DOS-500

SISTEMA DE OPERAÇÃO DE DISCO CP-500

DOS-500 SISTEMA DE OPERAÇÃO DE DISCO

EDITELE

ADVERTÊNCIA

- Não acione o equipamento sem antes consultar, no manual do **CP-500**, o procedimento correto para fazê-lo.

- Quando ligar ou desligar o computador, certifique-se de que nenhum drive contém disquete, pois isto pode danificar as informações contidas nele.

- Não deixe os disquetes expostos a campos magnéticos provocados por objetos metálicos ou aparelhos elétricos, tais como: imãs, ferramentas magnetizadas, transformadores ou motores elétricos, televisores, rádios, etc. Esses campos podem apagar as informações gravadas no disquete.

- Introduza totalmente o disquete no drive antes de fechar a porta do mesmo. O posicionamento correto se dá quando o ejetor do disquete é travado, deixando de haver o efeito de mola sobre o disquete. A introdução incorreta poderá danificar, não só o disquete, como também o drive.

- Leia nos capítulos **1** e **2** deste livro sobre o modo correto para instalar e operar o **DOS-500**.

DOS-500
SISTEMA DE OPERAÇÃO
DE DISCO

EDITELE

DOS-500
SISTEMA DE OPERAÇÃO
DE DISCO

1.ª Edição
9.ª Impressão
1985

EDITELE
Editora Técnica Eletrônica Ltda.
São Paulo - Brasil

Editele - Editora Técnica Eletrônica Ltda.
Av. Eng.º Luís Carlos Berrini, 1168 - 5.º andar
04571 - São Paulo - SP - Brasil

Cx. Postal 30141

Impresso no Brasil - Printed in Brazil

Índice Geral

INTRODUÇÃO

Neste manual, procuramos satisfazer dois grandes grupos de interessados na utilização do DOS-500. O primeiro grupo, dos principiantes, deseja uma explicação mais detalhada e seqüencial de suas aplicações; o segundo, dos usuários experientes, prefere uma análise mais sintética, portanto mais específica em sua forma.

Os exemplos práticos são dirigidos especialmente para o primeiro grupo e as informações técnicas para o segundo.

Não é necessário ler todo o manual para operar o sistema de disco. Se você estiver interessado em BASIC-DISCO, leia os três primeiros capítulos, que são básicos, e os referentes ao BASIC-DISCO (8.º e 9.º).

Aqui você encontrará toda a potência do CP-500 aliada às seguintes características:

1 — Seu computador poderá ser controlado pelo DOS-500, que está incluso num disquete com o Sistema de Disco.
2 — Você pode executar uma grande variedade de programas utilizando o DOS, como por exemplo, o Intérprete de BASIC-DISCO, incluso no Disquete do DOS-500.
3 — Cada disquete de sistema tem aproximadamente 126 mil bytes disponíveis para armazenar seus programas e informações, enquanto que cada disquete de dados possui 178944 bytes disponíveis.
4 — Você pode carregar e armazenar dados numa velocidade de aproximadamente 250000 bits/segundo.

PARA OPERAÇÕES COM O DISCO:

O manual do DOS-500 completa o do CP-500. Use o primeiro como fonte de informações e ele lhe indicará quando você deve utilizar o outro.

Para Operações sem Disco, consulte o manual do CP-500.

PARA INFORMAÇÕES SOBRE PROGRAMAÇÃO:

O manual do CP-500 contém todas as informações que concernem a ele, exceto aquelas pertinentes à entrada e saída de disco. Neste manual, partimos da premissa de que você já está familiarizado com as definições de programação BASIC e com os detalhes no manual do CP-500.

TERMOS ESPECIAIS

Mesmo em seções que não são técnicas, nos vimos obrigados a utilizar vários termos especiais, ao invés de repetir definições no decorrer deste manual. Esses termos foram definidos em outra parte do manual e são impressos em extra light. Procure a palavra ou frase no índice para descobrir onde ela foi definida.

DICAS PARA O CARREGAMENTO DO BASIC-DISCO

Há vários meios de acionar o BASIC-DISCO, além dos explanados neste manual.

O modo mais simples é:

BASIC programa-f:arquivos-m:endereços

Programa é uma especificação do arquivo do DOS-500 para programa de BASIC-DISCO. Após acionado, o BASIC-DISCO executa o programa. Este comando é opcional.

-f:arquivos diz ao BASIC-DISCO o número máximo de arquivos que pode ser abertos simultaneamente.

Arquivos é um número entre 0 e 15, também opcional. No caso de omissão será usado o número 3.

Se forem necessários arquivos de comprimentos variáveis, deve-se sufixar os arquivos com a letra V, senão os arquivos terão comprimentos fixos.

-m:endereço diz ao BASIC-DISCO para não utilizar memória acima desse endereço.

Também é opcional. Se for omitido, o BASIC-DISCO usará toda a memória.

Se todas as opções forem omitidas, o BASIC-DISCO perguntará: QUANTOS ARQUIVOS? e MEMÓRIA? As opções permitem a especificação de qualquer um, ou de todos os itens abaixo:

1 — Qual o programa a ser executado após o acionamento do BASIC-DISCO.

2 — O número máximo de arquivos de dados que podem ser abertos simultaneamente. Quanto maior o número de arquivos, menor a área disponível para armazenagem e execução de seus programas.

NOTA

Cada arquivo de comprimento fixo ocupa até 360 bytes e cada um de comprimento variável até 616 bytes de memória.

3 — O mais alto endereço a ser usado pelo BASIC-DISCO durante a execução de um programa. Você deve omití-lo a não ser que vá chamar sub-rotinas em linguagem de máquina.

Exemplos

Na condição DOS-500 Ativo, se você digitar:

BASIC ENTER . O BASIC-DISCO entrará no modo imediato assim que você responder às perguntas sobre número de arquivos e memória.

BASIC-F:1 ENTER . O BASIC-DISCO abrirá um arquivo de comprimento fixo e não protegerá as memórias.

BASIC-M:32000 ENTER . O BASIC-DISCO abrirá 3 arquivos de comprimento fixo e usará memória inferior a 32000.

BASIC FOLHAPAG -F:3V ENTER o BASIC-DISCO será acionado; Carregará e executará o programa BASIC chamado FOLHAPAG (FOLHA DE PAGAMENTO). Serão abertos 3 arquivos de comprimentos variáveis e o BASIC poderá usar toda a memória disponível.

NOTA

Se você modificar o conteúdo do BASIC-DISCO para ele lhe questionar sobre a velocidade de Transferência quando você introduzir BASIC-DISCO, você ainda obterá o sinal : CASS? não importando a opção utilizada.

INSTALAÇÃO

Primeiramente, ajuste o computador de acordo com as instruções existentes no manual do CP-500.

Se você possui um sistema de um ou dois drives, a instalação já está realizada. Os drives internos já vêm prontos para sua utilização.

No caso de um sistema de 3 ou 4 drives, será necessária a conexão de drives externos.

DRIVES EXTERNOS DE DISCOS

Os 2 drives externos não são permutáveis.
Eles possuem algumas diferenças internas.

	nome do sistema
Primeiro Drive Externo Adquirido	"Drive 2/3"
Segundo Drive Externo Adquirido	"Drive 2"

Procedimento de Instalação

1. Localize o cabo plano tipo "fita", que foi incluso no primeiro drive.
   Veja figura 1 mostrando as etiquetas dos plugues.
2. Conecte o plugue simples do computador à tomada de expansão de disco na parte traseira do computador. Veja figura 2.
3. Em referência a figura 3.
   Conecte os drives externos à extremidade do cabo com se segue:

3.A — Se você possuir um drive externo, faça a sua conexão ao plugue do drive 2 próximo ao meio do cabo tipo "fita".

3.B — Se você possuir 2 drives extérnos, faça a conexão do plugue do Drive 3 no final do cabo e do plugue do drive 2 próximo ao meio do cabo.

4. Faça a conexão dos drives externos em uma fonte conveniente de corrente alternada. As especificações relativas à conexão à rede elétrica são fornecidas na própria unidade e neste manual.

Você já está suficientemente informado para estudar o sistema de disco.

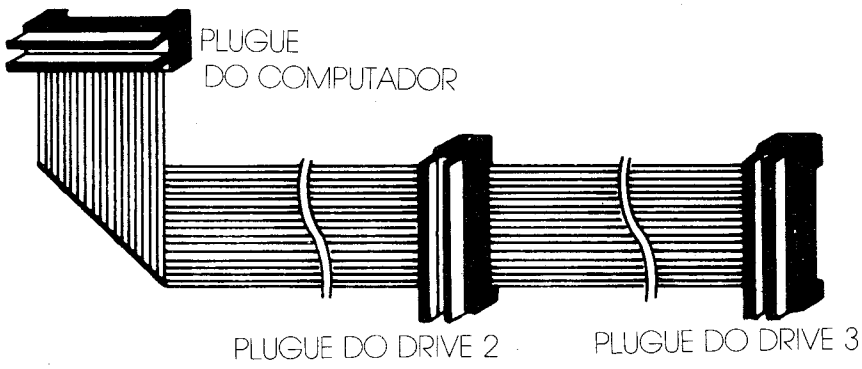


Figura 1 — Cabo externo de disco com plugues etiquetados

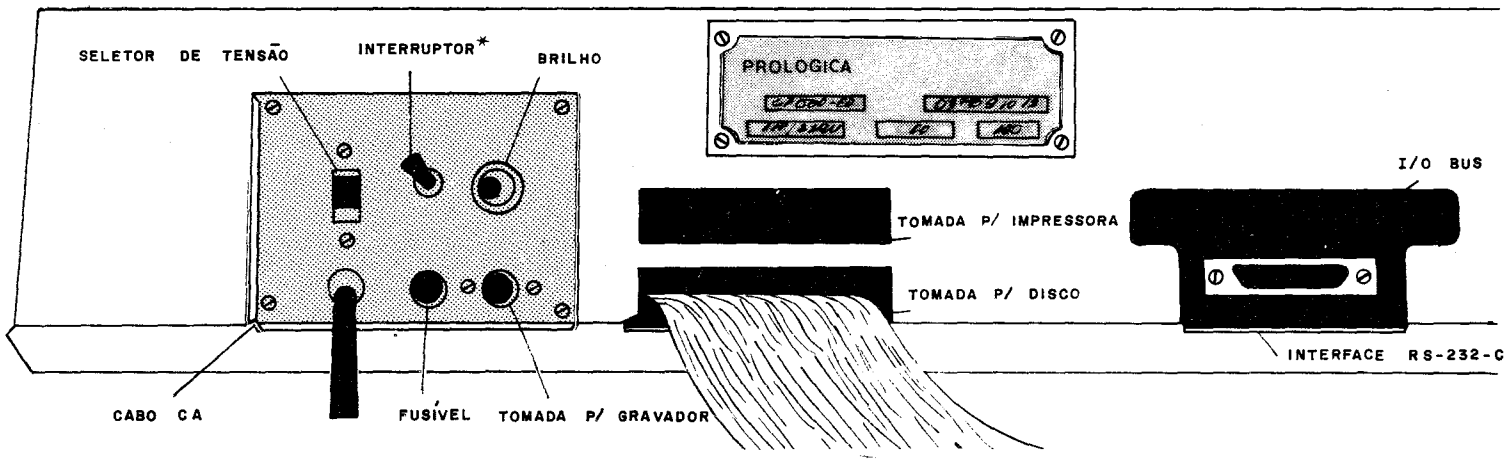


Figura 2 — Conexão de cabo externo de disco ao CP-500
* — Interruptor de força. Todos os drives deverão estar vazios no momento em que você ligar ou desligar o computador, caso contrário a informação no disquete poderá ser destruída.

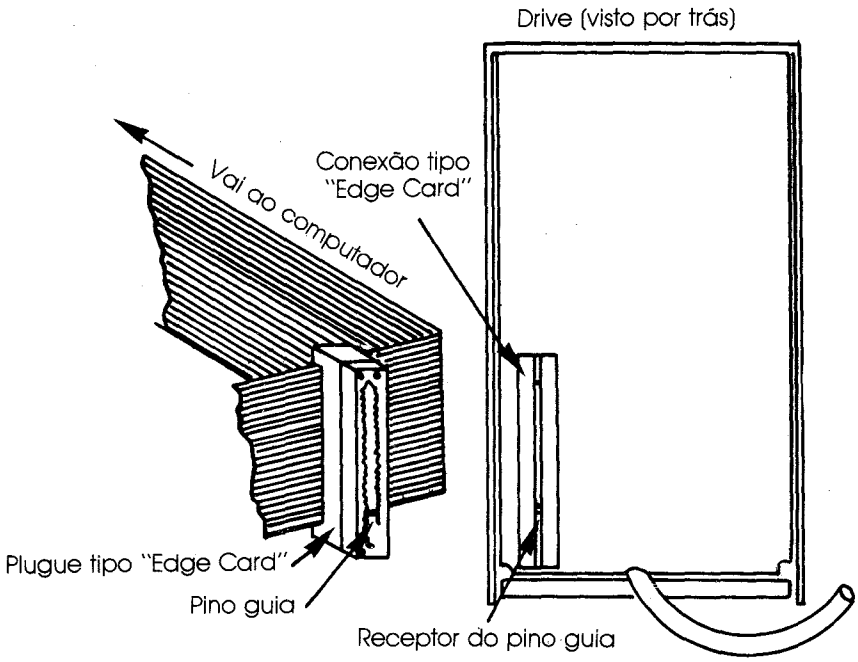


Figura 3 — Conexão de Drives externos de discos

MANIPULAÇÃO DOS DISQUETES

Primeiramente, tente se familiarizar com os vários elementos do seu sistema de disco. Veja as figuras 4 e 5. Este primeiro contato é muito importante, pois há a possibilidade de danificar seu disquete, caso você não saiba manipular o seu computador corretamente.

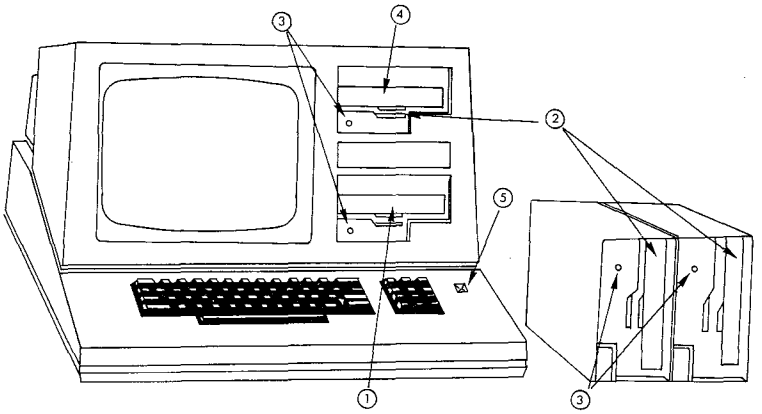


Figura 4 — Sistema de Disco com Drives Externos 'Opcional/Extra.
1 — Drive 0. O sistema de disquete do DOS-500 é conectado neste drive.
2 — Drive 1,2,3. Esses podem conter disquetes de dados. Esses tipos de disquete são descritos resumidamente neste capítulo.
3 — LED de seleção de drives. No caso de acesso a um drive, o seu LED é ativado.
4 — Porta de Drive. Para inserção de disquete abra esta porta. Nunca o remova enquanto o seu LED estiver ativado ou quando ele contiver arquivos em aberto.
5 — Botão RESET Ao se pressionar este botão, o computador tentará copiar do drive o software do sistema de operação. O disquete DOS-500 deverá estar no drive 0, quando se pressionar esse botão.

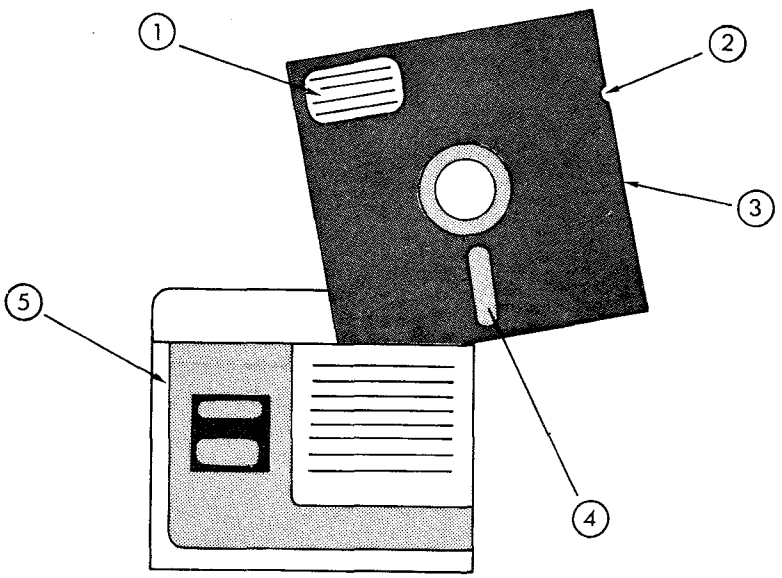


Figura 5 — Um Disquete
1 — Envelope de armazenagem.
Mantenha o disquete neste receptáculo quando não estiver em uso.
2 — Entalhe de proteção contra gravação.
Quando este entalhe está coberto, os drives de disco não podem gravar (modificar informação) no disquete. Não pressione a alça para dentro do entalhe, quando for utilizá-lo. Se a alça se tornar saliente, o drive do disco não poderá "sentir" se o disco está protegido contra gravação. Deixe o entalhe descoberto se desejar gravar ou modificar informações no disquete.
3 — Jaqueta.
O disquete está permanentemente selado dentro de um envoltório de proteção aqui denominado jaqueta. Não tente removê-la.
4 — Janela de gravação/leitura.
O drive tem acesso à superfície do disquete através desta janela. Não toque nesta área.
5 — Etiqueta.
Para escrever nesta etiqueta, somente utilize caneta de ponta de feltro. Qualquer outra caneta ou lápis poderá causar danos ao disquete.

DISQUETES

Em geral, manuseie os disquetes cuidadosamente, observando as mesmas precauções referentes a cassetes e discos de alta fidelidade. Uma pequena saliência, partícula de poeira ou arranhão pode inutilizar definitivamente o disquete.

— Mantenha o disquete no envelope de armazenagem, sempre que não estiver em uso.
— Não coloque o disquete no drive enquanto se ativa ou desativa o sistema.
— Mantenha os disquetes afastados de campos magnéticos (transformadores, motores de corrente alternada, magnetos, TVs, rádios, etc.). Campos magnéticos potentes apagarão os dados armazenados do disquete.
— Manuseie os disquetes somente pela jaqueta. Não toque as superfícies expostas. Não tente esfregar ou limpar a superfície do disquete pois ela é fácil de ser arranhada.
— Mantenha os disquetes fora do contato direto do sol e longe de fontes de calor.
— Não escreva diretamente na jaqueta do disquete com dispositivos de ponta dura, tais como; caneta esferográfica ou lápis. Somente use canetas com ponta de feltro.
— Evite a contaminação de disquetes com cinzas de cigarro, poeira e outras partículas.
— Armazene os disquetes num arquivo de elementos de disposição vertical onde eles estejam protegidos de danos devido à pressão lateral como arquivo de discos comuns.
— Em ambientes empoeirados, há a necessidade da filtração do ar na sala do computador.

SUGESTÕES NA ROTULAÇÃO DE DISQUETES

Cada disquete tem uma etiqueta na sua jaqueta. Esta etiqueta se destina à gravação de dados importantes, que nunca sofrerão modificações. É aconselhável atribuir um único número a cada disquete para uma melhor organização desses. Escreva este número na etiqueta juntamente com seu nome e data de sua primeira utilização. Lembre-se de usar caneta com ponta de feltro. Esta etiqueta não é o local apropriado para se gravar o conteúdo do disquete, pois esse deverá ser modificado e a supressão ou eliminação da informação contida nela é indesejável.

OPERAÇÃO

SISTEMA DO ACIONAMENTO

1 — Ative todos os periféricos.
2 — Ligue o computador. Espere a parada dos motores dos drives de disco.
3 — Localize o disquete do DOS-500 que contém o sistema de disco.
Introduza o drive 0 com o lado etiquetado para cima e a janela de gravação/leitura em direção à entrada do drive (Veja figura 6).
4 — Feche a porta no drive quando o disquete estiver completamente introduzido.
5 — Pressione RESET. O computador deve carregar o DOS-500 e começar o diálogo inicial descrito na próxima seção.

Se nada aparecer na tela ou se a mensagem DISQUETE? for visualizada, verifique o seguinte:
— Você está usando o disquete do sistema DOS-500?
— O disquete foi inserido na posição certa?
— O disquete está convenientemente introduzido no Drive?
— No caso da utilização de drives externos: Eles estão conectados e ativados apropriadamente?

Figura 6 — Introduzindo um Disquete

Caso não solucione o problema, consulte o capítulo de manutenção e deteção de falhas.

DIÁLOGO INICIAL DO DOS-500

Ao se ativar o sistema de disco do BASIC, ele carregará o DOS-500 e começará o diálogo inicial.
1 — O número da versão do DOS-500 e sua data de criação serão mostrados, seguidos pela quantidade de memória RAM (32K ou 48K) e o número de drives do sistema.
2 — O DOS-500 solicitará a introdução da data na forma MM/DD/AA. Por exemplo, 07/04/80 para 04 de Julho de 1980. Digite a data correta e pressione ENTER. O DOS-500 não continuará se a data não for digitada corretamente.
3 — O DOS-500 solicitará, então, a introdução da hora na forma HH:MM:SS. Por exemplo: 14:45:00 para 2:45 da tarde. Digite a hora correta e pressione ENTER. Se não desejar ajustar a hora, pressione ENTER no início da linha. O DOS-500 a ajustará em 00:00:00.

4 — Neste momento o DOS-500 mostrará a mensagem:

```
DOS500 ATIVO
```

Quando esta mensagem for visualizada, você estará no modo READY do DOS-500 e deverá digitar um comando.

OPERAÇÕES IMPORTANTES DE DISCO

Nesta seção, descrevemos 3 operações:
1 — Cópia do disquete dos DOS-500. (BACKUP).
2 — Inicialização de um disquete de dados (FORMAT).

Todos os principiantes devem fazer o procedimeto de BACKUP (cópia) do DOS-500 agora.

Os possuidores de sistemas com mais de um drive devem executar a operação FORMAT para vários disquetes.

Informações mais detalhadas serão fornecidas posteriormente, aqui apenas delinearemos esses procedimentos.

FAZENDO UM BACKUP (cópia)

Sua primeira operação será copiar o disquete do DOS-500 que você recebeu da PROLÓGICA. Este disquete contém um programa de uso geral chamado BACKUP, criado para esse fim.

1 — Coloque o disquete do DOS-500 no Drive 0 e um outro, virgem, no drive 1. O disquete do DOS-500 será chamado de FONTE e o outro de DESTINO durante esta operação.
2 — Acione o DOS-500 como foi explicado anteriormente e aparecerá na tela:

```
DOS500 ATIVO
```

3 — Digite: BACKUP ENTER

4 — O DOS-500 carregará e iniciará o BACKUP. Então, você visualizará a pergunta: NÚMERO DRIVE FONTE?
Especifique o drive que contém o disquete original do DOS-500, digitando 0 ENTER

5 — A próxima pergunta será: NUMERO DRIVE DE DESTINO?

Agora, esquecifique o drive que será utilizado na cópia. Se você tiver 2 ou mais drives no sistema, digite: 1 ENTER

6 - Agora, a pergunta será:

```
SENHA MESTRA DO DISCO FONTE?
```

Digite: SENHA ENTER

(SENHA é a senha de disquete fornecida)

7 — O processo de cópia começa agora:
Se o disquete destinatário não estiver formatado, o BACKUP o formatará antes de continuar (antes da utilização de qualquer disquete, este deve ser inicializado, isto é, seu espaço deve ser formatado — as áreas dos arquivos definidas e rotuladas e um índice ou diretório criado).
Se você estiver usando um sistema de drive simples, o DOS-500 lhe solicitará a permuta de disquetes FONTE e DESTINO várias vezes, durante o processo de Formatação/Cópia.

Após um BACKUP de drive simples, o DOS-500 mostrará essa mensagem na tela: INSIRA DISQUETE SISTEMA

Certifique-se que o disquete no Drive 0 seja um do sistema DOS-500, e então pressione ENTER.

O processo de cópia está completo agora. Sugerimos a estocagem do disquete original e a uti-

lização da réplica em seu trabalho. Se algo ocorrer à cópia, você pode produzir outra a partir da original.

FORMATANDO UM DISQUETE DE DADOS (FORMAT).

Essa seção só é útil a sistemas multidrive.

O drive 0 deve conter um disquete DOS-500, pois assim o computador pode ter acesso aos programas de sistema ali armazenados. Grande parte da capacidade de armazenagem deste disquete é utilizada por esses programas. Entretanto, os outros drives podem ocorrer disquetes de dados com total capacidade disponível de armazenagem para os seus programas e informações.

O comando FORMAT inicializa ou formata um disquete. Se o disquete tiver sido formatado anteriormente, toda essa informação precedente poderá ser perdida.

O disquete resultante não contém os arquivos do sistema e só pode ser utilizado nos DRIVES 1,2 ou 3.

1 — No modo DOS-500 ATIVO, digite:
FORMAT ENTER.

2 — O DOS-500 iniciará um programa formatador e procederá a uma série de perguntas:

```
FORMATAR O DRIVE?
```

Introduza um disquete virgem no Drive 1. Digite 1 ENTER .

```
NOME DISQUETE?
```

Este nome servirá como uma etiqueta para o disquete. Digite um nome apropriado de 1 a 8 letras e números, iniciando sempre por uma letra.

Pressione ENTER no final de uma senha.

A utilização da senha permite acesso aos comandos PURGE, PROT e BACKUP a todos os arquivos que não forem de sistema. Sugerimos o uso da senha SENHA, exceto no caso de uma proteção especial, mas ao escolher uma, não a esqueça!

Se o disquete contiver dados, o DOS-500 lhe avisará:

```
DISQUETE CONTEM DADOS, USO DISCO OU NAO?
```

O aviso é necessário, pois o FORMAT apaga toda a informação que o disquete contiver.

Caso deseje cancelar o FORMAT digite N ENTER; para continuar digite S ENTER ou U ENTER.

3 — O DOS-500 formatará e verificará o disquete. O disquete de dados estará pronto para ser usado no Drive 1, 2, ou 3.

NOTAÇÃO E ABREVIATURAS

Para maior nitidez e brevidade nas explicações deste manual, usamos uma notação e estilo de grafia especiais.

MAIÚSCULAS e pontuação.

Indica material que deve ser introduzido exatamente como aparece. (O único símbolo de pontuação que não é introduzido é a reticência explicada abaixo); Por exemplo:
na linha:

DUMP LISTA (START=7000, END=7100, TRA=7004)

toda letra ou caractere deve ser digitado como indicado.

extra light minúsculo

Representam palavras, letras, caracteres ou valores extraídos de um conjunto de valores aceitáveis para um comando especial. Por exemplo, a linha:
LIST nome do arquivo
indica que você pode fornecer qualquer especificação válida de arquivo após LIST.....

Reticências indicam que os itens anteriores podem ser repetidos. Por exemplo:

ATTRIB nome de arquivo (opção,....)

indica que várias opções podem ser repetidas dentro dos parênteses.

ø

Este símbolo é usado ocasionalmente para indicar em caractere de espaço em branco (ASCII 32 decimal, 20 hexadecimal)

PRINT "øOløløløø!"

X'**nnn**'

Indica que nnn é um número hexadecimal. Todos os outros números do texto deste manual estão na forma decimal, salvo observações em contrário.

X'7000'

indica o valor hexadecimal 7000 (decimal 28672).

Quaisquer palavras, letras ou números que apareçam na tela são escritos no tipo matriz de pontos. São usadas maiúsculas embora, na tela possa aparecer em letras minúsculas algumas vezes.

SISTEMA DE OPERAÇÃO DE DISCO-DOS-500

O que é o DOS-500?

O DOS-500 (pronuncia-se D-O-S 500) significa "Sistema de Operação de Disco do CP-500".

Desempenha 3 papéis:

1 — Programa mestre
2 — Interpretador de comando
3 — Gerente de programa

No primeiro caso, ele possibilita a integração eficiente do microprocessador e seus componentes.

Os componentes incluem:

— **Memória de Acesso Aleatório (RAM)**. O DOS-500 reserva espaço para sua própria necessidade e aloca espaço para os programas dos usuários.
— **Drives de Disco.** O DOS-500 interliga-se com o hardware de disco e fornece um arquivo de sistema para armazenagem do sistema e de informações dos usuários nos disquetes.
— **Dispositivos Entrada/Saída.** Englobam teclado, tela de vídeo, impressora e interface RS-232-C.

O DOS-500 é também um intérprete de comandos. Ao se visualizar: **DOS-500 ATIVO** você pode introduzir comandos que controlam o modo de operação do sistema. Esses comandos são chamados de biblioteca.

No papel de gerente de programa, O DOS-500 carregará e executará os programas de sistema ou os dos usuários. Durante este processo tanto um programa quanto o outro estarão sob controle do computador.

A figura 7 mostra as relações existentes entre estes três aspectos de funcionamento.

ONDE O BASIC É ÚTIL?

Com relação a figura 7, você pode observar que o BASIC-DISCO pertence à categoria dos "pacotes de linguagem".

O BASIC-DISCO consiste em alguns melhoramentos do BASIC-RESIDENTE e ainda capacidade de entrada/saída em discos. Ele utiliza o BASIC-RESIDENTE (armazenado em ROM) sempre que possível. Por exemplo, o BASIC-RESIDENTE engloba todas as funções matemáticas. Se você não está acostumado a sistemas com disco, há uma pequena diferença digna de nota:

Em sistemas sem disco, o BASIC-RESIDENTE tem o controle quando você aciona o equipamento, no entanto, em sistemas com disco o DOS-500 tem este controle. Você tem que instruir o DOS-500 a carregar e rodar o BASIC. Somente neste momento, você pode iniciar a execução do programa escrito em BASIC.

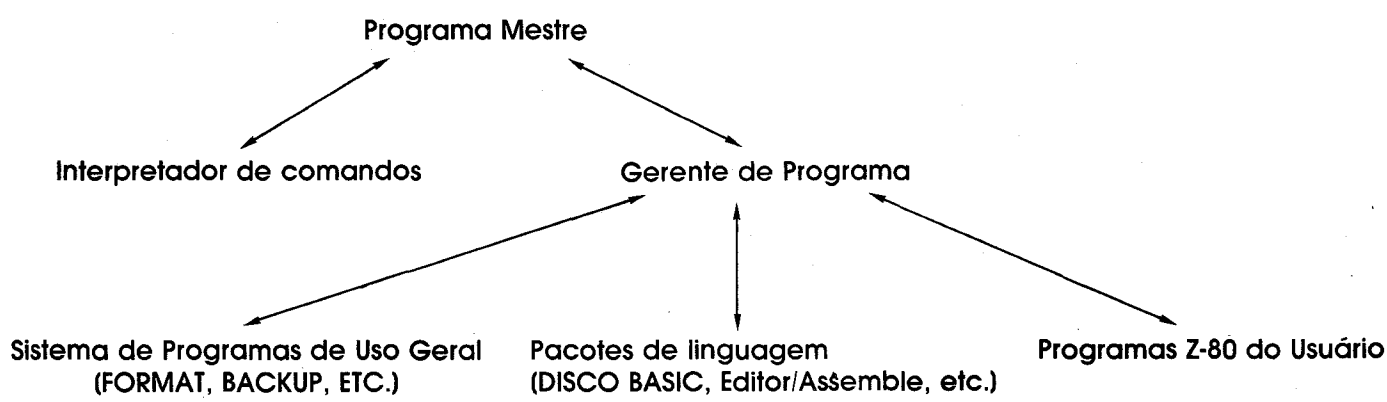


Figura 7 — Papéis do DOS-500

COMO USAR O DOS-500

INTRODUÇÃO DE UM COMANDO

Quando o DOS-500 mostra:

```
DOS500 ATIVO
```

você pode digitar um comando, que não exceda 63 caracteres de comprimento.

Deve-se pressionar ENTER para finalizar a linha, só então do DOS-500 "aceitará" o comando.

Por exemplo, digite: CLS ENTER

O DOS-500 limpará a tela e aparecerá **DOS-500 ATIVO**.

Geralmente, seus comandos exigirão mais que uma palavra. Por exemplo, para apagar um arquivo denominado MEUNOME, é necessário a especificação de um comando e de um nome de arquivo.

```
KILL MEUNOME
```

Esta instrução diz ao DOS-500 para procurar o arquivo chamado MEUNOME, eliminá-lo do disquete, liberando espaço no disco.

Ao se digitar uma linha, o DOS-500 segue este procedimento:

1 — Primeiramente, verifica se o nome digitado é um comando DOS-500.
2 — Se não for, ele prosseguirá verificando se o mesmo é um nome de arquivo de programa num dos drives.
3 — Na procura de um arquivo, o DOS-500 inicia pelo DRIVE 0, então Drive 1,... etc., ou pelo drive especificado pelo comando MASTER.

Se o DOS-500 encontrar um arquivo especificado pelo usuário, ele o carregará e executará se o mesmo for um arquivo de programa, caso contrário, será emitida uma mensagem de erro.

SINTAXE DE COMANDO

Sintaxe de comando é a forma geral de um comando. A sintaxe informa como utilizar as palavras chaves (tais como DIR, LIST, CREATE, etc.) aliadas aos parâmetros e à pontuação necessários. Se necessitar de auxílio na elaboração da sintaxe de comando ao operar o DOS-500, digite:
HELP comando ENTER.

Comando deve ser um comando de biblioteca específico com o qual você está tendo problemas. O DOS-500 lhe fornecerá o formato sintático, assim como uma definição concisa do comando.

COMANDOS

COMANDOS SEM ARQUIVO

COMANDO (opções) comentário

"Opções" é uma lista de um ou mais parâmetros que um comando pode necessitar. Alguns não tem "opções" Os parênteses delimitando as opções devem aparecer, exceto na ausência das mesmas.

"Comentário" é um campo opcional usado para documentar o objetivo do comando. Os comentários são úteis dentro de arquivos de introdução automática de comandos (Veja BUILD e DO).

COMANDOS DE UM ARQUIVO

COMANDO nome de arquivo (opções) comentário
Nome do arquivo é uma especificação padrão de arquivo do DOS-500.
Opções — Veja definição acima

COMANDOS DE DOIS ARQUIVOS

COMANDO nome de arquivo delimitador nome de arquivo (opções) comentário.
nome de arquivo é uma especificação padrão de arquivos de DOS-500.
Delimitador é um espaço em branco:
Opções
Comentário Veja definição acima

ESPECIFICAÇÃO DE ARQUIVO

O único meio de armazenar informações num disquete é colocá-la num arquivo de disco.

Posteriormente, as informações podem ser recuperadas através do nome do arquivo especificado.

Uma especificação de arquivo tem a forma:
nome de arquivo/ext senha:d
Nome de arquivo consiste em uma letra seguida de até 7 letras ou números opcionais
Ext é uma extensão opcional do nome. Consiste numa seqüência até 3 letras ou números, iniciada por uma letra.
Senha é opcional. Consiste numa seqüência de até 8 letras ou números, sendo iniciada sempre por letra.
:d é uma especificação opcional do drive. "d" pode ser um desses números: 0,1,2,3.

NOTA

Não é permitido espaços em branco dentro de uma especificação. O DOS-500 terminará a especificação de arquivo ao encontrar o primeiro espaço em branco.

Por exemplo: `ARQUIVO A/TXT.GERENTE:3` refere-se ao arquivo chamado ARQUIVOA/TXT com a senha GERENTE, no Drive 3.

O nome, a extensão e a especificação de drive fazem com que uma especificação de arquivo só possa ter uma única interpretação. A senha não tem esta característica, ela simplesmente controla o acesso ao arquivo.

NOMES DE ARQUIVO

Um nome de arquivo consiste em um nome e uma extensão opcional de nome. Você pode escolher qualquer letra para o nome, seguida por até 7 letras ou números adicionais. Para o uso de uma extensão, inicie com um barra diagonal e acrescente até 3 números ou letras, mas o primeiro caractere deve ser uma letra.

Por exemplo:

CP-500/TXT	RESUMO	ARQUIVO
TESTE	AGOSTO/VND	DADO1/BAS
NOME/Z10	TESTE1	TABELA18

Esses são todos nomes de arquivo válidos e distintos.

Embora as extensões de nomes sejam opcionais, elas são úteis na identificação do tipo de informação no arquivo. Por exemplo, você pode usar o seguinte conjunto de extensão:

/BAS para programa BASIC
/TXT para texto ASCII
/CMD para arquivo de comandos em linguagem de máquina
/DAT para informações (DADOS)

Uma vantagem ao utilizar-se extensões é que você pode saber o tipo de informação contida no arquivo, apenas consultando o "diretório", isto é, o índice.

Outra vantagem é que o DOS-500 pode reconhecer certas extensões. Por exemplo: se um arquivo tem a extensão /CMD, quando se digitar: NOMEARQ ENTER omitindo a extensão /CMD, o DOS-500 carregará e tentará a execução daquele arquivo.

ESPECIFICAÇÃO DE DRIVE

Se você fornecer ao DOS-500 um comando como:

`KILL TESTE/A` ele procurará o arquivo TESTE/A.

no drive 0 em primeiro lugar, e nos drives 1,2 e 3 seqüencialmente até encontrá-lo.

Sempre que se omitir uma especificação de Drive, DOS-500 seguirá esta seqüência, exceto se um comando MASTER for usado.

É possível dizer ao DOS-500 exatamente qual drive você quer usar especificando-o.

Essa especificação consiste de 2 pontos seguidos de um desses algarismos: 0,1,2 ou 3 que correspondem aos drives a ser utilizados.

Por exemplo: **KILL TESTE/A:3** diz ao computador para apagar o arquivo TESTE/A somente no drive 3.

Sempre que o DOS-500 tiver que abrir um arquivo (por ex.: para listá-lo), ele seguirá a mesma seqüência. Quando o DOS-500, tiver que escrever um arquivo, ele não considerará nenhuma disquete protegido contra gravação.

SENHA

Você pode proteger um arquivo contra acesso ou uso não autorizados atribuindo-lhe uma senha. Desta maneira, ninguém terá acesso ao arquivo sem saber a senha apropriada.

É muito importante saber que todo arquivo tem uma senha, mesmo que não tenham sido especificada quando foi criado o arquivo. Neste caso, a senha será 8 espaços em branco e o arquivo estará desprotegido.

Qualquer um terá acesso a ele simplesmente indicando o nome do arquivo.

O DOS-500 permite a determinação de 2 senhas ao arquivo:

— Uma "palavra atual" que confere total acesso à informação.
— Uma "palavra de acesso" que confere um acesso limitado à informação (veja ATTRIB).

Ao se criar um arquivo, as palavras atual e de acesso são igualadas a sua senha especificada: Você pode modificá-las posteriormente com o comando ATTRIB.

Uma senha consiste de um ponto seguido de uma a oito letras ou números. Se você não determinar uma senha ao arquivo. o DOS-500 utilizará uma senha de oito espaços em branco. Por exemplo, suponha que você tenha um arquivo chamado SEGREDO/BAS, e que sua palavra de acesso e atual seja MEUNOME.

Sendo assim, o comando: `KILL SEGREDO/BAS` não surtirá efeito, pois você tem que incluir a palavra MEUNOME na especificação do arquivo para alcançar seu intento.

Suponha agora, que você tenha um arquivo denominado DOMÍNIO/BAS e tem espaços em branco como senha. Nessas condições o comando:

KILL DOMINIO/BAS.ADVINHE não será obedecido, pois ADVINHE não é a senha especificada.

ALGUMAS DEFINIÇÕES IMPORTANTES

Disquete de Sistema X Disquetes de dados.

Um disquete de sistema é aquele que contém o software de sistema de operação de disco do DOS-500. Embora sujeito a limitações de espaço, ele pode conter também seus próprios arquivos. Este disquete deve sempre permanecer no DRIVE 0, quando o computador estiver em uso.

Por outro lado, um disquete de dados não contém o software de sistema operacional e portanto não pode ser utilizado no Drive 0, apenas nos drives 1,2,3. Este disquete tem um máximo espaço disponível para armazenagem dos seus próprios programas e dados.

ARQUIVOS DE SISTEMA, DE PROGRAMAS E DE DADOS

Os arquivos de sistema incluem o software de sistema operacional do DOS-500, o intérprete de linguagem BASIC, os arquivos FORMAT, BACKUP e outro software que seja desenvolvido pela PROLÓGICA. Estes arquivos aparecem no Diretório com um atributo "S" (Veja DIR).

Arquivo de programas são armazenados em formato especial que lhes permite ser carregados e executados diretamente da condição **DOS-500 ATIVO**

Por exemplo, um programa escrito em BASIC será armazenado como arquivo de dados. Ele poderá ser carregado e executado no BASIC, mas não na condição **DOS-500 ATIVO**

SENHAS MESTRAS

Uma senha mestra é determinada inicialmente a cada disquete, durante o FORMAT ou o BACKUP (sua senha mestra para o DOS-500 é SENHA). A senha mestra possibilita o uso do BACKUP, PURGE e PROT num disquete. Você pode modificar esta condição, utilizando a senha mestra do disquete (Veja PROT).

Comandos de uso geral do DOS-500

BACKUP

Cria uma Cópia Exata de um Disco Original

BACKUP: fonte:destino
:fonte especifica o drive que contém o disco original. Se omitido, o DOS-500 requisitará esta informação.
:destino especifica o drive que contém o disco que receberá a cópia. Se omitida, o DOS-500 perguntará por ela.
:fonte e :destino podem especificar o mesmo drive.

BACKUP copia os conteúdos do disco fonte no disco de destino. Isto lhe dá uma "2ª via" do disco. Mantenha sempre uma cópia extra de dados e programas que possui em disco.

NOTA
Ambos os discos, fonte e de destino, devem ser liberados para gravação (WP).

O DOS-500 perguntará por cada passo após você introduzir BACKUP.
Se forem omitidos o destino e a fonte o DOS-500 começará com esta questão:

```
NUMERO DRIVE FONTE?
```

Digite o número do drive que contém o disco original e pressione: ENTER

```
NUMERO DRIVE DE DESTINO?
```

Digite o número do drive que conterá a cópia e pressione: ENTER

```
SENHA MESTRE DO DISCO FONTE.
```

Digite a senha atribuída ao seu disco original.

```
DISQUETE CONTEM DADOS. USAR O DISQUETE?
```

Digite S (Sim) ou N (Não)

```
JA formatado. REFORMATAR?
```

Digite S (Sim) ou N (Não)

Se você especificar os números dos drives, o DOS-500 pedirá a senha, pulando as duas primeiras perguntas. O DOS-500 irá, então, se encarregar de formatar e verificar o disco cópia bem como de avisar se existe algum erro de gravação ou de trilhas defeituosas.

FORMAT

Prepara Um Disco De Dados

FORMAT:d

O valor de :d especifica o drive que contém o disco a ser formatado. Se for omitido, o DOS-500 pedirá esta informação.

Este comando permite que se preparem discos de dados (discos novos ou aqueles que contenham dados ou programas dispensáveis), deixando a máxima quantidade de espaço possível para seus arquivos de dados e de programas.

NOTA

Discos de dados podem ser usados apenas nos drives 1, 2 e 3 a menos que sejam usados para BACKUP ou FORMAT

FORMAT pega um disco virgem (ou magneticamente apagado), grava os limites dos setores e trilhas nele e os usa para inicializar o disco e, ainda, cria um diretorio.

Quando o FORMAT detecta um disco com informações, ele mostrará uma mensagem de aviso:

```
DISQUETE CONTEM DADOS. USAR O DISQUETE?
```

Digite S(Sim) e pressione ENTER se você quer reformatar ou N(Não) e ENTER se quiser manter a informação do disco.

FORMAT deixará de lado trilhas defeituosas para evitar que dados contidos nestas trilhas venham a se perder.

Se começarem a ocorrer erros READ (de leitura) durante o acesso, reformate o disco.

EXEMPLO

```
FORMAT :1
```

Depois de você ter respondido ao **NOME DISQUETE?** e **SENHA MESTRE?,** o DOS-500 formatará o disco que estiver no drive 1.

TESTMEM

Teste de memória do CP-500

TESTMEM

Este programa utilitário testa automaticamente todas as posições de memória do CP-500, qualquer que seja a configuração utilizada Em primeiro lugar, são testadas as posições da ROM. Se estiver tudo em ordem com as ROM, o comando passa automaticamente para o teste da RAM do vídeo. Estando em ordem, o comando passa a testar individualmente cada banco de 16k de memória.

Se durante este processo for detectado algum erro, será impressa na tela uma mensagem de erro informando qual o bit (e o banco) que está com defeito.

Deve-se repetir o teste para se ter certeza da existência e localização do erro.

Se o seu CP-500 estiver operando com 48k (3 bancos de 16k) não deverá ocorrer erro algum. Caso esteja com 16 ou 32k (1.º ou 1.º e 2.º bancos), o programa indicará erros em todas as posições de todos os bancos que não estiverem operando, devido à inexistência de memória nessas posições. Portanto devem ser levados em consideração apenas os erros indicados nas posições reais da RAM.

Uma vez confirmado o erro, deve-se copiar a mensagem e contatar o representante PROLÓGICA mais próximo.

Caso nenhum erro seja detectado, pressione RESET para se carregar novamente o DOS-500 e iniciar normalmente as operações.

NOTA

TESTMEM modifica todo o conteúdo da RAM, porisso, guarde em disco ou fita qualquer informação valiosa que estiver na memória.

COMANDOS DE MANIPULAÇÃO DE ARQUIVO

APPEND

Anexa arquivos

APPEND Arquivo fonte Arquivo destino

Arquivo fonte é a especificação do arquivo que será anexada (copiada) no final de um outro.
Arquivo destino é a especificação de arquivo que recebeu o anexo (o acréscimo).

NOTA

Tanto o arquivo-fonte quanto o arquivo-destino devem possuir o formato ASCII (arquivos de dados ou programas de BASIC armazenados (SAVE) com a opção A).

O comando APEND copia o conteúdo do arquivo fonte no final do arquivo-destino. O arquivo fonte não é alterado, enquanto que o arquivo destino é ampliado para incluir o arquivo fonte.

NOTA

Os comprimentos de gravação lógica devem combinar Para maiores detalhes sobre estes comprimentos de gravação, consulte DIR.

Exemplos:

```
APPEND PALAVRA/C PALAVRA/D
```

Uma cópia de PALAVRA/C será acrescida ao PALAVRA/D.

```
APPEND REGIAO1/DAT TOTAL/DAT.TENTE
```

Uma cópia de uma REGIÃO 1/DAT será acrescida à TOTAL/DAT, que é protegida pela senha TENTE.

EXEMPLOS PRÁTICOS

Suponha, que você tem dois arquivos de dados: FOLHAPAG/A e FOLHAPAG/B.

FOLHAPAG/A

Albuquerque, A. S.
Brito, J. J
Fernandez, F. L
Linz, M. A
Martinez, S. O

FOLHAPAG/B

Penteado, P. M
Pimentel, L. A
Silva, J. C
Silva, L. A
Silveira, W. L

Você pode combinar os 2 arquivos com o comando

```
APPEND FOLHAPAG/B FOLHAPAG/A
```

A folha de pagamento/A se apresenta assim:

FOLHAPAG/A
Albuquerque, A. S. . . . . .
Brito, J. J . . . . . .
Fernandez, F. L . . . . . .
Linz, M. A . . . . . .
Martinez, S. O . . . . . .
Penteado, P. M . . . . . .
Pimentel, L. A . . . . . .
Silva, J. C . . . . . .
Silva, L. A . . . . . .
Silveira, WL . . . . . .

A FOLHAPAG/B será inalterada. Para observar o arquivo gerado, digite LIST FOLHAPAG/B (ASCII).

NOTA
Não carregue um programa em BASIC depois de um comando APPEND.

ATTRIB

Modifica a senha de um arquivo

ATTRIB arquivo (visibilidade, ACC = nome, UPD = nome, PROT = nível)

arquivo é a especificação de arquivo.

visibilidade deve ser I ou N. Diz ao DOS-500 se o arquivo é invisível (I) ou não invisível (N) (Veja DIR). Se omitido, a visibilidade será inalterada.

ACC = nome diz ao DOS-500 a senha de acesso.

Se omitida, ela será inalterada. Se apenas o nome for omitido, (ACC=,) a senha de acesso será representada por espaços em branco (inexistente).

UPD = diz ao DOS-500 a senha para atualização do arquivo, se omitido, não será alterada. Se apenas o nome for omitido (ACC=,) a senha de atualização será representada por espaços em branco (inexistente).

PROT = nível informa ao DOS-500 a senha de acesso. Caso seja omitido, o "nível" permanecerá o mesmo.

Nível Grau de acesso dado à palavra de acesso*
FULL Acesso total, nenhuma proteção.
KILL Supressão (KILL), alteração de nome, leitura, execução e gravação (permite acesso total, isto é, o menos protegido).

NAME Alteração de nome, leitura, execução e gravação
WRITE leitura, execução e gravação.
READ leitura e execução.
EXEC Somente execução.

NOTA
Cada nível permite acesso a ele mesmo e aos níveis inferiores.

O comando ATTRIB permite a mudança das senhas para um arquivo existente e o torna invisível ou não. As senhas são determinadas quando o arquivo é criado. Nesse momento, as palavras atual e de acesso são igualadas a um mesmo valor (por especificação ou por omissão).

EXEMPLOS

```
ATTRIB ARQUIVO (I, ACC = 1N/JULHO,UPD = RATO, PROT = READ)
```

Esse comando torna o arquivo invisível e determina as senhas de acesso (JUNHO/14) e a atualização (RATO). A utilização da senha de acesso possibilitará somente a leitura e a execução do arquivo.

```
ATTRIB FOLHAPAG/BAS,SECRETO (N,ACC=,)
```

Neste caso, a senha de acesso é constituída de espaços em branco. O arquivo é não-invisível e o nível de proteção determinado para senha não é alterado:

```
ATTRIB VELHA/DAT,MACAS (UPD =,)
```

Neste caso, a senha de atualização consiste em espaços em branco.

```
ATTRIB FOLHAPAG./BAS,SN (PROT=EXEC)
```

Aqui, as senhas de atualização e de acesso ficam inalteradas, mas modifica-se o nível de acesso

EXEMPLOS PRÁTICOS

Suponha que você tem um arquivo de dados, FOLHAPAG (FOLHA DE PAGAMENTO) e deseja que seu subordinado o utilize na preparação de cheques de pagamentos, e que ele seja capaz de lê-lo mas não de modificá-lo. Então, você deve usar esse comando:

```
ATTRIB FOLHAPAG (I,ACC=DIAPAG,UPD=ABACATE,PROT=READ)
```

Agora diga a ele para usar a senha DIAPAG (DIA DE PAGAMENTO) (que só permite leitura), enquanto só você sabe a senha (ABACATE), que dá acesso total ao arquivo.

PROTEÇÃO DE PROGRAMA DO BASIC

Você pode dar uma proteção de somente-execução para programas em BASIC, utilizando o comando ATTRIB. Por exemplo, suponha que o programa é denominado TESTE (sem senha).

Na condição

DOS500 ATIVO,

execute este comando:

```
ATTRIB TESTE (ACC=,UPD=VALE,PROT=EXEC)
```

Agora, TESTE tem uma senha de acesso em branco, uma atual (VALE) e um nível de acesso de somente-execução. Só existe um meio de se executar o programa, sem usar a senha atual e é esse:

1 — Acione o BASIC

2 — Digite: `RUN "TESTE"`

(este é o único meio de se ter acesso ao programa. Se o operador tante carregá-lo, o BASIC apagará o programa da memória antes de retornar à condição READY).

Após a execução do comando RUN "TESTE", o BASIC carregará e executará o programa. Se o operador pressionar a tecla BREAK ou o programa finalizar normalmente, o BASIC apagará o programa antes de retornar com a mensagem READY. Isto impossibilita a obtenção de uma listagem do programa a não ser que a senha atual seja usada. Neste caso, você terá acesso completo ao programa.

AUTO

Comando automático após o acionamento do sistema.

AUTO linha de comando

Linha de comando fornece ao DOS-500 um comando ou o nome de arquivo executável de programa criado pelo comando BUILD.

Se a linha de comando por dada, esta será executada o acionamento do DOS-500 ou depois de RESET

Se a linha de comando for omitida, o comando AUTO anterior será apagado do disquete.

AUTO possibilita a execução de um comando sempre que o DOS-500 é ativado (acionamento ou inicialização). Você pode utilizá-lo para obter a execução de um dado programa, sem necessidade de operador, exceto para digitação de data e hora.

Ao se introduzir um comando AUTO, o DOS-500 gravará uma linha de comando em seu procedimento de acionamento. O DOS-500 não verifica se os comandos são válidos; se a linha de comando estiver errada, o erro será detetado na próxima vez que o sistema for acionado.

EXEMPLOS

```
AUTO DIR (SYS)
```

Diz ao computador para executar o comando DIR(SYS) após o procedimento de acionamento.

Toda vez que o sistema for inicializado ou ativado, ele executará automaticamente aquele comando após a introdução da data e da hora.

AUTO BASIC

Diz ao computador para carregar e executar o BASIC, toda a vez que o sistema for acionado.

```
AUTO FORMS (WIDTH = 80)
```

Diz ao computador para reajustar o parâmetro de largura de impressão em 80 caracteres, cada vez que o sistema for acionado.

```
AUTO FOLHAPAG/CMD (PARA FOLHA DE PAGAMENTO)
```

Diz ao computador para carregar e executar FOLHAPAG/CMD (deve ser um programa em linguagem de máquina) após cada acionamento do sistema.

```
AUTO INICIAL
```

Diz ao computador para executar uma seqüência de comandos de um arquivo chamado INICIAL, após cada acionamento do sistema. Veja BUILD e DO.

PARA APAGAR UM COMANDO AUTO

Digite: AUTO ENTER

Isso dirá ao DOS-500 para apagar qualquer comando automático. O comando será apagado quando você ligar, ou inicializar o sistema novamente. A mensagem: AUTO = " " aparecerá na tela depois que o comando for apagado.

Para cancelar um comando AUTO.

Você pode ignorar qualquer comando automático, retende a tecla ENTER enquanto se aperta RESET . Você deve continuar a reter ENTER até a solicitaão da data durante o processo de inicialização.

BUILD

Cria um Arquivo de introdução automática de comando

BUILD arquivo

arquivo é uma especificação de arquivo que não pode conter uma extensão.

Este comando possibilita a criação de um arquivo de introdução automática de comandos que não pode ser executada através do comando DO.

O arquivo deve conter dados que normalmente seriam digitados através do teclado no modo **DOS500 ATIVO**

O objetivo do comando BUILD é transferir as linhas de comando para o DOS500 exatamente como elas foram digitadas na condição **DOS500 ATIVO**

NOTA

CLEAR não pode ser usado num arquivo DO.

Ao se introduzir o comando BUILD, este cria um arquivo e imediatamente solicita a inserção de linhas. Ao completar uma linha, pressione ENTER

Você pode utilizar as teclas normais de controle do cursor para apagar e fazer correções.

Para finalizar o arquivo BUILD, simplesmente pressione a tecla BREAK no começo da linha.

Primeiro digite: **BUILD ARQUIVO**

Então você será solicitado a digitar o texto do comando, que poderá conter até 63 caracteres e logo após você deverá pressionar a tecla ENTER .

Você pode introduzir tantas linhas quanto desejar.

Pressione BREAK para terminar e retornar à condição **DOS500 ATIVO**

Exemplo de BUILD-arquivo.

Aqui apresentamos um comando hipotético BUILD-arquivo que inicializa o interfase serial e o driver de impressora:

```
SETCOM (BAUD = 1200; WAIT)
FORMS (WIDTH = 80)
PAUSE INICIALIZADAS IMPRESSORA E INTERFACE RS-232-C
```

CLEAR

Limpa Memória do Usuário

CLEAR (START=aaaa, END=bbbb, MEM=cccc)

START=aaaa indica onde começar a limpar a memória do usuário. aaaa é um número hexadecimal de 4 dígitos compreendido entre 6000 e o final dessa memória.

Se esta opção for mantida, será usado 6000. Caso contrário, END = bbb também deverá ser utilizado.

END=bbbb indica onde terminar a limpeza dessa memória. bbbb é um número hexadecimal maior que o número inicial e menor que o máximo permitido na memória.

A utilização dessa opção implica no uso do START=aaaa

MEM=cccc determina o endereço de proteção de memória, cccc, é um número hexadecimal de 4 dígitos compreendido entre OOOO e FFFF. O endereço de proteção da memória será determinado como a final da RAM utilizada, se sua opção for omitida.

Se todas as opções forem omitidas, toda a RAM disponível será apagada e a proteção da memória ajustada para o final da mesma, a tela será limpa e todos os drives serão reajustados (Veja. Requisitos de Memória do DOS-500). Este comando proporciona um novo começo; dependendo das operações selecionadas, ele:

1 — Inicializará a memória do usuário (cada endereço da memória acima de 6000 será zerado).
2 — Limpará a tela.
3 — Desprotegerá toda a memória.

Consulte os requisitos de memória do DOS-500 para maiores detalhes sobre endereço de proteção de memória.

NOTA

CLEAR não pode ser utilizado em arquivos DO.

Exemplo:

```
CLEAR (START=9000,END=0A000)
```

NOTA

Números hexadecimais prefixados por uma letra devem ser precedidos por um zero (veja exemplo acima).

```
CLEAR (MEM=7000)
```

CLOCK

Ativa visualização do relógio

CLOCK (chave)

chave - fornece 2 opções (ON-OFF) ao DOS-500.

Na ocasião da opção, será considerado a condição ON (ligado).

Este comando controla a visualização da hora real no canto superior direito da tela.

Se ativado, a hora-minuto-segundo será mostrada em ciclos de 24 horas e haverá um incremento a cada segundo, não importando o programa em execução.

A visualização do relógio está desativada na hora do acionamento do DOS-500.

NOTA

O relógio de tempo real, está sempre funcionando, não importando se sua visualização está ativada ou não (excessão: Durante a entrada e saída de disco e cassete).

Exemplos

```
CLOCK
```
Ativa a visualização do relógio

```
CLOCK (ON)
```
Ativa a visualização do relógio

```
CLOCK (OFF)
```
Desativa a visualização do relógio

Veja TIME e DATE

CLS

Limpa a tela.

Esse comando CLS limpa a tela e o coloca no modo de 64 caracteres/linha.

Exemplo:

```
CLS
```

COPY

Cópia de um ou mais arquivos.

Três formas:

A) **COPY** arquivo-fonte arquivo-destino

arquivo-fonte é uma especificação do arquivo a ser copiado.

arquivo-destino é uma especificação de arquivo do nome e drive do arquivo copiado.

B) **COPY** arquivo-fonte: d

arquivo-fonte (definido-acima).

:d diz ao computador para copiar o arquivo no drive d, usando o mesmo nome de arquivo.

C) **COPY**/EXT:d

/ext é uma especificação de arquivo arbitrária na qual o nome de arquivo é omitido e a extensão é dada. O DOS-500 copiará todos os arquivos que tenham uma extensão compatibilizante não importando o nome de arquivo.

:d (definido acima).

Este comando copia o arquivo-fonte, no novo arquivo, definido pelo arquivo-destino. Ele possibilita a cópia de um disco para outro utilizando um único drive se necessário. (No último caso, deve se incluir especificações de drive em ambas especificações de arquivo). Para sistemas de drive simples (DRIVE 0), ambos os disquetes devem ser do DOS-500. (Isto é, disquetes de dados não são permitidos no DRIVE 0).

Exemplos:
COPY ARQANTIG/BAS ARQNOVO/BAS

Esse comando copia o ARQANTIG/BAS em outro chamado ARQNOVO/BAS. O DOS-500 procurará em todos os drives pelo ARQANTIG/BAS e o copiará no primeiro disco sem proteção de gravação.

```
COPY ARQNOME/TXT:1
```

Este comando especifica o arquivo chamado ARQUIVO DE NOME/TXT para um outro disco.

```
COPY ARQ/EXT:0 :1
```

Este comando copia o ARQ/EXT do drive 0 para o drive 1.

```
COPY /BAS:0 :1
```

Diz ao computador para copiar todos os arquivos do drive 0 que tenham a extensão/BAS. Os arquivos serão copiados no drive 1, usando as extensões e nomes de arquivos presentes.

EXEMPLO PRÁTICO

Use COPY para conseguir a cópia do seu arquivo em outro disco, sempre que seu arquivo for atualizado.

Você também poderá utilizar este comando para reestruturar um arquivo para acesso mais rápido. Certifique-se que o disco destino esteja menos segmentado que o disco-fonte; do contrário o arquivo novo poderá ser mais segmentado que o antigo. (Veja FREE para informações sobre segmentação de arquivos).

Para dar um outro nome ao mesmo arquivo, use RENAME, não COPY.

7

CREATE

Cria um arquivo pré-alocadq.

CREATE nome de arquivo (LRL=aaa, REC=bbb)

nome de arquivo é uma especificação de arquivo.

LRL=aaa é o comprimento lógico de uma gravação. aaa é um número decimal compreendido entre zeo e 255. No caso de omissão do mesmo, será considerado o número 256.

REC=bbb é o número de gravações a armazenar. bbb é o númreo de gravações desejadas.

No caso de omissão do mesmo, nenhuma gravação será alocada.

Este comando possibilita a criação de arquivos e a pré-alocação (separação) de espaço para futuras informações. Ele é diferente do procedimento de omissão (normal) do DOS-500, no qual o espaço é alocado a um arquivo dinamicamente, isto é, de acordo com a necessidade de se escrever dados do arquivo.

Se você abrir o arquivo para gravações seqüências, o DOS-500 recuperará (deslocará) qualquer bloco em desuso, caso o arquivo esteja fechado. Se você abrí-lo para acesso aleatório, O DOS-500 não recuperará espaço, enquanto o arquivo estiver fechado. Você pode usar CREATE para preparar um arquivo que contenha uma quantia conhecida de dados. Geralmente, isto acelera as operações de gravação de arquivos. A leitura de arquivos será também mais rápida pois arquivos pré-alocados são menos segmentados ou dispersos no disco, necessitando de menos movimento do mecanismo de leitura/gravação para localizar as gravações.

Exemplos:

```
CREATE ARQDADOS/BAS (REC=300,LRL=0)
```

Cria um arquivo chamado ARQDADOS/BAS e aloca espaço para 300 gravações de 256 bytes.

```
CREATE NOMES/ TXT.IRIS (LRL= 64,REC= 50)
```

Cria um arquivo chamado NOMES/TXT protegido pela senha IRIS. O arquivo será amplo o bastante para conter 50 gravações, cada uma com 64 bytes de comprimento.

```
CREATE FOLHAPAG/BAS
```

Cria um arquivo FOLHAPAG/BAS não aloca espaço algum nele.

EXEMPLO PRÁTICO

Suponha que você vá armazenar informações sobre até 250 empregados, cada gravação dos dados será desta forma:
NOME (até 25 letras)
CARTEIRA DE IDENTIDADE (11 caracteres)
DESCRIÇÃO DO SERVIÇO (até 92 caracteres)

Sendo assim, sua gravações necessitariam ter 128 bytes (25+11+92) de comprimento.

Você poderia criar um arquivo apropriado com este comando:

```
CREATE PESSOAL/TXT (REC=250,LRL=128)
```

Uma vez criado, este arquivo pré-alocado permitiria gravações mais rápidas do que um arquivo alocado dinamicamente, pois o DOS-500 não teria que parar a gravação periodicamente para alocar mais espaço (exceto se você exceder a quantia pré-alocada).

DATE

Acerta ou obtêm a data.

DATE mm/dd/aa

mm/dd/aa é a especificação do mês (mm), dia (dd) e ano (aa).

Cada termo deve ser um número decial de 2 algarismos compreendidos nesta faixa:

mm	01-12
dd	00-31
aa	00-99

As especificações são opcionais, mas se uma for utilizada, todas deverão ser.

Se mm/dd/aa for omitido, o DOS-500 mostrará a data corrente.

Este comando possibilita o reajuste da data e sua visualização:

Inicialmente, ajusta-se a data ao se acionar o DOS-500. Após esta operação, o DOS-500 atualiza a data automaticamente, utilizando seu calendário interno. Você pode introduzir qualquer ano (aa) a partir de 1900. Quando necessitar de uma data, o DOS-500 visualizará 07/25/82 para 25 de julho de 1982.

EXEMPLOS

```
DATE
```

Mostra a data corrente

```
DATE 07/18/80
```

Reajuste a data em 18 de julho de 1982.

DEBUG

Início de Monitorização de Detecção-Correção de erros.

DEBUG

Este comando aciona a monitorização de detecção-correção de erros, que o permite introduzir, testar, detectar e corrigir programas de linguagem de máquina.

Suas características são:

— Visualização total ou de meia tela dos conteúdos de memória.
— Comando para modificações no conteúdo da RAM e dos registros.
— Execução de programa passo à passo.
— Ponto de interrupção da execução de um programa.
— Transferência de controle (jump).
— Edição de arquivos

DEBUG usa a área de memópria de X'4E00' a X'54FF' (veja Mapa de memória do DOS-500).

Este comando só pode ser utilizado na área de memoria do usuário compreendida entre X'5500 e o topo.

Exemplos

DEBUG

Ativa o comando DEBUG. Pressione Q para suspender a operação e retornar ao DOS-500.

Q

Desativa o DEBUG

Os comandos DEBUG geralmente são introduzidos através de uma tecla. Na maior parte dos casos, não é necessário pressionar ENTER após a digitação do comando. Ou um sinal será imediatamente visualizado ou o comando DEBUG executará a operação sem outra instrução. Em alguns casos você terá que introduzir um endereço ou valor hexadecimal específico (Veja comandos R e J, por exemplo). Ao invés de pressionar ENTER após a digitação do endereço, deverá ser pressionada a barra de espaço.

Uma vez introduzidos o programa de DEBUG, você poderá usar qualquer desses comandos especiais:

D (Visualização do conteúdo da memoria)

Pressione D para mostrar o conteúdo da memória. O DOS-500 responderá com a mensagem:

```
D ENDERECO=
```

Você deve digitar o endereço hexadecimal da locação da memória que deseja ver.
A visualização será total ou de meia tela conforme o formato em uso (veja abaixo).

X (Visualização de meia tela)

Pressionando X coloca-se à mostra somente meia tela ou seja um bloco de 128 bytes será mostrado começando com o próximo endereço múltiplo de 16.

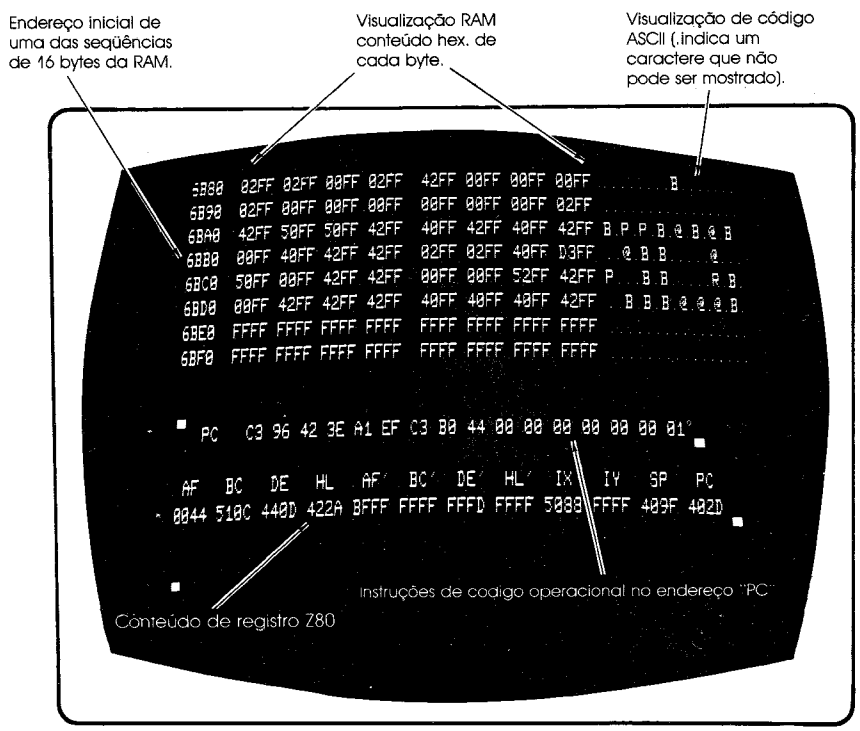


Figura 8 — Formato de meia tela.

S (Visualização tela toda)
Pressione S para visualizar o conteúdo do bloco de 256 bytes de memória iniciando pelo próximo endereço múltiplo de 256.

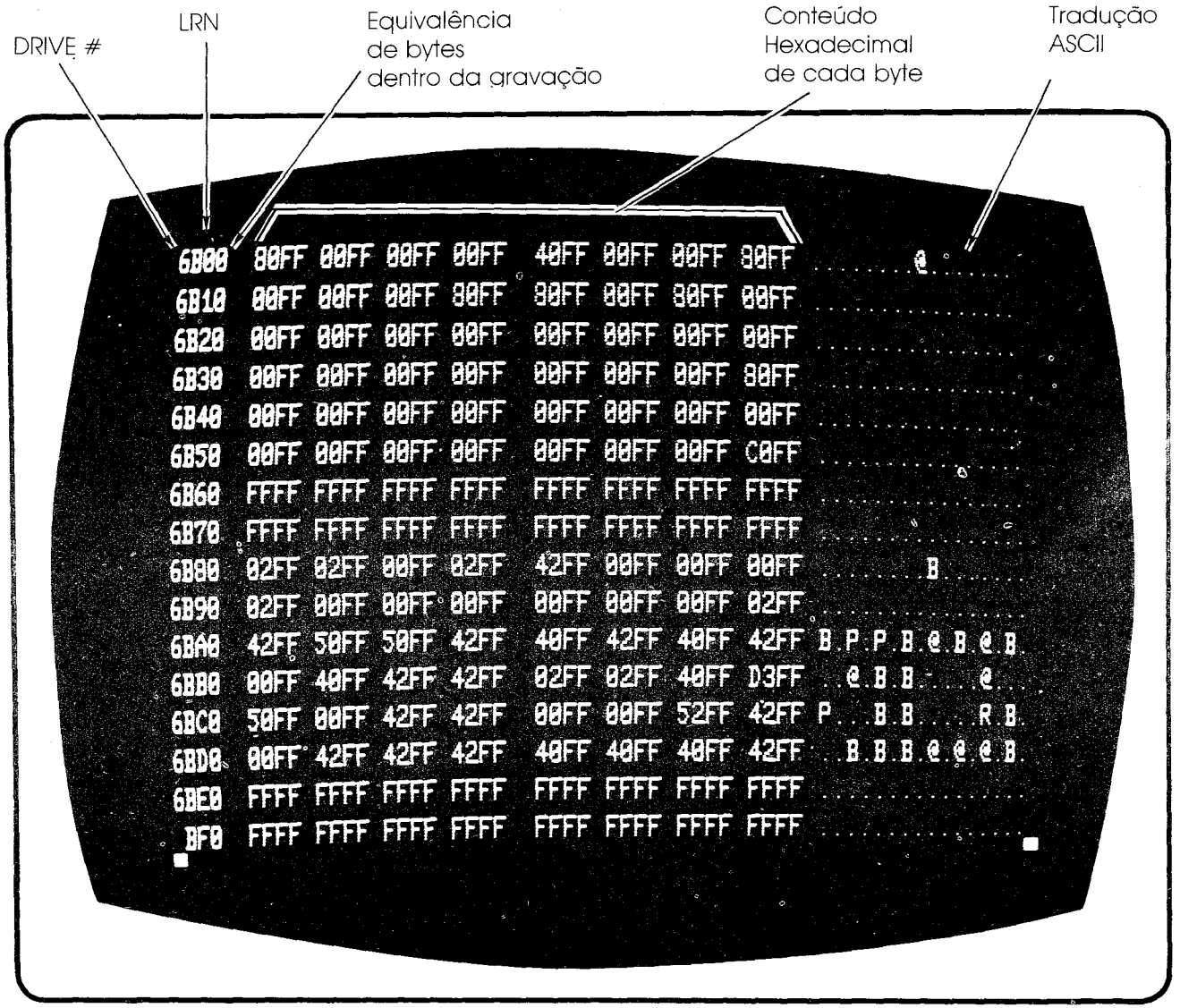


Figura 9 — Formato de visualização total da tela.

NOTA

Os últimos 16 bytes na tela devem ser encobertos por qualquer linha de comando digitada depois que a visualização total de tela tenha sido atualizada.

M (Modificação de RAM)

Pressioner M para transferir para o formato de visualização de utilidade de disco. (Veja comando F). O DOS-500 responderá com um sinal:

M ENDERECO=

Você deve digitar o endereço hexadecimal de 4 dígitos da locação de memória que você deseja modificar, seguido por um espaço em branco (qualquer outra coisa que não seja um espaço abortará o comando).

A tela passará para o formato de edição da memória. O cursor aparecerá como um caractere intermitente na locação especificada.

Para sair do modo de modificação, pressione ENTER para conservar todas as mudanças feitas.

R (modificação do Conteúdo dos Registros)

Digite:

Raa,bbbb **BARRA DE ESPAÇO**

aa é o nome de um dos pares de registros.

AF, BC, DE, HL ou PC

bbbb é o valor hexadecimal de 4 dígitos que será carregado em aa.

Se menos que 4 dígitos forem digitados antes de se pressionar a barra de espaços haverá o acréscimo de zeros a frente do número.

I (Instrução Passo a Passo)

Pressionando a tecla I, você permitirá que o computador execute uma simples instrução Z80. A instrução visualizada será atualizada.

A instrução do conteúdo da memória, que o contador de programa fez referência, é executada. O contador do programa é incrementado pelo valor apropriado e o controle retorna 20 DEBUG.

DEBUG não passará por uma chamada nem entrará num endereço ROM.

C (Chamada Passo à Passo)

Se você deseja completar uma seqüência inteira de chamada/retorno, pressione C.

A chamada será então executada e o controle será mandado de volta para DEBUG, quando a sub-rotina retornar.

De outro modo, esta instrução atua como o comando I.

Você não será capaz de dar um passo através de CALL ou pular para um endereço da ROM

U (Atualização)

Pressionando-se U faz com que a visualização seja atualizada repetidas vezes. Pressione qualquer tecla para sair deste modo.

; (Incremento do endereço de visualização)

Se a visualização for de meia tela, a primeira locação mostrada será incrementada por 16, ao se pressionar a tecla ";".

Se a visualização for total, o endereço inicial será incrementado por 256.

— (Decremento do Endereço de Visualização)

Se a visualização for de meia tela, a primeira locação será decrementada por 16, ao se pressionar a tecla —. Se a visualização for total, o endereço inicial será decrementado por 256.

J (Transferência)

Pressione J para transferir o controle para um programa de linguagem de máquina determinando pontos de interrupção opcionais.

O comando DEBUG responderá com o sinal:

J ENDERECO? =

Você pode digitar um endereço de transferência e outro de ponto de interrupção.

O comando termina ao se pressionar ENTER Digite os endereços em um desses 3 formatos:

J ENDEREÇO? = aaaa, bbbb ENTER
J ENDEREÇO? = aaaa ENTER
J ENDEREÇO? =,bbbb ENTER

aaaa é um endereço hexadecial de 4 dígitos que especifica o destino da transferência. Se omitido, o endereço usado é o do registrador PC

bbbb é um endereço hexadecimal de 4 dígitos que especifica um ponto de interrupção.

Antes da execução de uma instrução nesse endereço, o computador transferirá o controle ao DEBUG.

NOTAS

- Pontos de interrupção devem ser determinados no início de instruções do Z80.
- Não se pode determinar pontos de interrupção em endereços ROM.
- O endereço de ponto de interrupção conterá um X'17' até que o ponto de interrupção seja encontrado. Então o conteúdo original será restaurado e o DEBUG terá novamente o controle.

Q (Parar)

Pressionando-se a tecla Q desativa-se o DEBUG e retorna o controle ao DOS-500.

F (Alteração de arquivos)

Este comando possibilita a carga e a modificação do conteúdo de arquivo de disquete.

Ao se pressionar a tecla F , DEBUG responderá com o sinal;

FILESPEC?

Introduza o nome do arquivo a ser alterado.

O DEBUG formará uma visualização total da tela mostrando os primeiros 256 bytes no arquivo.

Você pode "folhear" o arquivo, utilizando as teclas ; e — . A figura 9 mostra uma visualização típica.

No modo de visualização de arquivo, tanto o código hexadecimal quanto o ASCII são fornecidos para cada byte. Se o código não tiver nenhum caractere visualizável, um ponto será mostrado na ára ASCII.

Os comandos de controle de visualização são similares àqueles no modo normal de visualização de arquivo:

; próxima página
— Página anterior

Para modificação do conteúdo do arquivo, pressione a tecla M . Esta operação o colocará no modo de modificação de memória, como o descrito anteriormente. Utilize as teclas de setas para modificar a posição do curso (um caractere intermitente) e digite o conteúdo correto como um valor hexadecimal. Quando você terminar de modificar uma "página" na sua tela, pressione ENTER e o arquivo de disquete será atualizado e você retornará ao modo de visualização de arquivo.

Para cancelar as mudanças feitas, não pressione ENTER e sim BREAK . Isto colocará de volta ao modo de visualização de arquivo sem atualização do arquivo de disquete. Você pode pressionar ; e então — para recolocar à visualização da página em seu conteúdo real.

Para terminar a alteração do arquivo, pressione BREAK , enquanto estiver no modo de visualização de arquivo. O DEBUG lhe solicitará uma nova especificação de arquivo.

Pressione BREAK e você retornará à condição

DOS500 ATIVO

DIR

Lista o diretório do disquete.

DIR: d(INV,SYS,PRT)
:d é o diretório do drive desejado.
No caso de omissão, será considerado o drive 0.
INV — Lista os arquivos invisíveis do usuário. No caso de omissão, serão listados os arquivos não invisíveis do usuário.
SYS — Lista arquivos de sistema e de usuários. No caso de omissão, apenas os arquivos não invisíveis do usuário serão listados.
PRT — Lista o diretório na impressora. No caso de omissão, o diretório será listado no vídeo somente.
Se nenhuma opção for dada, o DOS-500 listará os arquivos não invisíveis do usuário na tela.
Este comando fornece informações a respeito do disco e dos arquivos nele contidos.
Para paralisar a listagem, pressione @ . Para continuar, pressione ENTER e para finalizar, a tecla BREAK

EXEMPLOS

DIR
Mostra o diretório de arquivos não-invisíveis do usuário do Drive 0.

DIR:1 (PRT)
Lista o diretório dos arquivos do usuário no Drive 1 através da impressora.

Exemplo de listagem do diretório
(Veja figura 10).
Definição de cabeçários de colunas.

1) Nome de Arquivo — O nome e a extensão atribuídos a um arquivo, quando este foi criado. A senha (se houver) não será mostrada.
2) Especificações — um campo de 4 caracteres.
O primeiro caractere é a letra I (Invisível) ou N (Não-invisível).
O segundo caractere é a letra S (Sistema) ou o * asterisco (arquivo-de-usuário).
O terceiro caractere fornece a condição de proteção por senha (figura 10 listagem do diretório):

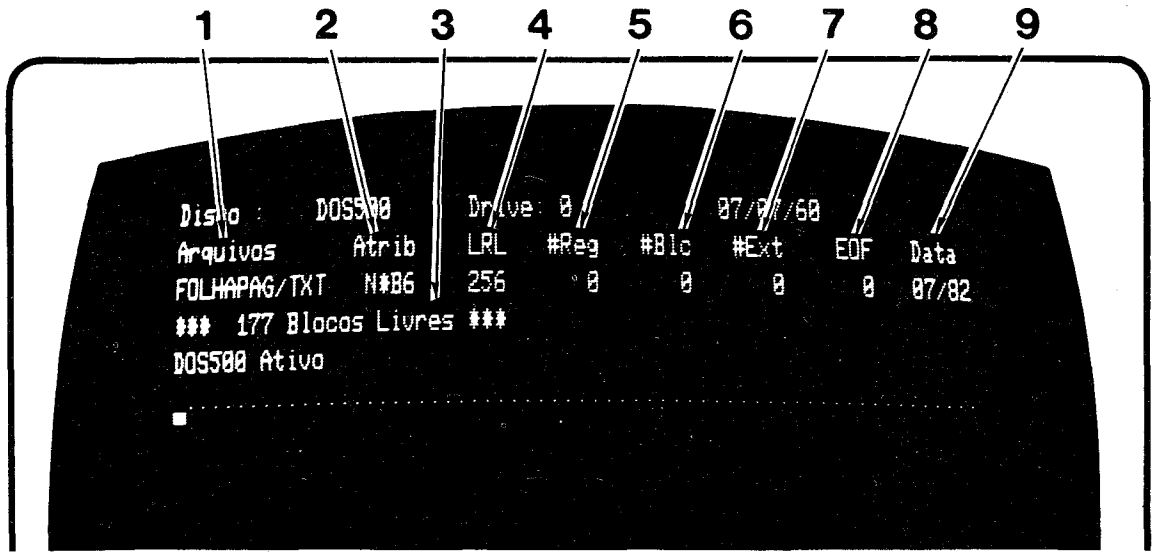


Figura 10 — Listagem do Diretório

X — O arquivo está desprotegido (sem senha)
A — O arquivo tem uma palavra de acesso.
U — O arquivo tem uma palavra atual, mas não tem a palavra de acesso.
B — O arquivo tem ambas.
O quarto caractere especifica o nível de acesso atribuído à palavra de acesso:

0 — Acesso total
1 — Elimine o arquivo e tudo listado abaixo
2 — Dê novo nome ao arquivo e a tudo listado abaixo
3 — Essa designação não é usada
4 — Gravação e tudo listado abaixo
5 — Leitura e tudo listado abaixo
6 — Execução somente
7 — Nenhum acesso

3) Número de blocos livres — Quantos blocos livres permanecem no disquete.
4) Comprimento do registro lógico. Determinado quando o arquivo foi criado.
5) Número de registros — Quantos registros lógicos foram escritas.
6) Números de blocos — Quantos blocos foram usados naquele arquivo.
7) Números de Extensões — Quantos segmentos (áreas contíguas de até 32 blocos) de espaços de disco são colocados ao arquivo.
8) Fim de arquivo (EOF). Mostra o número do último byte do arquivo.
9) Data de criação — Quando o arquivo foi criado.

DO

Inicia a execução dos comandos de um arquivo BUILD.

DO nome do arquivo.

Nome do arquivo é o nome do arquivo criado com BUILD. Não há especificação de extensões, automaticamente, a extensão /BLD será fornecida ao arquivo.

Este comando lê e executa as linhas armazenadas em um arquivo de formato especial criado com o comando BUILD. O sistema executa os comandos como se eles tivessem sido digitados.

As linhas de comando em um arquivo BUILD podem incluir comandos de biblioteca ou especificações de arquivo para programas do usuário.

Ao alcançar o fim do arquivo de execução automática de comandos, ele devolverá o controle ao DOS-500.

Os comandos DEBUG e CLEAR não podem ser incluídos num arquivo BUILD.

Com o comando DO, além de executar comandos de biblioteca do DOS-500, você pode carregar e executar programas de usuário.

Provavelmente, você desejará que o nome de seu programa seja a última linha do arquivo especificado no comando DO.

Exemplos:

```
DO INICIO
```

O DOS-500 começará a execução automática de comandos do arquivo INÍCIO.

```
AUTO DO INICIO
```

O DOS-500 começará a execução automática de comandos do arquivo INÍCIO, assim que você acionar o DOS-500 e responder aos pedidos de data e hora.

EXEMPLO PRÁTICO

Suponha que você deseja iniciar as seguintes funções do DOS-500 automaticamente no acionamento:

```
FORMS (LARGURA = 80)
CLOCK (ON)
```

Então use BUILD para criar esse arquivo.

Se você o chamar de início, use o comando: AUTO DO INÍCIO para executar os comandos cada vez que o DOS-500 for acionado.

DUAL

Cópia saída do Vídeo e na Impressora.

DUAL (chave)

Chave é uma das duas condições, ON ou OFF. No caso de omissão o DOS-500 usará a condição ON.

Este comando copia toda saída de vídeo na impressora e vice-versa. Ele se efetua imediatamente.

NOTAS

1 — As saídas de vídeo e impressora podem ser diferentes devido às diferenças intrínsecas entre os dispositivos de saída e o software para saída.
2 — O uso do comando DUAL implica na diminuição da velocidade do processo de saída de vídeo.
3 — O comando DUAL não pode ser usado durante ROUTE e vice-versa.
4 - A impressora deverá estar preparada quando você executar o comando.

EXEMPLO PRÁTICO

Para se conseguir uma cópia de todo o diálogo sistema/operador, digite:

DUAL ENTER

Para desativar o processo DUAL, digite:

DUAL (OFF) ENTER

DUMP

Armazena um programa em um arquivo de disco.

DUMP arquivo (START=aaaa, END=bbbb, TRA=cccc, RELO=dddd)

arquivo é a especificação do arquivo.

START=aaaa é o endereço inical do bloco de memória. aaaa deve ser um número hexadecimal de 4 dígitos igual ou maior que X'7000':

END=bbbb é o endereço final do bloco de memória. bbbb deve ser um número hexadecimal de 4 dígitos.

TRA=cccc é o endereço de transferência no qual a execução inicia quando o programa é carregado. cccc deve ser um número hexadecimal de 4 dígitos.

Se esta especificação for omitida, o comando assumirá como reentrada para o DOS-500.

RELO=dddd é o endereço inicial para relocar ou carregar o programa de volta à memória. dddd dever ser um número hexadecimal de 4 dígitos. Se esta especificação for omitida, não haverá relocação.

NOTA

Os endereços devem estar na forma hexadecimal, sem a notação X' '.

Você deve acrescentar o prefixo "0" a qualquer número hexadecimal que inicie com uma letra.

Este comando copia um programa em linguagem de máquina da memória para um arquivo de programa. Você pode, desta maneira, carregar e executar o programa a qualquer hora, introduzindo o nome de arquivo no modo **DOS500 ATIVO**

EXEMPLOS

```
DUMP LISTER (START=7000,END=7100,TRA=7004)
```

Cria um arquivo de memória chamado LISTER/CMD contendo o programa nas locações da memória de X'7000' e X'700'0.

Quando carregado, LISTER/END ocupará os mesmos endereços e DOS-500 protegerá a memória a partir do endereço X'7000'.

O programa é exequível no modo **DOS500 ATIVO**

```
DUMP PROG2 (START=7000,END=7F00,TRA=8010,RELO=8000)
```

Cria um arquivo de programa chamado PROG2/CMD contendo o programa em endereços de X'7000' a X'7F00'. Quando carregado, o PROG2/CMD se localizará de X'8000' a X'8F00'. A execução começará em X'8010'.

FORMS

Determina os Parâmetros de Impressão

FORMS (WIDTH=w, LINES=l)

WIDTH=w é um número máximo de caracteres por linha de saída. Se uma linha alcança este comprimento, o DOS-500 introduzirá um retorno do carro para forçar uma nova linha. Se esta especificação for omitida, a largura máxima usada no momento será utilizada.

Para desativar a característica de largura máxima de linha, use WIDTH=255.

O DOS-500 não forçará novas linhas.

LINES=l é o número de linhas por página.

O DOS-500 não utiliza este valor, no entanto o BASIC o usará na computação do deslocamento necessário de páginas para execução ou se LPRINT CHR$(12) for executado.

Se LINES=l for omitido, o valor atual será usado.

Este comando permite a modificação das características de controle do formato de impressão.

Os valores arrumados em caso de omissão são:

largura máxima da linha = 132

linhas/página = 60

FORMS também posiciona a contagem da linha em zero.

EXEMPLOS

Se você estiver usando o formulário 8½ polegadas de largura, sua largura deverá ser provavelmente 80:

```
FORMS (WIDTH=80)
```

Se estiver usando o formulário de 14 polegadas de comprimento, você poderá fazer LINES = 78.

```
FORMS (LINES=78)
```

Esta modificação possibilitará que a instrução do BASIC, LPRINT CHR$(12) avance uma página pelo número correto de linhas.

NOTAS

1 — A largura, que você especifica em WIDTH, é armazenada na posição 16427 na RAM.
As linhas que você especifica em LINES, são armazenadas na posição 16424 da RAM.

2 — A impressora deve estar preparada ao se executar este comando.

FREE

Mostra o mapa do Alocação de Disco.

FREE: d (PRT)

:d é a especificação de Drive.

Se for omitida, o Drive 0 será usado.

(PRT) diz ao DOS-500 para enviar o mapa à impressora.

Se houver omissão o DOS-500 enviará o mapa somente ao vídeo.

Este comando fornece um mapa de alocação de blocos no disquete. (Um bloco, 768 bytes, é a unidade de alocação de espaço).

Ele também mostra a localização do diretório e de qualquer setor defeituoso.

Quando um disquete é muito usado (atualização, eliminação e extensão de arquivos etc.), os arquivos se tornam segmentados (dispersos ou fragmentados). Isto aumenta o tempo de acesso, uma vez que o mecanismo de gravação e leitura de disco deve retroceder e avançar no disquete para ler ou gravar um arquivo.

O comando FREE o auxilia a determinar exatamente como seus arquivos estão segmentados.

Se você decidir reorganizar um arquivo, para permitir um acesso mais rápido, você deverá copiá-lo em um disquete relativamente limpo.

EXEMPLOS

`FREE`

Mostra um mapa de espaço livre do disquete no DRIVE 0.

`FREE (PRT)`

Lista o espaço livre do DRIVE 0 através da impressora.

`FREE:1 (PRT)`

Lista o mapa do DRIVE 1 através da impressora.

Uma visualização típica do comando FREE

Quatro símbolos especiais são usados no mapa FREE.

•	bloco não utilizado
Direto	Informação do diretório
X	bloco alocado
ruim	bloco que contém setores defeituosos (sem possibilidade de uso).

Uma visualização de um mapa típico de FREE é mostrado na figura 11.

HELP

Explicação do comando do DOS-500.

HELP comando

Comando é o comando ou assunto específico do DOS-500 em que você necessita ajuda.

Se for omitido ou se um assunto inválido for dado, o DOS-500 listará todos os assuntos disponíveis.

EXEMPLO

Se você digitar: `HELP BACKUP` ENTER

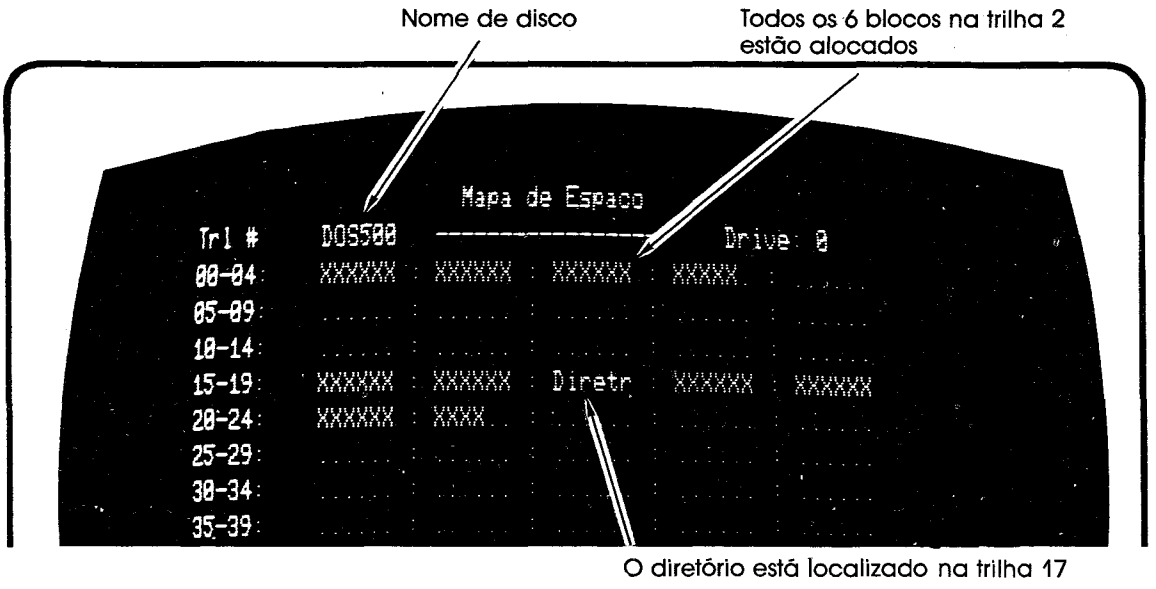


Figura 11 Mapa de memória (FREE)

O DOS-500 responderá com o formato do comando BACKUP, uma definição do comando e uma explicação das abreviaturas.

HELP ENTER diz ao DOS-500 para fornecer a lista dos comandos HELP.

KILL

Apaga um arquivo ou um grupo de arquivos.
Duas formas:

a) **KILL** arquivo
arquivo é uma especificação de arquivo.

b) **KILL/** ext:d
/ext é uma extensão do arquivo que deve conter 3 caracteres.
d é uma especificação do DRIVE, que deve ser fornecida.

Este comando apaga um arquivo ou um grupo de arquivos, dependendo da forma utilizada. A forma (a) apaga o arquivo especificado. Se nenhuma especificação for fornecida, o DOS-500 apagará o arquivo do primeiro disquete que o contiver.

A forma (b) apaga todos os arquivos com a extensão especificada, sem considerar os seus nomes de arquivo.

Se nenhuma especificação de drive for fornecida, os arquivos serão apagados do primeiro drive que contiver a extensão especificada.

EXEMPLOS

```
KILL TESTPROG/BAS
```

Apaga o arquivo mencionado do primeiro drive que o contiver.

```
KILL JOBFILE/IDY.SENHA:1
```

Apaga o arquivo citado do drive 1. A senha do arquivo é SENHA.

```
KILL/BAS:0
```

Apaga todos os arquivos com a extensão /BAS do drive 0.

LIB

Mostra os Comandos de Biblioteca

LIB

Este comando lista na tela todos os comandos de biblioteca. Se precisar de auxílio, utilize o comando HELP.

EXEMPLO:

```
LIB
```

LIST

Lista o conteúdo de um arquivo

LIST arquivo (PRT, SLOW, ASCII)

arquivo é uma especificação de arquivo.

PRT diz ao DOS-500 para fazer a listagem através da impressora. Se for omitido somente será usada a visualização através do vídeo.

SLOW diz ao DOS-500 para fazer uma breve pausa após cada registro.

Se for omitido, a listagem será contínua.

ASCII diz ao DOS-500 para listar o arquivo no formato ASCII. Se for omitido, será utilizado o formato hexadecimal.

Essa rotina lista o conteúdo de um arquivo. A listagem mostra o conteúdo tanto em hexadecimal como em caracteres ASCII, correspondentes a cada valor.

Será visualizado um ponto para valores fora da faixa (X'20' — X'7F').

Utilize a especificação ASCII para arquivos de texto e programas BASIC armazenados com a especificação A.

NOTA

Somente os códigos ASCII de X'00' a X'7F' são enviados á impressora.

O bit 7 sempre é ajustado em O

Durante a listagem, pressione @ para obtenção de uma pausa, ENTER para continuar e BREAK para sair do comando.

EXEMPLOS

```
LIST DADOS/TXT (ASCII)
```

Lista o conteúdo de DADOS/TXT no formato ASCII

```
LIST ARQUIVO/A (SLOW)
```

Lista o conteúdo do ARQUIVO/A com pausas após cada registro.

```
LIST PROGRAMA/CMD (PRT)
```

Lista o arquivo PROGRAMA/CMD através da impressora

LOAD

Carrega um arquivo de programa Z-80

LOAD arquivo
arquivo é uma especificação de arquivo para um arquivo criado pelo comando DUMP.

Este comando carrega na memória um arquivo de programa de linguagem de máquina.

Após a carga do arquivo, o DOS-500 retorna ao modo DOS500 ATIVO.

Você não pode utilizar esse comando para carregar um programa BASIC ou qualquer arquivo criado pelo BASIC. Consulte o Manual de Referências do BASIC para instruções a respeito da carga de programas do BASIC.

NOTA

O arquivo será carregado na área do usuário (X'7000' até o TOPO)

EXEMPLOS

```
LOAD FOLHAPAG/PT1
```

EXEMPLO PRÁTICO

Geralmente, vários módulos de programas devem ser carregados na memória para uso de um programa mestre. Por exemplo, suponha que FOLHAPAG/PT1 e FOLHAPAG/PT2 sejam módulos e MENU seja o programa mestre. Você poderia utilizar os comandos:

```
LOAD FOLHAPAG/PT1

LOAD FOLHAPAG/PT2
```

para armazenar os módulos na memória e então digitar: MENU para carregar e executar MENU.

MASTER

Determina o Drive Mestre de leitura/gravação.

MASTER (DRIVE=a).
a é a especificação do drive. No caso de omissão o drive 0 será considerado mestre.

Este comando possibilita a especificação de um certo drive como sendo drive mestre de leitura ou gravação.

Se um arquivo não for encontrado no drive especificado, o DOS continuará a procurá-lo nos drives de numeração superior.

EXEMPLO

Após a introdução do comando: `MASTER (DRIVE=1)`, o Drive 1 se torna mestre.

PATCH

Modifica o conteúdo do arquivo do Disco.

PATCH ARQUIVO (ADD=aaaa, FIND=bb, CHG=cc)

ARQUIVO arquivo é a especificação do arquivo.

ADD=aaaa especifica o endereço no qual a informação se encontra. aaaa é um número hexadecimal de 4 dígitos.

FIND=bb especifica o string que você deseja achar (ou comparar). bb é uma seqüência hexadecimal.

CHG=cc especifica o novo conteúdo dos bytes. cc é uma seqüência hexadecimal.

NOTA

Esta utilizade se refere somente a programas em linguagemd e máquina.

Este comando permite pequenas correções em qualquer arquivo de disco, considerando-se que:

1 — Você conhece o conteúdo existente e a locação de dados que deseja modificar.

2 — Você deseja substituir um string de código ou dados por outra de mesmo comprimento.

Você pode utilizar PATCH para pequenas modificações em seus próprios programas de linguagem de máquina; você não será obrigado a mudar o código fonte, re-assemblá-lo ou recriar o arquivo.

Outra aplicação para o comando PATCH é possibilitar a complementação de quaisquer modificações no DOS-500 que poderão ser fornecidas pela PROLÓGICA. Deste modo, você não precisa esperar por um novo lançamento de um sistema operacional.

EXEMPLO PRÁTICO

Suponha que você deseja modificar 7 bytes em um arquivo de programas de linguagem de máquina. Primeiro, determine onde a seqüência de 7 bytes se localiza na RAM quando o programa é carregado.

Então, certifique-se que o comprimento de sua string é o mesmo da original.

Por exemplo: você deve escrever a informação deste modo:

Arquivo a ser modificado: VREAD

Endereço inicial: X'5280'

Sequência de códigos a ser modificada:
X'CD2D25E5'

Código de substituição: X'00000009'

Assim, você pode usar o seguinte comando:

```
PATCH VREAD (ADD=5280,FIND=0CD2D25E5,CHG=00000009)
```

PAUSE

Interrompe a Execução para Ação do Operador

PAUSE mensagem
mensagem é a mensagem a ser visualizada durante a pausa na execução

Ela é opcional. Se for omitida, PAUSE será visualizado sozinho:

Este comando é destinado a uso interno de um arquivo DO, assim sendo o DOS-500 pode imprimir uma mensagem ou lembrete.

Para continuar a execução após a pausa, o DOS-500 lhe dará a mensagem:

```
PRESS <ENTER> PARA CONTINUAR
```

EXEMPLO

```
PAUSE INSIRA DISQUETE # 21
```

O DOS-500 mostra PAUSE e a mensagem, e logo após aguarda você pressionar ENTER para continuar a execução.

Veja BUILD e DO para exemplos práticos.

PROT

Utiliza ou modifica a senha mestra de um disquete

PROT :d (PW, LOCK)

:d é uma especificação de drive opcional. Quando omitida, será usado o drive 0.

PW diz ao DOS-500 que você deseja modificar a senha mestra.

LOCK diz ao DOS-500 para atribuir a senha mestra a todos os arquivos de usuário desprotegidos. Se for omitido, tais arquivos desprotegidos assim permanecerão.

PROT possibilita o uso da senha mestra para proteger de uma vez, todos os arquivos desprotegidos, ou modificar a senha mestra.

A senha mestra será necessária na cópia de um disquete; sendo assim certifique-se de não esquecê-la!

NOTA

A senha mestra no disquete do DOS-500 recém fabricado é SENHA.

EXEMPLOS

```
PROT:0 (PW)
```

Diz ao DOS-500 para modificar a senha mestra do disquete no DRIVE 0.
O DOS-500 aguardará primeiramente a senha mestra antiga e depois a nova.

```
PROT:1 (LOCK)
```

Diz ao DOS-500 para atribuir a senha mestra a todos os arquivos desprotegidos do usuário. O DOS-500 aguardará você digitar a senha mestra.

PURGE

Apaga arquivos

PURGE :d (tipo de arquivo)

:d é o drive que contém o disco a se limpo.

tipo de arquivo deve ser um dos seguintes:

SYS todos os arquivos do Sistema e de Usuários (não Invisíveis)

INV todos os arquivos Invisíveis e de Usuários (menos do Sistema).

ALL todos os arquivos em disco (Usuário, Sistema, Invisível)

Se o tipo de arquivo for omitido, o DOS-500 assumira como os arquivos de usuários.

Este comando possibilita a limpeza rápida de arquivos de um certo disquete. Para utilizar PURGE, você deve saber a sua senha mestra. (Os disquetes do sistema DOS-500 tem a senha SENHA).

O DOS-500 solicitará a senha do disquete, ao se introduzir o comando. Digite até 8 caracteres. Pressione ENTER, se você digitiou menos que 8. O sistema mostrará na tela nomes de arquivos de usuários de uma só vez, fornecendo um sinal para eliminar ou abandonar cada um dos arquivos.

EXEMPLO:

PURGE:1

O DOS-500 apagará os arquivos de usuário presentes no drive 1. Isto incluiria os programas BASIC.

```
PURGE :1 (INV)
```

O DOS-500 apagará todos os arquivos invisíveis no DRIVE 1.

NOTA

Os disquetes de sistema contém alguns arquivos que não são mostrados em nenhuma das listagens do diretório.

Você pode apagá-los com uma forma especial de PURGE:

PURGE* : d (tipo de arquivo)

O asterico diz ao DOS-500 para lhe perguntar se deseja que os arquivos de sistema sejam apagados.

Se você realmente os apagar, o disquete se tornará um disquete de dados e poderá ser usado nos drives 1, 2 ou 3.

As outras partes deste comando são como as explicadas anteriormente. Entretanto, certifique-se de executar o PURGE utilizando os DRIVES 1, 2 ou 3, uma vez que o disquete se tornará sem sistema, durante o PURGE.

RELO

Modifica o local de carga do programa na memória.

RELO arquivo (ADD=aaaa)

arquivo é uma especificação de arquivo.

ADD = aaaa especifica o novo endereço de carga. aaaa é um número hexadecimal de 4 dígitos que indica um endereço na memória do usuário. aaaa deve ser um endereço de memória do usuário da RAM.

Este comando possibilita a modificação do endereço no qual o programa carrega na memória. Ele não muda o programa em si.

NOTA

Este comando pode ser útil junto com o DUMP.

Exemplo:

```
RELO PROGRAMA/CMD (ADD = 6578)
```

O DOS-500 irá carregar o programa PROGRAMA/CMD no novoendereço da memória (6578).

RENAME

Troca o nome de um arquivo

RENAME nome antigo nome novo.

nomo antigo é o novo nome do arquivo.

O nome antigo do arquivo pode ainda conter uma especificação de drive e/ou uma senha.

O novo nome do arquivo não deve conter nem especificação de drive nem senha.

Este comando possibilita a redenominação de um arquivo ou de um programa.

Somente o nome/extensão é modificado; os dados no arquivo e sua locação física são inalterados.

RENAME não pode ser usado para modificar uma senha de proteção de senha de um arquivo. Utilize ATTRIB para isso.

RENAME também verifica se o novo nome desejado não copia um nome de arquivo presente no mesmo disquete.

Se a verificação acusar essa situação, o comando será cancelado e uma mensagem de erro será mostrada.

EXEMPLOS:

```
RENAME MATEMPAC MATEMPAC/BAS
```

Diz ao DOS-500 para acrescentar a extensão ao nome de arquivo.

```
RENAME ABCDE/DAT ABCDEF/DAT
```

Diz ao DOS-500 para modificar somente o nome de arquivo

```
RENAME FOLHAPAG1/TXT.GSR FOLHPAG2/TXT
```

Diz ao DOS-500 para modificar o nome de arquivo: a senha é retida automaticamente.

```
RENAME ARQUIVO:3 ARQUIVO2
```

Diz ao DOS-500 para modificar o nome de arquivo no DRIVE 3.

ROUTE

Mudança de Dispositivos de E/S

ROUTE (SOURCE=aa, DESTIN=bb)

SOURCE=aa, especifica o equipamento de E/S de origem.
DESTIN=bb especifica o equipamento de E/S de destino (DESTINation).
aa e bb podem ser quaisquer combinações das seguintes abreviaturas:
DO (tela)
PR (Impressora)
KB (Teclado)
RI (Entrada RS-232)
RO (Saída RS-232)

Se as opções SOURCE e DESTIN forem omitidas, o DOS-500 posicionará os drivers de E/S em seus equipamentos originais. Os equipamentos em SOURCE e DESTIN devem ser ambos de entrada ou ambos de saída.

Este comando permite que você mude os equipamentos de E/S automaticamente.

Por exemplo: o DOS-500 pode enviar para a tela (DO é o destino) uma informação que normalmente iria para a impressora (no caso, PR é a origem).

EXEMPLO:

```
ROUTE (SOURCE=PR,DESTIN=DO)
```

O DOS-500 mudará a saída da impressora para a tela.

```
ROUTE
```

faz com que os equipamentos voltem aos seus estados originais.

Para maiores detalhes sobre mudança de E/S veja "Alterando Entradas/Saídas" no Manual do CP-500.

SETCOM

Inicialização de comunicações RS-232-C

SETCOM (OFF,WORD=a,BAUD=b,STOP=c,PARITY=d,mode)

OFF desliga o RS-232-C

WORD=a o número de bits por byte desejado. O número a pode ser 5,6,7 ou 8, dependendo da necessidade. Se omitido, o comprimento da palavra não será mudado.

BAUD=b especifica a velocidade de transferência (taxa baud). O valor de b pode ser um número múltiplo de dez entre 50 e 9600. Se omitido, a taxa baud não será alterada.

STOP=c especifica o número de bits de parada. O valor de c deve ser 1 ou 2. Se omitido, os bits não serão alterados

PARITY=d determina se a paridade é ímpar, par ou nenhuma. Pode ser 1 (ímpar), 2 (par) ou 3 (nenhuma). Se omitido, a paridade não será alterada.

mode = digite WAIT para o modo de "espera"; e NOWAIT para o modo de "não-espera".

As opções devem ser introduzidas na ordem mostrada.

Se todas as opções forem omitidas, o DOS-500 mostra as condições atuais.

Este comando inicializa as comunicações RS-232-C via canal serial. Antes de executá-lo, deve-se conectar o equipamento de comunicação ao CP-500.

Veja o Manual do CP-500 a descrição dos sinais RS-232-C usados.

Veja, também, "Usando a Interface RS-232-C" no mesmo manual.

EXEMPLOS:

```
SETCOM (WORD=7, BAUD=300, STOP=1, PARITY=3, WAIT)
```

colocará o RS-232-C com palavras de 7 bits, 300 baud, um bit de parada, sem paridade, e no modo WAIT.

```
SETCOM
```

sem especificação mostrará as condições atuais.

O programa a seguir permitirá a você usar seu computador como um terminal.

Para maiores informações, leia a seção de Operação do seu Manual de Operação.

7

NOTA

Este programa opera a 300 bauds.

```
5 DEFINT A-Z                'VARIAVEIS INTEIRAS PARA VELOCIDADE
10 POKE 16890,0             'NAO ESPERA I/O PELO RS-232-C
20 POKE 16888,(5*16)+5      'BAUD RATE DE TRANSMICAO/RECEPCAO = 300
40 X=USR0(0)
60 DEFUSR1 = &H50           'END. CHAMADA DA ROTINA DE RECEPCAO ($RSRCV)
65 DEFUSR2 = &H55           'END. CHAMADA DA ROTINA DE TRANSMICAO
70 CI = 16872               'BUFFER DE INPUT(RECEPCAO)
80 CO = 16880               'BUFFER DE OUTPUT(TRANSMISSAO)
90 'VERIFICA ENTRADA PELO RS-232-C
110 X=USR1(0)               'CALL ($RSRCV)
120 C$=CHR$(PEEK(CI))       'VERIFICA BUFFER DE INPUT
130 PRINT C$                        'SE C$ = 0 NADA ACONTECE
140 'VERIFICA ENTRADA PALO TECLADO
150 C$ = INKEY$
160 IF C$ = "" THEN 110 'SE NADA DIGITADO, VERIFICAR RS-232-C
165 PRINT C$                'ECO DO TECLADO
170 POKE CO,ASC(C$)         'COLOCA CARATER NO BUFFER DE SAIDA
190 X=USR2(0)               'CALL $RSTX
200 GOTO 110                'VERIFICAR RS-232-C
```

TAPE

Transferência Fita/Disco

TAPE (S=fonte, D=destino)
fonte e destino são abreviaturas dos equipamentos de armazenamento a serem usados.
T fita
D Disco
R memória RAM

NOTA

TAPE só pode ser usada com programas em linguagem de máquina.

Programas em BASIC devem usar CLOAD e CSAVE.

Este comando transfere programas Z-80 em linguagem de máquina e um equipamento para outro. As seguintes transferências são possíveis:
Da fita para o disco
Do disco para a fita
Da fita para a RAM
EXEMPLO:

```
TAPE (S=T,D=D)
```

Inicia a transferência fita-disco. O DOS-500 vai questionar você com CASS?

Selecione a velocidade desejada de transferência para a fita (A para alta, B para baixa); o DOS-500 aguardará você pressionar ENTER quando o gravador estiver pronto para operar.

NOTA

Se os asteriscos na tela não piscarem, o volume do gravador deve ser ajustado ou a velocidade de transferência alterada.

O DOS-500 lerá o nome de arquivo na fita e o usará no arquivo do disco.

Ele copiará o programa no primeiro disquete que estiver desprotegido para gravação, começando pelo drive master (veja MASTER).

```
TAPE (S=D,D=T)
```

Inicia uma transferência disco-fita. O DOS-500 questiona você sobre a velocidade de transferência, então, sobre a especificação do arquivo no disco. Então você deve pressionar ENTER quando o gravador estiver pronto para receber o arquivo.

```
TAPE (S=T,D=R)
```

Inicia uma transferência fita-RAM. O DOS-500 questionará sobre a velocidade de transferência e indicará que você deve pressionar ENTER quando o gravador estiver pronto para enviar os dados. Após carregar o programa, o DOS-500 começará a execução a partir do endereço de transferência contido na fita.

TIME

Acerta ou Obtém As Horas

TIME hh:ms:ss especifica as horas hh, os minutos mm e os segundos ss.
Cada um deve ser um número inteiro dentro das seguintes faixas:

hh 0 a 23
mm 0 a 59
ss 0 a 59

Se hh:mm:ss for dado, o DOS-500 acerta as horas.
Se for omitido, será mostrada a hora certa na tela.
Este comando permite a você acertar ou ver as horas.

TIME usa o sistema de 24 horas. Por exemplo, 1:00 da tarde é mostrada 13:00.
Você deve acertar as horas quando ativar o DOS-500. A partir daí, o DOS-500 atualizará a hora automaticamente, através de um relógio interno.
Quando você pede a hora, o DOS-500 a fornece no formato: hh:mm:ss
Exemplo:

```
TIME
```

Fornece a hora atual.

```
TIME 13:20:00
```

Acerta a hora.

NOTA

Quando a hora passar de 23:59:59 ela será zerada, a data incrementada e a contagem continuará.

WP

Proteção Por Software Contra Gravação

WP (DRIVE=d)
O número d especifica o drive que se quer proteger. Se for omitido, todos os drives ficarão desprotegidos.

Com este comando pode-se proteger o conteúdo gravado nos discos.
É uma proteção por software e não por hardware, ou seja, a proteção é feita por programação e não por um dispositivo físico como é o caso do selo de proteção.
Apenas um drive é protegido por vez.
Para desproteger um drive, fazendo-o acessível à gravação, simplesmente entre com o comando WP sem opções ou com um número de drive diferente. O comando WP não desativa a proteção física (selo de proteção).

EXEMPLOS

```
WP (DRIVE=1)
```

O DOS-500 protegerá contra gravação o disco no drive 1.

```
WP
```

O DOS-500 eliminará a proteção de todos os drives.

BASIC-DISCO

PRIMEIRO PASSO

Estando em DOS-500 ATIVO, digite:

BASIC ENTER

O DOS-500 carregará o BASIC e começará o "diálogo inicial".

Se quiser recuperar um programa BASIC do disco, depois de ter estado no modo DOS-500 para um diretório ou outro comando DOS-500, use este comando quando em DOS500 ATIVO:

BASIC * ENTER

Desta maneira, você colocará o sistema diretamente em BASIC READY, sem passar pelo diálogo inicial. Se tiver um programa na memória, ele não se perderá, porém pode ser que você não possa rodá-lo. Para ter certeza de poder fazê-lo, grave o programa em um disco, passe para DOS-500 ATIVO, e recomece o modo BASIC (sem asterisco).

NOTA

Se você ultrapassar os limites da memória do usuário quando em DOS-500, seu programa será apagado. Nesse caso, você não poderá reiniciar o BASIC, mas poderá usar o procedimento normal descrito no item "primeiro passo", acima.

INICIALIZAÇÃO

Quando você começa o BASIC do disco, a primeira pergunta que deve responder é Cass?, como no BASIC Residente. Em seguida virá:

Quantos Arquivos? Isto permite que você especifique quantos arquivos estarão "abertos" ou em uso ao mesmo tempo (veja OPEN). Digite o número apropriado e pressione ENTER, ou simplesmente pressione ENTER e o DOS-500 estabelecerá automaticamente três arquivos como resposta.

Por exemplo, se o seu programa necessita de um arquivo de entrada e um de saída, você deve especificar dois arquivos.

NOTA

Normalmente o BASIC-DISCO faz todos os seus arquivos de dados com um comprimento de 256 (vide "Técnicas de Acesso de Arquivo").

Se você quiser especificar o comprimento de cada arquivo individualmente, use o sufixo V para identificar "variável". Por exemplo:

```
QUANTOS ARQUIVOS? 3V ENTER
```

faz com que o BASIC forneça 3 buffers de arquivo e permita que você selecione o comprimento de gravação de cada um na ordem em que forem sendo abertos.

O BASIC-DISCO cria automaticamente um buffer para carregar, salvar e combinar programas em BASIC. Este buffer existe na RAM antes dos buffers de arquivo de dados que você venha a requisitar. Ele está sempre disponível para E/S de programas, a despeito de como você responder a questão sobre os Arquivos.

Após responder a pergunta sobre os arquivos, o BASIC-DISCO perguntará:
Mem. Usada? Simplesmente pressione ENTER sem digitar qualquer número. Você terá então toda a capacidade disponível de RAM para uso em BASIC.

Se você pretender carregar programas ou rotinas em linguagem de máquina, terá que proteger sua memória BASIC destes programas em linguagem de máquina. Neste caso, responda com o endereço mais alto de memória (na forma decimal) que você quer que o BASIC-DISCO use para armazernar e executar seus programas em BASIC. Endereços acima do especificado serão então protegidos contra o uso pelo BASIC-DISCO.

EXEMPLO:
Mem. Usada? 32000 ENTER

faz com que o BASIC-DISCO proteja os endereços acima de 32000. Se você tem 16K de RAM, isto significa que sobram 32767 — 32000 = 767 bytes protegidos para armazenar suas rotinas em linguagem de máquina.

Após responder a questão Mem. Usada?, o BASIC-DISCO vai mostrar as seguintes informações:
1. Qual a versão do BASIC-DISCO você está usando.
2. Informação sobre direitos autorais.
3. O número de bytes disponíveis.
4. O número de arquivos simultâneos que você requisitou.

Para sair do BASIC-DISCO e retornar ao DOS-500, sem ter que recomeçar o sistema, digite:

CMD"S" ENTER

Isto fará com que o sistema passe para DOS-500, sem a necessidade do diálogo inicial. Você pode recuperar seu programa, se não tiver alterado a memória do usuário enquanto estava no modo DOS-500. Para retornar para o BASIC-DISCO, use BASIC*

ARMAZENAGEM DE UM PROGRAMA

Duas condições: Você deve ter um programa na memória e estar no BASIC.
Então digite SAVE "PROGRAMA" ENTER

O BASIC armazenará o programa no arquivo de disco que foi arbitrariamente denominado "PROGRAMA". Qualquer outro nome também seria aceito.

CÓPIA DE UM PROGRAMA

A título de exemplo, nesta seção copiaremos o programa recém armazenado. Primeiro, digite: NEW ENTER para apagá-lo da memória (este procedimento se destina a comprovar a recuperação do programa através do arquivo de disco).

Agora digite: LOAD "PROGRAMA" ENTER e o BASIC copiará o programa especificado. Você agora pode listá-lo e rodá-lo.

Para maiores detalhes a respeito do uso do BASIC-DISCO consulte a seção 3 deste manual.

AJUSTE DA TAXA BAUD (BAUD RATE) DO CASSETE EM BASIC-DISCO

O DOS-500 coloca em alta a taxa baud de transferência para fita cassete. Se você desejar modificar esta condição, use o seguinte comando:

```
PATCH BASIC / CMD (ADD = 5202, FIND = 00, CHG = FF) ENTER
```

Em conseqüência disso, o computador perguntará:
CASS? sempre que você iniciar o BASIC-DISCO. Neste ponto, você deverá digitar A para alta ou B para baixa, na escolha da relação.

Para reverter o sistema disquete ao seu valor original, simplesmente, use o comando PATCH do DOS-500 novamente, mas desta vez invertendo os valores de FIND e CHG. Se não resolver o problema, leve a PROLÓGICA e nós o ajustaremos e o devolveremos em tempo hábil.

CARACTERÍSTICAS ESPECIAIS DO BASIC—DISCO

O BASIC-DISCO possui muitas características que não estão relacionadas com o BASIC-RESIDENTE (aquele que é interno ao CP-500); Estas características estão relacionadas abaixo com descrições resumidas. Descrições mais detalhadas são dadas a seguir, em ordem alfabética.

&H	Prefixo de constante hexadecimal
&O	Prefixo de constante octal
Abreviaturas	Muitos comandos possuem abreviaturas
CMD"A"	Retorna para o DOS-500 com uma mensagem de erro
CMD"B"	Ativa/Desativa a tecla BREAK
CMD"C"	Elimina espaços e notas em um programa (compressão)
CMD"D"	Mostra o diretório de um drive específico
CMD"E"	Mostra o erro anterior do DOS-500
CMD"I"	Executa em BASIC um comando do DOS-500
CMD"J"	Converte data de calendário
CMD"L"	Carrega sub-rotinas Z-80 ou arquivos de programas na RAM
CMD"O"	Coloca em ordem alfabética uma matriz string
CMD"P"	Verifica as condições da impressora
CMD"R"	Ativa o relógio na tela
CMD"S"	Retorno normal para o DOS-500 (salta para a rotina EXIT)
CMD"T"	Desliga o relógio da tela
CMD"X"	Fornece a listagem com referência das palavras chave, variáveis string ou strings em um programa
CMD"Z"	Duplica a saída para tela e impressora
DEFFN	Define função de instrução BASIC
DEF USR	Define o ponto de partida para uma rotina externa em linguagem de máquina
INSTR	Função "Instring"; encontra uma sub-rotina dentro de uma string dada
LINE INPUT	Introduz uma linha pelo teclado
MID$ =	Substitue uma parte do string dado (colocado à esquerda do sinal de igual)
NAME	Renumera um programa na RAM
USRn	Chama rotina externa (n = 0,1,2,..., 9)

&H E &O (CONSTANTES OCTAL E HEXADECIMAL)

Muitas vezes é preferível usar constantes hexadecimais (base 16) ou octais (base 8) no lugar de suas equivalentes decimais. Por exemplo, endereços de memória e valores de bytes são mais fáceis de manipular no formato hexadecimal.

&H e &O permitem que você introduza estas constantes no seu programa.

&H e &O são usados como prefixos para os numerais que os seguem imediatamente:

&Hdddd

dddd é uma seqüência de 1 a 4 dígitos composta de numerais hexadecimais, que são: 0,1,2,3,...,9,A,B,...F

&Oddddd

ddddd é uma seqüência de numerais octais 0,1,2,...,7 e não pode ultrapassar o valor 177777 decimal.

NOTA

A letra "O" pode ser omitida. Portando, &Oddddd = &ddddd.

As constantes sempre representam inteiros positivos ou negativos. Portanto, qualquer número hexa maior que &H7FFF, ou octal maior que &O77777, será interpretado como um valor negativo. A tabela a seguir ilusta isto:

Octal	Hexa	Decimal
&1	&H1	1
&2	&H2	2
&77777	&H7FFF	32767
&100000	&H8000	—32768
&100001	&H8001	—32767
&100002	&H8002	—32766
&177776	&HFFFE	—2
&177777	&HFFFF	—1

Constantes hexa e octal não podem ser introduzidas em resposta a um comando INPUT, nem ser colocados em listas DATA. A maior parte das vezes, devem estar entre parenteses para se prevenir a ocorrência de erros de sintaxe.

EXEMPLOS:

```
PRINT &H5200,&051000
```

imprime os equivalentes decimais das duas constantes (ambas iguais a 20992)

```
POKE &H3C00,42
```

coloca o decimal 42 (código ASCII para asterisco) no endereço hexadecimal 3C00 da memória da tela.

ABREVIATURAS DO BASIC-DISCO

Abreviatura	Significado
↑	Lista a linha anterior do programa
↓	Lista a próxima linha de programa
.	Lista a linha atual de programa
,	Prepara a edição de linha atual de programa
SHIFT ↑	Lista a primeira linha de programa
SHIFT Z	Lista a última linha de programa
LXX	Lista a linha XX do programa
EXX	Prepara a edição da linha XX do programa
DXX	Elimina a linha XX do programa
AXXX,YYYY	Numeração automática de linha começando pela linha XX, com incremento de YYYY

CMD"A"

Retorna ao DOS-500

CMD"A"

Este comando permite que você retorne ao DOS=500 com uma mensagem de erro

```
TAREFA ABANDONADA
```

Uso Típico

Após ocorrer um erro de entrada/saída em um programa BASIC, você pode querer sair da condição de erro para o DOS-500 e imprimir uma mensagem.
CMD"A"

Será mostrado na tela o seguinte:

```
TAREFA ABANDONADA
DOS-500 ATIVO
.....
```

CMD"B"

Ativa/Desativa a tecla BREAK

CMD"B", "chave"

chave pode ser ON ou OFF e deve estar entre aspas.

Este comando ativa ou desativa a tecla BREAK. Quando a função contém a chave "OFF", a tecla BREAK será ignorada exceto durante saídas para gravador ou impressora ou ainda durante uma entrada/saída serial.

A tecla BREAK permanecerá desativada, mesmo depois de terminado o programa.

Para ativá-la, use o comando CMD"B", "ON". Retornar para o DOS-500 através dos comandos CMD"S" e CMD"I" também ativa esta tecla.

EXEMPLOS:

```
CMD"B","OFF"
```

Desativa a tecla BREAK

```
CMD"B","ON"
```

Devolve à tecla BREAK sua função normal.

9

CMD"C"

Comprime o Programa

CMD"C", opções

opções podem ser R (elimina notas "REM") ou S (elimina espaços). Se ambas as opções forem omitidas, as notas (observações) e os espaços serão eliminados.

Este comando permite que se comprima um programa, para se obter um melhor aproveitamento de espaço no disco. Você pode optar entre eliminar todas as instruções de orientação (que comecem com REM ou ') ou eliminar todos os espaços entre as palavras-chaves BASIC. Os espaços entre aspas não são eliminados.

EXEMPLO:
Suponhamos que se tenha o programa abaixo na memória:

```
850 RESTORE : ON ERROR GOTO 800          'TRATAMENTO DE ERROS
860 READ COMPANHIA$                      'NOME DA EMPRESA
870 PRINT RIGHT$ (COMPANHIA$,2) : GOTO 860
880 END
```

Se você quiser deletar as observações (notas), digite o comando:
CMD"C",R
e o programa será mudado para:

```
850 RESTORE : ON ERROR GOTO 800
860 READ COMPANHIA$
870 PRINT RIGHT$ (COMPANHIA$,2) : GOTO 860
880 END
```

Se quiser então eliminar os espaços, digite:
CMD"C",S
e o programa ficará:

```
850 RESTORE:ONERRORGOTO800
860 READCOMPANHIA$
870 PRINTRIGHT$(COMPANHIA$,2):GOTO860
880 END
```

Pode-se obter o mesmo resultado digitando apenas:
CMD"C"

NOTA

Coloque sempre a segunda aspa em string literais nos seus programas BASIC. De outra maneira, CMD"C" poderá funcionar erradamente. Por exemplo, na linha.

```
10 PRINT "ISTO E UM TESTE"
```

a segunda aspas deve ser usada, apesar da sua omissão não causar problemas no programa.

CMD"D"

Mostra o diretório de um drive específico

CMD"D:d"

d é o número do drive desejado.

Introduzindo-se o comando CMD"D':d", pode-se conseguir o diretório de um dado drive do modo BASIC-DISCO, sem ser necessário voltar ao modo DOS-500.

Apenas os arquivos não protegidos e visíveis serão mostrados. O número do drive não é opcional e deve ser especificado inclusive para o drive 0.

EXEMPLO

Se você digitar o comando:

```
CMD"D:1"
```

o diretório do drive 1 será mostrado.

CMD"E"

Mostra o Erro Anterior do DOS-500

CMD"E"

Este comando mostra a última mensagem de erro DOS-500. Se nenhum erro ocorreu antes do comando, a mensagem será "NO ERROR FOUND".

EXEMPLO

Se você tiver um sistema com dois drives (0 e 1) e digitar:

```
SAVE "PROGRAMA:3"
```

O BASIC-DISCO fornecerá uma mensagem **DISK I/O ERROR**. Para se saber qual o tipo de erro de E/S que ocorreu, digite: CMD"E" ENTER e aparecerá então a mensagem: **DISK DRIVE NOT IN SYSTEM**.

CMD"I"

Executa comandos DOS-500 no modo BASIC-DISCO

CMD"I", comando

comando é uma expressão string que contém um comando ou um nome de arquivo com programa Z-80. Caso seja uma constante string, deve ser colocada entre aspas.

Você pode executar comandos DOS-500 diretamente do modo BASIC valendo-se do CMD"I"; Considerando-se que o BASIC-DISCO não será afetado pela execução do programa ou comando, o controle voltará para o BASIC-DISCO. Caso contrário, o controle voltará para o DOS-500 (Os comandos DOS-500 sempre se sobrepõem aos do BASIC; seu programa Z-80 não poderá fazê-lo se estiver em uma área protegida da memória).

EXEMPLO

```
CMD"I","PROGRAMA"
```

coloca o sistema no modo DOS-500 e executa o arquivo de programa chamado PROGRAMA..

```
CMD"I",A$
```

coloca o sistema no modo DOS-500 e executa o comando contido em A$

CMD"J"

Conversão de Data de Calendário

CMD"J", original, conversão

original é uma expressão string que contém a data a ser convertida. Seu conteúdo pode estar em um dos dois formatos:

A) mm/dd/aa

B) - aa/ddd

O formato A contém a data na seqüência mês-dia-ano. O formato B contém a data com relação ao dia do ano (de 1 a 365, ou 366 para anos bissextos). No formato B é necessário hífen. conversão é uma variável string que contém a data convertida. Se a origem estiver no formato A, a conversão estará no formato B, e vice-versa.

Este comando converte datas entre estes dois formatos. O conteúdo da origem determina qual será realizada.

EXEMPLO

```
CMD"J","11/30/82",D$
```

Calcula o dia do ano e o armazena em D$

```
CMD"J","-79/300",D$
```

Calcula o mês, dia e ano equivalente e os armazena em D$ (a data para o 330º dia de 1979).

Programa Demonstrativo:

```
 10 CLEAR 50
 20 LINE INPUT "DIGITE A PRIMEIRA DATA (MM/DD/AA)";PD$
 30 LINE INPUT "DIGITE A SEGUNDA DATA (MM/DD/AA)";SD$
```

```
40 CMD "J",PD$,D1$
50 CMD "J",SD$,D2$
60 A1=VAL(RIGHT$(PD$,2))
70 A2=VAL(RIGHT$(SD$,2))
80 J1=VAL(RIGHT$(D1$,3))
90 J2=VAL(RIGHT$(D2$,3))
100 S1=A1*365+J1
110 S2=A2*365+J2
120 PRINT "O INTERVALO ENTRE AS DATAS E'",
130 PRINT ABS(S1-S2); "DIAS";
140 PRINT "(IGNORANDO ANOS BISSEXTO)";
150 INPUT "<ENTER> PARA CONTINUAR";A$
160 GOTO 20
```

CMD"L"

Carrega Rotina Z-80 na RAM

CMD"L" rotina

rotina é uma expressão string que contém uma especificação de arquivo para uma rotina Z-80 ou programa criado pelo comando DUMP. Se a rotina for uma constante string, deve ser colocada entre aspas.

CMD"L" carrega uma rotina Z-80 (em linguagem de máquina) na RAM. Normalmente será usada para carregar uma sub-rotina Z-80 que será acessada diretamente do BASIC-DISCO.

A rotina Z-80 deve ser armazenada nos endereços mais elevados da RAM, e não deve sobrecarregar a área reservada para ela (a reserva de espaço na memória é feita com a pergunda Mem. Usada?); Se as rotinas BASIC e DOS-500 na memória não forem alteradas com o carregamento da rotina Z-80, o controle retornará para o BASIC, após o programa ter sido carregado.

EXEMPLO

O comando:

```
CMD"L","PROG"
```

carregará um arquivo de programa chamado PROG na RAM.

```
CMD"L",P$
```

carregará um programa que está especificado em P$.

CMD"O"

Coloca em Ordem Alfabética os Elementos de uma Matriz String Uni-Dimensional

CMD"O", x, matriz (partida)

x é uma variável inteira que contém o número de itens a ser ordenado.
matriz (partida) especifica um elemento da matriz. A matriz contém os dados a serem ordenados, e partida é o subscrito do primeiro elemento a ser ordenado. A matriz deve ser uni-dimensional do tipo string. Os elementos string na matriz podem ser de qualquer tamanho.

Este comando coloca em ordem alfabética uma matriz string uni-dimensional; em outras palavras, ordena uma lista. Pode-se ordenar toda uma matriz ou apenas uma parte, dependendo dos valores atribuídos à partida e a x

PROGRAMA DE EXEMPLO

```
10 CLEAR 10*25+50
20 DIM A$(9)
30 FOR WD =0 TO 9
40 PRINT "DIGITE A PALAVRA NR.";WD+1
```

```
50 INPUT A$(WD)
60 NEXT WD
70 N% = 10 : CMD "O", N%, A$(0)
80 PRINT "AQUI ESTA A LISTA CLASSIFICADA"
90 FOR WD = 0 TO 9
100 PRINT A$ (WD)
110 NEXT WD
```

CMD"P"

Verifica as condições (status) da impressora

CMD"P", status

status é uma variável string

CMD"P" torna possível que o BASIC-DISCO verifique as condições da impressora. Ao contrário da tela, a impressora não fica permanentemente à disposição. Ela pode estar desligada, desconectada, sem papel, etc. Nestes casos, quando se tentar enviar informação para ser impressa, o computador vai esperar até que a impressora seja preparada. Durante este tempo ele ficará "bloqueado". Para se retomar o controle do teclado (e cancelar a operação da impressora), pressione BREAK.

Suponha que você tenha um programa que utilize a impressora, mas não queira que o processamento seja interrompido, caso a impressora não esteja preparada. Digamos que o programa neste caso deva emitir uma mensagem como "IMPRESSORA NÃO PREPARADA" e passe a executar outras operações. Para se realizar isso, deve-se verificar as condições da impressora.

CMD"P" pode ser usado para este fim a qualquer hora. Seu resultado é um número decimal em ASCII e depende da impressora em particular que se está usando, bem como das condições da mesma em um determinado momento.

Este valor pode ser impresso na tela ou examinado pelo programa.

Apenas os quatro bits mais significativos desse "byte de status" são usados.

Seu valor, em binário, deve ser 0011, caso contrário a operação de impressão não poderá ser realizada. Para se verificar esta condição de liberação, deve-se exècutar a operação lógica AND entre o byte de status e o número 240, comparando o resultado com 48. O significado de cada bit de status depende de qual impressora você usa. Veja no manual do proprietário da impressora para conhecer a função de cada bit.

PROGRAMA EXEMPLO

```
10 CMD "P", X$
20 ST% = VAL(X$) AND 240
30 IF ST% <> 48 THEN PRINT "INPRESSORA NAO DISPONIVEL" : STOP
40 PRINT
50 REM AQUI O PROGRAMA DEVE CONTINUAR
```

CMD"R"

Ativa o relógio na tela

CMD"R"

Esse comando controla a colocação do relógio de tempo no canto superior direito do vídeo. Quando estiver ativado, o relógio será mostrado e acertado a cada segundo, sem se considerar o programa em execução.

NOTA

O relógio de tempo real está sempre funcionando (exceto durante E/S de cassete ou disco), não importando se ele aparece ou não na tela

EXEMPLO
Para ativar o relógio na tela, digite: CMD"R" e para desligar:
CMD"T"
CMD"S"
Retornar ao DOS-500

CMD"S"

Para sair do BASIC-DISCO e devolver o controle ao DOS-500, simplesmente digite o comando:

CMD"S"

Para retornar ao BASIC e recuperar o seu programa, use **BASIC ***. Entretanto, a recuperação não será sempre possível.
Veja BASIC*

EXEMPLO
O sinal BASIC abaixo, lhe diz que você está operando no BASIC-DISCO.

```
READY
>
```

Para sair dessa condição, digite:
CMD"S" e o sinal dos DOS-500 aparecerá:

```
DOS500 ATIVO
```

.....

CMD"T"

Desativa a visualização do relógio na tela.

```
CMD"T"
```

Esse comando desativa a função de colocação do relógio de tempo real, entretanto, o relógio continua funcionando.

EXEMPLO
Para apagar o relógio de tela, digite **CMD"T".**
Para fazê-lo retornar, digite: **CMD"R"**

CMD"X"

Referência das linhas de Programa

CMD"X", objetivo

objetivo é uma palavra reservada do BASIC (como por exemplo, PRINT) ou uma constante string.

Se for um palavra reservada, ela não deverá estar entre aspas; se for uma constante string, as aspas deverão estar presentes.

Esse comando encontra todas as ocorrências de uma palavra reservada ou outra constante string no programa residente. Os "achados" são listados na tela como números de linha de 5 dígitos. Para procurar por qualquer palavra reservada do BASIC (inclusive operadores aritméticos reservados), use a palavra-chave sem modificações.

Para procurar qualquer outra coisa (incluive nomes de variáveis e texto), coloque o texto entre aspas.

Por exemplo, sumponha que você possua o seguinte programa na memória:

```
10 PRINT "ESTE E' UM TESTE"
20 INPUT "PRESSIONE <ENTER> PARA A PROXIMA MENSAGEM COM PRINT" ; Z$
30 A = A + 1
40 PRINT "*******"
```

CMD"X",PRINT encontrará todas as ocorrências de PRINT, exceto nos casos onde PRINT seja parte de um string entre aspas: linhas 10 e 40.
CMD"X","PRINT" encontrará todas as ocorrências de PRINT, como uma constante string: linha 20.
CMD"X" + listará a linha 30, mas CMD"X", "*" listará a linha 40. CMD"X", "A" listará as linhas 10,20 e 30. Observe que as variáveis e o texto são tratados como constantes string.

CMD"Z"

Duplica a saída para vídeo e Impressora

CMD"Z", "interruptor"

interruptor é a condição, ligado ou desligado (ON/OFF). Deve sempre estar entre aspas.

Esse comando ativa e desativa a saída vídeo/Impressora. Enquanto a função está "ligada", toda a saída de vídeo é copiada na impressora e vice-versa. (A impressora deve estar pronta para funcionar quando você ativar a saída dupla).

A saída de vídeo e impressora podem diferir, devido a diferenças intrínsecas nos dispositivos.

EXEMPLOS

```
CMD"Z",ON
```

Ativa saída dupla vídeo/impressora

```
CMD"Z","OFF"
```

Desativa saída dupla vídeo/impressora

DEF FN

Define funções

DEF FN nome da função (argumento, 1...) = fórmula

Nome da função e qualquer nome válido de variável.
argumento-1 e subseqüentes são usados para definir a função.
Fórmula é uma expressão, que geralmente envolve argumento(s) escrito(s) no lado esquerdo do sinal de igualdade.

A instrução DEF FN permite a criação da sua própria função, isto é, você precisa apenas chamar a nova função pelo nome e as operações associadas serão automaticamente executadas. Se uma função for definida por uma instrução DEF FN, você poderá chamá-la colocando FN na frente do nome da função. Você pode usá-la, como uma das funções internas (SIN, ABS e STRING$).

O tipo de variável usado para nome de variável, determina o tipo de valor que a função fornecerá como resultado. Por exemplo, se o nome de função for de precisão simples, temos como resultado dessa função um valor de precisão simples, não importando a precisão dos argumentos.

As variáveis especiais que você utilizar como argumentos na instrução DEF FN (argumentos-1...) não são pertinentes à função. Ao chamar a função posteriormente, qualquer nome de variável do mesmo tipo poderá ser usado. Além disso, não haverá alteração no valor da variável, se ela for usada como argumento de DEF FN, assim você poderá usá-la em outra parte de seu programa sem se preocupar com interferência da DEF FN.

A função pode ser definida sem argumento algum, se não for necessário.

```
DEF FNR = RND (90) + 9
```

Define uma função que deve resultar em um valor aleatório entre 10 e 99.

EXEMPLOS

```
DEF FNR (A,B) = A + INT ((B - (A - 1))*RND (0))
```

Essa instrução define a função FNR que resulta em um número aleatório entre os inteiros A e B. Os valores para A e B são utilizados quando a função for "chamada" isto é, usada em uma instrução como:

```
Y = FNR (R1, R2)
```

Se designarmos os valores 2 e 8 a R_1 e R_2 respectivamente, essa linha determinará um número aleatório entre 2 e 8 para Y.

```
DEF FNL$(X) = STRING$ (X, "-")
```

Define a função FNL$ que resulta em uma seqüência de hífens de comprimento x. O valor de x será utilizado, quando a função for chamada:

```
PRINT FNL$ (30)
```

Essa linha imprimirá uma seqüência de 30 hífens.
Aqui, apresentamos um outro exemplo mostrando a DEF FN um número de precisão dupla.

```
DEF FNX # (A#, B#) = (A#-B#) * (A# - B#)
```

Define a função FNX# que resulta do valor de precisão dupla, resultante de diferença entre A# e B# elevado ao quadrado. Os valores A# e B# são utilizados quando a função é chamada.

```
S# = FNX# (A#, B#)
```

Aqui, partimos da premissa de que A# e B# foram determinados em alguma outra parte do programa.

PROGRAMA EXEMPLO

```
710 DEF FNV(T) = (1087 + SQR (273 + T))/16.52
720 INPUT "TEMP. DO AR EM GRAUS CELSIUS";T
730 PRINT "A VELOCIDADE DO SOM NUM AR DE "T"GRAUS CELSIUS E' ";
FNV(T)*3.28;"METROS POR SEGUNDO."
```

DEFUSR

Define o ponto de Entrada da Rotina USR

DEFUSRn = endereço

n deve ser um algarismo de 0 a 9; se for omitido, será considerado igual a zero.
endereço especifica o endereço de entrada da rotina de linguagem de máquina. Ele deve estar compreendido na faixa de —32768 a 32767, podendo ser uma expressão numérica ou uma constante.

Essa instrução possibilita a definição de pontos de entrada para até 10 rotinas de linguagem de máquina.

Os endereços não podem ser introduzidos (POKE) na RAM do BASIC-DISCO

EXEMPLOS

DEFUSR3 = &H7D00 determina o ponto de entrada X'7D00' (32000 no código decimal) para chamada da função USR 3. Ao chamá-la o controle desviará para a sub-rotina cujo início é X'7D00'.

`DEFUSR = (BASE + 16)` determina o endereço inicial BASE + 16 para a rotina USR0.

NOTA

Quando utilizamos endereços decimais, eles são avaliados como inteiros de 2 bytes com um sinal antecedendo-os.

Sendo assim, para endereços acima 32767, use o endereço decimal —65536.

Veja USRn

INSTR

Procura por um string Especificado

INSTR (posição, string1, string2)

Posição especifica a posição no string1 onde a procura deve começar.

Essa especificação é opcional, se ela for omitida, a procura começará automaticamente no primeiro caractere de string 1. (Posição 1 é o primeiro caractere do string 1).

String 1 é o string onde se verifica a procura.

String 2 é o sub-string procurado.

Essa função possibilita a procura de um string contido em um outro.

Se o string 1 contiver o 2, a instrução INSTR fornecerá como resultado a posição inicial do segundo; caso contrário, obteremos como resultado o zero.

Atente também para o fato de que a instrução INSTR encontra a primeira ocorrência de um sub-string na posição especificada.

EXEMPLOS

Nesses exemplos, `A$ = "LINCOLN"`

`INSTR (A$, "INC")` resulta no valor 2

`INSTR (A$, "12")` resulta em zero

`INSTR (A$, "LINCONLABRAAO")` resulta em zero.

Para o uso ligeiramente diferente de INSTR, observe:

`INSTR (3, "1232123" , "12)` que resulta em 5.

PROGRAMA EXEMPLO

```
10 CLEAR 1000
20 CLS
30 INPUT "TEXTO PRINCIPAL";S$
40 INPUT "TEXTO A SER PROCURADO";T$
45 CLS
50 C = 0 : P = 1                    'P= POSICAO, C= CONTADOR
60 F = INSTR(P,S$,T$)
70 IF F = 0 THEN 120
80 C = C + 1
90 PRINT @0, LEFT$(S$,F-1)+ STRING$(LEN(T$),191)+ RIGHT$(S$,
LEN(S$)- F - LEN(T$) + 1)
100 P = F + LEN(T$)
110 IF P < = LEN(S$) - LEN(T$) + 1 THEN 60
120 PRINT "ENCONTRADO ";C;"OCORRENCIAS "
```

LINE INPUT

Introdução de linhas através do teclado.

LINE INPUT "mensagem"; variável

mensagem é uma identificação do que deve ser introduzida.
variável é o nome que será atribuído à linha digitada.
LINE INPUT (ou LINEINPUT; o espaço é opcional) é semelhante a instrução INPUT, entrento:

— Não será visualizado o ponto de interrogação indicando no operador para iniciar a introdução (INPUT) de dados.
— Cada instrução LINE INPUT poderá determinar um valor a apenas uma variável.
— Vírgulas e aspas podem ser usadas como parte da introdução de um string.
— Os espaços em branco dianteiros não são ignorados; eles se tornam parte integrante de uma variável
— O único meio de finalizar uma introdução de string é pressionar ENTER

Essa instrução é um meio conveniente de introdução de dados instring, sem a preocupação de inserções acidentais de delimitadores (vírgulas, aspas, dois pontos, etc.).

A tecla ENTER serve como único delimitador.

Se você deseja a introdução de informações no seu programa, sem instruções especiais, use a instrução LINE INPUT.

Algumas situações exigem a introdução de vírgula, aspas e espaços em branco como parte dos dados. Nesses casos, essa instrução é ideal.

EXEMPLOS

```
LINE INPUT A$
```

Introduz A$ sem mostrar nenhuma mensagem na tela.

```
LINE INPUT "SOBRENOME, NOME ?" ;N$
```

Mostra a mensagem e espera a introdução de dados. As vírgulas não cancelarão o string de entrada, isto é, elas serão consideradas parte dele.

PROGRAMA EXEMPLO

```
205 CLEAR 1000
207 PRINT
210 LINE INPUT "DIGITE SEU NOME "; A$"
220 LINE INPUT "VOCE GOSTA DO SEU COMPUTADOR ?; B$"
230 LINE INPUT "PORQUE ?"; C$"
235 PRINT
240 PRINT A$ : PRINT
250 IF B$ = "NAO" THEN 270
260 PRINT "EU GOSTO DO MEU COMPUTADOR PORQUE "; C$:END
270 PRINT "EU NAO GOSTO DO MEU COMPUTADOR PORQUE "; C$
```

Observe que quando a linha 210 é executada, o ponto de interrogação é visualizado após a instrução "Digite seu nome".

Note que, na linha 230, você pode responder a pergunta "Por que" com uma instrução cheia de delimitadores, vírgulas e aspas.

MID$ =

Substitue uma parte de um string

MID$ (string antigo, posição, comprimento) = string-substituto.

String antigo é o nome da variável string que sé deseja modificar.
Posição é a expressão numérica que especifica a posição do primeiro caractere a ser modificado.
Comprimento é a expressão numérica que especifica o número de caracterdes a ser substituídos.
String-substituto é a expressão string que substitui a parte especificada do string antigo.

NOTA

Se o string-substituto for menor que o comprimento especificado, então a substituição será total.

Essa instrução possibilita a substituição de qualquer parte de string por um novo string especificado, proporcionando-lhe uma enorme capacidade de correção de string.

Observe que o string resultante e o original têm o mesmo comprimento.

EXEMPLOS A$ = "LINCOLN"

```
MID$ (A$, 3, 4) = "12345" : PRINT A$
```
resulta em LI1234N.
```
MID$ (A$, 1,2) = " " : PRINT A$
```
resulta em NCOLN
```
MID$ (A$, 5) = "12345" : PRINT A$
```
resulta em LINC123
```
MID$ (A$, 5) = "01" : PRINT A$
```
resulta em LINC01N
```
MID$ (A$, 1, 3) = "***" : PRINT A$
```
resulta em ***COLN

PROGRAMA EXEMPLO

```
770 CLS:PRINT:PRINT
780 LINE INPUT "DIGITE MES E DIA MM/DD. ";S$
790 P=INSTR(S$,"/")
800 IF P = 0 THEN 780
810 MID$(S$,P,1) = CHR$(45)
820 PRINT S$ "E MAIS FACIL DE LER, NAO E'?"
```

Esse programa usa INSTR para procurar a barra ("/"). Quando a encontra ele usa MID$ = para substituí-la por um hífen "-" (CHR$ (45)).

NAME

Renumera o programa corrente

NAME nova linha, linha inicial, incremento.

Nova linha especifica o novo número de linha da primeira linha a ser renumerada. No caso de omissão usa-se o 10.
Linha inicial, especifica o número de linha no programa original onde a renumeração terá início. Se for omitida, o programa inteiro será renumerado.
Incremento especifica o incremento a ser usado entre números sucessivos de linhas.
Se for omitido, será usado 10.

EXEMPLOS

```
NAME
```

Renumera o programa inteiro: 10, 20, 30, ...

```
NAME 6000, 5000, 100
```

Renumera todas as linhas a partir do número 5000.
Sendo que essa se transformará em 6000 e as seguintes sofrerão acréscimos de 100.
Todas as referências de linha de seu programa sofrerão as alterações correspondentes.

USRn

Chamada para subrotina Externa de Usuário

USRn (nmexp)

onde n especifica uma das dez chamadas disponíveis de USR, n— 0, 1, 2,...9.
Se n for omitido, será usado zero.
nmexp é um inteiro de —32768 a 32767 e é usado como um argumento inteiro na rotina.
Essas funções (de USR0 a USR9) transferem o controle para rotinas de linguagem de máquina previamente definidas por instruções DEFUSRn.
Quando uma chamada USR é encontrada em uma instrução, o controle passa para o endereço especificado na instrução DEF USRn. Esse endereço especifica o ponto de entrada da sua rotina de linguagem de máquina.

NOTA

Ocorrerá um erro de chamada ilegal de função, se você chamar uma rotina USRn antes de definir o seu ponto de entrada com uma DEFUSRn.
Você pode utilizar um argumento e obter um valor de saída diretamente através de um argumento do USR, ou então utilizar e obter argumento indiretamente, via instruções POKE e PEEK.

EXEMPLO

```
10 DEFUSR1=&H7D00
20 REM... MAIS LINHAS DE PROGRAMA AQUI
100 A = USR1(X)
```

Esta seqüência é composta de:
1 — Definição da USR: uma rotina com um ponto de entrada em 7D00 hexadecimal (linha 100)
2 — Transferência de controle para a rotina; o valor x pode ser utilizada na rotina, se essa executar a chamada (CALL) descrita abaixo (linha 100).
3 — Quando a rotina retorna ao BASIC, a variável A pode conter o valor utilizado na rotina (se sua rotina executar o JUMP descrito abaixo); caso contrário, define o valor de x para A (linha 100).

Transferência de argumentos entre o BASIC e rotinas USR

Existem vários meios de passar argumentos do BASIC para rotina USR e vice-versa.
Esses são os mais usados:
1 — Introduza (POKE) o(s) argumento(s) em posições fixas de RAM. A rotina de linguagem de máquina pode ter acesso a esses valores e colocar os resultados em outras posições da RAM.
Quando a rotina transfere o controle para o BASIC, seu programa poderá "ler" (PEEK) esses endereços e escolher os valores de saída.
Esse é o único meio de obtermos uma transferência de dois ou mais argumentos nos dois sentidos.
2 — Tranfira um argumento para rotina como o argumento em uma chamada de USRn, então use chamadas especiais de ROM para se ter acesso a esse argumento e forneça um valor ao BASIC.
Esse método se restringe a enviar um argumento e obter como resultado um valor, sendo que ambos são inteiros.

CHAMADAS ROM

Call 0A7FH coloca o argumento de USR no par de registros HL; H e L contém, respectivamente. MSB e LSB. Essa instrução deveria ser a primeira em sua rotina USR.

JP 0A9AH — Use esse JUMP para retornar ao BASIC; o inteiro em HL se torna a saída da chamada de USR. Se o valor contido em HL não for importante, execute uma simples instrução RETURN, ao invés desse JUMP.

O programa listado abaixo serve para deixar a tela toda verde (o inverso da tecla CLEAR). Não digite. Digite o programa BASIC que o segue.

```
0               16          26          34          46          56

                00100       ;
                00110       ; FUNÇÃO MANUSEIO DA TELA
                00120       ;
7D00            00130                   DRG         7D00H
                00140       ;
                00150       ; EQUATES
                00160       ·
3C00            00170       VIDEO       EQU         3C00H       ; END MEN DO VIDEO
00BF            00180       WHITE       EQU         0BFH        ; TODOS CARACTERES BRANCOS
03FF            00190       COUNT       EQU         3FFH        ; NR CARACTERES PARA MOVER
                00200       ;
                00210       ; MOVE 'BF' PARA TODA A MEMÓRIA DE VIDEO
                00220       ;
7D00 21003C     00230 ZAP   LD          HL, VIDEO               ; ENDEREÇO FONTE
7D03 36BF       00240       LD          (HL),'WHITE             ; PRIMEIRO BYTE
7D05 11013C     00250       LD          DE, VIDEO+1             ; ENDEREÇO DESTINO
7D08 01FF03     00260       LD          BC, COUNT               ; NR DE INTERAÇÕES
7B0B EDB0       00270       LDIR
                00280       ;
7D0D C9         00290       RET                                 ; VOLTA AO BASIC
                            END         ZAP
```

Essa rotina pode ser introduzida (POKE) na RAM e podemos ter acesso a ela como a uma rotina USR. Primeiro inicialize o BASIC e responda a pergunta Men. Usada?, com 31999. Depois, execute o programa.

```
100 'PROGRAMA : USR1
110 'EXEMPLO DO USO DE LINGUAGEM DE MAQUINA
115 'PARA PARAR PRESSIONE A TECLA '@'
130 '*** PROGRAMA COLOCADO NA MEMORIA ATRAVES DO POKE ***
150 DEFUSR1 = &H7D00
160 FOR X = 32000 TO 32013    '7D00 EM HEXA = 32000 DECIMAL
170     READ A
180     POKE X,A
190 NEXT X
195 '*** LIMPA TELA E IMPRIME NUMEROS DE 1 ATE 100 ***
200 CLS
205 PRINT TAB(15); "ROTINA DE INVERSAO DA TELA";:PRINT
210 FOR X = 1 TO 100
220     PRINT X
225     A$ = INKEY$ : IF A$ = "@" THEN END
230 NEXT X
250 '*** SALTA PARA ROTINA DE INVERSAO ***
```

9

```
270 X = USR1(0)
280 FOR X = 1 TO 1000 : NEXT X          'ESPERA DENTRO DO LOOP
290 GOTO 200
310 '*** DATA CONTEM CODIGO EM DECIMAL DO PROGRAMA EM HEXA ***
330 DATA 33,0,60,54,191,17,1,60,1,255,3,237,176,201
```

Execute o programa. Como podemos observa, uma rotina de BASIC para tornar a tela verde exige um tempo muito maior.

CARACTERÍSTICAS RELATIVAS AO DISCO

O DISCO-BASIC fornece um conjunto potente de comandos, instruções e funções relativas à E/S de disco no DOS-500. Esse conjunto pode ser dividido em 2:

1 — Manipulação de Arquivo: Os arquivos são considerados unidades, não importando os detalhes de seus registros.

2 — Acesso de arquivo: Preparação de arquivos de dados para E/S; leitura e gravação de registros dos arquivos.

Sob o título manipulação de arquivo, teremos os seguintes comandos:

KILL	Apaga um programa ou um arquivo de dados de um disco.
LOAD	Carrega um programa BASIC do disco.
MERGE	Combina um programa BASIC de formato ASCII no disco, com o atual na RAM.
RUN "programa"	Carrega e executa um programa BASIC armazenado no disco.
SAVE	Armazena o programa BASIC residente no disco.

Sob o título Acesso de Arquivo, teremos as seguintes instruções e funções:

OPEN	Abre um arquivo para acesso (Cria um se necessário)
CLOSE	Fecha o acesso ao arquivo
INPUT	Lê o disco no modo seqüencial
LINE INPUT	Lê uma linha de dados no modo seqüencial
PRINT	Grava no disco, no modo seqüencial.
FIELD	Determina nomes dimensões de campos buffer de um arquivo de acesso aleatório.
GET	Lê no disco no modo de acesso direto.
PUT	Grava no disco no modo de acesso direto.
LSET	Coloca um valor num campo especificado no buffer e acrescenta espacos em branco à direita para preencher o campo.
RESET	Coloca um valor num campo especificado do buffer e acrescenta espaços em branco à esquerda para preencher o campo.
FUNÇÕES	
CVD	Transforma um número de precisão dupla que se encontra em forma de string para a forma numérica, após sua leitura (GET) do disco.
CVI	Transforma um inteiro que se encontra em forma de string para a forma numérica, após a leitura (GET) do disco.
CVS	Transforma um número de precisão simples que se encontra em forma de string para a forma numérica, após a leitura (GET) do disco.
EOF	Verifica se o fim de arquivo foi alcançado durante a leitura.
LOC	Lê (GET) o número correto de registro.
LOF	Fornece o número do último registro do arquivo.
MKD$	Converte o número de precisão dupla em string, assim ele pode ser gravado (PUT) no disco.
MKI$	Converte o inteiro em string, assim ele poderá ser gravado (PUT) no disco.
MKS	Converte o número de precisão simples em string, assim poderá ser gravado (PUT) no disco.

Manipulação de arquivo.

KILL

Apaga um arquivo no disco

KILL exp$

exp$ define uma especificação de arquivo para um arquivo já existente. Esse comando funciona como o KILL do DOS-500 (veja comados de biblioteca do DOS-500).

EXEMPLO
KILL "ARQANTIG/BAS.PSW 1" apaga o arquivo especificado do primeiro drive que o contenha.
Não apague (KILL) um arquivo aberto, ou você destruirá o conteúdo do disquete (primeiro, feche (CLOSE) o arquivo).

LOAD

Carrega o arquivo de Programas BASIC do disco

LOAD exp $ [,R]

exp $ define uma especificação de arquivo para um arquivo de programas BASIC armazenados no disco.

P diz ao BASIC para executar o programa após ser carregado. Se a opção R for usada, o BASIC procederá à execução do programa automaticamente; senão, ele retornará ao modo Comando. LOAD sem a opção R limpa todas as variáveis e fecha todos os arquivos abertos. LOAD com a opção R limpa todas as variáveis, mas não fecha os arquivos abertos. LOAD com a opção R é equivalente ao comando RUN exp $,R. Ambos os comandos podem ser usados dentro de programas, para possibilitar o encadeamento deles, isto é, um programa chama o outro e este, por sua vez, um terceiro, etc.

EXEMPLO

```
LOAD "PROG1/BAS:2"
```

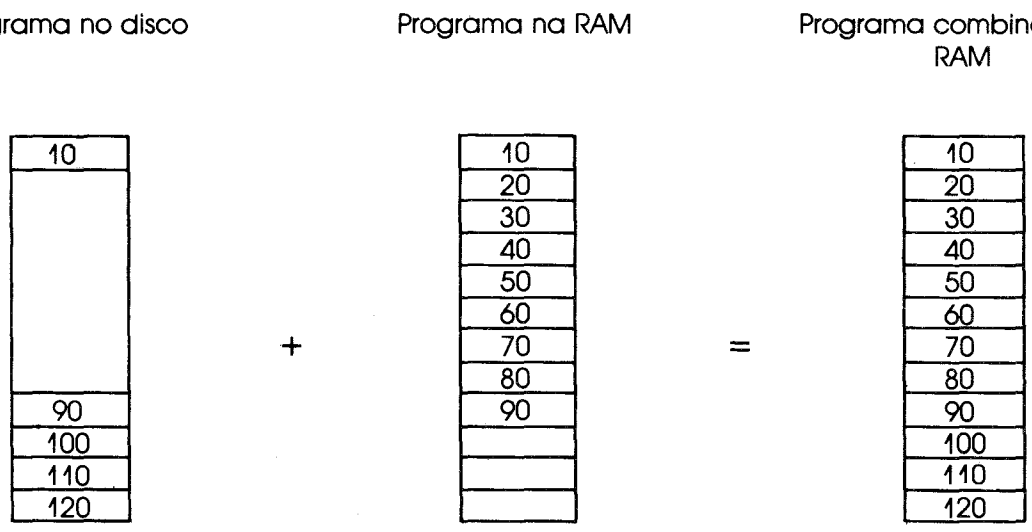
Limpa o programa BASIC residente e carrega o PROG1/BAS do DRIVE 2 e então retorna ao modo de comando BASIC.

MERGE

Combina programa de disco com programa residente.

MERGE exp$

exp$ é uma especificação de arquivo para arquivo de disco BASIC em formato ASCII, por ex., um programa armazenado com a opção A. MERGE é similar a LOAD, exceto pelo fato do programa residente não ser apagado antes que o novo programa exp$ seja copiado. Ao invés disso, o novo programa é associado ao programa residente, isto é, as linhas do programa em exp$ serão simplesmente inseridas no programa residente em seqüências. Se os números das linhas de exp$ coincidirem com os do programa residente, as linhas deste serão substituídas pelas do exp$.



EXEMPLO PRÁTICO

Armazene este programa em formato ASCII

```
1000 REM *** SUBROTINA PARA DIZER OLA' ***
1010 PRINT "OLA'!!!"
1020 RETURN
```

Digite NEW ENTER, e então digite esse programa.

```
100 CLS
110 PRINT "VAMOS CHAMAR A SUBROTINA ..."
120 PRINT "DIALOGANDO AGORA..."
130 FOR I = 1 TO 1000 : NEXT
140 GOSUB 1000
150 PRINT "RETORNO DA SUBROTINA"
160 END
```

Agora digite MERGE "ARQUIVO", usando o nome de arquivo fornecido ao primeiro arquivo. Liste o programa. Então execute-o.

RUN "programa"

Carrega e executa um programa do Disco

RUN arquivo [,R]

Arquivo é o nome de um arquivo de programas BASIC. Ele é uma expressão string. Se uma constante string for usada, ela deve ser disposta entre aspas.

A opção R faz com que o BASIC deixe aberto os arquivos abertos. Se for omitida, os arquivos abertos serão fechados antes da execução do programa. Esse comando carrega e executa um programa BASIC armazenado no disco. Ele pode ser usado dentro de um programa para possibilitar um encadeamento (um programa chamar outro).

EXEMPLOS

```
RUN "PROG"
```

Carrega e executa PROG (primeiramente, todos os arquivos abertos são fechados).

```
A$ = "PROGNOVO"
RUN A$, R
```

Carrega e executa PROGNOVO (todos os arquivos abertos permanecem abertos)

SAVE

Armazena Programa no Disco

SAVE arquivo [,A]

arquivo é o nome de um arquivo de programas BASIC. Ele é uma expressão string. (Se uma constante string for usada, ela deverá ser disposta entre aspas).

"A" faz com que o arquivo seja armazenado em ASCII e não em formato comprimido.

Esse comando possibilita o armazenamento dos seus programas BASIC no disco. Você pode armazenar o programa em formato comprimido ou em ASCII. O uso do formato comprimido ocupa menos espaço de disco e é mais rápido durante SAVE e LOAD. O uso da opção ASCII possibilita a realização de certas coisas, que não poderiam ser feitas com arquivos BASIC de formato comprimido.

Por exemplo:

1 — O comando BASIC necessita que o arquivo de disco esteja na forma ASCII.
2 — Programas que lêem outros programas como se fossem dados, normalmente necessitarão que os programas de dados sejam armazenados em ASCII.
3 — O comando do DOS-500 APPEND também requer que os arquivos de disco estejam na forma ASCII.

EXEMPLOS

```
SAVE "ARQ1/BAS. PROLOGICA:3"
```

armazena o programa BASIC residente em formato comprimido com o nome de arquivo, ARQ 1. extensão a BAS e senha PROLOGICA. O arquivo é colocado no Drive 3.

```
SAVE "MATHPAK/TXT", A
```

armazena o programa residente na forma ASCII, usando o nome MATHPAK/TXT, no primeiro disco não protegido contra gravação.

Ao se completar um SAVE, o BASIC retorna ao modo comando.

ACESSO A ARQUIVO

Este capítulo é dividido em 4 partes:

1 — Criação de arquivos e determinação de buffers-OPEN e CLOSE.
2 — Instruções e Funções.
3 — Técnicas de E/S seqüenciais.
4 — Técnicas de E/S aleatórias.

Se esse for seu primeiro contato como o acesso de arquivo em disco, você deverá desviar sua atenção para as partes 1, 3, e 4 e apenas folhear a parte 2, para ter uma idéia geral do conteúdo. Posteriormente, você poderá voltar a lê-la mais atentamente.

CRIAÇÃO DE ARQUIVOS E DETERMINAÇÃO DE BUFFERS

Durante o diálogo de inicialização, você deverá digitar um número em resposta a:

Quantos arquivos?

O número que digitar, dirá ao BASIC quantos buffers devem ser criados para manipular seus acessos ao disco (leituras e gravação).

Um número de 1 a 5 é fornecido a cada buffer.

Se você digitar:

QUANTOS ARQUIVOS? 3V ENTER , o BASIC separará 3 buffers, numerados 1,2,3, de tamanhos variáveis.

O buffer é uma área de espera pela qual os dados passam quando vão e voltam do arquivo de disco. Se você desejar ter acesso a um arquivo em especial, você deve dizer ao BASIC qual buffer deve ser usado, quando tover acesso a esse arquivo.

Você deve dizer ao BASIC que tipo de acesso deseja: saída/entrada seqüencial ou saída/entrada aleatória.

Tudo isso é feito com a instrução OPEN, e desfeito com a CLOSE:

OPEN

Abre um arquivo

OPEN, modo, buffer, arquivo, comprimento de registro.

Modo é uma expressão string contendo uma das seguintes características abaixo:

I — Introdução seqüencial com início no primeiro registro. Se o arquivo não for encontrado, ele será então, criado.
O — Saída seqüencial com início no primeiro registro. Se o arquivo não for encontrado, ele será criado.
E — (extensão) saída seqüencial com início no final do arquivo; se o arquivo não for encontrado, ele será criado.
R — entrada/saída aleatória. Se o arquivo não for encontrado, ele será criado.

Se modo é uma constante, esta deverá ser disposta entre aspas.

Buffer é uma expressão numérica que especifica o buffer a ser utilizado.

Arquivo é uma expressão string que contém uma especificação de arquivo. Se for usado uma constante, esta deverá vir entre aspas.

comprimento do registro — é uma expressão numérica de 0 a 256 que especifica um comprimento lógico de registro. 0 é o mesmo que 256. Essa opção somente poderá ser usada se os registros de comprimento variável forem necessários durante a inicialização (Quantos arquivos?):

Se o comprimento do registro for omitido, 256 será usado. O comprimento do registro somente é usado em acesso aleatório.

Essa instrução possibilita a criação de um arquivo, a gravação de registros de dados, sua correção e sua leitura. Para maiores detalhes, consulte Métodos de Acesso nesta seção. Se o "arquivo"

contiver uma especificação de drive, o BASIC somente usará o drive especificado. Se nenhum drive for especificado, o BASIC procurará por um arquivo compatibilizante, iniciando no drive mestre (geralmente Drive 0).

EXEMPLOS

```
OPEN "O",1,"ARQDADOS"
```

Abre um arquivo, chamado ARQDADOS (cria se ele não existir) para saída seqüencial.
A saída será efetuada através do buffer #1. Os registros serão de 256 bytes.
Se o modo "O" for especificado, a saída iniciará no primeiro registro do arquivo.
Se o modo "E" for usado ao invés de "O", a saída iniciará no final do arquivo.

```
OPEN "R",2,"FOLHAPAG/A:1",64
```

Abre ou cria um arquivo chamado "FOLHAPAG/A" para saída e entrada aleatória. O acesso será através do buffer #2. Os registros serão de 64 bytes (Se o BASIC foi inicializado para registros de comprimento variável).

```
BUFFER = 3: ARQUIVO$ ="DADOS" : RECLN = 128
OPEN "R",BUFFER, ARQUIVO$, RECLN
```

Abre ou cria o arquivo chamado DADOS para saída e entrada aleatória. O acesso será através do buffer #3. Os registros serão de 128 bytes (Se o BASIC foi inicializado para registro de comprimento variável).

CLOSE

Fecha o acesso no Arquivo.

CLOSE [nmexp [nmexp,...]]

nmexp é um valor de 1 a 15 e indica o número do buffer do arquivo (determinado ao se abrir o arquivo). Se nmexp for omitido, todos os arquivos abertos serão fechados.

Esse comando termina o acesso a um arquivo, através de um ou mais buffers especificados.

Se um exp não for determinado numa instrução OPEN anterior, então CLOSE nmexp não surtirá efeito.

Se nmexp for omitido, todos os arquivos abertos serão fechados.

Esse comando termina o acesso a um arquivo, através de um ou mais buffers especificados.

Se um exp não for determinado numa instrução OPEN anterior, então CLOSE nmexp não surtirá efeito.

Exemplos:

```
CLOSE 1, 2, 8
```

Cancela as especificações de arquivo dos buffers 1, 2, e 8. Esses buffers podem ser agora designados a outros arquivos, utilizando instruções OPEN.

```
CLOSE ARQ01% + CONT%
```

Cancela a especificação de arquivo do buffer especificado pela soma (1% + CONT %). NÃO RETIRE UM DISQUETE QUE CONTENHA UM ARQUIVO ABERTO PARA GRAVAÇÃO (MODO O, E ou R). PRIMEIRO, FECHE O ARQUIVO. Isto, devido à impossibilidade de se escrever os últimos 256 bytes de dados no disco.

O fechamento de arquivo registrará os dados, se não o tiver feito.

Qualquer modificação no programa residente (NEW, edição, LOAD, MERGE, etc.) farão com que os arquivos abertos sejam fechados.

INPUT

Leitura seqüencial do Disco

INPUT # nmexp, var, [, var...]

nmexp especifica o buffer de arquivo com entrada seqüencial, **nmexp** onde nmexp = 1,2,...15.

Var é o nome de variável que contém os dados do arquivo.

Esta instrução introduz seqüencialmente dados de um arquivo de disco, isto é, quando for aberto pelá primeira vez, um indicador será colocado no começo do arquivo. Cada vez que há introdução de dados, o indicador avança. Para recomeçar a ler no início, você deve fechar o arquivo e reabrí-lo

A instrução INPUT # não considera o meio de inserção de dados no disco, isto é, quando for aberto pela primeira vez, um indicador será colocado no começo do arquivo. Cada vez que há introdução de dados, o indicador avança. Para recomeçar a ler no início, você deve fechar o arquivo e reabri-lo.

A instrução INPUT # não considera o meio de inserção de dados no disco, isto é, se apenas uma ou 10 instruções PRINT # diferentes os introduziram. O importante para ela é o símbolo de final de arquivo (EOF) e as posições dos caracteres de finalização.

Para uma boa leitura de dados em um disco, necessitamos primeiramente saber o formato dos dados a serem lidos. Aqui, temos uma descrição de como a INPUT # interpreta os vários caracteres que ela encontra ao ler os dados.

Ao introduzir dados em uma variável, o BASIC ignora os espaços em branco dianteiros.

Qualquer caractere diferente dos precedentes será considerado início de um ítem de dados.

O final de um item de dados é demarcado por um caractere de finalização ou por condições de término. Os caracteres especiais de finalização variam, dependendo do tipo de variável que o BASIC estiver usando (numérica ou string).

NOTA ESPECIAL

Aqui apresentamos uma exceção importante:
Ao se pressionar a tecla ENTER (retorno do carro) precedido da tecla ↓ (alimentação de linha), o ENTER não será considerado um finalizador.

INTRODUÇÃO NUMÉRICA

Suponha que a imagem de dados seja `1.234 -33 27` ENTER

Meste caso, ENTER é um caractere de retorno de carro (ASCII código decimal 13).

Então a instrução `INPUT #1, A, B, C` ou a seqüência de instruções:

`INPUT #1, A: INPUT #1, B: INPUT #1, C` Atribuirão os valores 1.234, —33 e 27 as variáveis A, B e C respectivamente.

Isto funciona porque, tanto os espaços em branco como ENTER, servem como finalizadores de leitura de variáveis numéricas.

O espaço em branco anterior ao número 1.234 é ignorado e o posterior ao mesmo será considerado finalizados; portanto, o BASIC inicia a leitura da segunda variável ao detetar o caractere —, lê o número —33 e considera os 2 espaços em branco seguintes como sendo finalizadores. A terceira leitura coemça no 2 e termina no 7.

INTRODUÇÃO DE STRING

Ao ler os dados numa variável string, a instrução INPUT ignora todos os espaços em branco dian-

teiros; o primeiro caractere diferente do anterior a ser detetado marcará o início de um ítem de dados.

Se o primeiro caractere for aspas ("), a INPUT considerará os dados como um string entre aspas (ela lerá todos os caracteres subseqüentes até as próximas aspas). Vírgulas, espaços em branco, e caracteres ENTER serão incluídos no string. As aspas não se tornam parte integrante do string.

Se o primeiro caractere de um item string não for aspas, a INPUT considerará os dados como sendo um string sem aspas (ela lerá todos os caracteres subseqüentes até a primeira vírgula ou ENTER se as aspas forem encontradas, elas serão inclusas no string. Por exemplo, se a informação no disco é: BECOS, TEXAS "BONS MELÕES", então a instrução **INPUT # 1, A$, B$, C$** atribuirá os valores BECOS, TEXAS, "BONS MELÕES", as variáveis A$ e B$ ficando C$ uma string nula respectivamente.

Se uma vírgula for introduzida na imagem de dados antes da primeira aspas, C$ adquirirá o valor BONS MELÕES.

Estes são exemplos simples, para lhe dar uma idéia de como a INPUT funciona.

Entretanto, existem vários outros meios de se introduzir dados, (finalizadores diferentes, tipos de variáveis de objetivos diferentes, etc,).

Tentaremos dar uma descrição geral de como a instrução INPUT funciona e quais são as condições e caracteres de finalização e então fornecer vários exemplos.

Quando o BASIC encontra um caractere de finalização, ele avança para verificar quantos caracteres de finalização a mais podem estar coligados ao primeiro.

Isto garante o início da procura do próximo item de dados no lugar correto.

A lista abaixo define os vários conjuntos de finalização que a INPUT # procurará. Ela sempre tentará abranger o maior conjunto possível.

COJUNTO DE FINALIZADORES DE ENTRADA NUMÉRICA

Fim de arquivo encontrado
255.º caractere de dados encontrado.
(vírgula)
ENTER
ENTER ↓
[...] [ENTER]
[...] [ENTER ↓]

CONJUNTO DE FINALIZADORES DE STRINGS ENTRE ASPAS

Fim de arquivo encontrado
255.º caractere de dados encontrado.
" (aspas)
"[...] [,]
"[...] [ENTER]
"[...] [ENTER ↓]

CONJUNTO DE FINALIZADORES DE STRINGS SEM ASPAS

Fim de arquivo encontrado
255.º caractere de dados encontrado.
ENTER ↓

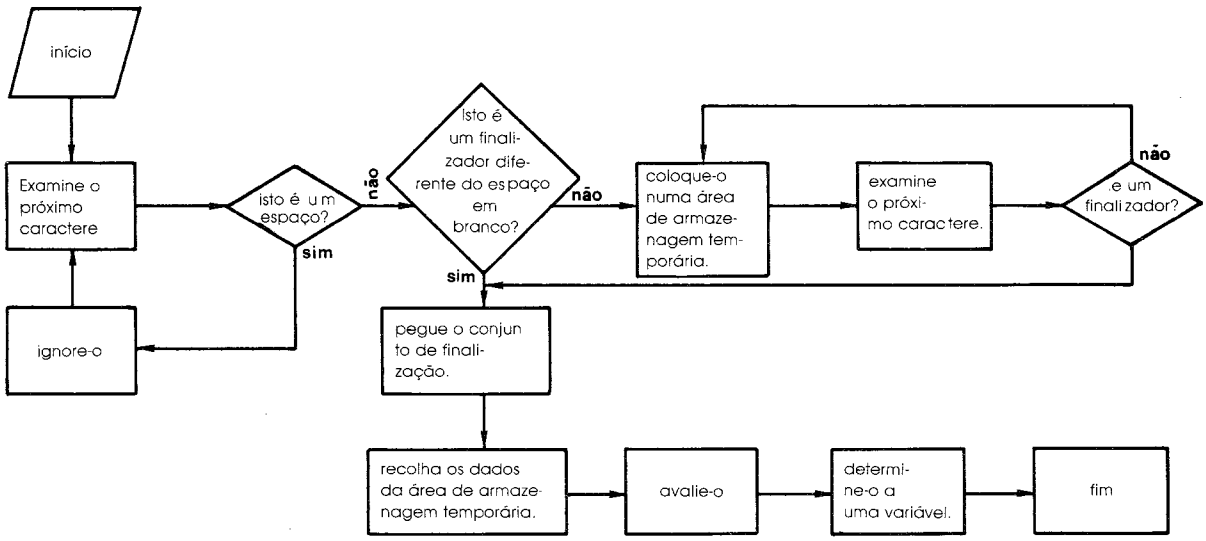


Figura 12 descreve como a INPUT # atribui dados a uma variável

A tabela seguinte mostra como várias imagens de dados serão lidas pela instrução: INPUT # 1, A, B, C.

Ex. #	Imagem do disco	Valores atribuídos
1	123.45 ENTER ↓ 8.2E4 7000 ENTER	A = 123.45 B = 82000 C = 7000
2	3 ↓ ENTER 4 ENTER 5 ENTER A12 eof	A = 34 B = 5 C = 0
3	1, 2, 3, 4 ENTER	A = 1 B = 0 C = 2
4	1, 3, eof	A = 1 B = 3 C = 0 eof error

(eof = fim de arquivo)

Por que a variável C tem o valor 0 no exemplo 2 acima? Quando a entrada alcança o fim de arquivo, ela finaliza o último ítem de dados, que conterá "A12" neste momento. Essa será avaliada por uma rotina exatamente como a função BASIC VAL, que retorna um zero desde que o primeiro caractere de "A12" não seja numérico.

No exemplo 3, quando a INPUT # procura por um segundo ítem de dados, ele encontra imediatamente um finalizador (vírgula); portanto, é dado o valor zero à variável B.

A segunda tabela mostra como as várias imagens de dados no disco serão lidas pelas instruções INPUT # 1, A$, B$.

Ex #	Imagem no disco	Valores atribuídos
1	"ROBERTO,J, "ROBERTO,M.N. eof	A$: ROBERTO,J. B$:ROBERTO,M.N
2	ROBERTO,J., ROBERTO,M.N. ENTER	A$:ROBERTO B$:J.
3	A PALAVRA "MEL", 12345.789 ENTER	A$:A PALAVRA "MEL" B$:12345.789
4	BYTE ↓ ENTER UNIDADE DE MEMÓRIA eof	A$ BYTE ↓ ENTER UNIDADE DE MEMÓRIA B$:nulo (erro eof)

No exemplo 3, o primeiro ítem de dados é um string sem aspas, portanto, elas não são finalizadoras e sim, parte de A$.

No exemplo 4, o ENTER é precedido por ↓, logo, ele não finaliza o primeiro string; Tanto ↓. como ENTER são incluídos em A$.

LINE INPUT

Lê uma linha de texto de um Disco.

LINE INPUT # nmexp, var$

onde nmexp especifica um Buffer de arquivo de saída seqüencial, nmexp 1,2,...15

var$ é o nome de variável que conterá dados string.

Semelhante à instrução LINE INPUT do teclado, essa lê uma "linha" de dados string na var$. Essa característica se torna útil, quando se deseja ler um arquivo de programas BASIC em formato ASCII como dados, ou quando se quer ler dados sem seguir as restrições usuais, considerando caractere e finalizadores dianteiros.

LINE INPUT (ou LINEINPUT — o espaço é opcional)

Lê tudo,do primeiro caractere ao:

1 — Caractere ENTER que não seja precedio por ↓

2 — fim de arquivo

3 — 255° caractere de dados (este caractere está incluso no string)

Outros caracteres encontrados — aspas, vírgulas, espaços em.branco dianteiros, ↓ ENTER são inclusos no string.

Por exemplo, se os dados formam:

```
10 CLEAR 500 ENTER
20 OPEN "I",1,"PROG" ENTER
'
'
'
```

Então a instrução `LINEINPUT #1, A$` poderia ser usada repetitivamente para ler cada linha do programa individualmente.

PRINT

Gravação seqüencial no arquivo do disco.

PRINT #nmexp, [formato$ USING;] exp [p exp...]

nmexp especifica um buffer de arquivo de saída seqëncial, nmexp = 1, 2,...15.

format$ é uma seqüência de especificadores de campo, usados com a opção USING

p é um delimitador colocado entre 2 expressões a serem impressas no disco; Tanto o ponto-e-vírgula como a vírgula podem ser usados (o ponto-e-vírgula é preferível).

exp é a expressão a ser executada e gravada no disco.

Essa instrução grava dados seqüencialmente em um arquivo especificado. Ao se abrir pela primeira vez um arquivo para saída seqüencial, um indicador é colocado no começo do arquivo; portanto, seu primeiro PRINT # posiciona os dados no início do arquivo. Ao final de cada operação PRINT #, o indicador avança, assim os valores são escritos em seqüência:

Uma instrução PRINT # cria uma imagem no disco semelhante à criada pela PRINT comum, na

tela. Lembre-se disso, e você poderá organizar suá lista PRINT # corretamente, para acesso de uma ou mais instruções INPUT.

PRINT # não comprime informações antes de gravá-las no disco; ela grava uma imagem de daods codificadas pelo ASCII.

Por exemplo, se A = 123.45

PRINT #1,A escreverá uma seqüência de caracteres em 9 bytes do disco:

123.45 ENTER

A pontuação em uma lista PRINT #é muito importante. Vírgulas e ponto-e-vírgulas sem aspas tem o mesmo efeito·que teriam em instruções normais PRINT para visualização.

Por exemplo, se A = 2300 e B = 1.303, então

PRINT # 1, A, B coloca os dados no disco como

2300 1.303 ENTER

A vírgula entre A e B na lista PRINT# proporciona 10 espaços extras no arquivo de disco; assim sendo, deve-se usar ponto-e-vírgulas ao invés de vírgulas.

PRINT #1, A,B escreve os dados como:

2300 1.303 ENTER

A PRINT# com dados numéricos é contínua (lembre-se de separar os itens com ponto-e-vírgulas).

A PRINT # com dados string necessita de muita atenção, pois você tem que introduzir delimitadores de modo que os dados possam ser relidos corretamente.

Você deve separar os itens string com delimitadores explícitos, se você desejar introduzí-los (INPUT #) como strings distintos.

Por exemplo, suponha que:

A$ = "JOAO C. DIAS" B$ = "100-01-001"

Então: PRINT #1, A$; "," ; B$ produzirá esta imagem no disco:
JOAO C. DIAS 100-01-001 , que não pode ser reintroduzido em 2 variáveis, A instrução:
PRINT #1, A$; B$ produziria:
JOAO C. DIAS , 100-01-001 que pode ser reintroduzido em 2 variáveis.

Esse método será eficiente à medida que os dados string não contiverem caracteres delimitadores (vírgulas ou ENTER). Mas, se os dados realmente contiverem delimitadores ou espaços em branco, que você não deseja ignora, você deverá colocar aspas nos dados. Por exemplo, suponha que A$ = "DIAS, JOAD C." B$ = "100-01-001" se você usar:
PRINT #1, A$; "," ; B$ a imagem de disco será: DIAS, JOAO C., 100-01-001

Se você tentar introduzir essa informação com uma instrução, como: INPUT#2, A$, D$

A$ terá valor DIAS e B$ o valor JOAO C, devido à vírgula após o DIAS na imagem do disco.

Como 34 é o decimal do código ASCII para aspas, use CHR$ (34) desta forma:
PRINT #1, CHR$(34); A$; CHR$(34); B$

Essa instrução criará imagem de disco:
"DIAS, JOAD C." 100-01-001
que poderá ser lida com uma simples INPUT #2, A$, B$

10

NOTA

Você pode utilizar a função CHR$ para inserir outros delimitadores e códigos de cantrole no arquivo por exemplo:

CHR$(10)	↓ Alimentação de linha
CHR$ (13)	Retorno de carro (caracteres ENTER)
CHR$ (11) ou CHR$ (12)	SALTO DE PÁGINA (TOP OF FORM)

OPÇÃO USING

Essa opção facilita escrever arquivos num formato cuidadosamente controlado.
Por exemplo, suponha:

```
A$ = "LUDWIG"
B$ = "VAN"
C$ = "BEETHOVEN"
```

Então a instrução
`PRINT #1, USING "!. !.%  %" ;A$ ; B$ ; C$` escreverá a informação em forma abreviada.
`L.V. BEET <ENTER>`
Neste caso, não desejamos acrescentar nenhum delimitador explícito.

INTRUÇÕES DE ACESSO ALEATÓRIO

FIELD

Organiza um BUFFER Aleatório de arquivo em campos

FIELD nmexp, nmexpl. AS VAR1$ [,nmexp2 AS VAR2 $...]

nmexp especifica um buffer de arquivo de acesso aleatório
nmexp = 1, 2,...15
nmexpl especifica o comprimento do primeiro campo.
VAR1$ define um nome de variável para o 1.º campo, nmexp 2 especifica o comprimento do 2.º campo.
VAR2$ define o nome da variável do 2.º campo.
... pares de nmexp AS VAR$ subseqüentes define outros campos no buffer.

NOTA

A soma de todos os componentes de campo não devem exceder o comprimento do registro, mas igualá-lo

Antes de se fornecer campo ao buffer, deve-se usar uma instrução OPEN a fim de determinar a esse buffer um arquivo de disco específico (deve-se usar o modo aleatório).

Então, use a instrução FIELD para organizar um buffer aleatório de arquivo, de modo que se transferir dados do BASIC para armazenagem de disco e vice-versa.

Cada buffer aleatório de arquivo contém até 256 bytes, que podem armazenar dados para essa transferência. (Quando arquivos de comprimento variável são usados, o máximo se situa entre 1 a 256). No entanto, você necessita de um meio de acesso para esse buffer do BASIC, de modo a poder ler os dados nele existentes, ou colocar novos no mesmo.

A instrução FIELD proporciona os meios para esse acesso.

Você pode usar a instrução FIELD quantas vezes desejar para reorganizar um buffer de arquivo. Esta organização não limpa o conteúdo do buffer; somente os meios de acesso do buffer (os nomes de campo) são modificados. Além disso, 2 ou mais nomes de campo podem indicar uma mesma área do buffer.

EXEMPLOS

```
FIELD 1, 128 AS A$, 128 AS B$
```

Esta instrução diz ao BASIC para distribuir 128 bytes para a variável string. A$ e 128 para B$. Se você imprimir A$ e B$ agora, você verá o conteúdo do buffer. É claro que este valor seria inexpressivo, se você não usasse GET para ler um registro de 256 bytes do disco.

NOTA

Todos os dados (string e números) devem ser colocados no buffer na forma de string. Existem 3 pares de funções (MKI$/CVI, MKS$/CVS, MKD$/CVD) para conversão de números em strings e vice-versa.

Veja as funções abaixo:

```
FIELD 3, 16 AS NM$, 25 AS AD$, 10 AS CY$, 2 AS ST$, 7 AS ZP$
```

Os primeiros 16 bytes do buffer 3 foram reservados para NM$ (nome) nos próximos 25, CY$ nos próximos 10, ST$ nos próximos 2 e ZP$ nos próximos 7.

Não são fornecidos campos aos 196 bytes restantes do buffer.

NOMES DE CAMPO

Nomes de campo, como NM$, AD$, CY$, ST$ e ZP$, não são variáveis string no sentido ordinário. Eles não ocupam espaço string disponível ao BASIC. Ao invés disso, eles indicam o campo do buffer que você determinou com a instrução FIELD.

Esta é a razão de se usar:

`100 FIELD 1,255 AS A$` , sem se preocupar com a disponibilidade do espaço string de 255 bytes para A$. Se você usar um nome de campo de buffer no lado esquerdo de uma instrução normal de designação, aquele nome não indicará mais o campo do buffer, e você não poderá ter acesso a ele, usando o nome anterior de campo.

Por exemplo, A$ = B$ anula o efeito da instrução FIELD acima (linha 100). Durante a entrada aleatória, a instrução GET coloca dados no buffer de 255 bytes, onde se pode ter acesso a ele usando nomes de campo determinado para aquele buffer. Durante a saída aleatória, LSET e RSET colocam dados no buffer; assim, você pode gravar (PUT) o conteúdo do buffer no arquivo de disco.

Normalmente, deseja-se usar uma variável pró-forma em uma instrução FIELD, a fim de passar por uma certa quantidade de buffer e iniciar a organização de um campo em algum lugar adiante.

Por exemplo:

```
FIELD 1, 16 AS CLIENTE$ (1), 112 AS HIST$ (1)
FIELD 1, 128 AS PROFORMA$, 16 AS CLIENTE$ (2)
```

Na segunda instrução FIELD, PROFORMA$ serve para mover a posição inicial de CLIENTE$ (2) para 129. Desse jeito, 2 sub-registros idênticos são definidos no buffer número 1. Na realidade, não usamos PROFORMA$ para colocar ou retirar dados do buffer.

O buffer agora se apresenta assim:

16	112	16	112
CL$	HIST$	CL$	HIST$
(1)	(1)	(2)	(2)

PROFORMA$

GET

Lê um registro do disco-Acesso aleatório

GET nmexp1 [nmexp2]

nmexp1 especifica um buffer de arquivo de acesso aleatório, nmexp = 1, 2,...15.
nmexp2 especifica qual o registro se deve ler no arquivo; se omitido, o registro corrente será lido.

Esta instrução lê dados de um arquivo de disco e os coloca em um buffer especificado. Antes de lê-los, você deve abrir o arquivo e determinar-lhe um buffer, isto é, uma instrução da forma:

"OPEN "R", nmexp1, Arquivo" é necessária antes da "GET nmexp2, nmexp2".

A instrução GET diz ao BASIC para ler o registro nmexp 2 no arquivo e colocá-lo no buffer nmexp. Se for omitido o número registro na GET, o BASIC lerá o registro corrente.

O "registro corrente" é o registro cujo número é maior do que aquele do último registro a que se teve acesso. A primeira vez que se tem acesso a um arquivo através de um buffer especial, o registro corrente é igualado a 1.

EXEMPLO

Instrução de programa	Efeito
`1000 OPEN "R", 1, "NOME/BAS"`	Abrir "NOME/BAS" para acesso aleatório usando buffer 1
`1010 FIELD 1,`	Estruturar o buffer
`1020 GET 1,`	Ler registro 1 no buffer 1
`1025 REM-BUFFER DE ACESSO`	
`1030 GET 1,30`	Ler registro 30 no buffer 1
`1035 REM-BUFFER DE ACESSO`	
`1040 GET 1,25`	Ler registro 25 no buffer 1
`1046 REM-BUFFER DE ACESSO`	
`1050 GET 1`	Ler registro 26 no buffer 1

Se você estiver usando registro de comprimento variável (Não comprimento fixo) e tentar ler após o final do arquivo, ocorrerá um erro.

No caso de se usar comprimento fixo nessa situação, teremos um retorno de um registro nulo e ausência de erros.

Para impedir que isso aconteça, você pode usar a função LOF e determinar o número do registro de maior numeração.

PUT

Escreve um registro no Disco-Acesso Aleatório.

PUT nmexp1 [nmexp2]

nmexp1 especifica um buffer de arquivo de acesso aleatório, nmexp = 1, 2,...15.
nmexp2 especifica o número de registro do arquivo, nmexp2 é o registro que você deseja gravar. No caso de omissão de nmexp2, o número corrente de registro será usado.

Esta instrução movimenta os dados de um buffer de arquivo para um lugar específico no arquivo (disco).

Antes de gravar (PUT) os dados no arquivo, você deve:

1 — Abrir o arquivo. Para isso determina-se um buffer e se define o modo de acesso (deve ser R).
2 — Fornece os campos ao buffer. Assim você poderá:
3 — Colocar dados no buffer utilizando instruções LSET e RSET

O BASIC faz o seguinte ao encontrar a instrução PUT nmexp, nmexp2:

— Lê a informação necessária para se ter acesso ao arquivo em disco.
— Verifica o modo de acesso para esse buffer (deve ser R)

— Arranja mais espaço em disco para o arquivo, se for necessário para comodar o registro indicado por nmexp2.
— Copia o conteúdo do buffer no registro especificado do arquivo de disco.
— Atualiza o número corrente de registro para nmexp2+1:

O "registro corrente" é o registro cujo número é maior do que o do último registro que se teve acesso: A primeira vez que você tiver acesso a um arquivo através de um buffer específico, o registro corrente se igualará a 1.

Se o número de registro que você gravar for maior que o número de registro do fim do arquivo, então nmexp2 substituirá este último.

LSET e RSET

Coloca dados em um campo do buffer do arquivo de acesso aleatório.

LSET var$ = exp$ e **RSET** var$ = exp$

var$ é um nome do campo especificado na instrução FIELD.
exp$ contém os dados a ser colocados no campo de buffer chamado var$.

Estas duas instruções possibilitam a colocação de dados de caracteres string em campos anteriores formados por uma instrução FIELD.

Por exemplo, suponha que NM$ e AD$ tenham sido definidas como nomes de campo para um buffer aleatório de arquivo. NM$ e AD$ têm comprimentos de 18 e 25 caracteres respectivamente.

Agora, desejamos colocar a seguinte informação nos campos de buffer, de modo que ela possa ser escrita no disco:

```
NOME: PROLOGICA MICROCOMPUTADORES
ENDEREÇO: LUIZ CARLOS BERRINI 1168
```

Isto é realizado com as 2 instruções:

```
LSET NM$ = "PROLOGICA MICROCOMPUTADORES"
LSET AD$ = "LUIZ CARLOS BERRINI 1168"
```

Essas instruções colocam os dados no buffer como a seguir:

PROLOGICA MICROCOMPUTADORES — **NM$**

LUIZ CARLOS BERRINI 1168 — **AD$**

Observe que os espaços preenchedores foram colocados à direita dos strings de dados em ambos os casos.

Se nós tivéssemos usado a instrução RSET ao invés da LSET os espaços preenchedores teriam sido colocados à esquerda. Esta é a única diferença entre LSET e RSET.

Por exemplo:

RSET NM$ = **"PROLOGICA MICROCOMPUTADORES"**
RSET AD$ = **"LUIZ CARLOS BERRINI 1168"**

colocam dados nos campos dessa forma:

PROLOGICA MICROCOMPUTADORES — **NM$**

LUIZ CARLOS BERRINI 1168 — **AD$**

Se o item string é muito grande para caber em um campo especificado de buffer, ele será sempre truncado à direita, isto é, os caracteres extras à direita serão ignorados.

CVD, CVI e CVS

Converte a forma string em numérica

CVD (exp$)

exp$ define um string de 8 caracteres, exp$ é normalmente o nome de um campo de buffer contendo um string numérico. Se LEN (exp$) < 8, um erro de chamada ilegal de função ocorrerá; se LEN (exp$) > 8, somente os primeiros 8 caracteres serão usados.

CVI (exp$)

exp$ define um string de 2 caracteres; exp$ é normalmente o nome de um campo de buffer contendo um string numérico. Se LEN (exp$) < 2, um erro de chamada ilegal de função ocorrerá; se LEN (exp$) > 2, somente os dois primeiros caracteres serão usados.

CVS (exp$)

expç$ define um string de 4 caracteres. Normalmente, ele é o nome de um campo de buffer que contém um string numérico. Se LEN (exp$) < 4, ocorrerá um erro de chamada ilegal de função; se LEN (exp$) > 4, somente serão usados os primeiros 4 caracteres.

Essas funções possibilitam a convenção de dados para sua forma numérica, após sua leitura do disco.

Geralmente, os dados são lidos através de uma instrução GET e são armazenados em um buffer de arquivo de acesso direto.

As funções CVD, CVI e CVS são inversas das MKD$, MKI$ e MKS$ respectivamente:

Por exemplo, suponha que o nome DESPBRUT$ indique um campo de 8 bytes em um buffer de arquivo de acesso direto, e que depois de se ler um registro o DESPBRUT$ contenha uma representação MKD$ do número 12123.38.

Então, a instrução:

`PRINT CVD (DESPBRUT$)-TAXAS` imprimirá o resultado da diferença, 12123.38 - TAXAS, enquanto que `PRINT(DESPBRUT$)-TAXAS` produzirá um erro de casamento de tipos, pois valores strings não podem ser usados em expressões aritméticas.

Utilizando-se o mesmo exemplo, a instrução:

`A#=CVK(DESPBRUT$)` designa o valor numérico 12123.38 à variável A# de precisão dupla.

EOF

Detetor de fim de arquivo.

EOF (nmexp)

nmexp especifica um buffer de arquivo, nmexp = 1, 2,..15.

Essa função verifica se teve acesso a todos os caracteres até o sinalizador de fim de arquivo, pois assim se pode evitar erros por ultrapassagem do número máximo de registros durante uma entrada seqüencial. Se nmexp especificar um arquivo aberto, EOF (nmexp) resultará em 0 (falso) quando o registro EOF não tiver sido lido, e em —1 (verdadeira) quando ele tiver sido lido.

EXEMPLOS

```
IF EOF (5) THEN PRINT "FIM DE ARQUIVO" ARQNM$
IF EOF (NM%) THEN CLOSE NM%
```

A seguinte seqüência de linhas lê dados numéricos do "DADOS/TXT" no array A( ).

Quando o último registro de dados for lido, o teste de EOF na linha 30 será positivo (—1) e o programa desviará do loop de acesso de disco, impedindo a ocorrência de erros de ultrapassagem do limite de registros.

Observe que a variável I contém o número de elementos introduzidos no array A( ).

```
5 DIM A(100) 'PRESUMINDO QUE  ESTE SEJA UM VALOR SEGURO
10 OPEN "I",1,"DADOS/TXT"
20 I% = 0
30 IF EOF (1) THEN 70
40 INPUT #1, A (I%)
50 I% = I% + 1
60 GOTO 30
70 REM PROGRAMA CONTINUA AQUI APOS A ENTRADA DE DADOS DO DISCO
```

LOC

Determinação do Número Corrente de registro

LOC (número de arquivo)

Número de arquivo é a expressão numérica especificada do buffer para arquivos atualmente abertos.

LOC é usado para determinar o número corrente do registro lido desde a abertura do arquivo LOC só será válida após a instrução GET.

EXEMPLO

```
PRINT LOC(1)
```

PROGRAMA-EXEMPLO

```
1310 A$ = "LUZO DANTAS JUNIOR"
1320 GET 1,X : X=X+1
1330 IF N$ = A$ THEN PRINT "ENCONTRADO NO REGISTRO" LOC (1)
: CLOSE : END
1340 GOTO 1320
```

Essa é apenas uma parte do programa, em alguma outra, o arquivo foi aberto e organizado um campo (FIELD) para ele. Se N$ é uma variável do campo, o número de REGISTRO no qual ele foi achado será impresso.

LOF

Número do registro de fim de arquivo.

LOF (nmexp)

nmexp especifica um buffer de aceso aleatório nmexp = 1, 2,...15.

Essa função lhe diz o número do último registro no arquivo, isto é, do registro de número mais elevado. Ela é útil em acesso aleatório e seqüencial.

Por exemplo, durante acesso direto a arquivos pré-existentes, geralmente necessita-se saber quando foi lido o último registro válido. LOF foi criada para este fim.

LOF torna-se válida quando o arquivo usado anteriormente é aberto. Se o arquivo for expandido, LOF não será válida até a execução de GET.

EXEMPLOS

```
10 OPEN "R",1,"DADOS/TXT"
20 FIELD 1,255 AS A$
30 FOR I% = 1 TO LOF(1)
40 GET 1,5
50 PRINT A$
60 NEXT
```

Na linha 30, LOF (1) especifica o maior número de registro que se teve acesso.

NOTA

Se você tentar ler números de registro além do registro de fim de arquivo, o BASIC preencherá o buffer com zeros hexadecimais e não haverá nenhum erro.

Se você desejar acrescentar ao final de um arquivo, a LOF lhe dirá onde iniciar esse acréscimo:

```
100 I% E LOF (1) + 1 "MAIOR REGISTRO EXISTENTE"
110 PUT 1, I%        "ACRESCENTE PROXIMO REGISTRO"
```

MKD$, MKI$ e MKS$

Conversão de dados numéricos em string

MKD$ (nmexp)

nm exp é considerado um número de precisão dupla.

MKI$ (nm exp)

nmexp é considerado inteiro.

(−32768 < = nm exp < 32768); se nm exp exceder esta faixa, ocorrerá um erro de chamada ilegal de função. Qualquer parte fracionária de nm exp será truncada.

MKS$ (nmexp)

nmexp é considerado um número de precisão simples.

Essas funções convertem um número em um string. Na realidade, os valores dos bytes que formam o número são inalterados; somente um byte, o especificador interno de tipo de dados, é mudado, de modo que os dados numéricos possam ser colocados em uma variável string.

Isto é:

MKD$ Resulta num string de 8 bytes.
MKI$ Resulta num string de 2 bytes.
MKS$ Resulta num string de 4 bytes.

EXEMPLOS

```
LSET DUPLICAT$ = MKI$ (I%)'
```

O nome de campo duplicatas conteria agora uma representação em 2 bytes do inteiro I%.

```
A$ = MKI$ (8/I)
```

A$ se torna uma expressão em 2 bytes da porção inteira de 8/I.

A parte fracionária é ignorada. Observe que A$, nesse caso, é uma variável string normal, não um nome de campo do buffer.

Suponha que BASEBALL/BAT (uma extensão não padronizada de arquivo) foi aberto para acesso direto usando o buffer 2, e ele foi organizado em campos desta forma:

campo:	NM$	YRS$	AUG$	HR$	AB$	RENDA$
comprimento:	16	2	4	2	4	4

NM$ deve conter caracteres string: AUG$, AB$, RENDA$, valores convertidos de precisão simples.

YRS$ e HR$, inteiros convertidos.

Suponha que se deseja escrever o seguinte registro de dados:' João jogou baisebol durante 38 anos, sua média de rebatidas foi de 123; ataques 11; defesas 32768,..., renda —13.75.

Usaremos as funções conversoras de string assim:

```
1000 LSET NM$ = "JOAO"
1010 LSET AN$ = MKI$ (38)
1020 LSET MD$ = MKS$ (.123)
1030 LSET HR$ = MKI$ (11)
1040 LSET AB$ = MKI$ (32768!)
1050 LSET GANHOS$ = MKS$ (-13.75)
```

Após essa seqüência, você poderá escrever as informações sobre João no disco, usando a instrução PUT. Quando você relê-las com a GET, haverá recessidade de conversão de dados string para numéricos, usando as funções CVI e CVS.

MÉTODOS DE ACESSO

O BASIC-DISCO fornece dois meios de acesso de arquivo:

— Aleatória, no qual você inicia a leitura ou a gravação em qualquer registro especificado. (Acesso aleatório é também conhecido por acesso direto).

Acesso seqüencial é orientado pelo fluxo; isto é, o número de caracteres lidos ou escritos pode variar, e geralmente dependem de delimitadores de dados.

Acesso aleatório é orientado pelo registro isto é, os dados são sempre lidos ou gravados em blocos de comprimento fixo chamados de registros.

Para entrada ou saída de um arquivo de disco, você deve primeiramente abrir o arquivo. Ao abri-lo, deve-se especificar o tipo de acesso desejado:

— O para saída seqüencial
— I para entrada seqüencial
— R para entrada/saída aleatória
— E (estender) para saída seqüencial iniciando pelo final do arquivo.

Você determina também um buffer de arquivo para o BASIC, para uso durante as entradas no arquivo. Este número está compreendido entre 1 e 15, mão não pode exceder o número de arquivos simultâneos, que você requisitou ao iniciar o BASIC do DOS-500. Por exemplo, se você começou o BASIC com 3 arquivos, você poderá utilizar números de buffer, 1, 2 e 3.

Se você designar um número de buffer a um arquivo, você não poderá redesigná-lo a outro, antes de fechar o primeiro.

Exemplos:

```
OPEN "O", 1, "TESTE"
```

Cria um arquivo de saída seqüencial chamado "TESTE" no primeiro drive disponível. Se esse arquivo já existir, seu conteúdo anterior se perderá. O BUFFER 1 será usado para este arquivo.

```
OPEN "R", 1, "TESTE"
```

Abre o arquivo chamado "TESTE" para uma entrada seqüencial, usando o BUFFER1.

```
OPEN "R", 1, "TESTE"
```

Abre o "TESTE" para acesso direto, utilizando o BUFFER 1. Se "TESTE" não existir, ele será criado no primeiro drive disponível. Desde que o comprimento de registro não é especificado, registros de 256 bytes serão usados.

```
OPEN "R", 1, "TESTE", 40
```

Semelhante ao exemplo anterior, mas com registros de 40 bytes.

```
OPEN "E", 1, "TESTE"
```

Abre "TESTE" seqüencialmente para escrita e posiciona em EOF (fim de arquivo).

ACESSO SEQÜENCIAL

Esse é o mais simples meio de armazenagem de dados de forma livre, sem disperdício de espaço entre os ítens dados. Você lê os ítens na mesma seqüência em que eles foram gravados.

1 — Você deve iniciar o registro no começo do arquivo.
Se os dados que você procura estão em algum ponto desconhecido, você deve ler tudo até o ponto desejado.

2 — Sempre que você abrir arquivo para saída seqüencial, você perderá o conteúdo anterior do arquivo, exceto se você usar o modo "E" ao invés de "0".

3 — Para atualizar (modificar) um arquivo seqüencial, leia o arquivo e escreva os dados alterados em um novo arquivo de saída.

4 —Informações escritas seqüencialmente geralmente inclui delimitadores (marcas) para indicar o início e o fim de ítem de dados. Para ler um arquivo seqüencialmente, você deve conhecer de antemão o formato dos dados. Por exemplo:
O arquivo consiste de linhas de texto terminadas com retornos do carro?
Ele consiste de números separados por espaços em branco? Ele consiste em informações numéricas e em texto alternados?

5 — Arquivos seqüenciais são sempre gravados como texto codificado em ASCII, um bute para cada caractere de dados. Por exemplo, o número 1.2345 necessita de 8 bytes de armazenagem em disco, incluindo os espaços em branco dianteiros e traseiros que são fornecidos. O string de texto LUZO, DEBORAH necessita de 13 bytes de armazegem em disco.

6 — Arquivos seqüenciais são sempre gravados com um comprimento de registro de 256.

Saída seqüencial: Um exemplo:

Suponha que você deseja armazenar uma tabela de conversão das unidades do sistema inglês de medidas para as unidades do sistema decimal de medidas:

Unidades do Sistema Inglês:	Unidade do Sistema Decimal:
1 polegada	2.54001 centímetros
1 milha	1.60935 quilômetros
1 acre	4046.86 metros
1 polegada cúbica	0.01638716 litros
1 galão	3.785 litros
1 quarto de galão	09463 litros
1 libra	0.45359 quilogramas

Primeiramente, temos que decidir qual será a imagem de dados a aparecer. Digamos que ela deve ser desta forma:

Unidade do sistema inglês -- Unidade sistema Decimal, fator ENTER

Por exemplo: os dados armazenados começariam:

IN — > CM, 2.54001 ENTER

O programa seguinte criará este arquivo de dados.

NOTA

X' OD' representa um retorno do carro

```
10 OPEN "O",1,"METRICO/TXT"
20 FOR I% = 1 TO 7
30       READ UNID$, FATOR
40       PRINT#1, UNID$;",";FATOR
```

```
50 NEXT
60 CLOSE
70 DATA IN->CM,2.5400, MI->KM,1.60935, ACRE->SQ.KM,4046.86E-6
80 DATA CU.IN->LTR,1.638716E-2, GAL->LTR,3.765
90 DATA LIQ.QT->LTR,0.9463, LB->KG,0.45359
```

A linha 10 cria um arquivo em disco chamado METRICO/TXT e determina o buffer 1 para saída seqüencial daquele arquivo.

A extensão/TXT foi usada devido às saídas seqüenciais sempre armazenam dados como texto codificado em ASCII.

NOTA

Se o arquivo METRICO/TXT já existir, a linha 10 fará com que todos os seus dados se percam. Aqui está o motivo: Sempre que um arquivo for aberto para saída seqüencial, será determinado o fim de arquivo (EOF) no seu início. Efetivamente, o DOS-500 esquecerá qualquer coisa escrita após este ponto. Para evitar este efeito, você pode usar "E" no lugar de "O" na linha 10.

A linha 40 imprime o conteúdo atual da UNID$ e da FACTR no arquivo.

Considerando que os ítens de string não possuem delimitadores, não é necessário imprimir aspas para explicitá-lo; a vírgula será o suficiente.

A linha 60 fecha o arquivo. O fim de arquivo (EOF) é posterior ao último ítem de dados, isto é, 0.45359.

Assim, durante a introdução, ao BASIC saberá quando terão sido lidos todos os dados.

INTRODUÇÃO SEQÜENCIAL: Um exemplo:

O programa seguinte lê os dados do arquivo METRICO/TXT em dois arrays paralelos, e depois lhe solicita a entrada do problema de conversão:

```
5 CLEAR 500
10 DIM UNID$(9), FATOR(9)     'PERMITE ATE DEZ PARES DE DADOS
20 OPEN "I",1,"METRICO/TXT"
25 I% = 0
30 IF EOF (1) THEN 70
40 INPUT #1, UNID$(I%), FATOR(I%)
50 I% = I% + 1
60 GOTO 30
70 CLOSE          'FATORES DE CONVERSOES FORAM LIDOS
100 CLS : PRINTTAB (5) "CONVERSAO DO SISTEMA INGLES PARA METRICO"
110 FOR ITEM% = 0 TO I%-1
120     PRINT TAB (9); USING " (**)    %          %    "; ITEM%,
UNID$(ITEM%)
130 NEXT
140 PRINT @ 704, "QUAL CONVERSAO (0-6)";
150 INPUT ESCOLHA%
160 INPUT "DIGITE QUANTIDADE (SISTEMA INGLES)";V
170 PRINT"O EQUIVALENTE AO SISTEMA METRICO E'"; V*FATO(ESCOLHA %)
180 INPUT "PRESSIONE <ENTER> PARA CONTINUAR"; X
190 PRINT @ 704,CHR$(31)        'LIMPA ATE FIM DA TELA
200 GOTO 140
```

A linha 20 abre o arquivo para entrada seqüencial. A entrada inicia no começo do arquivo.

A linha 30 verifica se o registro de fim de arquivo foi alcançado. Se foi, o controle desviará do loop de entrada de disco para o programa, que utiliza os dados adquiridos recentemente.

A linha 40 lê um valor no array de string UNID$ ( ) e um número no array de precisão simples FACTOR ( ). Observe que a lista INPUT emparelha com a lista PRINT # que criou o arquivo de dados (veja a seção "saída sequencial: um exemplo").

O emparelhamento não é um pré-requisito; nós poderíamos ter usado a linha abaixo e teríamos alcançado o nosso intento.

```
40 INPUT #1, UNID$ (I%): INPUT #1, FATOR(I%)
```

COMO ATUALIZAR UM ARQUIVO

Suponha que você deseja acrescentar outras unidades, no arquivo de conversão supra-citado. Você poderia simplesmente reabrir o arquivo com o modo =E e gravar (PRINT #) os dados extras, ou deixar o velho arquivo intacto e fazer um outro:

1 — Abra o arquivo por introdução seqüencial (Modo = 1)
2 — Abra um novo arquivo de dados e atualize-os, se for necessário.
3 — Introduza um bloco de dados e atualize-os, se for necessário.
4 — Envie-os para o novo arquivo.
5 — Repita as etapas 3 e 4 até a leitura completa e a atualização dos dados, e envio-os para o novo arquivo; depois proceda à etapa 6.
6 — Feche ambos os arquivos.

INTRODUÇÃO SEQÜENCIAL DE LINHAS: UM EXEMPLO

Utilizando uma entrada por linhas, você poderá escrever programas que editem outros arquivos de programas BASIC, renumerá-los, mudar LPRINTS para PRINTS etc., desde que estes programas principais sejam armazenados no formato ASCII.

O programa a seguir conta o número de linhas em qualquer (arquivo de disco de formato ASCII com a extensão/TXT).

```
10 CLEAR
20 INPUT "QUAL O NOME DO PROGRAMA"; PROG$
30 IF INSTR(PROG$,"/TXT")=0 THEN 110   'EXIGE EXTENSAO /TXT"
40 OPEN "I",1,PROG$
50 I% = 0
60 IF EOF(1) THEN 90
70 I% = I% + 1: LINE INPUT #1, TEMP$
80 GOTO 60
90 PRINT PROG$ "TEM" I% "LINHAS."
100 CLOSE: GOTO 20
110 PRINT " O ARQUIVO DEVE TER EXTENSAO  /TXT"
120 GOTO 20
```

Para programas BASIC armazenados em ASCII, cada linha de programa termina com um caractere de retorno de carro, sem ser precedido por uma alimentação de linha.

Sendo assim, a instrução LINE INPUT, na linha 70, lê automaticamente uma linha por vez na variável TEMP$. A variável I% faz, na realidade, a contagem.

Para experimentar o programa, primeiro armazene qualquer programa BASIC, usando o opção A (ASCI) (VEJA SAVE). Use a extensão/TXT

TÉCNICAS DE ACESSO ALEATÓRIO

O acesso aleatório oferece varias vantagens em relacão ao seqüencial.

1 — Ao invés de ser obrigado a iniciar a leitura no começo do arquivo, você pode ler qualquer registro especificado.

2 — Para atualizar um arquivo, você não necessita ler todo o arquivo, depois atualizar os dados, e então reescrevê-lo. Você pode reescrever ou acrescentar a qualquer registro que escolher, sem necessidade de passar por todas as outras.
3 — Acesso aleatório é mais eficiente, isto é, os dados ocupam espaço menor e são lidos e gravados mais rapidamente.
4 —A abertura de um arquivo para acesso direto permite a você gravar e ler do arquivo através do mesmo buffer.
5 —O acesso aleatório proporciona muitas instruções e funções eficientes na estruturação dos seus dados.

Ao se determinar uma estrutura, a entrada e saída diretas tornam-se muito simples.

A última vantagem, aqui listada, é também a parte mais difícil de acesso direto. Há necessidade de um pouco mais de imaginação.

Em relação aos acessos diretos, você pode imaginar um arquivo de disco como um conjunto de caixas postais dispostas em uma parede. Tanto um quanto outro são numerados e nós os denominaremos "registros".

Você pode colocar dados em um registro, com instruções semelhantes a essa:

PUT 1,5 buffer de gravação — 1 conteúdo para registro 5

GET 1,5 leia o conteúdo do registro 5 no buffer-1.

Na figura 13, assumimos um comprimento de registro como 256

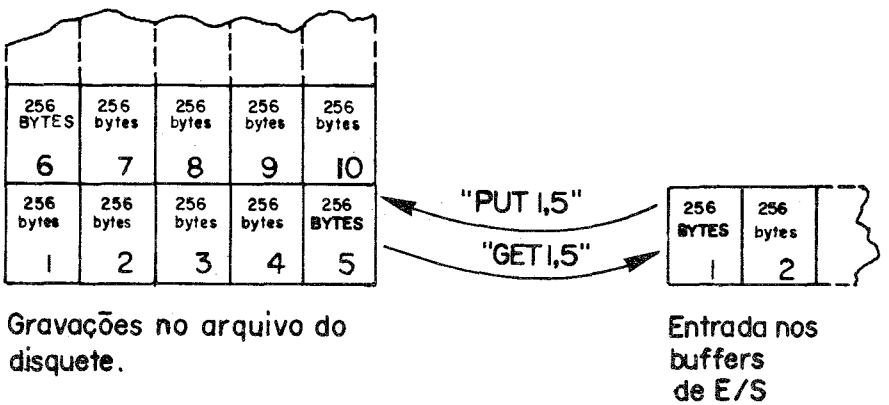


Figura 13 GET e PUT

O buffer é uma área de espera para os dados.

Antes de gravar os dados em um arquivo, eles devem ser colocados em um buffer especificado para o arquivo. Após a leitura de dados de um arquivo, você deve retirá-los do buffer.

Como você pode observar, os exemplos das instruções PUT e GET acima, os dados são transferidos para o disco através de registros.

O tamanho de cada registro é determinado por uma instrução OPEN.

ARMAZENAGEM DE DADOS EM UM BUFFER

Você deve colocar todo o registro num buffer antes da inserção do seu conteúdo no arquivo de disco. Isto é realizado:

1 — Dividindo o buffer em campos e dando-lhes nomes.
2 —Colocando dados numéricos ou strings em campos.

Por exemplo: Suponha que desejamos armazenar um glossário no disco. Cada registro consistirá de uma palavra seguida por essa definição. Nós iniciamos assim:

```
100 OPEN "R",1,"RESUMO/BAS
110 FIELD 1,16 AS WD$, 240 AS SIGNIF$
```

A linha 100 abre o arquivo chamado RESUMO/BAS (se ele não existir é criado) e cancela o acesso direto ao arquivo.

A linha 110 define dois campos no buffer 1:

WD$ consiste dos primeiros 16 bytes do buffer.

SIGNIF$ consiste dos últimos 140 bytes.

WD$ e SIGNIF$ são nomes de arquivo, agora.

O que faz nomes de arquivo diferentes?

Grande parte das variáveis string são dirigidas a uma área na memória chamada espaço string.

Este é o local onde o valor do string é armazenado.

Por outro lado, nomes de campo são dirigidos à área de buffer determinada na instrução FIELD. Assim, a instrução:

```
10 PRINT WD$ ; ":" ; SIGNIF$
```

mostra o conteúdo de 2 campos de buffer definidos acima.

Estes valores não são significativos a não ser que, primeiramente, sejam colocados dados em um buffer. LSET, RSET e GET podem ser usados, na realidade, para esta função. Iniciaremos com LSET e RSET, que são usados na preparação da saída de disco.

Nosso primeiro registro é a expressão "alinhamento a esquerda" seguida de sua definição:

```
100 OPEN "R",1,"RESUMO/BAS
110 FIELD 1,16 AS WD$, 240 AS SIGNIF$
120 LSET WD$ = "ALINHAMENTO A ESQUERDA"
130 LSET SIGNIF$ = "COLOCACAO DE UM VALOR EM
UM CAMPO DA ESQUERDA PARA A DIREITA; SE OS DADOS   NAO
PREENCHEREM O CAMPO, ESPACOS EM BRANCO SERAO ACRESCIDOS
A DIREITA; SE A INFORMACAO FOR  MUITO   COMPRIDA    OS
CARACTERES EXTRAS A DIREITA SERAO IGNORADOS. LSET,   E'
UMA FUNCAO DE ALINHAMENTO A ESQUERDA."
```

A linha 120 alinha da esquerda para a direita o valor entre aspas no primeiro campo no buffer 1. A linha 130 faz a mesma coisa com seu string entre aspas.

NOTA

RSET coloca espaços em branco à esquerda do item. O truncamento ainda seria à direita.

Agora que os dados estão no buffer, podemos gravá-los no disco com uma simples instrução PUT:

```
140 PUT 1,1
150 CLOSE
```

Esta instrução grava primeiro registro no arquivo RESUMO/BAS.

Para lê-la e imprimí-la, use a seguinte seqüência:

```
160 OPEN "R",1, "RESUMO/BAS"
170 FIELD 1, 16 AS WD$, 240 AS SIGNIF$
180 GET 1,1
190 PRINT WD$ ; PRINT SIGNIF$
200 CLOSE
```

As linhas 160 e 170 só são necessárias porque fechamos o arquivo na linha 150.

Se não o tivéssemos fechado, poderíamos nos dirigir diretamente à linha 180.

ACESSO ALEATÓRIO: UM PROCEDIMENTO GERAL

O exemplo anterior mostra a seqüência necessária para a leitura e registro usando acesso aleatório. Isto não demonstra as vantagens primordiais desta forma de acesso em especial, além de não mostrar como atualizar arquivos existentes, dirigindo-se diretamente ao registro desejado.

O programa abaixo (RESUMO/BAS), desenvolve o exemplo do glossário para mostrar algumas das técnicas de acesso aleatório para manutenção de arquivos. Mas antes de observar o programa, estude este procedimento geral para criar e manter arquivos através deste tipo de acesso.

Veja RESUMO/BAS

ETAPA	NÚMERO DE LINHA
1 — Abrir arquivo	110
2 — Fornecer um campo ao BUFFER	120
3 — Obter o registro a ser atualizado	140
4 — Visualizar conteúdo do registro corrente na tela (use CVD, CVI, CVS) antes da impressão de dados numéricos.	145-170
5 — Use LSET e RSET para colocar novos valores nos campos (use MKD$, MRI$, MKS$ com dados numéricos antes de colocá-los no buffer)	210-230
6 — Grave (PUT) o registro atualizado	240
7 — Para atualizar outro registro proceda à etapa 3, senão passe para a etapa 8	250-260
8 — Feche o arquivo	270

```
10 REM ..............RESUMO/BAS.........
100 CLS : CLEAR 300
110 OPEN "R", 1, "RESUMO/BAS"
120 FIELD 1, 16 AS WD$, 238 ZS SIGNIF$, 2 AS NX$
130 INPUT "QUE REGISTRO DESEJA ACESSAR"; R%
140 GET 1, R%
145 NX% = CVI(NX$)          'LIGACAO PARA PROXIMA ENTRADA ALFABETICA
150 PRINT "PALAVRA:         ";WD$
160 PRINT "DEFINICAO:       ";PRINT SIGNIF$
170 PRINT "PROXIMA ENTRADA ALFABETICA:.REGISTRO NR.:";NX%:PRINT
180 W$ = "": INPUT "DIGITE NOVA PALAVRA <ENTER> OU <ENTER> SE OK";W$
190 D$ = "": PRINT "DIGITE NOVA DEFINICAO <ENTER> OU <ENTER> SE OK?"
191 LINE INPUT D$
200 INPUT"DIGITE NOVA SEQUENCIA DE NUMEROS OU <ENTER> SE OK";NX%
210 IF W$ <> "" THEN LSET WD$ = W$
220 IF D$ <> "" THEN LSET SIGNIF$ = D$
230 LSET NX$ = MKI$ (NX%)
240 PUT 1, R%
245 R% = NX%                'USAM PROXIMO ALFA.
250 CLS: PRINT "DIGITE <ENTER> PARA LER PROXIMA ENTRADA ALFA,"
251 PRINT "OU REGISTRO <ENTER> PARA UMA ENTRADA ESPECIFICADA,"
252 INPUT "OU 0 <ENTER> PARA FINALIZAR";R%
260 IF 0 < R% THEN 140
270 CLOSE
280 END
```

Observe o acréscimo de um campo (NX$) ao registro (linha 120). Esse campo contera o registro seguinte em ordem alfabética. Isto possibilita o avanço através do glossário em ordem alfabética, desde que saibamos qual registro contém o primeiro verbete.

Por exemplo: suponha que o glossário contenha:

Registro	Palavra (WD$)	Defn	Indicador do próximo verbete (NX$)
	Alinhamento	.....	3
2	Byte	.....	4
3	Alinhamento Sinistrógrado	.....	0
4	Hexadecimal	.....	1

Ao lermos o registro 2 (Byte), ele nos diz que o registro 4 (Hexadecimal) é o próximo, que por sua vez nos diz que o registro 1 (alinamento destrógrado) é o próximo, etc. O último verbete, o registro 3 (alinhamento a direita, indica o zero, que significa o fim).

Desde que NX$ conterá um inteiro, primeiramente temos que converter aquele número em uma representação string de 2 bytes, utilizando MKI$ (linha 230 acima).

O programa a seguir mostra o glossário em ordem alfabética:

```
300 REM ....RESUMO/BAS....
310 CLS : CLEAR 300
320 OPEN "R",1,"RESUMO/BAS"
330 FIELD 1, 16 AS WD$, 238 AS SIGNIF$, 2 AS NX$
340 INPUT "QUAL REGISTRO E' O PRIMEIRO ALFABETICAMENTE"; N%
350 GET 1,N%
360 PRINT : PRINT WD$
370 PRINT SIGNIF$
380 N% = CVI (NX$)
390 INPUT "PRESSIONE <ENTER> PARA CONTINUAR";X
400 IF N% <> 0 THEN 350
410 CLOSE
420 END
```

ESPECIFICAÇÕES

Disquetes	5¼" MINI-FLOPPY
Organização de disquetes (disquete formatado)	FACE ÚNICA DENSIDADE DUPLA 40 TRILHAS 18 SETORES POR TRILHA 256 BYTES POR SETOR
Temperatura de operação	13 a 27°C
Requisitos de Potência (Drives externos)	110 V AC, 60 Hz, 100 W

INFORMAÇÕES TÉCNICAS

Conteúdo deste capítulo
Organização do Disco
Estrutura do Arquivo
Rotina de Sistema para Coordenação de E/S
    Blocos de Controle de Dados/Equipamento
    Gravações Lógicas e Físicas
    Chamadas das E/S DOS-500 Fundamentais.
    Rotinas Adicionais e Endereços de Armazenagem

ORGANIZAÇÃO DO DISCO

Cada disco de sistema DOS-500 contém um sistema DOS-500, uma biblioteca de comandos de uso geral e um diretório de arquivos.

Cada disco é de face simples e tem 40 trilhas de informações. Cada trilha é dividida em 18 setores de 256 bytes cada.

Normalmente, operações de escrita/leitura de dados podem começar apenas nos limites de setores, e devem envolver exatamente 256 bytes. Entretanto, o DOS-500 permite ao usuário conseguir uma máxima flexibilidade com um mínimo esforço, bloqueando e liberando automaticamente todos os acessos a arquivos para comprimentos de gravações lógicas especificados pelo usuário mesmo que isto requeira uma "interligação" entre dois setores.

A estrutura dos arquivos em um disco de sistema permite um melhor uso do espaço para arquivos no disco, pois segmenta os arquivos em um disco em vários pequenos pedaços. Estes pedaços são interrelacionados, formando um único arquivo lógico contínuo, sem que você precise saber a localização física do mesmo.

Esta estrutura elimina perda de tempo com operações de "condensação" (packing) do disco.

ESTRUTURA DE ARQUIVO

Um arquivo DOS-500 é composto por um ou mais segmentos de espaço de armazenagem. Cada segmento é composto de um a 32 "blocos" fisicamente contínuos.

Um "bloco" é a menor unidade alocável de armazenamento e é composta de três setores (Cada setor tem 256, portanto um bloco tem 768 bytes).

Como um arquivo é composto por blocos, geralmente sobra algum espaço livre no fim de cada arquivo. Este espaço livre permite que pequenas adições sejam feitas sem que se aumente fisicamente o arquivo.

Cada vez que um arquivo é ampliado (inicializado ou aumentado), blocos extras podem ser alocados para ele, dependendo do comprimento acumulado do arquivo, espaço do disco, saturação, etc. Estes blocos extras, juntamente com todos aqueles que estiverem após o que contém a marcação de EOF, são recuperados e recolocados no sistema quando o arquivo é fechado.

Um arquivo DOS-500

ARQUIVO:

NRL1	NRL2	NRL3	NRLN	EOF
EXTENSÃO 1		EXTENSÃO 2		

SEGMENTO:

BLOCO	BLOCO 2	...	BLOCO 32

BLOCO:

SETOR X	SETOR X+1	SETOR X+2

SETOR:

BYTE 1	BYTE 2	BYTE 3	...	BYTE 256

NRL: Número de registro lógico, usado para especificar um registro lógico individual definido pelo usuário.
Este registro lógico é a menor unidade de informação que pode ser endereça durante entrada ou saída do disco.

Este Registro lógico é a menor unidade de informação que pode ser endereçada durante entrada ou saída do disco (um registro físico é a unidade que realmente é lida ou escrita no disco.)

ARQUIVO: Um grupo de Registros lógicos; é a maior unidade de informação que pode ser endereçada por um comando DOS-500.

SETOR: Um registro físico, composta de 256 bytes contínuos.

BLOCO: A menor unidade alocável de armazenagem de qualquer arquivo.

EXTENSÃO: Uma seqüência física contínua de blocos.

ROTINA DE SISTEMA PARA E/S EM LINGUAGEM ASSEMBLY

Esta informação é destinada aos usuários que queiram escrever suas próprias rotinas de E/S a nível de máquina. São dadas explicações sobre a seqüência de chamada e sobre os parâmetros para cada rotina de E/S necessária. É necessário um prévio conhecido do código de máquina de Z-80.

As seguintes notações são padrão nesta seção:

(HL)=xxxx Os registros HL contém o endereço (indicam) xxxx no formato de máquina (se o endereço xxxx for 34B2H, então os valores nos registros são: H=34; L=B2). Outros pares de registros também serão indicados assim.

A=xxx O registro A contém o valor numérico de letra x em binário.
Este registro é usado para informar ao DOS-500 o código de erro para chamadas de E/S. Uma lista completa dos códigos de erros e seus significados está no final deste capítulo. Outros registros também serão indicados desta forma.

Z=OK Flag zero é ativado (OK) se houver um retorno bem sucedido das rotinas de sistema.

X'nnnn'
ou
nnnnH Endereço físico de RAM em hexadecimal (exemplo: 34B2 é X'34B2').

LRL Comprimento de Registro Lógico de 1 a 255 apenas. Você pode definir qualquer comprimento de Registro que queira até 255 bytes no máximo. O comprimento zero é um caso especial apenas para registros físicos, indicando que LRL-256 bytes.

BUFFER 256 bytes designados pelo usuário na RAM para o DOS-500 ler e escrever setores neles. Se LRL = 0, esta área fica sob responsabilidade do usuário de controlá-la antes e depois de entradas/saídas. O DOS-500 controla esta área se o LRL estiver entre 1 e 255. Não altere esta área quando usando o processamento de Registro lógico.

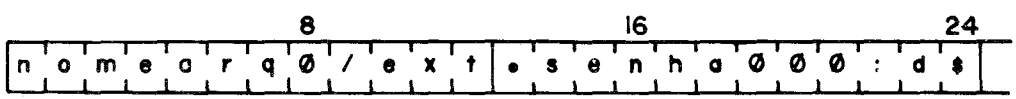
UREC Registro do Usuário (User Record) é o endereço da seqüência contínua de bytes na RAM especificado pelo usuário como sua área de Registro lógico. Seu comprimento deve ser igual a LRF. É uma área diferente de BUFFER.

LSB/MSB Byte menos significativo seguido do mais significativo.
Este é o formato padrão Z-80 para endereçamento.

$nome O "$" antecede a todas as posições no sistema e rotinas de chamada, para que não sejam confundidas com comandos DOS-500 ou utilitários. Por exemplo: $OPEN.

BCE antes de $OPEN e depois de $CLOSE

O BCE (bloco de controle do equipamento) é definido como sendo 50 bytes contínuos de RAM designados pelo usuário. Antes de $OPEN e após $CLOSE, ele é um string ASCII comprimido à esquerda (sem espaços), como em uma especificação de arquivo comum. O string é finalizado com um retorno de carro.



NOTA

/ext, senha e :d são opcionais.
$ significa um retorno de carro (X'0D')

BCE durante $OPEN

Endereço	Comprimento	Explicação
BCE +0	3	Reservado
+3	2	Endereço do Buffer Físico (LSB/MSB)
+5	1	Equivalência do Delimitador no fim do Registro corrente.
+6	1	Posição do Número do Drive do Arquivo.
+7	1	Reservado
+8	1	Equivalência do EOF do último delimitador no último Registro físico.
+9	1	CGL (comprimento do registro lógico)
+10	2	NPR (número do próximo registro $OPEN = X'0000' -LSB/MSB
+12	2	ERN (número do registro final-LSB/MSB último no arquivo).
+14	50	Reservado

NPR: O Número do Próximo Registro define qual registro será lido ou escrito na próxima chamada do sistema para uma rotina $READ ou $WRITE. Ele é automaticamente incrementado de um após cada chamada do sistema. Para se processar arquivos aleatórios, use a chamada $POSN para colocar o DOS-500 no próximo registro que você deseja transferir.

NRF o Número do Registro Final é o número do último registro presente no arquivo. Ele é colocado no diretório quando o arquivo é fechado ($CLOSE), portanto para que ele esteja correto, o usuário deve fechar os arquivos após fazer adições nos mesmos. Este valor pode, ainda, ser usado para posicionar no final do arquivo para que se façam alterações no mesmo. Para esta operação, use uma chamada para $POSN com o número de NRF +1.

$POSN é descrito mais a frente.

REGISTROS FÍSICOS E LÓGICOS NOS DOS-500

Um registro físico é definido como um setor no disco. Um setor do disco contém 256 bytes de dados do usuário. O termo "bloco" é definido como sendo 3 setores de espaço no disco. Existem 6 blocos em cada uma das 40 trilhas do disco. Um bloco é a menor quantidade que pode ser alocado pelo DOS-500. Para propósitos de programação, os registros físicos em um arquivo terão então três vezes o número de blocos alocados menos um ((3*B)-1). Todas as alocações de blocos do DOS-500 são feitas conforme a necessidade do momento de escrever e não quando é criado o arquivo.

BYTES	SETORES	BLOCOS	TRILHAS	DISCO
256	1	—	—	
768	3	1	1	—
4608	18	6	1	—
184320	720	240	40	1

Tabela de distribuição do espaço do Disco: Para um drive de disco de 5 1/4".

Uma gravação lógica é definida pelo usuário do DOS-500. Pode ter de 1 a 255 bytes de comprimento. Uma vez que o arquivo é aberto ($OPEN) com um LRL (Comprimento do registro lógico), específico, o comprimento não pode ser alterado enquanto o arquivo não for fechado ($CLOSE). Para mudar um dado arquivo, deve-se fechá-lo ($CLOSE) e reabrí-lo ($OPEN) novamente com um novo LRL

Cada abertura do arquivo especifica um único e inalterável comprimento lógico. O DOS-500 vai "acomodar" registros lógicos em um registro físico para um máximo aproveitamento do espaço do disco.

Acomodar (block) é colocar mais que um registro lógico em um registro físico. Por exemplo, quatro registros lógicos de 64 bytes podem ser acomodados em um único registro físico de 256 bytes. Um registro lógico pode ser dividido em duas partes pelo DOS-500, para que se possa preencher inteiramente um registro físico, antes de se começar a usar o próximo registro físico (isto é, os registros são interligados). Isto ocorre quando o comprimento do registro físico não é um múltiplo inteiro do comprimento do registro lógico.

Se o usuário quiser fazer sua própria acomodação, deve especificar um comprimento de registro lógico nulo (LRL=0) na hora em que for inicializar ($INIT) ou abrir ($OPEN). Desta forma, o usuário ficará com a responsabilidade de controlar os conteúdos da área de 256 bytes destinada ao buffer de registros físicos. O DOS-500 não moverá um registro lógico para o usuário se o LRL for zero; neste caso em particular ele apenas vai ler ou escrever o registro físico no buffer. Uma vez que o controle seja transferido para o seu programa voce terá cerca de 20 bytes de espaço sobrando no "stack".

Chamadas Fundamentais de E/S de DOS-500

Existem 17 rotinas DOS-500 envolvidas na manipulação de E/S de arquivos. São as seguintes:

$BACKSPACE	$POSN
$CLOSE	$PUTEXT
$DIVIDE	$RAMDIR
$IMULT	$READ
$FILPTR	$REWIND
$INIT	$SYNTAX
$KILL	$VERF
$OPEN	$WRITE
$POSEOF	

As seqüências detalhadas de chamadas e as discussões sobre cada uma dessas rotinas serão dadas a seguir. Note que todas estas chamadas de sistema usam o registro F e não recuperam seu valor quando terminam. Para se utilizar deste dado apropriadamente, você deve ler todas estas descrições e esclarecer todos os pontos que achar duvidosos usando outros livros como referência. Se você for bem sucedido nesta leitura, vai chegar a conclusão que o DOS-500 é uma ferramenta muito útil nas suas idéias de programação.

$INIT — 17440/X'4420'

$INIT é um ponto de partida para o DOS-500 que criará uma nova entrada de arquivo no diretório e abrirá o BCE para este arquivo. $INIT procura no diretório pelo nome "nomearg" dado no BCE. Se for encontrado tal nome, a rotina $INIT apenas vai abrir o arquivo para uso. Se o nome não for encontrado, um novo arquivo é criado com o nome dado.

Condições de entrada
(HL)=BUFFER (veja o começo desta seção para esclarecimentos)
(DE)=BCE
B=LRL
CALL $INIT
Condições de Saída
IY=alterado
Z=OK
Carry flag C estará na condição "ON" se um novo arquivo foi criado
A = Código de Erro DOS-500 (Os códigos de erro são enumerados no final deste manual).

$OPEN - 17444/X'4424'

$OPEN fornece um modo de abrir o BCE de um arquivo que já exista no diretório. O BCE deve conter o filespec do arquivo a ser aberto, antes de se ativar a rotina $OPEN.

Condições de Entrada
(HL) = BUFFER
(DE) = BCE
B = LRL
CALL $OPEN

Condições de Saída
Z = OK
Z = 0 se o arquivo não existe.
A = Código DOS-500 de erro
IY = Alterado

$POSN — 17474/X'4442'

$SPOSN posiciona um arquivo para ler ou escrever um registro lógico aleatoriamente escolhido. Como se trata de registro lógico deve-se fazer os cálculos adequados para se localizar aquele que contenha os dados desejados. Uma rotina $POSN seguida de $READY ou $WRITE transferirá o registro de ou para a RAM.

Note que posicionar no registro lógico zero faz com que o arquivo seja preparado para ler o primeiro registro lógico. Para posicionar no final do arquivo, para que se possa acrescentar novos registros nele, use o número NRF + 1.

Condições de Entrada
(DE) = BCE (deve ser preparado previamente)
BC = Número do registro lógico a ser posicionado.
CALL $POSN

Condições de Saída
Z = OK
A = Código DOS-500 de erro

$READ — 17462/X'4436'

Se NRL for diferente de zero, $READ transfere o registro lógico identificado pelo número NPR (número do próximo registro) do BCE para a área endereçada como UREC, com o comprimento LRL, como foi estabelecido no momento da abertura. O registro vem do "BUFFER" definido também no momento da abertura do arquivo.

Se o DOS-500 precisar ler um novo registro físico para satisfazer as necessidades, ele o fará. Registros lógicos "separados" serão remontados quando necessário. $READ automaticamente incrementa NPR de 1 no BCE após completar a transferência. $INIT e $OPEN fazem NPR = X'0000' para que se leia o primeiro registro com o primeiro $READ.

Se LRL for zero, $READ transfere um registro físico do arquivo no drive para o BUFFER definido na abertura. Os registros HL são ignorados. $READ incrementa NPR como na condição anterior.

Condiçõesa de Entrada
(HL) = UREC se NRL for diferente de zero e não é usado em caso contrário.
(DE) = BCE
CALL $READ

Condições de Saída
Z = OK
A = Código DOS-500 de erro (EOF = X'1C' ou X'LD')
(veja os erros 28 e 29 para EOF ou NRF).

$WRITE — 17465/X'4439'

Se LRL for diferente de zero, a rotina $WRITE transfere o registro lógico da área de RAM endereçada como UREC, com o comprimento LRL, definido na abertura.

O registro vai para o BUFFER, que também é definido na abertura. O DOS-500 escreverá um registro físico, se necessário. A "separação" de registros será manipulado pelo DOS-500, conforme necessário. As rotinas $INIT e $OPEN fazem o valor do NPR do BCE igual a X'0000' para que o primeiro registro seja transferido pelo primeiro $WRITE. Após cada registro lógico ser transferido, o valor de NPR é incrementado de 1.

Se LRL é zero, $WRITE transfere um registro físico do BUFFER para o disco arquivo usando o NPR no BCE. O BUFFER é definido na abertura do arquivo ($INIT ou $OPEN). O valor de NPR é atualizado no BCE, como no caso anterior, após o $WRITE.

Condições de entrada
(HL) = UREC se LRL for diferente de zero e não é usado se for igual.
DE = BCE
CALL $WRITE

Condições de saída
Z = OK
A = Código DOS-500 de erro

$VERF — 17468/X'443C'

A única diferença entre $VERF e $WRITE é que a primeira escreve um registro físico no disco e então lê para se certificar da correção do que foi transferido. Esta leitura é realizada em uma área de RAM que não é definida pelo usuário.

Esta área especial e o buffer original são comparados byte a byte, para se certificar de que a transferência foi bem sucedida.

Condições de Entrada
(HL) = O mesmo que no $WRITE
(DE) = BCE
CALL $VERF

Condições de Saída
Z = OK
A = Código DOS-500 de erro

$PUTEXT — 17483/X'444B'

Esta rotina coloca uma extensão no nome do arquivo se não houver. Uma extensão em um nome de arquivo pode ser útil para se identificar o tipo de dados que ele contém.

Condições de Entrada
(DE) = BCE
(HL) = a extensão a ser adicionada ao arquivo
CALL $PUTEXT

Condições de Saída
Nenhuma

$BKSPC — 17477/X'4445'

Esta rotina posiciona o indicador de registro do arquivo no registro anterior.

Condições de Entrada
(DE) = BCE
CALL $BKSPC

Condições de Saída
Z = Posição Válida
NZ = Posição inválida no arquivo

$REWIND — 17471/X'443F'

Recoloca no início do arquivo. Esta rotina posiciona o indicador de arquivo no primeiro registro do arquivo. Isto é útil quando o mesmo arquivo deve ser processado mais que uma vez.

Condições de Entrada
(DE) = BCE
CALL $REWIND

Condições de Saída
Z = Especificação boa de arquivo
NZ = Especificação má de arquivo

$POSEOF — 17480/X'4448'

Posiciona no final do arquivo. Esta rotina posiciona o indicador de arquivo no último registro de arquivo. Isto pode ser usado para se ampliar um arquivo de acesso sequencial.

Condições de Entrada
(DE) = BCE
CALL $POSEOF

Condições de Saída
Z = Especificação boa de arquivo
NZ = Especificação má de arquivo

$SYNTAX — 17436/X'441C'

Move uma especificação de arquivo para o BCE. Esta rotina toma uma especificação de arquivo, verifica sua validade e a coloca no BCE para que o arquivo seja aberto.

Condições de Entrada
(HL) = nome do arquivo
(DE) = BCE
CALL $SYNTAX

Condições de Saída
Z = Especificação boa de arquivo
NZ = Especificação má de arquivo

$DIVIDE — 17489/X'4451'

Esta rotina pega um dividendo de 16 bits e um divisor de 8, após a divisão, o quociente substitui o dividendo De 16 bits e o resto substitui o divisor.

Condições de Entrada
HL = dividendo
A = divisor
CALL $DIVIDE

Condições de Saída
HL = quociente
A = resto (0 Indica sem resto)

$DMULT — 17486/X'444E'

A rotina de multiplicação usa um multiplicando de 16 bits e um multiplicador de 8. Após a multiplicação se realizar, o produto toma o lugar do multiplicando de 16 bits.

Condições de Entrada
HL = multiplicando
A = multiplicador
CALL $DMULT

Condições de Saída
H = byte e maior ordem
L = byte de ordem média
A = byte de menor ordem

H	L	A
Alta	Média	Baixa

$RAMDIR — 17040/X'4290'

Esta rotina permite a você examinar o diretório de um disco (uma determinada entrada ou todo o diretório), ou o espaço livre no disco. A informação é escrita em um buffer determinado pelo usuário. Apenas arquivos que não sejam de sistema são incluidos no diretório da RAM.

Condições de Entrada
HL = Buffer da RAM. Se C=0, a dimensão = 1761 (número # max *22+1). Se C=1 a 96, dimensão = 22. Se C = 255, dimensão = 64.
B = Número do drive especificado.
C = Chave de Função:

Conteúdo do Registro	**Resultados:**
0	Coloca todo o diretório na RAM (Veja "Formato do Diretório da RAM)
1-96	Fornece registro específico do diretório, se existir
255	Fornece a informação sobre o espaço livre.

CALL $RAMDIR

Condições de Saída
NZ = Erro ocorrido
Z = Sem erro
(HL) = diretório ou informação em espaço livre

FORMATO DO DIRETÓRIO DA RAM

O diretório é composto de vários registros, um por arquivo. Todos os valores são hexadecimais. Cada registro colocado na memória RAM do usuário tem o seguinte formato:

Número do byte	**Conteúdo**
0-14	nome do arquivo/ext: d (comprimidos a esquerda e seguidos de espaços)
15	Nível de proteção, binário de 0 a 6
16	byte 'fim de arquivo', binário de 0 a 255
17	Comprimento do registro lógico; binário de 0 a 255
18-19	último número de setor no arquivo; binário LSB/MSB
20-21	Número de blocos alocados (LSB/MSB) em binário
22	"+" (marca o final da lista do diretório)

FORMATO DA MENSAGEM DE ESPAÇO LIVRE

nnnn BLOCOS LIVRES
Onde nnnn é um número decimal. A mensagem é codificada em ASCII.

$FILPTR — 17037/X'428D'

Esta rotina fornece informação sobre qualquer arquivo que esteja aberto no momento. Ela permite que você obtenha o número do drive e o número de arquivo lógico para qualquer arquivo, e pode ser usada em conjunto com $RAMDIR.

Condições de Entrada
(DE) = Bloco de Controle de Dados (BCD) definido quando o arquivo foi aberto.
CALL $FILPTR

Condições de Saída
NZ = Erro ocorrido
Z = Sem erro. Os seguintes registros são ativados:
B = Drive que contém o arquivo (0,1,2 ou 3)
C = Número lógico do arquivo (de 1 a 96)

NOTA

Esta rotina opera com arquivos feitos pelo usuário.

$CLOSE — 17448/X'4428'

$CLOSE fecha um arquivo depois que o processamento termina. É muito importante fazer um $CLOSE em todo arquivo aberto, antes do programa terminar.

Se o arquivo não for fechado, a entrada do diretório para este arquivo ficará incorreta se algum registro for feito nele. Outros casos podem ocorrer, mas não os discutimos aqui; porém, é muito importante para o DOS-500 que todos os cuidados sejam tomados para gerenciamento dos arquivos.

Condições de Entrada
(DE) = BCE
CALL $CLOSE

Condições de Saída
Z = OK
A = Código DOS-500 de Erro

$KILL — 17452/X'442C'

$KILL elimina a entrada do diretório para um arquivo e libera a armazenagem em disco. O arquivo pode estar aberto ou fechado que $KILL opera da mesma maneira.

Condições de Entrada
(DE) = BCE
CALL $KILL

Condições de Saída
Z = OK
A = Código DOS-500 de Erro

Rotinas Adicionais e Endereços de Armazenagem

$JP2DOS — 16429/X'402D'

Esta rotina transfere o controle para o DOS-500 ATIVO

Condições de Entrada
JP $JP2DOS

Condições de Saída
Nenhuma

$DATE — 12339/X'3033'
$TIME — 12342/X'3036'

Estas rotinas fornecem a data e a hora no seguinte formato:
Data: MM/DD/AA
Hora: HH:MM:SS

Condições de Entrada
(HL) = Buffer de oito bytes para receber o texto de data ou hora
CALL $DATE
CALL $TIME

Condições de Saída
(HL) = Texto de hora ou data

$DATLOC — 16922/X'421A'
$TIMLOC — 16919/X'4217'

Estas posições armazenam a data e a hora em formato binário:
$DATLOC (três bytes): MM DD AA
$TIMLOC (três bytes): SS MM HH

$ERRDSP — 17417/X'4409'

Esta rotina emite uma mensagem de erro determinada pelo conteúdo do acumulador (A). Este registro contém o código de erro zero (nenhum erro) após se completar qualquer rotina de sistema.

Condições de Entrada
A = Código DOS-500 de erro. Em um código DOS-500 de erro, os bits 6 e 7 são normalmente desativados (off). Desta forma, $ERRDSP os interpreta como controle.

Bit	Ativados (Set)	Desativados (condição normal)
7	Retorna à rotina de chamada para conclusão	Retorna ao DOS-500 para conclusão
6	Fornece mensagem detalhada	Fornece apenas o número do erro.

CALL $ERRDSP

Condições de Saída
Nenhuma

EXEMPLO DE UTILIZAÇÃO

```
1 6                    13              28

   CALL                $SYSRTN         ; Qualquer rotina do sistema
   JR                  Z,OKGO          Verifica erro
   Aproxima instrução seta
   bit 6 (mensagem de erro
   detalhada) e bit 7 (retor-
   na ao ponto de chama-
   da após imprimir mensa-
   gem)
   OR                  COH             ; Binário 11000000
   CALL                $ERRDSP         ; Chama rotina de emissão da mensagem
;  De erro
;  Continua aqui após retornar da rotina de erro
;  A rotina de tratamento erro deve ser incluída aqui
;  ;
;  ;
;  ;
OKGO; continuação após $SYSRTN sem erro
```

$DSPDIR — 17433/X'4419'

Este comando mostra na tela a listagem de diretório de todos os arquivos do usuário que não estiverem protegidos em um dado drive.

Condições de Entrada
(X'4271') = Número do drive em ASCII "0", "1", "2", ou "3"
CALL $DSPDIR

Condições de Saída
Todos os registros são alterados

$COMDOS — 17049/X'4299'

Esta rotina executa um comando DOS-500 e retorna para DOS-500 ATIVO

Condições de Entrada
(HL) = Texto do comando DOS-500, terminado por X'0D.
JP $COMDOS

Condições de Saída
Nenhuma

$CMDDOS — 17052/X'429C'

Esta rotina executa um comando DOS-500 e retorna para a rotina de chamada.

Condições de Entrada
(HL) = Texto de comando DOS-500 finalizado por X'0D'

Condições de Saída
Todos os registros são alterados.
Cuidado: Os comandos DOS-500 irão ocupar a RAM até a posição X'6FFF'

$CMDTXT — 16933/X'4225'

Este é o endereço inicial de um buffer que contém a última linha de comando introduzida em DOS-500 ATIVO. Usando este buffer, seu programa poderá recuperar parâmetros que foram introduzidos com a linha de comando anterior.

Por exemplo, dado um programa chamado EDITOR/CMD, queremos que o operador selecione um nome de arquivo de entrada quando o programa é carregado e executado em DOS-500 ATIVO.

DOS 500 ATIVO
EDITOR MEUARQ

O programa EDITOR, pode executar o nome de arquivo no buffer $CMDTXT.

NOTA

Quando for usado como entrada para um programa, (HL) = Primeiro caractere que não seja um espaço após o nome programa.

$MEMEND — 17425/X'4411'

Esta posição de memória contém o maior endereço disponível. Geralmente é o próprio fim físico da RAM, mas pode ser mudado para propósitos específicos.

O endereço está colocado, respectivamente, LSB e MSB.

MANUTENÇÃO E DETECÇÃO DE FALHAS

Se, por acaso, você tiver problemas de operação no seu CP-500, verifique os seguintes ítens referentes ao problema apresentado, conforme a tabela abaixo. Caso seja necessário, recorra ao Manual do CP-500.

PROBLEMA	SOLUÇÃO
Os motores dos drives de disco funcionam ininterruptamente, quando o computador está ligado.	Verifique a seqüência de conexões externas dos drives
Computador não carrega o DOS-500.	1 — Certifique-se da correta introdução do disquete DOS-500 no Drive 0. 2 — Certifique-se da correta conexão dos periféricos.
Mensagens de erro.	Procure a mensagem no capítulo de mensagens de Erro do DOS-500 ou do BASIC. A solução deve estar relacionada.
Erros freqüentes de E/S de disco.	1 — Disquete parcialmente apagado. Copie o disquete e reformate-o. 2 - Disquete gasto. Copie um novo disquete através do original. 3 — Os drives de disco necessitam limpeza e alinhamento feitos pela assistência técnica da PROLÓGICA.

MANUTENÇÃO

Os drives de disco devem ser mantidos limpos e alinhados convenientemente para obtenção de operações confiáveis. Esses procedimentos deveriam ser feitos pela assistência técnica da PROLÓGICA conforme o esquema abaixo:

TIPO DE USO	INTERVALOS DE MANUTENÇÃO
Processamento de Dados no Comércio	Todo mês para uso médio.
Uso doméstico ocasional.	Cada 8-10 meses; com maior freqüência se necessário..

Para instruções mais detalhadas, consulte a seção de Manutenção e Deteção de falhas no manual do CP-500.

UTILIZAÇÃO DA MEMÓRIA PELO SISTEMA OPERACIONAL

O DOS-500 se encontra armazenado no disquete de sistema do seu Sistema de Disco.

Cada vez que o computador for ativado, ou reiniciado, o programa DOS-500 mestre será carregado da RAM, para que possa ser executado.

O DOS-500 ocupa 40000 bytes de espaço no disquete, no entanto, somente uma parte está na RAM. Isto é possível, pois o DOS-500 é dividido em vários módulos independentes.

O módulo residente está sempre na memória. Ele consiste em drives de Entrada/Saída tabelas, interpretadores de comandos e outras rotinas importantes.

Os módulos adicionais são carregados quando necessário e substituídos por outros conforme a necessidade.

NOTA

Após a introdução de um comando qualquer geralmente se ouvirá o DOS-500 acio-nando o Sistema de Disco. Ele estará carregando um módulo substituto que contém o código necessário para completar o comando

O mapa de memória na figura 14 mostra como o DOS-500 utiliza o espaço de memória disponível

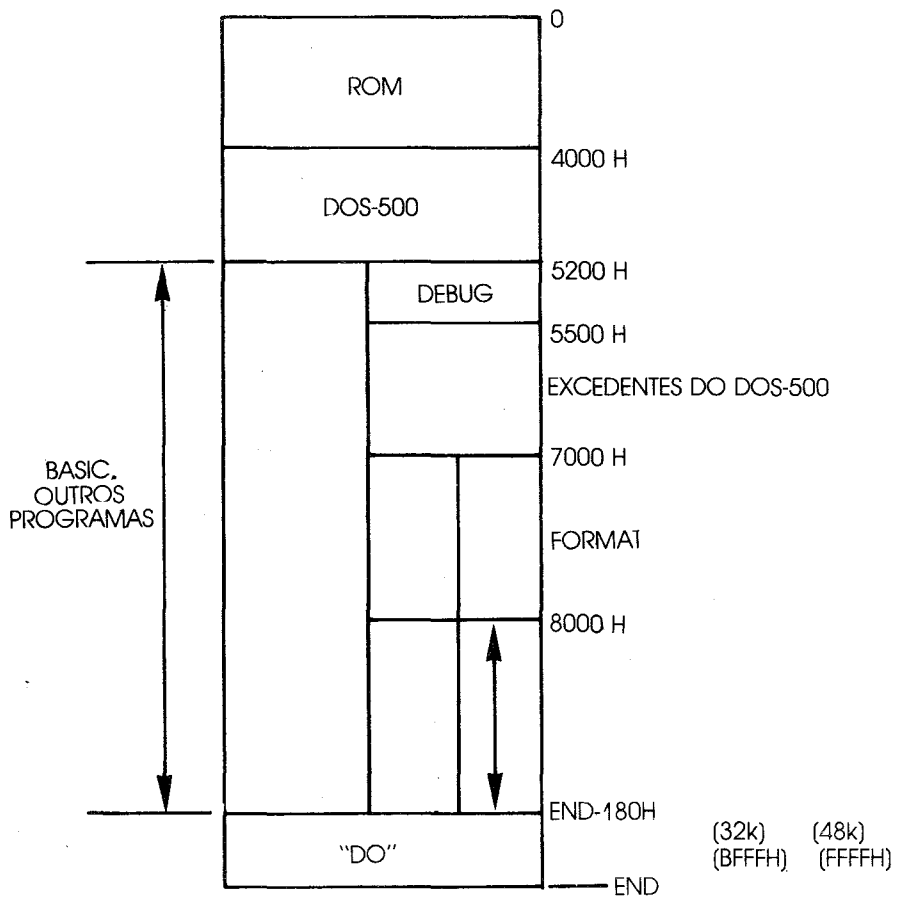


Figura 14
Mapa de Memória de CP-500

Códigos e Mensagens DOS-500 de erro

0	Nenhum erro foi encontrado
1	Erro CRC Durante E/S do Disco
2	Drive de Disco Não Está no Sistema
3	Perda de Dados Durante E/S do Disco
4	Erro CRC Durante E/S do Disco
5	Setor do Disco Não Foi Encontrado
6	Falha no Hardware do Disco
7	**Código de Erro Indefinido**
8	Drive Não Está Preparado
9	Tentativa Ilegal de E/S
10	Parâmetro de Comando Pedido Não Foi Encontrado
11	Parâmetro Ilegal de Comando
12	Tempo Esgotado no Drive
13	Tentativa de E/S Ilegal Para um Disco Sem DOS-500
14	Falha de Gravação na E/S de Disco
15	Disco Com Proteção Contra Gravação
16	Número do Arquivo Lógico Ilegal
17	Erro na Leitura do Diretório
18	Erro na Gravação do Diretório
19	Nome Inválido de Arquivo
20	Erro de Leitura GAT
21	Erro de Gravação GAT
22	Erro de Leitura HIT
23	Erro de Gravação HIT
24	Arquivo Não Encontrado
25	Acesso a Arquivo Negado Devido a Proteção por Senha
26	Espaço do Diretório Completo (cheio)
27	Espaço do Disco Completo (cheio)
28	Tentativa de ler após final do arquivo
29	Tentativa de ler fora dos limites do arquivo
30	Não há mais extensões disponíveis
31	Programa não encontrado
32	Número Inválido de Drive
33	**Código de Erro Indefinido**
34	Tentativa de usar um arquivo que não é programa, como se fosse
35	Falha de memória durante carregamento de programa
36	**Código de Erro Indefinido**
37	Acesso a arquivo Negado Devido a Proteção por Senha
38	Tentativa de E/S para um arquivo não aberto
39	Parâmetro de Comando Inválido
40	O Arquivo já Está no Diretório
41	Tentativa de abrir um arquivo já aberto
42	Tamanho inválido de registro

Código e Mensagens de Erro do BASIC- DISCO.

51	Overflow de Campo
52	Erro interno
53	Número indevido de arquivo
54	Arquivo não encontrado
55	Modo indevido de arquivo
58	Erro de E/S de disco
62	Disco completo (cheio)
63	Entrada após o fim do programa
64	Número indevido de registro
65	Nome indevido de arquivo
67	Instrução direta no arquivo
68	Arquivos em demasia
69	Disco protegido contra gravação
70	Acesso ao arquivo

NOTA

Erros de disco não podem ser simulados através de instruções de ERRO

ÍNDICE REMISSIVO

Tópico Página Capítulo

RELATÓRIO DE SOFTWARE

Utilize este formulário para descrever erros encontrados na documentação ou problemas no equipamento.

DATA: _____/_____/_____ VERSÃO: _____

Descrição do Problema

Erros de Documentação:

Manual: _____

Número das páginas: _____

Descrição dos Erros:

Preencha os Dados a Seguir:

NOME: _____

TELEFONE: _____

ENDEREÇO:_____ CEP:_____ ESTADO:_____

Envie este Farmulário para:

PROLÓGICA Indústria e Comércio de Microcomputadores Ltda.
Av. Eng.º Luís Carlos Berrini, n.º 1.168 - 1.º andar
Centro de Treinamento
CEP 04571 - São Paulo - SP

Este livro foi impresso pela
artes gráficas guaru s/a.
Rod. Presidente Dutra, km 214
Fone: 912-1388 - Guarulhos