "FUNÇAO 1":RETURN
200 PRINT "FUNÇAO 2":RETURN
300 PRINT "FUNÇAO 3":RETURN
```

O programa acima mostra uma tabela com o arco e seu cosseno. Se forem pressionadas as teclas de função **[F1]** a **[F3]**, o fluxo interrompe e é iniciada a execução da subrotina correspondente à tecla pressionada, e ao final retorna para o fluxo normal.

ON SPRITE GOSUB linha

Caso dois "*sprites*" se sobreponham, então o programa salta para a sub-rotina especificada por "linha", desde que tenha sido ligado com **SPRITE ON**.

Exemplo:
```
100 SCREEN2,1:BASE(2) = BASE(14):SCREEN 0 :SCREEN 2
110 ON SPRITE GOSUB 150
120 FOR X=0 TO 255:FOR S=3TO 6:SPRITE ON
130 PUTSPRITE S,(X,S*20-(S-4.5)*X),15-S,S: NEXT S,X:GOTO 120
150 SPRITE OFF:BEEP:RETURN
```

ON STOP GOSUB linha

Desvia o fluxo do programa, quando for detectado o pressionamento de **[CONTROL]** e **[STOP]**. É ativado pelo **STOP ON** (Veja **STOP**).

Exemplo:
```
10 ON STOP GOSUB 100
20 STOP ON:PRINT"Pressione CTRL+STOP": GOTO 20
100 STOP OFF
110 PRINT "CTRL+STOP PRESSIONADA!!!" :RETURN
```

ON STRIG GOSUB linha1 [,linha2,...]

Desvia o processamento quando detectado o pressionamento da barra de espaço ou os botões do "*Joystick*". "Linha1" indica a barra de espaço; "linha2" botão 1 do *joystick A*; "linha3" botão 1 do *joystick B*; "linha4" e "linha5" para o botão 2 de **A** e **B**, respectivamente.

Deve ser ativado por **STRIG (n) ON** (Consulte **STRIG (n) ON/OFF/STOP**)

Exemplo:
```
10 ON STRIG GOSUB 100,200,300,400,500
20 FOR F=1 TO 5:STRIG (F) ON:NEXT
30 PRINT " PRESSIONE ESPAÇO OU BOTÃO DE TIRO":GOTO 30
100 PRINT "BARRA DE ESPAÇO":RETURN
200 PRINT "BOTÃO 1 – A":RETURN
300 PRINT "BOTÃO 1 – B":RETURN
400 PRINT "BOTÃO 2 – A":RETURN
500 PRINT "BOTÃO 2 – B":RETURN
```

OPEN "[disp] [nome]" [FOR modo] AS [#]num

Comando que abre um arquivo, ou seja, os torna acessível pelos comandos disponíveis no **BASIC MSX**, destinados a manipulação destes.

disp indica o dispositivo no qual o arquivo será aberto (CAS:, CRT:, GRP:, LPT:, etc.).

nome indica o nome do arquivo a ser aberto no dispositivo especificado. Só se faz necessário quando o dispositivo for o gravador cassete ou quando houver um drive conectado ao micro.

"Modo" indica o tipo de acesso e pode ser especificado por: **INPUT** para leitura sequencial; **OUTPUT** para escrita sequencial;.

"Num" indica qual o número a relacionar com o nome do arquivo, pois os comandos do **BASIC** se referenciam ao arquivo pelo número e não pelo nome deste. O número pode variar de 1 até o limite especificado por **MAXFILES**.

Exemplo:
```
10 MAXFILES = 1:OPEN "CAS:TESTE" FOR OUTPUT AS#1
20 PRINT #1,"GRADIENTE E EDITORA ALEPH"
30 CLOSE #1
```

Cria um arquivo em cassete chamado TESTE com o conteúdo GRADIENTE E EDITORA ALEPH".

OR

Operador que faz a operação "*ou lógico*" bit a bit de dois números inteiros. Obtém um bit 1 se pelo menos um dos bits operados for igual a 1.

Exemplo: PRINT BIN$(&B10101010 OR &B11110000)

Deve resultar 11111010.

X \ Y	1	0
1	1	1
0	1	0

OUT porta, expressão

Comando que envia à porta especificada o valor dos 8 bits menos significativos do resultado da expressão.

A comunicação interna entre os circuitos de som, vídeo e outros dispositivos são feitas por várias portas, e se o programador tem um bom conhecimento, da arquitetura do padrão **MSX**, então, este comando será extremamente útil.

Exemplo:
```
COLOR 15,1,1:SCREEN 0
OUT&H99,&H1F:OUT&H99,&H87
```

COLOR define branco como frente e preto como fundo, e em seguida os dois **OUT's** invertem a combinação.

PAD (n)

Retorna o estado do "*touch pad*". Se n estiver entre 0 e 3, refere-se ao terminal A; entre 4 e 7, refere-se ao terminal B.

0 ou 4 – Retorna se o "*touch pad*" foi tocado ou não (0 = não –1 = sim);

1 ou 5 – Retorna coordenada **X** do lugar tocado

2 ou 6 – Retorna coordenada **Y** do lugar tocado

3 ou 7 – Estado do interruptor (0 = pressionado –1 não pressionado)

PAINT [STEP] (X,Y) [,cor da pintura] [,cor do limite]

Preenche com uma cor específica, uma área pré-definida. Em **SCREEN 2** a cor que limita a área, e a cor que preencherá a área devem ser iguais. Isto não é necessário na **SCREEN 3**.

"STEP" faz com que a origem do sistema seja à partir do último ponto marcado (origem relativa).

X e Y são as coordenadas do local, a partir de onde será iniciada a pintura.

Exemplo:
```
10 SCREEN 2:CIRCLE (128,96),80,6: PAINT (128,96),6,6
30 GOTO 30
```

PDL (n)

Retorna o valor determinado pelo *paddle*; "n" pode estar entre 0 e 12 e os valores obtidos estarão entre 0 e 255.

Se "n" for *impar* então os dados são correspondentes ao terminal A, senão, ao terminal B.

Exemplo:
```
10 CLS:C = 0
20 COLOR,C,C
30 IF PDL(1) = 0 THEN 20
40 C = (C + 1)AND15:GOTO 20
```

PEEK (endereço)

Retorna o conteúdo do "endereço" da memória.

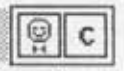
O computador armazena todos os dados em uma memória. O seu **Expert PLUS** sai de fábrica com 64 KBytes destinados ao uso geral.

O endereço pode variar de 0 a 65535 (0000 a FFFF em *hexa*), e cada posição dessas pode armazenar um dado de 8 bits, que sozinhos ou em conjunto, podem representar algo para o micro; para programadores com vastos conhecimentos (veja a bibliografia recomendada), pode ser uma excelente ferramenta (veja também **POKE, VPEEK, VPOKE**)

Exemplo:
```
10 REM EXEMPLO DO PEEK
20 FOR F = &H8000 TO &H807F
30 PRINT F;"...";PEEK(F);"... ";CHR$(PEEK(F))
40 NEXT F
```

Veja o exemplo acima, e repare como o micro armazena os programas **BASIC MSX** na memória. É muito difícil reconhecer o programa ?

PLAY "subcomandos"



Comando que toca sequências de notas e acordes de até 3 notas, através de uma sub-linguagem.

Veja abaixo a relação dos subcomandos para controle de volume tempo duração, pausas, tons, oitavas, etc:

Tn – Determina o andamento da música, e pode variar de 32 a 255 (inicial = T120);

On – n varia de 1 a 8 e determina a oitava em questão. (inicial = O4);

Ln – Determina a duração da nota e n pode variar de 1 a 64. (inicial = L4);

Nn – n pode variar de 0 a 96 e especifica uma nota musical pelo seu número.

A– G – Especifica a nota musical dentro de uma oitava pré-determinada. Se for seguido por um

número entre 1 e 64 a sua duração varia conforme no sub-comando **L**. Se for seguido por "#" ou "+", significa um *sustenido*, se seguido por "–" determina um *bemol*.

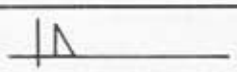
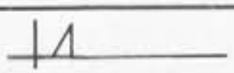
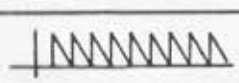
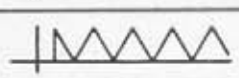
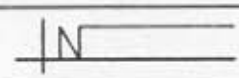
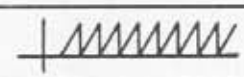
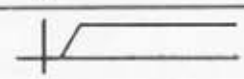
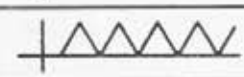
Rn – Com n de 1 a 64 e determina uma pausa.

. – Aumenta a duração da nota em 50%.

Vn – Com n de 0 a 15 determina o volume de saída, e aumenta com o valor de n (inicial = V8);

Mn – Determina o período da variação de volume durante a execução da nota, e n pode variar de 0 a 65535;

Sn – Com n entre 0 e 15 determina o formato do envelope.

VALOR ATRIBUÍDO AO REGISTRO 13 (valores atribuídos a Sn da função PLAY)	FORMA DO ENVELOPE
Ø,1,2,3 ou 9	
4,5,6,7 ou 15	
8	
10	
11	
12	
13	
14	

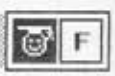
PLAY(n)

Função que retorna o estado das filas musicais. Se "n" for 0, testa os três canais; se for 1, testa o canal A; se for 2, testa o canal B; e se for 3, testa o canal C.

Esta função retorna –1 se o canal estiver ocupado, e 0 se estiver livre.

Exemplo:
```
10 PLAY "CDEFGFEDCDEFGFEDCDEFGFEDCDEFGFEDCCCC"
20 IF PLAY(0) = -1 THEN PRINT"AGUARDE":GOTO 20
```

POINT (X,Y)



Função que retorna o código da cor de um ponto especificado na tela gráfica.

X e Y determinam as coordenadas do ponto a ser verificado.

Exemplo:
```
10 SCREEN 2
20 PSET (128,96),4:C1 = POINT(128,96):C2 = POINT(127,96)
30 SCREEN 0:PRINT C1;C2
```

POKE endereço,expressão

Comando que carrega num endereço de memória, os 8 *bits* menos significativos do resultado da expressão.

Exemplo: POKE &HF3B1,10:KEYON:SCREEN 0

Veja o que acontece com a tela! Para colocá-la na sua condição original redigite a linha-exemplo substituindo 10 por 24.

POS(argumento)

Retorna, para qualquer "argumento", a abcissa X que o cursor ocupa.

Exemplo: PRINT "0123456";POS(0)

PRESET [STEP] (X,Y) [,cor]

Coloca um ponto colorido nas telas gráficas setadas com **SCREEN 2** ou **SCREEN 3**.

Se for especificada uma cor diferente da cor de fundo, então ele funciona igual a **PSET**; se não for especificada uma cor, então o ponto é preenchido com a cor de fundo, dando a impressão de estar sendo apagado.

X e Y definem a coordenada do ponto na tela.

Veja **PSET** para detalhes sobre **STEP**, e as coordenadas das duas telas gráficas.

Exemplo:
```
10 SCREEN 0:LINE (10,10)-(245,182),6,BF
20 FOR F=0 TO 90:X=235*RND(1)+10:Y=172*RND(100)+10
30 PRESET (X,Y):NEXT F
40 GOTO 40
```

PRINT expressão

Apresenta na tela os dados listados na expressão.

O comando **PRINT** é um dos comandos mais básicos, pois permite que o programador envie mensagens e resultados para a tela.

A expressão pode ser uma constante, uma variável numérica, uma expressão numérica, uma variável string e uma expressão string. Se for uma constante string, ela deve estar entre *aspas*.

O comando **PRINT** possue dois separadores: o ponto-e-vírgula (;), que faz com que a mensagem seguinte (que pode estar inclusive em outro PRINT) seja impressa exatamente a partir do fim da mensagem anterior (dentro do mesmo **PRINT** o (;) pode ser omitido); e a vírgula (,) que divide as linhas em dois campos iguais, e as mensagens são impressas a partir do campo seguinte. O primeiro campo tem 16 colunas.

Veja o exemplo, e procure entender. Se você pretende se tornar um programador, este comando é uma das ferramentas mais úteis.

Exemplo:
```
10 PRINT "CONSTANTES STRING E NUMERO";1989
20 A=2000:A$="ISTO É UMA STRING"
30 PRINT A$,A,A$,A;A$
40 PRINT "PODE-SE OMITIR O (;) "A$A"DENTRO DE UM COMANDO PRINT"
```

PRINT USING formato;expressão

Apresenta os dados com um formato específico na tela. O formato deve estar entre *aspas*, e são os seguintes:

"!" – Apresenta apenas o primeiro caracter de uma *string*;

"\n-espaços\" – Apresenta n + 2 caracteres de uma *string*;

"&" – Apresenta todos os caracteres de uma *string*;

"#" – Especifica o formato e o número de dígitos apresentados de um dado numérico;

"+" – Acrescenta o sinal + ou –antes ou após números conforme forem positivos, nulos ou negativos;

"–" – Acrescenta o sinal – após números negativos;

"**" – Preenche com asteriscos os espaços ocupados por um dado numérico;

"$$" – Acrescenta o símbolo $ antes de dados numéricos;

"**$" – Acrescenta o símbolo $ antes de dados numéricos e preenche com asteriscos os espaços não ocupados;

"," – Acrescenta vírgula a cada três dígitos à esquerda do ponto decimal;

"^ ^ ^ ^" – Apresenta dados numéricos em ponto flutuante.

Exemplo: PRINT USING "###.##",23.4392

PRINT #num,expressão

Escreve os dados da expressão no dispositivo (fita, disco, impressora, etc.) aberto por **OPEN**.

O número é o especificado no **OPEN**.

A expressão tem os mesmos parâmetros do **PRINT** convencional. Aliás em computação as telas de texto são vistas como um dispositivo de saída "*default*" (por isso não é necessário abrir o arquivo de impressão de tela). Nas telas gráficas, porém, deve-se abrir o arquivo gráfico (GRP:) para que se possa mandar dados para elas.

Exemplo:
```
10 OPEN "GRP:" AS #1
20 SCREEN 2:CIRCLE (128,96),60
30 PRESET(105,90):PRINT#1,"Expert"
40 PRESET(104,90):PRINT#1,"Expert"
50 GOTO 50
```

PRINT #num,USING"formato";expressão

Este comando é uma ampliação do **PRINT USING**, com a vantagem de poder acessar arquivos externos, abertos pelo **OPEN**.

Exemplo:
```
10 SCREEN 2:OPEN"GRP:"AS #1
20 LINE (20,20)–(235,172),6,BF
30 PRESET (110,90):PRINT#1,USING"##.##";12.5464
40 GOTO 40
```

PSET [STEP] (X,Y) [,cor]

Marca um ponto na *tela gráfica*, com a cor especificada. Se a cor não for especificada então será marcado com a cor de frente definido pelo **COLOR**.

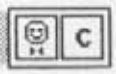
X e Y definem as coordenadas do ponto na tela gráfica. Nas telas gráficas **SCREEN 2** e **SCREEN 3** X varia de 0 a 255 e Y de 0 a 191. Na **SCREEN 3** os pontos têm o quádruplo da altura e largura da **SCREEN 2.**

STEP transfere a origem do sistema para a última coordenada impressa, de maneira que podem existir números negativos e positivos. Observe que apos o término do comando o sistema de coordenadas é devolvido à posição original.

O número da cor deve variar entre 0 e 15, e a tabela de correspondencia se encontra listada no comando **COLOR**.

Exemplo:
```
10 FOR A=2 TO 20 :CLOSE
20 SCREEN A MOD 2+2 : OPEN "GRP:" AS #1
30 PRESET (0,0): PRINT #1,"SCREEN "; A MOD 2+2
40 FOR X=0 TO 255 STEP 8 : FOR Y=33 TO 191 STEP 8
50 PSET (X,Y): NEXT Y,X,A
```

PUT SPRITE camada [[STEP] (X,Y)], [cor], [num]



Comando que permite colocar na tela as figuras móveis definidas pelo usuário.

camada – O MSX possue 32 camadas a disposição de maneira que cada camada pode ter apenas um *sprite*. Este número deve estar entre 0 e 31.

STEP – Esta opção transfere a origem do sistema de coordenadas, para o último ponto plotado. A origem está no canto superior esquerdo da tela.

X – Define o valor da absissa, e deve estar entre 0 e 255.

Y – Define o valor da ordenada, e deve estar entre –8 a 192.

cor – Define a cor do *sprite*, e deve estar entre 0 e 15. Veja o comando **COLOR** para saber a relação número/cor.

num – Define qual o desenho do *sprite*. Esse desenho é definido com o comando **SPRITE$**, e deve estar entre 0 e 255 se for de tamanho 8x8, ou entre 0 63 se for do tamanho 16x16 (veja comando **SCREEN**).

Exemplo:
```
10 SCREEN 1,1:FOR T=1 TO 8:READ A$
20 S$=S$+CHR$(VAL("&H"+A$)):NEXT T
30 SPRITE$(1)=S$:PUTSPRITE 0,(128,N),8,1
40 PRINT " EXPERT PLUS ";:N=N+1: GOTO 30
50 DATA C7,E3,73,3B,DC,CE,C7,C3
```

Digitando o programa acima, você terá uma idéia da vantagem do uso de sprites.

READ var1 [,var2...]

Lê dados sequencialmente armazenados em linhas **DATA's**.

Os dados lidos são colocados na(s) variável(s) listadas no comando **READ**.

Veja o exemplo no comando **PUT SPRITE** ou o comando **DATA**.

Este comando, em conjunto com **RESTORE** e **DATA**, monta um conjunto de comandos, ideais para armazenar dados constantes (No exemplo do verbete **PUT SPRITE** o formato do *sprite* é armazenado em linhas **DATA**).

REM comentário

Este comando é útil para *documentação*, e tem como intenção básica facilitar a descrição interna dos programas. O comando não executa absolutamente nada, e pode ser seguido de um comentário. Este comentário pode conter qualquer tipo de informação.

O comando **REM** pode ser substituído pelo apóstrofo da tecla 8 (').

Exemplo:10 REM Este comando não executa nada

20 ' e pode ser substituído por "'"

RENUM [lin1] [,lin2] [,inc]

Renumera as linhas de um programa. "Lin1" é o número inicial da linha, "lin2" indica a partir de que linha deve ser renumerado e "inc" corresponde ao incremento entre linhas.

O **RENUM** renumera inclusive **GOTO, GOSUB, RESUME** e outros comandos que façm referêcia à linhas de programas.

Para ver como funciona, coloque um programa na memória e execute:

RENUM 1,,1

RESTORE [linha]

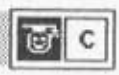
Este comando é usado em conjunto com **READ**, e atualiza a partir de que **DATA** devem ser considerados os dados lidos pelo **READ**. Se "linha" for omitido, atualiza a partir da primeira linha **DATA**.

Exemplo:10 RESTORE 30:READ A$:PRINT A$

20 DATA linha de dados 1

30 DATA linha de dados 2

RESUME [linha/NEXT]



Indica a linha de retorno de uma sub-rotina de erro.

Quando ocorre algum tipo de erro no **BASIC**, este pode ser detectado (veja **ON ERROR GOTO**) e, após o seu tratamento, pode-se retornar, para uma linha do programa. Se "linha" não for especificada, então retorna para a linha onde ocorreu o erro. Se for usada a opção NEXT, então retorna para a linha seguinte à ocorrência do erro.

Veja o exemplo de **ON ERROR GOTO**.

RETURN linha

Veja o comando **GOSUB**.

RIGHT$(string,n)

Função que retorna os n-caracteres pela direita de uma variável string.

Exemplo: PRINT RIGHT$("TRABALHO",4)

RND(argumento)

Retorna um número *aleatório* maior ou igual a 0, mas menor que 1.

Se o argumento é positivo fornece sempre um valor diferente mas a mesma sequência a cada vez que for iniciado. Se for 0, então retorna o último valor gerado. Se for negativo, retorna o primeiro valor da sequência indicada pelo número.

O modo de uso ideal, é setar uma determinada sequência, através de um número negativo, e os seguintes através de argumentos positivos.

Para números realmente *aleatórios* veja o exemplo:

```
Exemplo:10 X=RND(-TIME):FOR F=1 TO 10
        20 PRINT RND(9999):NEXT F
```

A variável **TIME** varia constantemente, e colocando, então como argumento negativo do **RND**, gera um sequência imprevisível, formando assim sempre números diferentes.

RUN [linha | "nomearq"]

Permite executar um programa que está residente na memória.

Se for especificada linha, então inicia a execução a partir da linha indicada.

Se for especificado o nome do arquivo (nomearq), carrega o arquivo, e o executa (como no **LOAD** com opção **",R"**).

Para testá-lo, digite qualquer programa, linha a linha, e comande:

RUN (e **[RETURN]**)

SAVE (veja sintaxe abaixo)

Comando que grava um programa da memória em um dispositivo com um determinado nome.

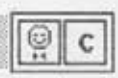
Este comando tem sintaxe diferente, para as versões disco ou cassete.

A sintaxe da versão cassete é a seguinte:

SAVE "cas:[nome]"

O nome do arquivo pode ter no máximo 6 letras, sendo as minúsculas diferentes das maiúsculas. O comando SAVE, quando executado no cassete sempre grava o programa no formato ASCII. Note que da mesma forma que você pode abrir arquivos em outros dispositivos (CRT:, GRP:, PRN:) pode também "salvar" o programa neles.

SCREEN [modo],[sprite],[clic],[bauds],[impressora]



Este comando configura vários recursos do **MSX**:

modo – Seta as várias telas. 0 para textos 40x24; 1 para textos 32x24; 2 para alta-resolução 256x192; e 3 para multicolorido baixa-resolução 64x48.

sprite – Define os tipos de sprite's. 0 para tamanho 8x8 normal; 1 para 8x8 ampliado; 2 para 16x16 normal; 3 para 16x16 ampliado.

clic – Ativa ou não o clic no alto-falante, para as teclas pressionadas. 0 – desliga 1 – liga.

bauds – Seleciona velocidade de transferência de dados para o cassete. 1 para 1200 *bauds*; 2 para 2400 *bauds*.

impressora – Seleciona o tipo de impressora. 0 para o **ABNT**; 1 para impresora **MSX**.

Exemplo:
```
10 A$ = "01234567890123456789012345678901234567890"
20 SCREEN 0:PRINT A$:GOSUB 100
30 SCREEN 1:PRINT A$:GOSUB 100
40 SCREEN 2:LINE(9,9)-(99,99),3,B:GOSUB 100
50 SCREEN 3:LINE(9,9)-(99,99),3,B:GOSUB 100
60 END
100 FOR F = 1 TO 2000:NEXT:RETURN
```

SGN (argumento)

Função que retorna o sinal do argumento.

Se o argumento for qualquer número positivo, o resultado é +1. Se for zero, o resultado é 0 e, se for negativo, o resultado é –1.

Exemplo: PRINT SGN(–4);SGN(0);SGN(200)

SIN (argumento)

Função que retorna o valor do seno do arco, em radianos, fornecido pelo argumento.

Exemplo: PI = 4*ATN(1):PRINT SIN(PI/2)

SOUND registro,expressão

Acessa cada um dos registros do PSG (processador de som programável) individualmente.

O PSG possue 16 registros, sendo que os registros de 0 a 13 são usados para emissão de sons. Para maiores detalhes, veja a bibliografia aconselhada (Apendice E).

Exemplo: SOUND 6,10:SOUND 8,16:SOUND 12,3:SOUND 13,14:SOUND 7,&B11110111

Você obterá o som de uma locomotiva.

SPACE$ (argumento)

Função que retorna uma *string* com espaços em branco, dependendo do argumento definido.

Exemplo: FOR F = 1 TO 19:A$ = SPACE$(F):PRINT A$;"ALEPH":NEXT F

SPC (argumento)

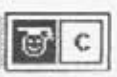
Função que abre espaços num comando de saída, dependendo do argumento. Este resultado não pode ser inserido em uma variável *string*.

Exemplo: FOR F = 1 TO 19:PRINT SPC(F);F:NEXT F

SPRITE

Veja **SPRITE$, ON SPRITE GOSUB, SPRITE ON/OFF/STOP** e **PUT SPRITE.**

SPRITE ON/OFF/STOP



Comando que habilita, desabilita ou adia a interrupção por sobreposição de *sprites*.

Veja o exemplo de **ON SPRITE GOSUB.**

SPRITE$(numero),string

Define um formato de *sprite*. A string deve ter os caracteres no formato **ASCII**, que atribuídos ao *sprite* definem seu formato. O número do sprite pode variar de 0 a 255 para tamanho 8x8 ou de 0 a 63 para o tamanho 16x16 (Veja **SCREEN**).

Veja o exemplo de **PUT SPRITE**.

SQR(argumento)

Função que retorna o valor da raiz quadrada do argumento.

Exemplo: PRINT SQR(9)

STEP

Veja **PSET, PRESET, CIRCLE, FOR/NEXT, PUTSPRITE** e **LINE.**

STICK (N)

Retorna o estado do *joystick* ou das teclas cursoras.

Se N for 0, então trata-se das teclas do cursor. Se for 1 então trata-se do *Joystick A*, se for 2, então trata-se do *joystick B*.

Pode-se obter os seguintes resultados:

Exemplo:
```
10 SCREEN 3:X=128:Y=96
20 PSET(X,Y),6:C=STICK(0)
30 X=X+(C>5)-((C>1)AND(C<5))
40 Y=Y+((C=8)OR(C=1)OR(C=2))-((C>3)AND(C<7))
50 GOTO 20
```

Se você quer ser um programador, observe os *parêntesis lógicos* nas linhas 30 e 40.

STOP

Interrompe a execução de um programa. A execução pode ser prosseguida com **CONT.**

Exemplo:
```
10 PRINT "ALO":STOP
20 PRINT "ESTA LINHA NÃO É EXECUTADA"
```

Veja também **INTERVAL, KEY(n), ON STOP GOSUB, SPRITE, STOP** e **STRIG(n).**

STOP ON/OFF/STOP

Habilita, desabilita ou adia a interrupção mediante o pressionamento de **[CONTROL] + [STOP].**

Exemplo:
```
10 ON STOP GOSUB 100:STOP ON
20 PRINT "TECLE CONTROL+STOP":GOTO 20
100 STOP OFF:PRINT "PARA TERMINAR PRESSIONE CONTROL+STOP"
110 GOTO 110
```

STRIG (N)

Função que retorna o estado dos disparadores. Se for detectado o pressionamento do disparador, então retorna o valor –1, senão 0.

N pode variar de 0 a 4, e indica qual o disparador, que deseja-se ler:

0 – barra de espaços. 1 – disparador 1 do *joystick A*.

2 – disparador 1 do *joystick B* 3 – disparador 2 do *joystick A*.

4 – disparador 2 do *joystick B*.

Exemplo:
```
10 CLS:PRINT"PRESSIONE ESPAÇO"
20 IF STRIG(0) =0 THEN 20 ELSE END
```

STRIG (N) ON/OFF/STOP

Comando que habilita, desabilita ou adia a interrupção pelos disparadores. N indica qual o disparador em questão (veja **STRIG**).

Exemplo:
```
10 ON STRIG GOSUB 100:STRIG(0) ON
20 PRINT "PRESSIONE ESPAÇO":GOTO 20
100 STRIG(0) OFF:PRINT "****PRESSIONADO****"
110 STRIG(0) ON:RETURN
```

STR$(expressão)

Converte dados numéricos em string.

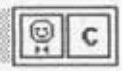
Exemplo: `A = 12:B = 13:C = A + B:C$ = STR$(A) + STR$(B):PRINT C$;C`

STRING$(n,caractere)

Esta função retorna uma string com n-caracteres iguais.

Exemplo: `PRINT STRING$(20,65),STRING$(30,"X")`

SWAP var1,var2



Comando que permuta o conteúdo de duas variáveis.

Exemplo: `A = 1:B = 2:SWAP A,B:PRINT A;B`

TAB (n)

Move o cursor n espaços para a direita. "n" não deve exceder 255.

Exemplo: `10 FOR A = 1 TO 20:PRINT TAB(A);A:NEXT A`

TAN (argumento)

Função que retorna o valor da tangente do arco, em radianos, definido pelo argumento.

Exemplo: `PI = 4*ATN(1):PRINT TAN(PI/4)`

THEN

Veja **IF... THEN ...ELSE.**

TIME

Variável reservada, que retorna o valor do temporizador interno.

Este temporizador é incrementado a cada 1/60 segundos, e ao atingir 65535, volta a 0, e assim repetidamente.

Como toda variável seu conteúdo pode ser alterado (sem o uso do **LET**).

Exemplo:
```
10 CLS: PRINT"PRESSIONE RETURN JÁ!!": TIME =0
20A$=INKEY$: IF A$< >CHR$(13)THEN 20
30 PRINT"VOÇÊ DEMOROU";TIME/60;"SEGUNDOS!"
```

TO

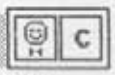
Veja **FOR.**

TROFF

Comando que desliga o **"TRACE".**

Quando o **"TRACE"** está ligado, toda número de linha que estiver sendo executada, é impresso na tela (Ideal para procura de erros).

TRON



Permite visualizar o número de linha que esta sendo executado. O inverso de **TROFF.**

USING

Veja **PRINT USING, PRINT# USING.**

USR[n] (argumento)

Executa uma das 10 subrotinas em *linguagem de máquina.*

O número "n" pode estar entre 0 e 9, e indica qual das 10 entradas, esta sendo usada.

O argumento é passado para a sub-rotina. Consulte um livro de linguagem de máquina, se desejar informações mais precisas.

A sub-rotina, em **ASSEMBLY,** deve terminar com "RET" (&HC9), para retornar ao **BASIC.**

O comando **DEFUSR[n]** define o endereço de entrada da sub-rotina.

Exemplo: DEFUSR=&H6C:POKE0,USR(0)

Acessa a sub-rotina que executa um SCREEN 0.

VAL(string)

Função que converte uma string em um número. Tem função inversa de **STR$.**

Note que, quaisquer caracteres diferentes de algarismos, serão ignorados!

Exemplo: A$="432+252":PRINT A$;VAL(A$)

VARPTR

Função que retorna o endereço do local onde esta armazenado uma variável.

Exemplo: A=0:PRINT "A = &H";HEX$(VARPTR(A))

VDP(reg) / VDP(reg) = expressão

Variável interna que contém o conteúdo do **VDP** (Processador de vídeo).

Elas podem ser modificadas e lidas como variáveis.

Exemplo: SCREEN0:A = VDP(7):VDP(7) = A XOR 255

Muda a cor da tela.

VPEEK (endereço)

Lê o byte correspondente ao endereço da memória de vídeo (VRAM).

O Expert tem, de fábrica, 80 KBytes de memória, sendo que destes, 16 KBytes se destinam exclusivamente à geração de imagens.

A função **VPEEK** possibilita que acessemos cada um desses 16384 bytes.

Exemplo: PRINT VPEEK(0)

VPOKE endereço,expressão

Coloca em um endereço da memória de vídeo (VRAM), o byte fornecido pela expressão.

Exemplo: 10 CLS:FOR F = 0 TO 255:VPOKE F,F:NEXT F

O programa coloca a tabela de caracteres do MSX na **SCREEN 0** (sem o uso do **PRINT**).

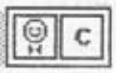
WAIT porta,expressão1 [,expressão2]

Aguarda até a ocorrência do valor especificado.

Um **XOR** ("ou" exclusivo) é executado entre os dados da porta de E/S especificada, e o valor da expressão2. Um **AND** ("e") é executado entre os resultados e o valor da expressão1. Se o resultado final for 0, então os dados serão lidos.

Este comando pode ser classificado como avançadíssimo, e de pouco uso em programas.

WIDTH n



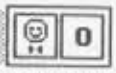
Determina o número de colunas que uma tela de texto deve ter.

Para **SCREEN 0** deve ser entre 1 e 40; em **SCREEN 1** entre 1 e 32.

Exemplo: WIDTH 15

O que faz com que a tela seja bastante reduzida.

XOR

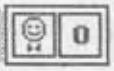


Operador que executa um "*ou exclusivo*" bit a bit entre dois números, ou seja, se ambos os bits forem iguais, resulta um bit 0; se forem diferentes, resulta um bit 1.

Exemplo: PRINT BIN$(&B01010000 XOR &B 10111100)

X \ Y	1	0
1	0	1
0	1	0

+



Operador de adição, permite efetuar a soma matemática de duas variáveis numéricas ou a concatenação de 2 strings.

Exemplos: PRINT 2+3

Resulta: 5

PRINT "Expert"+" PLUS"

Resulta: Expert PLUS

–

Operador de subtração, quando usado entre dois números. Pode ser usado para inverter o sinal de um número.

Exemplo: A=4:PRINT –A

Pode ser usado como separador em certos comandos (Veja **LINE, DELETE, LIST, LLIST**).

*

Operador de multiplicação.

Exemplo: A=3:B=2:PRINT A*B

É usado também como "coringa" para nomes de arquivos (Veja **FILES, DELETE** e **COPY**) ou para indicar, na numeração automática, que uma linha de programa já existe (Veja **AUTO**).

/

Operador de divisão.

Exemplo: A=10:B=3:PRINT A/B

Resulta: 3.3333333333333

Veja também "\"

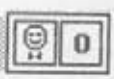
\

Operador de "divisão inteira": fornece o quociente inteiro da divisão de dois números.

Exemplo: A=10:B=3:PRINT A\B

Resulta: 3. Compare com o exemplo do "/".

^

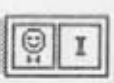


Operador de exponenciação.

Exemplo: PRINT 3 ^ 2

Resulta: 9

%

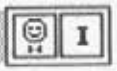


Posto ao final de um número ou variável numérica, indica ser de precisão inteira (ocupado 2 bytes de memória).

!

Posto ao final de um número ou variável numérica, indica ser de precisão simples (ocupando 4 bytes de memória).

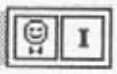
#



Posto ao final de um número ou variável numérica, indica ser de precisão dupla (ocupando 8 bytes de memória).

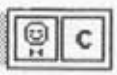
É usado também para numerar arquivos (veja **OPEN, CLOSE**). Em países anglo-saxônicos o "#" substitui o nosso "Nº".

$



No final do nome de uma variável, mostra ser ela alfa-numérica ("string").

'



Veja **REM**.

_

Veja **CALL**.

?

Veja **PRINT**.

&B

Indica que os algarismos que seguem (são permitidos apenas 0 e 1) representam um número em notação binária.

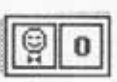
&H

Indica que os algarismos que seguem (são permitidos apenas 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E e F) representam um número em notação hexadecimal.

&O

Indica que os algarismos que seguem (são permitidos apenas 0,1,2,3,4,5,6,7) representam um número em notação octal.

>, <, >=, <=, = e <>



Correspondem aos operadores relacionais da matemática: Maior que, menor que, maior ou igual, menor ou igual, igual e diferente, respectivamente.

Exemplo: PRINT (4 < 8);(4 > 8)

MENSAGENS DE ERRO DO MSX-BASIC

BAD FILE MODE	O arquivo foi aberto de maneira incorreta.
BAD FILE NUMBER	Ou o arquivo não foi aberto ou está fora do MAXFILES.
CAN'T CONTINUE	Impossível executar o CONT.
DEVICE I/O ERROR	Ocorreu um erro de Entrada/Saída em algum dispositivo.
DIRECT STATEMENT IN FILE	O arquivo carregado não é um programa BASIC.
DIVISION BY ZERO	Tentou-se executar uma divisão por zero.
FILE ALREADY EXISTS	Tentou-se renomear um arquivo com um nome já existente.
FILE ALREADY OPEN	Tentou-se abrir um arquivo sequencial ainda aberto.
FILE NOT OPEN	Tentou-se acessar um arquivo que não foi aberto.
ILLEGAL DIRECT	O comando usado só funciona no modo programado.
ILLEGAL FUNCTION CALL	Erro no uso de um parâmetro ou argumento.
INPUT PAST END	Tentou-se ler algo de um arquivo que já terminou.
INTERNAL ERROR	Problemas com o funcionamento do micro.
LINE BUFFER OVERFLOW	Uma linha de BASIC possui caracteres demais.
MISSING OPERAND	Argumento errado ou faltante.
NEXT WITHOUT FOR	Há um NEXT sem que antes tenha se especificado o FOR.
NO RESUME	Não foi colocado um RESUME numa rotina de erro.
OUT OF DATA	Tentou-se ler um dado numa sequencia DATA já esgotada.
OUT OF MEMORY	Toda memória disponível ou reservada, esgotou-se.
OUT OF STRING SPACE	Excedeu-se o espaço reservado para strings.
OVERFLOW	A magnitude do número ultrapassa a capacidade do micro.
REDIMENSIONED ARRAY	Dois DIM foram usados para dimensionar a mesma matriz.
RESUME WITHOUT ERROR	Foi encontrado um RESUME sem ocorrer erro.
RETURN WITHOUT GOSUB	Foi encontrado um RETURN fora de uma sub-rotina.
STRING FORMULA TOO COMPLEX	Uma string é muito longa ou complexa.
STRING TOO LONG	Há uma string com mais de 255 caracteres.
SUBSCRIT OUT OF RANGE	Há um índice maior que o especificado no DIM.
SYNTAX ERROR	A sintaxe do comando ou função está errada.
TYPE MISMATCH	Tentou-se atribuir um valor numérico a uma string.
UNDEFINED LINE NUMBER	Um GOTO ou GOSUB envia a execução a uma linha inexistente no programa.
UNDEFINED USER FUNCTION	Tentou-se usar um FN antes do correspondente DEFFN.
UNPRINTABLE ERROR	Erro que não dispõe de mensagem.
VERIFY ERROR	Detectou-se um erro numa rotina de verificação.

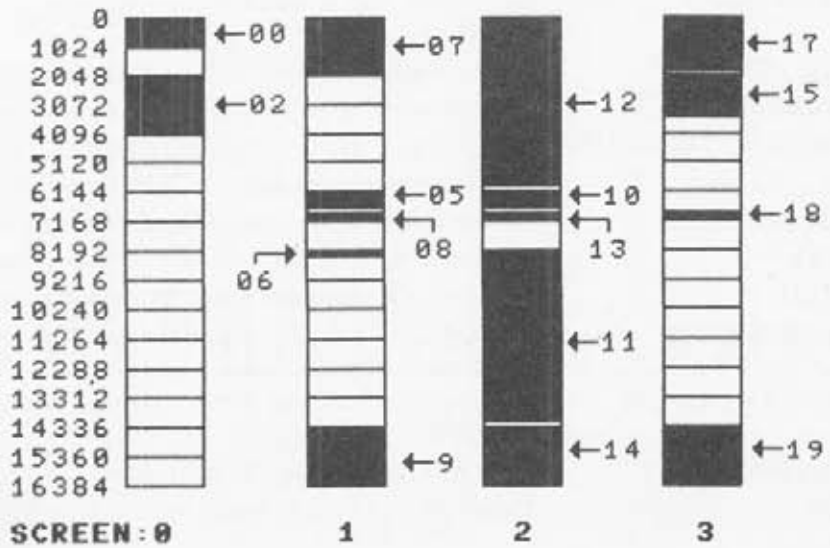
APÊNDICES

A-MAPA DA VRAM

A figura a seguir apresenta um esquema do mapeamento padrão da VRAM para as várias SCREENS e a função de cada uma dessas áreas.

Esses valores podem ser alterados usando-se o comando "BASE(N) =".

SCREEN	BASE	ENDEREÇO	CONTEÚDO DA TABELA
0	00	00000	Posição dos caracteres
	02	02048	Formato dos caracteres
1	05	06144	Posição dos caracteres
	06	08192	Cores dos caracteres
	07	00000	Formato dos caracteres
	08	06912	Atributos dos SPRITES
	09	14336	Formato dos SPRITES
2	10	06144	Posição dos "caracteres"
	11	08192	Cor de frente e de fundo
	12	00000	Formato dos "caracteres"
	13	06912	Atributo dos SPRITES
	14	14336	Formato dos SPRITES
3	15	02048	Posição dos "caracteres"
	17	00000	Padrão de cores
	18	06912	Atributos dos SPRITES
	19	14336	Formato dos SPRITES



B-TABELAS DE CARACTERES DO MSX

O Expert Plus envia dados à impressora através de duas tabelas: MSX e ABNT.

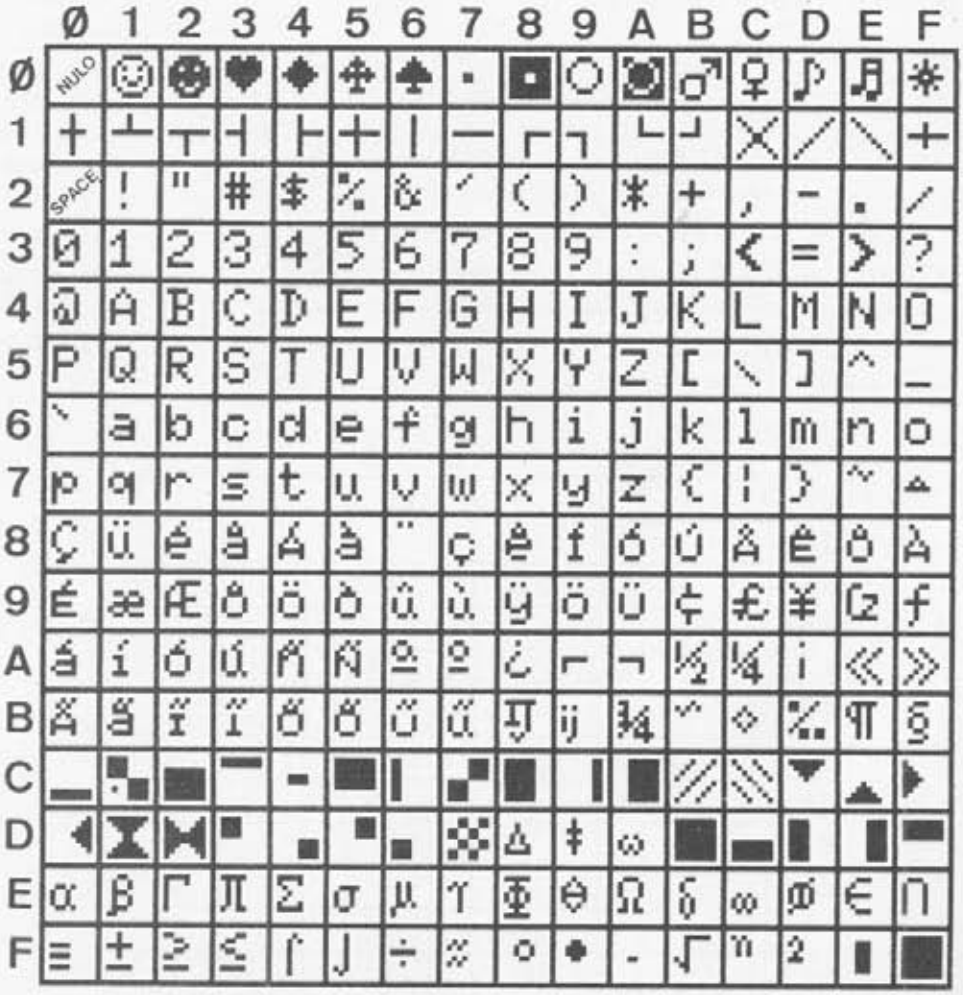
Ao ser ligado, o filtro ABNT é automaticamente ativado. Para configurar seu Expert para qualquer uma das duas tabelas, basta comandar:

```
POKE&HF417,X        ou        SCREEN,,,,X
```

com X = 0 para ABNT ou X = 255 para MSX.

A TABELA MSX

A tabela a seguir mostra os 256 caracteres da tabela residente do padrão MSX-Brasil. Para saber o código de qualquer um deles em Hexadecimal, basta usar como primeiro dígito a linha e como segundo dígito a coluna. Para imprimir o caractere "L", por exemplo, basta comandar PRINT CHR$(&H4C).



Os 32 primeiros caracteres

Para imprimir os caracteres das duas primeiras linhas (de &H00 a &H1F) devemos enviar a sequência :

```
PRINT CHR$(1)+CHR$(64+código)
```

Os 32 primeiros caracteres são reservados para contrôle do vídeo ou da impressora. Se você estiver com a tabela MSX ativa no Expert e tiver uma impressora configurada para MSX, ao enviar o comando:

```
LPRINT CHR$(&H02)
```

isso será interpretado como caracter de controle e terá um certo efeito sobre a impressora (consulte o manual da mesma para saber qual).

Em contrapartida, se você comandar:

```
LPRINT "☻" ☻ =CHR$(&H02)
```

a impressora receberá a sequência de códigos:

```
CHR$(1)+CHR$(64+&H02)
```

Algumas impressoras entenderão isso e imprimirão a "carinha". Outras não entenderão e imprimirão um "B" (chr$(64+2))!

O FILTRO ABNT

Quando seu Expert tiver o filtro ABNT ativo, ele poderá escrever letras acentuadas e "ç" em impressoras nacionais mais recentes, mesmo que elas não possuam uma configuração para MSX. Na tabela ABNT a seguir, os 32 primeiros caracteres servem como códigos de contrôle para a impressora, com duas observações:

(1) Este caractere não é enviado com o ABNT ativo.

(2) Este caractere produz o envio de uma sequência de espaços em branco suficiente para produzir uma tabulação de 8 em 8 na impressora.

Qualquer outro caractere que você tente enviar e não constar da tabela, será transformado em espaço em branco (32 = &H20).

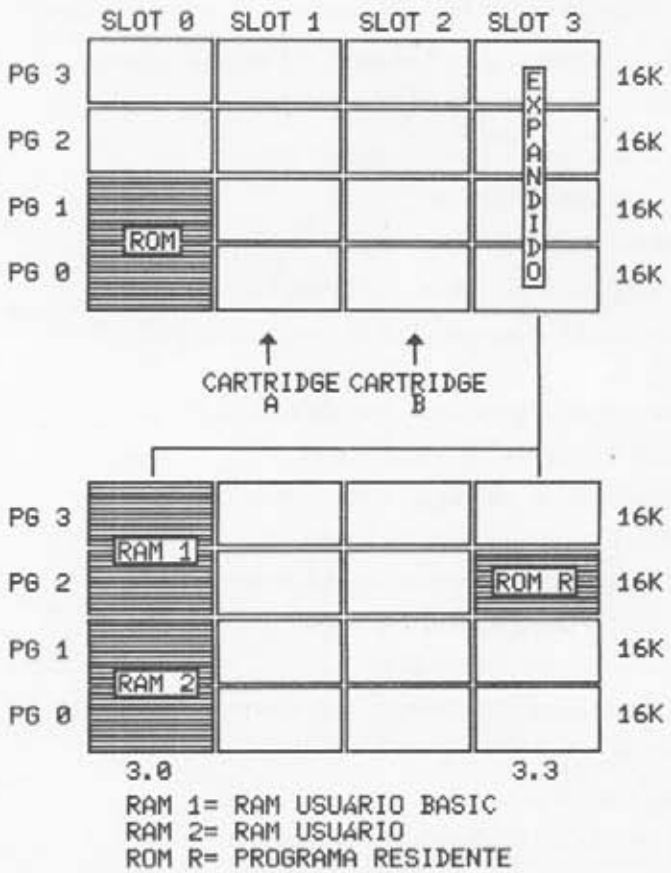
	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø	NULO	(1)								(2)						
1																
2	SPACE	!	"	#	$	%	&	´	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	▲
8																
9																
A		¡						§			º	«				
B	°	±	²			µ	¶				º	»	¼	½	¾	¿
C	À	Á	Â	Ã				Ç		É	Ê			f		
D		Ñ		Ó	Ô	Õ					Ú		Ü			ß
E	à	á	â	ã				ç		é	ê			í		
F		ñ		ó	ô	õ					ú		ü			

C – MAPA DA MEMÓRIA

O Expert PLUS possui 48 Kbytes de memória ROM (pré-gravada de maneira permanente) e 80 Kbytes de RAM (memória volátil para leitura e escrita). O microprocessador de vídeo (VDP) acessa 16 K e os outros 64 K são usados pelo microprocessador principal (Z80).

O Expert PLUS tem 4 slots, sendo dois para uso interno (0 e 3) e dois externos (slot 1= cartridge A ; slot 2= cartridge B). A prioridade dos slots vai do menor para o maior. Cada slot suporta 64 K e é dividido em 4 páginas de 16 K cada. Os slots externos podem ser expandidos em 4 sub-slots, suportando, portanto, 256 K cada um. No Expert PLUS o slot 3 (interno) já está expandido em sub-slots 3.0 e 3.3.

Veja na figura abaixo o esquema destas sub-divisões.



Para o usuário que programa em BASIC, ficam disponíveis as páginas 2 e 3 do sub-slot 3.0. As duas páginas inferiores podem ser acessadas ,em linguagem de máquina, mudando-se a configuração da PPI.

D – ESPECIFICAÇÕES TÉCNICAS

Microprocessador
- Z80A (3,58 MHz) - Compatível (*).

Memória residente
- ROM: 32 K (BASIC MSX) + 16 K (DISK-BASIC)
- RAM: 64 K (usuário) + 16 K (vídeo)

Linguagem residente
- BASIC MSX, Linguagem de Máquina Z80

Vídeo
- VDP : TMS-9128NL - Compatível (*).
- Texto 1: 32 colunas x 24 linhas
- Texto 2: 40 colunas x 24 linhas
- Gráficos: 256 x 192 pontos
- Sprites: 32 níveis
- Cor: 16 cores

Som
- PSG: AY-3-8910-A - Compatível (*).
- Recursos: 3 vozes, canal de ruído, 08 oitavas

Teclado
- Teclas: 89 com acentuação em Português (10 teclas programáveis)
- Níveis: 6 níveis para produção de 256 caracteres

Interfaces residentes
- Som: Alto falante interno com controle de volume
- Cassete: leitura, gravação, motor
- Printer: paralela padrão Centronics

Conexões externas
- Slots: 02 frontais para conexão de cartuchos, interfaces, etc.
- Vídeo: Conector RCA para monitor monocromático (1,2 VPP/75 ohms)
- Color: Conector RCA para vídeo colorido PAL/M
- RF: Conector RCA para TV (canal 3 ou 4)
- Áudio: Conector RCA para ligação em amplificador.
- D.CORDER: Conector circular de 5 pinos
- Joystick: 2 conectores 9 pinos para joystick, mouse ou paddle
- Printer: Conector finger duplo de 26 pinos
- Reset: Botão para reset por hardware
- Tomadas: Duas tomadas controladas pelo interruptor POWER (carga máx.100 VA) e alimentação 6 VDC (máx.240 mVA)

Alimentação
- Tensão: AC 120/240 V, 60 Hz, comutável
- Consumo: 30 VA (só CPU), máx.42 VA(com periféricos)

Acessórios
- Cabos: 1 cabo RF 75 ohms RCA-RCA
- Fusíveis: 1 para 120V e 1 para 240V
- Comutador: 1 comutador TV/Computer
- Etiquetas: 1 conjunto de etiquetas para as teclas de função
- Livro: 1 Manual de instruções

(*) Integrados no ENGINE T7937A

E – BIBLIOGRAFIA ACONSELHADA

LINGUAGEM BASIC

PARA PRINCIPIANTES

MSX GUIA DO USUÁRIO -HOFFMAN -McGRAW-HILL
MSX BASIC...SEM DOR -MARTELLO - CAMPUS
CURSO DE BASIC - PIAZZI E CARVALHO - ALEPH
MSX O MEU PRIMEIRO LIVRO - MARRIOT - McGRAW-HILL
MSX: PRÁTICA E DOMÍNIO- CASARI - ATLAS
JOGOS MSX V.1 - BURD E MOREIRA - McGRAW-HILL
JOGOS MSX V.2 - BURD E MOREIRA - McGRAW-HILL
JOGOS MSX V.3- BURD E MOREIRA - McGRAW-HILL
COLEÇÃO DE PROGRAMAS V-1 - VÁRIOS - ALEPH
COLEÇÃO DE PROGRAMAS V-2 - VÁRIOS - ALEPH
MSX:COMO PROGRAMAR - HARTNELL - CAMPUS
MSX BASIC-GUIA DE REFERÊNCIA - PIMENTEL - CAMPUS
RESUMO DE OPERAÇÕES DO EXPERT - VÁRIOS - ALEPH
MSX GUIA DO OPERADOR - BURD E MOREIRA - McGRAW-HILL
DOMINANDO O EXPERT - SANTORO - ALEPH
BASIC PARA CRIANÇAS - MARTELLO - ALEPH
LINGUAGEM BASIC MSX - SANTORO - ALEPH

PARA INICIADOS

APROFUNDANDO-SE NO MSX- VÁRIOS - ALEPH
PROGRAMAÇÃO PROFISSIONAL EM BASIC - WATANABE - ALEPH
PROGRAMAS PROFISSIONAIS PARA USO PRÁTICO - CASARI - ATLAS
MSX V.1 GUIA DO PROGRAMADOR - VÁRIOS - McGRAW-HILL
MSX V.2 GUIA TÉCNICO DE REFERÊNCIA - VÁRIOS - McGRAW-HILL
100 DICAS PARA MSX - VÁRIOS - ALEPH
+ 50 DICAS PARA MSX - VÁRIOS - ALEPH

APLICAÇÕES ESPECIAIS

SIMULAÇÕES MSX - BURD - McGRAW-HILL
MSX MÚSICA - BUSSAB - McGRAW-HILL
CURSO DE MÚSICA MSX - BARBIERI E PIAZZI - ALEPH
CIRCUITOS ELETRÔNICOS - FRIEDMANN - ALEPH
DESENHOS BÁSICOS - CARVALHO LIMA - ALEPH
ASTROLOGIA NO MSX- CARVALHO - ALEPH

LINGUAGEM DE MÁQUINA

PARA PRINCIPIANTES

JOGOS DE HABILIDADE - VÁRIOS - ALEPH
LINGUAGEM DE MÁQUINA MSX - FIGUEREDO E ROSSINI - ALEPH
ASSEMBLER PARA O MSX- CARVALHO - McGRAW-HILL
TABELA DE MNEMÔNICOS Z80 - VÁRIOS - ALEPH

PARA INICIADOS

O LIVRO VERMELHO DO MSX - AVALON - McGRAW-HILL
PROGRAMAÇÃO AVANÇADA EM MSX - FIGUEREDO ALEPH

USOS DO DISK DRIVE

MSX COM DISK DRIVE - CASARI - McGRAW-HILL
DRIVES – NOVOS HORIZONTES PARA SEU MSX - VÁRIOS - ALEPH
USANDO O DISK DRIVE NO MSX - PEREIRA - ALEPH
Expert DD Plus –Manual de Instruções -VÁRIOS - ALEPH
SISTEMA DE DISCO PARA MSX- OLIVEIRA E PEREIRA - ALEPH

DBASE II

DBASE II PLUS INTERATIVO - CASARI - ATLAS
DBASE II PLUS PROGRAMÁVEL-CASARI - ATLAS

LINGUAGEM LOGO

HOTLOGO-PRIMEIROS PASSOS - VÁRIOS - ALEPH

MANUAIS DE SOFTWARE

MSX: USANDO OS MELHORES APLICATIVOS V1 - SEABRA - CAMPUS
MSX: USANDO OS MELHORES APLICATIVOS V2 - SEABRA - CAMPUS
HOTDATA - WATANABE - ALEPH
HOTWORD - WATANABE - ALEPH
HOTPLAN - WATANABE - ALEPH
HOTWORD-HOTDATA-HOTPLAN - SEABRA - CAMPUS

EDITORAS CITADAS

ALEPH
Aleph Publicações e Assessoria Pedagógica Ltda
Av. Dr. Luiz Migliano, 1110 c.301/303 Portal Trade Center
05711 São Paulo SP (011) 843-3202 843-0514

ATLAS
Editora Atlas S.A.
R. Cons. Nébias, 1384 Campos Elísios
C.P. 7185 01203 São Paulo SP (011) 221-9144

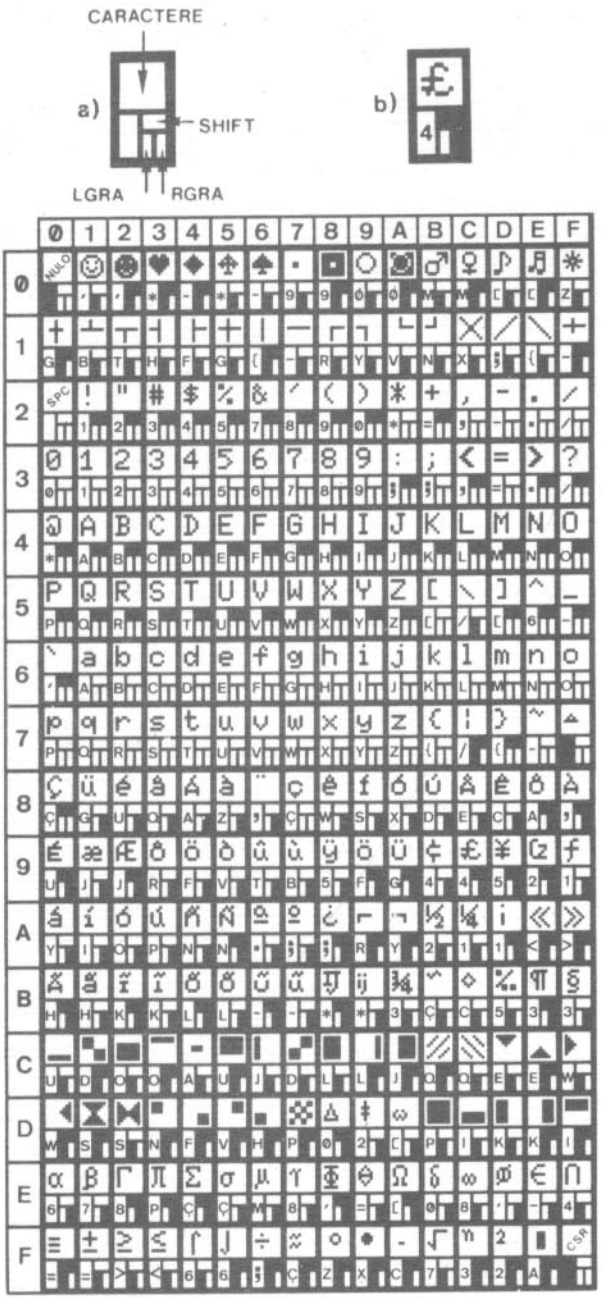
CAMPUS
Editora Campus Ltda
R. Barão de Itapagipe, 55 20261 Rio Comprido RJ
(021) 284-8443

McGRAW-HILL
Editora McGraw-Hill Ltda
R. Tabapuã, 1105 Itaim Bibi
04533 São Paulo SP (011) 881-8605 881-8528

Na figura abaixo, você tem todos os 256 caracteres relacionados com seus códigos hexadecimais.
Por exemplo, se você quiser saber qual o caractere de código hexadecimal 9C, basta procurar na linha 9, coluna C, e você encontrará o símbolo da libra esterlina. Se você comandar

PRINT CHR$ [&H9C]

obterá esse símbolo na tela.
No campo abaixo do caratere, você encontra a combinação de teclas que devem ser pressionadas, simultaneamente, para obtê-lo via teclado (figura [a]).
No caso da libra esterlina, por exemplo, você encontra o correspondente à figura [b]. Isto significa que, para obter o símbolo da libra esterlina você deverá pressionar, simultaneamente as teclas 4+RGRA+SHIFT.



Este não é um simples Manual de Instruções para microcomputador nem um Manual estruturado segundo os moldes usuais da microinformática no Brasil.

Partindo do princípio de que o Expert é o micro para "quem entende" e "para quem não entende" de microinformática, a equipe de redatores da ALEPH elaborou um texto extremamente didático e até divertido.

Além das indispensáveis instruções de instalação e apêndices de consulta, esta obra se destaca por 3 particularidades:

- Ela fornece uma completa orientação ao usuário principiante sobre os dois caminhos que ele pode tomar ao ingressar no mundo da informática: ser um "usuário de software pronto" ou um "programador".
- Para os que escolhem o segundo caminho, ela apresenta uma introdução ao MSX-BASIC, extremamente didática, na qual o leitor vai acompanhando, passo-a-passo, a construção de programas curtos, de fácil digitação e compreensão e, além disso, altamente elucidativos.
- Para os iniciados, este livro apresenta um dicionário de consulta onde todos os comandos, funções e operadores do MSX-BASIC são apresentados em ordem alfabética com sua sintaxe e exemplos de aplicação.

Trata-se portanto de um livro cuja leitura é obrigatória, não só para os usuários do Expert Plus, que já o recebem ao comprar o micro, mas também para todos aqueles que desejem se introduzir no mundo da informática e em particular no universo maravilhoso do MSX.

Editora Aleph

Aleph Publicações e Assessoria Pedagógica

Portal Trade Center Av. Dr Luis Migliano 1110

cj-301/303 CEP-05711 Morumbi - São Paulo - SP