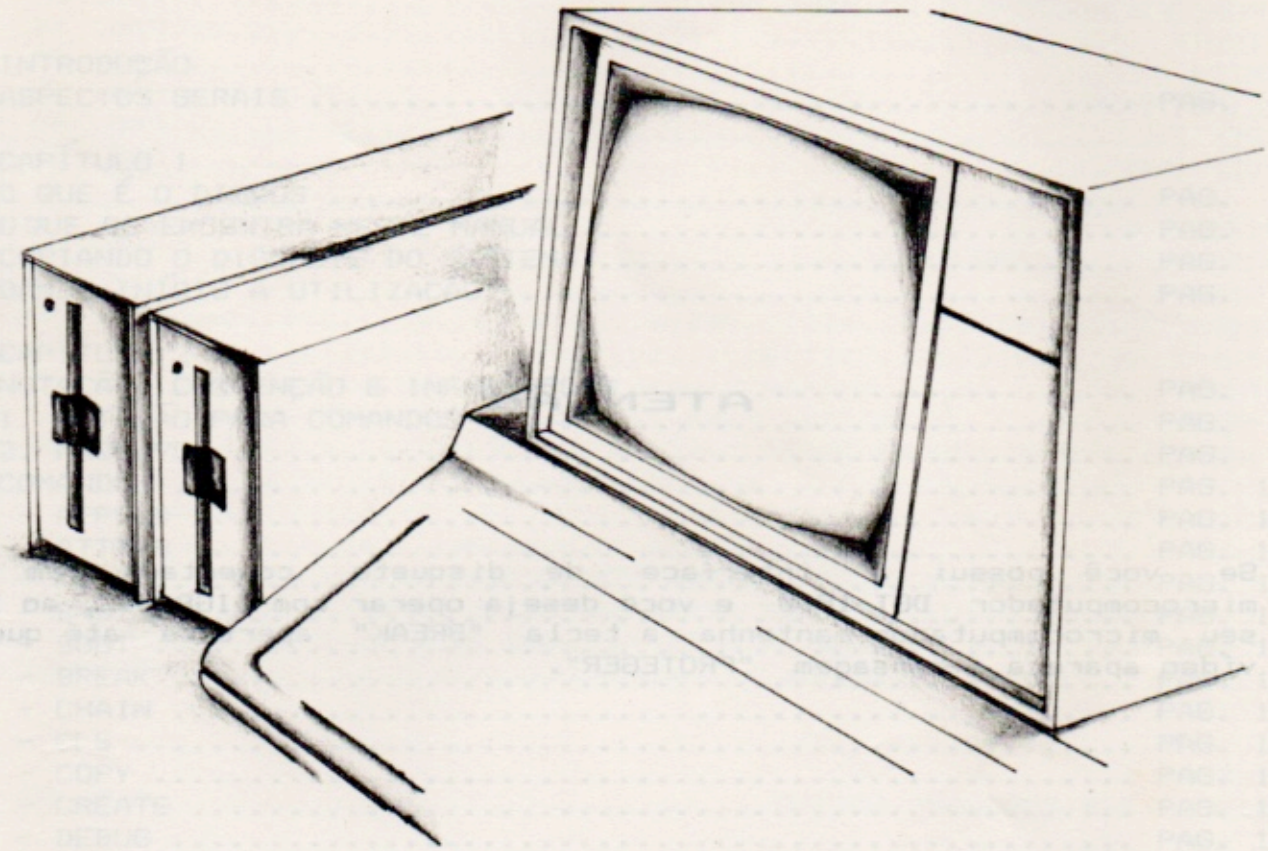
DIGDOS-100

DIGITUS



DIGDOS-100
DIGITUS



DIGITUS criou para o DGT-1000, a interface controladora de unidades de discos flexíveis, - Disquetes - que permitem o armazenamento de uma massa de dados de 184K bytes por unidade. Com isto, torna-se compatível a eficiência, velocidade e capacidade de armazenamento das unidades de discos magnéticos de sistemas de grande porte, com a facilidade de operação e reduzido volume físico de um gravador cassete.

ATENÇÃO

Se você possui a interface de disquete conectada em seu microcomputador **DGT-1000 e você deseja operar com DIGBASIC, ao ligar** seu microcomputador mantenha a tecla "BREAK" apertada até que no vídeo apareça a mensagem "PROTEGER".

SUMÁRIO

INTRODUÇÃO
ASPECTOS GERAIS ..... PAG. 1

CAPÍTULO 1
O QUE É O DIGDOS ..... PAG. 3
O QUE SE ENCONTRA NESTE MANUAL ..... PAG. 4
COPIANDO O DISQUETE DO SISTEMA ..... PAG. 5
DANDO INÍCIO A UTILIZAÇÃO ..... PAG. 7

CAPÍTULO 2
NOTAÇÃO, CONVENÇÃO E INFORMAÇÕES ..... PAG. 8
1. NOTAÇÃO PARA COMANDOS ..... PAG. 8
2. ARQUIVO ..... PAG. 9
COMANDOS ..... PAG. 10
- APPEND ..... PAG. 10
- ATTRIB ..... PAG. 10
- AUTO ..... PAG. 12
- BASIC2 ..... PAG. 12
- BOOT ..... PAG. 12
- BREAK ..... PAG. 13
- CHAIN ..... PAG. 13
- CLS ..... PAG. 14
- COPY ..... PAG. 14
- CREATE ..... PAG. 17
- DEBUG ..... PAG. 17
- DIR ..... PAG. 18
- DO ..... PAG. 19
- DUMP ..... PAG. 19
- ERROR ..... PAG. 20
- FORMAT ..... PAG. 21
- FREE ..... PAG. 22
- HIMEN ..... PAG. 22
- JKL ..... PAG. 22
- KILL ..... PAG. 22
- LIB ..... PAG. 23
- LIST ..... PAG. 23
- LOAD ..... PAG. 23
- PRINT ..... PAG. 23
- R ..... PAG. 23
- RENAME ..... PAG. 23

CAPÍTULO 3
DISK BASIC ..... PAG. 24
1- INTRODUÇÃO AO DISK BASIC ..... PAG. 24
2- CONSTANTES OCTAIS E HEXADECIMAIS ..... PAG. 27
3- COMANDOS DE EDIÇÃO DO DISK BASIC ..... PAG. 28
4- COMANDOS DE EDIÇÃO ADICIONAIS ..... PAG. 29
5- RENUM ..... PAG. 29

6- REF ..... PAG. 30
7- COMANDOS UTILITÁRIOS ..... PAG. 31
CMD "C" ..... PAG. 31
CMD "E" ..... PAG. 31
CMD "doscmd" ..... PAG. 32
CMD "J" ..... PAG. 32
CMD "O" ..... PAG. 33
CMD "S" ..... PAG. 34
8- FUNÇÕES ADCIONAIS DO DISK BASIC ..... PAG. 34
INSTR ..... PAG. 34
MID$ ..... PAG. 35
LINEINPUT ..... PAG. 35
DEF FN ..... PAG. 36
DEF USR ..... PAG. 37

CAPÍTULO 4
MANIPULAÇÃO DE ARQUIVOS ..... PAG. 38
1- SAVE ..... PAG. 38
2- LOAD ..... PAG. 38
3- RUN ..... PAG. 39
4- KILL ..... PAG. 39
5- MERGE ..... PAG. 39
6- AÇESSO A ARQUIVOS DE DADOS ..... PAG. 40

CAPÍTULO 5
ARQUIVOS ..... PAG. 43
A- COMANDOS DE MANIPULAÇÃO DE ARQUIVOS ..... PAG. 43
B- COMANDOS DE MANIPULAÇÃO DE ARQUIVOS DE AC DIRETO ..... PAG. 47
C- CARACTERÍSTICAS DOS ARQUIVOS DE ACESSOS DIRETOS E SEQÜENCIAIS
..... PAG. 51

APÊNDICE 1
CHAINBLD/BAS PROGRAMA EDITOR ..... PAG. 52

APÊNDICE 2
CONVERTE/JCL ..... PAG. 54

APÊNDICE 3
DEBUG ..... PAG. 55

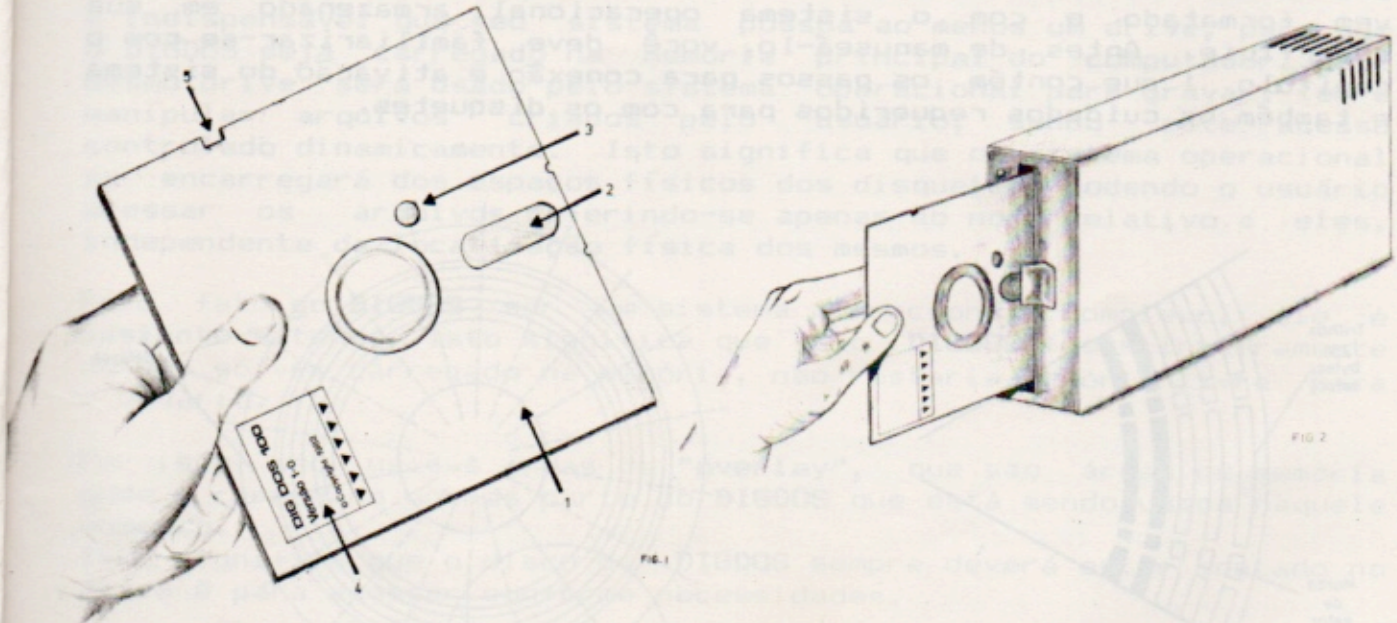
INTRODUÇÃO

ASPECTOS GERAIS

FIGURA 1 e 2

O disquete, ver fig.1, é similar a uma fita cassete, sendo uma superfície de plástico flexível, coberta por um material ferromagnético, dentro de um invólucro protetor (1 da fig.). Possui uma abertura no invólucro (2 da fig.) para permitir o contacto da cabeça de leitura e gravação do drive com a superfície magnética, um orifício (3 da fig.) utilizado para a localização relativa dos setores, uma etiqueta (4 da fig.) usada para a identificação do disquete e por fim, uma ranhura na aresta lateral (5 da fig.) que permite a proteção física contra gravações.

A figura 2 mostra como o disquete deve ser montado no drive.



ATENÇÃO:

1 - A luz do painel, indica que o motor do drive está acionado. Nunca retire ou coloque o disquete no drive com a luz acesa. Aguarde que ela se apague para a manipulação do disquete.

2 - A porta do drive deve estar fechada, sempre que o disquete estiver montado no drive. Abra-a e feche-a cuidadosamente, a fim de evitar danos ao disquete.

3 - Nunca desligue a alimentação do DRIVE ou do COMPUTADOR sem antes retirar o disquete de dentro do drive.

4 - Nunca toque a superfície magnética do disquete; exposta pela abertura do invólucro.

FIGURA 3

O disquete tem seu espaço físico dividido em trilhas e setores, conforme mostra a figura 3. Esta divisão física da superfície do disquete feita magneticamente pelo sistema, é chamada de FORMATAÇÃO, que segue um padrão rígido, determinado pelo sistema operacional (em 1.2 veremos o que é sistema operacional). O disquete,do modo como é fornecido pelo fabricante, isto é, em branco, não pode receber dados antes que seja formatado, ou seja, ter sua superfície dividida magneticamente em trilhas e setores. Após a formatação, o disquete fica dividido em 40 trilhas. Cada trilha é dividida em 18 setores de 256 bytes cada. Porém a menor unidade que se pode acessar no disquete é um grão, que corresponde a 5 setores. O disquete que acompanha a interface para disquetes já vem formatado e com o sistema operacional armazenado em sua superfície. Antes de manuseá-lo, você deve familiarizar-se com o capítulo 1 que contém os passos para conexão e ativação do sistema e também os cuidados requeridos para com os disquetes.

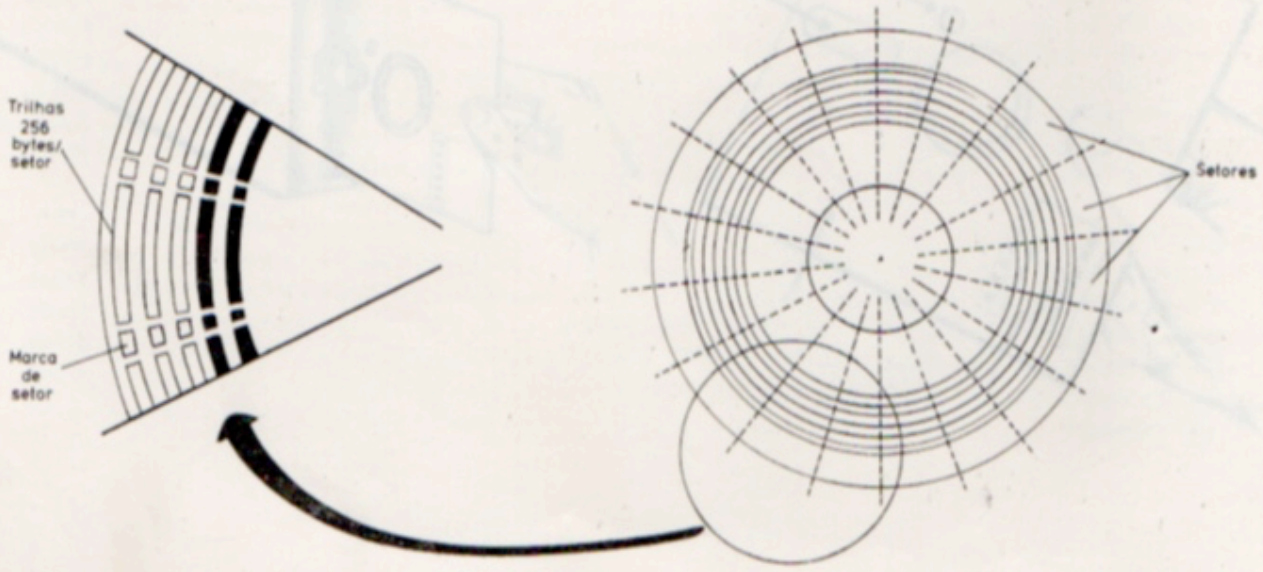


FIG.3

CAPÍTULO 1

O QUE É O DIGDOS ?

SISTEMA OPERACIONAL

- Sistema operacional é um programa principal agindo eficientemente sobre a complexidade do computador, fazendo com que sua operação se torne realmente simples.

Suas funções:
- Interfaciar a CPU e os periféricos.
- Receber e interpretar os comandos do operador.
- Fornecer rotinas fundamentais usadas em inúmeros programas.

DIGDOS

- O DIGDOS é um sistema operacional, compreendendo uma série de rotinas e utilitários de gerenciamento de arquivo. Ele torna invisível ao usuário, uma série de operações complexas que garantem a eficiente manipulação de arquivos armazenados nos disquetes.
Observe que todas as informações são gravadas nos disquetes em forma de arquivo, sendo esses, programas, dados, ou mesmo o próprio DIGDOS.
E indispensável que seu sistema possua ao menos um drive, para que o DIGDOS seja carregado na memória principal do computador. Este mesmo drive será usado pelo sistema operacional para gravar, ler e manipular arquivos criados pelo usuário, sendo este acesso controlado dinamicamente. Isto significa que o sistema operacional se encarregará dos espaços físicos dos disquetes, podendo o usuário acessar os arquivos referindo-se apenas ao nome relativo a eles, independente da localização física dos mesmos.

Pelo fato do **DIGDOS** ser um sistema operacional complexo, ele é bastante extenso. Isto significa que se o **DIGDOS** fosse inteiramente de uma só vez carregado na memória, não restaria memória livre para o usuário.

Por isso são usadas áreas de **"overlay"**, que são áreas de memória onde é carregado apenas parte do **DIGDOS** que está sendo usada naquele momento.
Isto significa que o disco do **DIGDOS** sempre deverá estar montado no drive Ø para acessar conforme necessidades.

OBS: O sistema operacional deve sempre estar montado no drive Ø

CONEXÃO DO SISTEMA

Veja o **MANUAL DE OPERAÇÕES** para informações sobre a conexão de interface para disco e dos proprios DISK-DRIVE.

O QUE SE ENCONTRA NESTE MANUAL

1 - COPIAR O DISQUETE ORIGINAL DE SISTEMA

Neste primeiro capítulo o usuário encontra as instruções para efetuar uma ou mais cópias do sistema, assegurando que o disquete original não seja ocasionalmente danificado.

2 - COMANDOS DO DOS

O segundo capítulo trata da Biblioteca dos Comandos do DIGDOS. Estes comandos são acionados quando a mensagem "DIGDOS READY" aparece no vídeo ou diretamente através do BASIC.

A familiarização com os comandos, se fará ao longo da utilização do sistema, de forma que o capítulo 2 será fonte de consulta, devendo o usuário mantê-lo sempre a mão.
Comandos tais como LIB, DIR, FREE, ERROR, SYSTEM, garantirão ao usuário acesso a informações relacionadas aos conteúdos dos disquetes, reservas de memória, e da configuração do sistema.

Outros comandos como CHAIN, DO, AUTO, BREAK, BOOT, CLS, etc, concatenam programas /arquivos, habilitam ou não a tecla BREAK, resetam o sistema ou limpam sua tela de vídeo.

COPY, CREATE, DUMP, LIST, KILL, LOAD, RENAME, etc, possibilitam total manipulação dos arquivos armazenados nos disquetes.

3 - O BASIC DISCO

O BASIC do DIGDOS, implementa o BASIC residente na ROM, adicionando novos comandos que o usuário terá contato nos capítulos 3,4 e 5.

MERGE, SAVE, DEFFN, LOAD, CMD, DEFUSR, FIELD, GET, PUT, INPUT #, PRINT #, etc, dão à linguagem BASIC as funções necessárias ao acionamento de arquivos aleatórios ou seqüênciais, e permitem gravações e leituras nos disquetes montados nos drives.

COPIANDO O DISQUETE DE SISTEMA

Após efetuar as conexões conforme instruções do apêndice A, siga minuciosamente os passos seguintes.

1- Proteja o disquete original com o selo contra gravações. Para isto, existe em uma das bordas do disquete, uma ranhura que envolvida com o adesivo que a acompanha, assegura que ele não sofrerá gravações em eventuais erros do operador.

2- Ligue o drive acionando o interruptor de força no painel traseiro do chassi. Depois, monte o disquete que acompanha o sistema no drive, inserindo-o na ranhura do painel frontal.

3- Ligue o computador.
O sistema armazenado no disquete será automaticamente carregado na memória principal do computador. Surgirá no vídeo, a mensagem "DIGDOS READY", que coloca o sistema a espera dos comandos do operador.

A partir desta mensagem, o computador aguarda os comandos do operador. Prepare um ou mais disquetes em branco para efetuar cópias do disquete original. Não preocupe-se por enquanto com a formatação. O sistema se encarregará de solicitá-la quando necessário.

Observe primeiramente:

- Se o seu sistema for MULTI-DRIVE (ou seja, se você tem mais de um drive), digite:

```
COPY,Ø,1,,SPW=DIGITUS
```

- Se o seu sistema for de apenas 1 DRIVE, digite:

```
COPY,Ø,Ø,,SPW=DIGITUS
```

Atente para as duas vírgulas seguidas. Mais a frente você entenderá o porque desta sintaxe.

Durante a execução da tarefa, o sistema enviará através do vídeo, mensagens, às quais o operador deve responder com precisão. Uma das mensagens será referente a formatação do disquete destino, a qual você deverá responder positivamente. (Digite "Y")

Em caso de falha durante a operação, assegure-se que o disquete de sistema esteja montado no DRIVE Ø e pressione o botão de RESET, reiniciando a operação até que obtenha sucesso.

Repita a operação com novas disquetes, quantas vezes convir-lhe e guarde o disquete original em lugar protegido de calor excessivo, campos magnéticos, poeira e qualquer agente que possa evetualmente danificá-lo.

NOTA -

1 - Evite desligar a força do computador ou do drive com disquetes montados, isto poderá eventualmente danificar os arquivos.

2 - SPW=DIGITUS, refere-se a uma "palavra chave" de acesso ao disquete fonte de sistema. Cópias de disquetes só serão realizadas se a "palavra chave" for especificada. DIGITUS é a chave para o disquete que acompanha a interface. Você verá com mais detalhes este assunto nos comandos COPY (2.9), FORMAT (2.16) e no tópico de ARQUIVOS (2.1.2).

DANDO INÍCIO À UTILIZAÇÃO

Monte no DRIVE Ø uma das cópias e pressione o botão de RESET, à esquerda do computador. Isto faz com que o sistema seja novamente carregado na memória principal. Observe que, para ativar o DIGBASIC II, sem carregar o sistema DIGDOS, basta pressionar o botão de RESET com a tecla BREAK pressionada. (Desta maneira o computador funcionará como se não possuisse interface para DISK DRIVES.)

- Aguarde a mensagem "DIGDOS READY".

O sistema estará a partir de agora aguardando os comandos do operador, ou seja, você!

Digite: LIB <RETURN>
Com este comando o sistema lista no vídeo toda a biblioteca de comandos do DIGDOS que serão vistos à frente no, capítulo 2.

Digite: DIR,Ø <RETURN>
Este comando lista no vídeo o diretório do disquete de sistema montado no DRIVE Ø. Programas como CHAINBLD/BAS, DEBUG/CMD, serão discutidos nos apêndices.

Digite: DIR,Ø,S,A,I,U <RETURN>
O diretório agora listado, é bem mais completo e além de outras coisas fornece os arquivos chamados "invisíveis". São uma série de rotinas utilizadas pelo DOS. Veja o ítem 2.12.

- Atente para o arquivo BASIC/CMD.

Este arquivo carrega na memória principal do computador o DISK BASIC. Chamando este programa do DOS, nós colocaremos o computador ao nível de BASIC.

Digite: BASIC <RETURN> (/CMD pode ser omitido)
Ao surgir na tela do vídeo a mensagem:

DISK BASIC READY

O computador estará ao nível de BASIC permitindo a você acionar os seus programas de BASIC com inúmeras vantagens que serão mostradas nos capítulos 3,4 e 5.

NOTA
Observe que os comandos do DIGDOS têm como delimitadores de campo, vírgulas ou espaços, que não poderão ser omitidos.
Caso sejam omitidos, o sistema operacional enviará uma mensagem de erro voltando em seguida ao "DIGDOS READY".

CAPÍTULO 2

NOTAÇÃO, CONVENÇÕES E INFORMAÇÕES

Neste capítulo serão apresentados os comandos da biblioteca do DIGDOS, usados quando em "DIGDOS READY" ou chamados do BASIC de maneira especial, acionando rotinas do sistema operacional.

Os comandos do DOS são limitados a um total máximo de 80 caracteres, incluindo nestes o "RETURN". O "RETURN" é um finalizador obrigatório, fazendo que as instruções sejam levadas a nível de execução. Em vista disto, a partir de agora não faremos mais referências ao "RETURN" após cada instrução, entendendo que o leitor finalizará por "RETURN", os exemplos que mostraremos neste capítulo.

1— NOTAÇÃO PARA COMANDOS

- < > - Estes dois sinais de "menor que" e "maior que" envolverão os parâmetros que chamamos "opcionais".

Esta notação é apenas didática, não sendo necessário digitar os sinais quando se fizer uso destes parâmetros nos comandos.

EXEMPLO:

<, PROT=xxx > <, ASE=yn > <, ASC=yn >

Podem ser digitados como:

,PROT=LOCK,ASE=Y,ASC=N

- As letras maiúsculas deverão ser digitadas exatamente como estão.

- As letras minúsculas evidenciam os campos dos parâmetros e devem ser substituídos conforme a conveniência e dentro dos valores considerados legais.

- Os comandos do DOS devem ter seus campos separados por delimitadores. Estes delimitadores podem ser "VÍRGULAS" ou "ESPAÇOS". A não inclusão destes delimitadores, tornam os comandos ilegais, de maneira que o sistema enviará mensagens de erro.

- Os valores numéricos serão sempre reconhecidos como valores decimais a menos que sufixados por "H".

2- ARQUIVOS

Os arquivos armazenados nos disquetes serão acessados e manipulados por uma especificação padrão a qual chamaremos de "arquivo".

O "arquivo" inclui:

NOME</ext> <.chave> <:dn>

Sendo:

NOME - Consiste em um nome obrigatório atribuído ao arquivo quando criado. Estes nomes são criados a critério do usuário. Observa-se que estes nomes poderão facilitar a manipulação dos arquivos sendo conveniente, assim, que os seus nomes sejam alusivos ao conteúdo ou função do arquivo.
O nome poderá conter de 1 a 8 caracteres sendo o primeiro de A-Z e os outros de A-Z ou Ø-9.

/ext - Trata-se de uma extensão ao nome do arquivo, sendo um parâmetro opcional, que deve, por conveniência, procurar fazer alusão ao tipo do arquivo.

EXEMPLO:
A extensão "/BAS" é usada quando damos nome a um programa em BASIC, já "/TXT" quando refere-se a um texto, "/DAD" quando o arquivo é de dados, "/CMD" quando o arquivo é um programa em linguagem máquina (Z8Ø) e "/SYS" de um programa do sistema operacional. A extensão poderá ter até 3 caracteres, sendo o primeiro de A-Z, e os outros, de A-Z ou de Ø-9.

chave - É um valor atribuído ao arquivo, usado em checagens feitas pelo sistema, permitindo ou não que o arquivo seja acessado, alterado e demais facilidades, determinadas pelo grau de proteção do arquivo. É um parâmetro opcional e do mesmo formato do "NOME".

:dn - Opcional, trata-se do número do "drive" onde está montado o disquete que possui o arquivo. Se omitido, o sistema procurará, a partir do "drive Ø" seguindo depois para o 1, 2 e 3, o primeiro arquivo com o nome, extensão e chave referida suspendendo a busca assim que encontra o primeiro arquivo que atenda a especificação.

EXEMPLOS:
São especificações válidas de arquivos:
- LOTERIA/BAS.ABC1:1
- CONTRATO/TXT.SCRT
- PGRTESTE:2
- TELEFONE/DAD:Ø
- PROGRAMA/CMD
- CHAINBLD/BAS

NOTA:

É importante que o usuário fixe bem a especificação padrão de "arquivos", criando para si próprio um método claro para especificá-los, tornando a manipulação dos mesmos mais rápida e livre de erros.

COMANDOS

1- APPEND

APPEND,arquivo1,<TO,>arquivo2

Este comando une o "arquivo1", ao final do "arquivo2". O "arquivo1" não terá seu conteúdo alterado, ficando o "arquivo2" com seu conteúdo inicial adicionado do "arquivo1".

EXEMPLO:

APPEND,AAA:1,BBB/DAD:0

O conteúdo de AAA, arquivo contido no disquete montado no drive 1, será adicionado ao final do conteúdo do arquivo BBB/DAD que se encontra no drive 0. Após este comando, o arquivo AAA ficará inalterado e o arquivo BBB/DAD terá o conteúdo original do arquivo BBB/DAD mais o arquivo AAA.

2- ATTRIB

Um arquivo depois de criado, poderá receber diversas atribuições, de acordo com a necessidade do usuário, usando-se o comando ATTRIB. (Ao menos um parâmetro opcional deverá ser incluído.)

ATTRIB,arquivo1<,INV><,VIS><,PROT=xxx><,ACC=chave1> <,UPD=chave2>

Das atribuições:

INV - Atribui ao "arquivo1", invisibilidade. Isto significa que este arquivo não será listado pelo comando DIR. O arquivo será invisível na listagem do diretório, podendo ser listado somente se o comando DIR for usado com a "opção I".

VIS - Exclui a atribuição INV. O "arquivo1" será visível ao comando DIR.

PROT =xxx - Atribui o grau de proteção ao acesso do arquivo. Determinado pelos parâmetros que seguem:

xxx=LOCK - Nível 7 - Nenhum acesso é permitido.

xxx=EXEC - Nível 6 - Acesso apenas para execução.

xxx=READ - Nível 5 - Acesso à execução e à leitura dos conteúdos do arquivo.

xxx=WRITE - Nível 4 - Acesso à execução, leitura e alteração de conteúdos do arquivo.

xxx=RENAME OU NAME - Nível 2 - Acesso à execução, leitura, alteração e troca de nomes do arquivo.

xxx=KILL - Nível 1 - Acesso à execução, leitura, escrita, troca, alteração e eliminação do arquivo.

xxx=FULL - Nível Ø - Acesso total. Qualquer operação é permitida.

ACC=chave1 - Trata-se da chave de acesso. Se omitida ela será inalterada, se omitida apenas a chave, <,ACC=>, a chave será retirada. Esta chave é usada nas checagens do comando COPY.

UPD=chave2 - É a chave para atualização do arquivo.

EXEMPLOS:

- ATTRIB,NOTAS/TXT.ZE,INV,PROT=LOCK,ACC=CARLO,UPD=SECRETO

Atribui a um arquivo de especificação NOTAS/TXT.ZE, uma nova chave de acesso, CARLO, uma chave de atualização, SECRETO. o nível de proteção 7, e invisibilidade no diretório.

Caso ao arquivo NOTAS/TXT.ZE estivesse atribuída anteriormente uma chave de atualização qualquer, esta deveria ser especificada. Para uma nova atualização do arquivo do exemplo, veja o seguinte:

- ATTRIB,NOTAS/TXT.CARLO.SECRETO,VIS,PROT=FULL,ACC=,UPD=

Eliminou as chaves de acesso e atualização do arquivo, atribuiu um nível de proteção Ø, e visibilidade no diretório.

3- AUTO

AUTO<,doscmd>

Permite gravar no disquete de sistema, uma série de comandos do DIGDOS, para que sejam executados automaticamente, assim que o sistema seja ativado. O comando AUTO é limitado em até 32 caracteres inclusive <RETURN>.
O campo <,doscmd> é usado para especificar o comando.

EXEMPLOS:

- AUTO,BASIC,RUN"PROGRAMA/BAS:2"

Quando ativado, o sistema automaticamente carrega o utilitário BASIC/CMD na memória principal, e executa em seguida o PROGRAMA/BAS, do disquete montado no drive 2.

- AUTO,BASIC,5V,57000,LOAD"PROG2/BAS:1"

Quando ativado, o sistema carrega o BASIC/CMD, reserva 5 buffers de campos variáveis, reserva a memória acima de 56999 para rotinas "USR", carrega do drive 1 o programa PROG2, colocando-o na área de texto.

- AUTO,DIR

Ativando o sistema, o diretório do drive Ø será listado automaticamente na tela.

4- BASIC2

Ativa o DIGBASIC II residente e retira o DIGDOS da memória principal. A mensagem de "Proteger?" aparecerá no vídeo, como inicialização do DIGBASIC II. O sistema só será carregado novamente na memória principal se o botão Reset for pressionado. Qualquer comando do DISKBASIC ou do DIGDOS, acionado do DIGBASIC II, não será reconhecido e o computador enviará uma mensagem de erro.

5- BOOT

Resseta o computador e aciona o programa monitor, i.e. o DIGBUG. É usado após alterações no sistema, ou após o comando AUTO.
Para retornar ao DIGDOS, digite:

G Ø<espaço>

6- BREAK

BREAK<,yn>

Ativa ou desativa a tecla "BREAK".

EXEMPLO:

- BREAK ou BREAK,Y

Ativa a tecla "BREAK", até a próxima mensagem DIGDOS READY, quando será selecionado conforme o parâmetro AG da "opção de sistema".

- BREAK,N

Desativa a tecla "BREAK", até a próxima mensagem DIGDOS READY, quando será selecionado conforme o parâmetro AG da opção de sistema.

7- CHAIN

CHAIN,arquivo1

O comando CHAIN e DO, permitem que um conjunto de comandos do DIGDOS que estejam agrupados num arquivo, sejam ligados em "elo" com o sistema, e executados como se fossem comandos digitados a partir do teclado, automaticamente.
Portanto, o "elo" será executado automaticamente até que um "fim de arquivo" seja encontrado.
O arquivo "arquivo1", fonte do "elo" de comandos, deverá ser criado através de um programa, o CHAINBLD/BAS.
Veja instruções detalhadas do CHAINBLD/BAS, no apêndice 1.

EXEMPLO:

- CHAIN,CHAINTST/JCL

O sistema será ligado em elo aos comandos do arquivo CHAINTST/JCL, até que encontre um "fim de arquivo". O arquivo referido no exemplo, é um utilitário que acompanha o sistema, afim de exemplificar o funcionamento dos comandos CHAIN e DO.

8- CLS

Limpa o vídeo e seleciona o formato de 64 caracteres por linha.

9- COPY

1 - COPY,arquivo1<,TO>,arquivo2

2 - COPY,d1,arquivo1<,TO>,arquivo2

3 - COPY,$arquivo1<,TO>,arquivo2

4 - COPY,d1,$arquivo1<,TO>,arquivo2

5 - COPY,d1<,TO>,d2,mm/dd/aa,SPW=chave1<,NDPW=chave2>
<,ODPW=chave3>

O comando COPY, permite copiar arquivos de disquetes chamados FONTE, em disquetes chamados DESTINO. Para tal, o sistema cria no disquete destino, o novo arquivo tal como especificado pelo comando. Caso a especificação seja a mesma de um arquivo existente, este será destruído.

O comando segue, conforme a configuração de seu sistema, as alternativas abaixo:

SISTEMAS MULTI-DRIVE

Formato 1 - Copia o arquivo especificado por "arquivo1" em um outro especificado por "arquivo2".

Formato 3 - O campo $, indica que,o disquete que contém o sistema operacional não estará montado no drive 0 durante toda operação. Desta forma o operador deverá atender as mensagens enviadas pelo sistema durante a operação.

SISTEMAS DE UM DRIVE

Formatos 2 e 4 - O sistema enviará durante toda a operação mensagens para a montagem dos disquetes DESTINO e FONTE, no drive Ø e quando os discos envolvidos não possuírem o sistema, usa-se o campo $.

FORMATO 5 - Faz cópias da íntegra de disquetes FONTE para disquetes DESTINO. Todos os arquivos serão copiados com as mesmas atribuições.

O campo SPW, especifica a chave de acesso do disquete FONTE, que fora especificada quando da formatação do mesmo.

O campo opcional <,NDPW=chave2> é utilizado quando o disquete DESTINO deverá ser formatado e a chave de acesso será a especificada por chave2, caso omitido a chave de acesso será a mesma do disquete FONTE, isto é, a chave especificada por SPW.

Se o disquete DESTINO já estiver formatado, deverá ser especificado o campo <,ODPW=chave3>, onde chave3 é a chave de acesso, conhecida, do disquete DESTINO. A não especificação de "chave3", quando o disquete DESTINO já estiver formatado, levará ao término da operação de cópia.

O campo mm/dd/aa corresponde à data e é obrigatório. Portanto se não utilizado deverá constar entre duas vírgulas seguidas. Ver exemplos, (note o formato mês/dia/ano).

EXEMPLO:

- COPY,Ø,Ø,,SPW=DIGITUS

Copia a íntegra do disquete FONTE no DESTINO. Somente o drive Ø será utilizado no processo. O campo SPW=DIGITUS, refere-se a chave de acesso do disquete FONTE, sendo utilizada para o DESTINO, pois NDPW não foi especificado. O disquete DESTINO deverá ser formatado. O campo mm/dd/aa está vazio e separado por duas vírgulas.

- COPY,1,2,Ø1/Ø4/83,SPW=DIGITUS,NDPW=DGT-1ØØØ

Copia a íntegra do FONTE, montado no drive 1,no DESTINO, montado no drive 2, atribuindo ao disquete DESTINO a chave de acesso DGT-1ØØØ e a data de 4 de janeiro de 1983. O disquete DESTINO deverá ser formatado.

- COPY,ARQUIVO1:Ø,TO,ARQUIVO1:1 ou COPY,ARQUIVO1:Ø,:1

Cria no DESTINO, montado no drive 1, a imagem do ARQUIVO1, FONTE, montado no drive Ø, com o mesmo conteúdo e nome.

- COPY,Ø,1,,SPW=SECRETO,ODPW=SENHA

Copia a íntegra do disquete montado no drive Ø no disquete do drive 1, sendo a chave de acesso do disquete FONTE, igual a SECRETO. O disquete DESTINO já se encontra formatado e sua chave de acesso é SENHA.

- COPY,Ø,ARQUIVO1,TO,ARQUIVO1

Copia ARQUIVO1 de um disquete FONTE, diferente do disquete DESTINO, utilizando na operação apenas o drive Ø.

- COPY,Ø,$ARQUIVO1,OUTROARQ

Copia um ARQUIVO1 de FONTE diferente de DESTINO, em um arquivo de nome OUTROARQ sendo que o disquete de sistema não estará envolvido na operação.

- COPY,CONTAS/DAD.D100:1,TO,CTSPAGAS/DAD:2

Copia o conteúdo do arquivo CONTAS/DAD.D100 montado no drive 1, em um arquivo CTSPAGAS/DAD que será criado com este nome no disquete montado no drive 2.

NOTAS:

1 - O sistema questiona no processo de cópia, se há ou não necessidade de formatação do disquete DESTINO. O operador deve estar seguro, caso não seja necessário formatá-lo, de que o DESTINO esteja com formatação adequada exigida pelo DIGDOS.

2 - Observe que os arquivos possuem chaves de acesso, e que são checadas quando estes arquivos forem FONTE para alguma cópia. No processo de cópia de todos os arquivos de um disquete, a chave torna-se a chave do próprio disquete, e todos os arquivos serão copiados checando apenas a chave do disquete FONTE especificada por SPW.

3 - Acompanha o seu DIGDOS, um utilitário, CONVERTE, que lhe permitirá compatibilizar o DIGDOS com outros sistemas. Veja explicações detalhadas no apêndice 2.

10- CREATE

Usado para pré-alocar espaços de um arquivo, substituindo o processo automático realizado pelo sistema.

CREATE,arquivo<,LRL=cmp><,REC=qtd>

LRL=cmp - Especifica em bytes, o campo de cada gravação. O valor de "cmp" pode variar de 1 a 256. Se omitido, o valor será de 256 bytes por gravação.

REC=qtd - Especifica o número de gravações que deverão ser alocadas.

EXEMPLO:

- CREATE,ENDERECO/DAD.PAULO:1,LRL=120,REC=100

Este exemplo aloca no disquete montado no drive 1, um espaço de 100 gravações de 120 bytes cada, portanto reserva 12000 bytes ao todo, para um arquivo que será acessado pela especificação ENDEREÇO/DAD.PAULO.

- CREATE,TELEFONE/PAR,LRL=7,REC=1000

Cria um arquivo de especificação TELEFONE/PAR, de 7 bytes por gravação, com um no. de 1000 gravações, no primeiro drive que possuir um disquete com espaço de 7000 bytes disponíveis.

- CREATE,ARQUIVO,REC=50

Omitindo a especificação de campo, o sistema alocará 50 gravações com o espaço de 256 bytes por gravação, sendo neste exemplo reservados 12800 bytes para o ARQUIVO.

11- DEBUG

Ao pressionar as teclas de 123 simultaneamente, será acionado um utilitário chamado de DEBUG. Veja explicações detalhadas no apêndice 3.

12- DIR

DIR<:><,dn><,A><,S><,U><,/ext><,P> < I >

Lista no vídeo o diretório do disquete montado no drive "dn". Se omitido o valor de "dn", a listagem será referente ao drive Ø.

Com referência à listagem, a primeira linha tráz o nome e data do disquete, (dados inseridos quando da formatação) e, também, o número de FDEs e GRÃOS disponíveis para gravações.
A listagem padrão trará 4 colunas com os nomes e extensões (se estas existirem) dos arquivos armazenados no disquete.

OPÇÕES

OPÇÃO "A" - O comando DIR com opção "A", listará arquivo por arquivo, trazendo em cada linha as seguintes especificações:

1 - Nome do arquivo.
2 - Extensão do arquivo.
3 - O EOF (fim de arquivo), no formato xxx/yyy onde xxx é o número do setor relativo dentro dc arquivo e yyy, é o número do byte relativo dentro do setor.
4 - O tamanho de cada gravação (LRL) em bytes.
5 - O número de gravações (REC) já feitas.
6 - O número de GRÃOS (GRANS) alocados ao arquivo.
7 - As extensões alocadas aos arquivos.
8 - 6 padrões referentes ao arquivo que seguem:

Padrão:
- S - Arquivo do sistema.
- I - Arquivo invisível.
- E - Não permite alocar mais espaço.
- U - Existência de "chave" de atualização.
- A - Existência de "chave" de acesso.
- L - Nível de proteção (ver ATTRIB).

OPÇÃO "S" - Listar inclusive arquivos do sistema.

OPÇÃO "I" - Listar inclusive arquivos invisíveis.

OPÇÃO "U" - Somente os arquivos a nível de "atualizáveis" serão listados.

OPÇÃO "/ext" - Somente os arquivos com a extensão indicada.

OPÇÃO "P" - Envia a listagem para a impressora.

NOTA
"DIR,$Ø" permite que seja listado o diretório de um disquete que não possui o DIGDOS, desde que ele tenha sido formatado de acordo com o formato DIGDOS.

13- DO

DO,arquivo
Veja o comando CHAIN.

14- DUMP

DUMP,arquivo,end-início,end-fim<,end-entrada<,end-relocado>>

Transfere um bloco dos dados atuais da memória principal do computador, a um arquivo de nome especificado pelo campo "arquivo", iniciando a transferência no byte endereçado no campo "end-início", e finalizando no byte endereçado no campo "end-fim".

Os campos "end-início", "end-fim", "end-entrada" e "end-relocado", são valores numéricos decimais ou hexadecimais (os valores em HEXA devem ser sufixados por H), podem assumir valores entre 0-65535 ou 0000H-0FFFFH.

Dependendo do valor do campo "end-entrada", este comando, assume duas maneiras de transferência.

No caso de "end-entrada"=65535 ou 0FFFFH, faz-se a transferência de todo o bloco endereçado, sem que o arquivo receba qualquer byte de controle. Apenas os 2 primeiros bytes do arquivo terão o "end-início" gravado. Este bloco de memória poderá posteriormente ser recuperado e desassemblado.

Caso o "end-entrada" seja diferente que 65535 ou o mesmo não seja especificado, o sistema assume que o bloco a ser transferido, trata-se de um programa em código de máquina e a transferência será feita de maneira que este possa ser executado posteriormente através do comando LOAD. O "end-relocado" permite que o seu bloco seja carregado novamente na memória principal relocando todo o bloco da posição que ele fora inicialmente transferido.

Observe que o "end-início" deve ser igual ou maior que 5200H, para que o DIGDOS não seja afetado.

EXEMPLO:

- DUMP,MÁQUINA/CMD:1,5200H,6200H,60A0H

Transfere o bloco da memória principal de 2.5 Kbyte, do endereço 5200H inclusive, ao endereço 6200H inclusive, considerando que o objeto da memória seja um programa em código de máquina que inicia sua execução no byte de endereço 60A0H.Este programa pode ser carregado na memória através do comando LOAD,MÁQUINA/CMD ou ser executado através do comando MÁQUINA/CMD, que carrega e inicializa o programa no byte 60A0H.

- DUMP,PROBLEMA/MEM:2,0,65535,65535

Supondo que durante uma tarefa o programa apresentou uma falha qualquer e os estudos para resolvê-la não podem ser feitos no momento, uma cópia exata da memória com 64 Kbytes é transferida para o disquete e, posteriormente através de técnicas, desassemblada e verificada, no sentido de sanar a falha.

NOTA
Um valor HEXA que se inicie por uma letra, deve ser prefixado por 0. Portanto valores como A34BH devem ser escritos 0A34BH.

15- ERROR

ERROR,xx

Fornece a mensagem de erro do DIGDOS associada aos números de 0 a 63.

EXEMPLO:

- ERROR,24

Envia a mensagem de erro 24 ao vídeo, "FILE NOT IN DIRECTORY".

16- FORMAT - FORMATA UM DISQUETE

Formatar um disquete é dividí-lo magneticamente em trilhas e setores, de forma a permitir o armazenamento de dados em sua superfície magnética.

Entre 15 e 3Ø% da capacidade de armazenamento do disco será usada em informações de controle, não podendo conter dados do usuário.

O DIGDOS formata o disquete selecionando trilhas e setores e armazenando os arquivos DIR/SYS e BOOT/SYS, obrigatórios em qualquer disquete. Só depois da formatação, os disquetes poderão ser usados para armazenamento de dados.

FORMAT,dn,nome,mm/dd/aa<,chave>

dn - É o número do drive onde está montado o disquete a ser formatado.

nome - Nome atribuído ao disquete. Se omitida o sistema atribuirá ao disquete o nome SEMNOME.

mm/dd/aa - Data da formatação. Observe o formato mês/dia/ano.

chave - Chave de acesso ao disquete. Esta chave será checada nos casos de cópia da íntegra do disquete, como foi visto no capítulo 1 e no comando COPY. Se omitida, o sistema atribuirá ao disquete a chave DIGITUS, para garantir o acesso do usuário ao disquete.

NOTA

Parâmetros nulos podem ser usados, mas devem ter seus campos separados por vírgula. Veja exemplos.

EXEMPLO:

1 - FORMAT,Ø,OBRAS,11/28/82
2 - FORMAT,1,,,CHAVE
3 - FORMAT,2,DISCO1
4 - FORMAT,1,,Ø7/Ø2/82

17- FREE

FREE<,P>

O comando FREE informa para todo o sistema, ou seja para todos os disquetes montados em todos os drives, o nome do disquete, a data de formatação, o número de trilhas formatadas, o número de FDEs e GRÃOS disponíveis em cada um deles. O parâmetro opcional "P", envia a listagem para a impressora, ao invés do vídeo.

18- HIMEM

HIMEM<,endereço>

Seleciona a reserva de memória que ficará disponível, sem acesso permitido para o BASIC. Sem o parâmetro "endereço", o comando simplesmente informa o endereço atual da reserva de memória. Ao ativar o computador, este endereço será 65535 ou ØFFFFH. Este comando é equivalente ao "Proteger ?" do DIGBASIC II.

19- JKL

JKL

Não possui parâmetros. A função do comando é enviar para a impressora, uma cópia do vídeo. A tecla BREAK interrompe a sua execução. Pode-se também acionar o comando, pressionando as teclas "JKL" simultaneamente.

2Ø- KILL

KILL,arquivo1

Deleta um arquivo contido em um disquete. Se o número do drive não for especificado, o sistema procura o arquivo, iniciando a busca pelo drive Ø e, deletando o primeiro arquivo que seja encontrado com a mesma especificação.

EXEMPLO:

- KILL,MMM/BAS:1
- KILL,PROG/BAS

O sistema deleta o primeiro arquivo com o nome PROG/BAS que encontrar, começando a busca pelo drive Ø.

21- LIB

LIB

Lista a biblioteca dos comandos do DIGDOS. Não requer parâmetros.

22- LIST

LIST,arquivo1<,linha inicial<,no. de linhas>>

Lista no vídeo o conteúdo de um arquivo. Observa-se que o conteúdo do arquivo pode estar armazenado no formato comprimido, de forma que a listagem será um pouco sem sentido. Os arquivos chamados TEXTOS terão suas listagens em caracteres ASCII, portanto, de compreensão imediata.

NOTA:
São considerados TEXTOS, programas BASIC armazenados com a opção "A" e demais arquivos que não estejam armazenados no formato comprimido.

23- LOAD

LOAD,arquivo1

Carrega um programa em código de máquina dentro da memória principal do computador. É usado para carregar programas que serão utilizados futuramente em outros programas, por exemplo, rotinas USR em programas BASIC.

24- PRINT

PRINT,arquivo1<,linha inicial<,no. de linhas>>

Executa a mesma função do LIST, exceto que a listagem é enviada para impressora.

25- R

R

Executa novamente o último comando que esteja no BUFFER. Se a opção de sistema "BE=N", o comando R não executará sua função retornando simplesmente o sistema ao DIGDOS READY.

26- RENAME

RENAME,arquivo1<,TO>,arquivo2

Atribui a um arquivo1 existente uma nova especificação, arquivo2, que consiste de um nome e opcionalmente uma extensão.

RENAME altera apenas o nome e extensão do arquivo1. Nada mais é alterado.

CAPÍTULO 3

DISK BASIC

2, INTRODUÇÃO

O DISK BASIC é um complemento do DIGBASIC II, que facilitará criar é arquivos de dados e programas em disco, agilizando assim sua programação.

Quando o DISK BASIC é carregado na memória do computador, este ocupa um espaço de aproximadamente 5.5 Kbytes, utilizando rotinas do DIGBASIC II residente na ROM de seu DGT, e outras adicionais particulares ao DISK BASIC. É bom lembrar, que este espaço ocupado pelo DISK BASIC, é largamente compensado pela versatilidade oferecida ao BASIC.

Neste capítulo, serão tratadas as novas funções que você poderá utilizar em programas editados em BASIC, e no capítulo 4 as funções de leitura, gravação e manipulação de arquivos.

1 - ATIVANDO O DISK BASIC

O DISK BASIC poderá ser ativado utilizando os seguintes formatos, através do DIGDOS:

```
1- BASIC
2- BASIC *
3- BASIC n
4- BASIC m
5- BASIC cmd
6- BASIC n,m,cmd
7- BASIC m,n,cmd
8- BASIC n,m
9- BASIC m,n
10-BASIC m,cmd
```

Onde:

t - Retorna ao BASIC não perdendo o programa existente antes de ir ao DOS, sem alterar os valores de m e n existentes. Existindo algum programa, este será forçosamente listado.
BASIC * não terá efeito se n for menor que 2, ou se o programa tiver menos que 3 linhas.

- Número de BUFFERS que deverão ser alocados para arquivos. Se omitido, 3 BUFFERS serão alocados e o máximo é de 15. Estes BUFFERS serão para E/S com o comprimento lógico do registro padrão de 256 bytes. Se o comprimento lógico do registro for menor que 256, deverá ser colocado o sufixo V (variável) ao número de BUFFERS especificados.

m - Reserva uma área de memória até o endereço m-1 para que o BASIC não se utilize dela. Acima do endereço m, inclusive, poderá ser utilizada pelas rotinas USR, etc.

cmd - Um ou mais comandos do BASIC, que serão considerados como uma entrada direta do teclado e serão executados após a ativação. Estes comandos estão limitados pelo DOS com o máximo de 80 caracteres incluindo RETURN, e 32 caracteres quando por via AUTO.

EXEMPLOS:

- BASIC

Carrega o BASIC, reserva 3 BUFFERS e aguarda comandos do operador.

- BASIC,RUN"MENU/BAS"

Carrega o BASIC, reserva 3 BUFFERS, carrega o programa com o nome de "MENU/BAS" e inicia sua execução.

- BASIC,5V,40100,LOAD"MENU/BAS"

Carrega o BASIC, reserva 5 BUFFERS com comprimento lógico variável, reserva a memória iniciando de 40100, para não ser usada pelo BASIC, e carrega o programa com o nome de "MENU/BAS" e aguarda o comando do operador.

- BASIC,CLEAR2000:A$="xxx":RUN"MENU/BAS",V

Carrega o BASIC, reserva 3 BUFFERS, reserva 2000 bytes para strings, permanece com o valor atual do HIMEM, carrega o programa "MENU/BAS", inicia sua execução sem apagar as variáveis, i.e., o valor de A$ permanece "xxx".

NOTA:

O "V" que aparece no final do comando do exemplo acima faz com que as variáveis, (numéricas ou strings) não sejam "zeradas" quando o programa é rodado. Observe que o programa inicia sua execução automaticamente.

2-CONSTANTES OCTAIS E HEXADECIMAIS

Em alguns casos pode ser mais interessante utilizar constantes octais (base 8) ou hexadecimais (base 16) em lugar de decimais, como por exemplo em endereços de memória.

&Hxxxx - Constantes hexadecimais, onde xxxx deverá ser composto de dígitos hexadecimais (0,1,2,...9,A,B,C,D,E,F), não ultrapassando então a FFFF.

&Oxxxxx - Constantes octais, onde xxxxx deverá ser composto de dígitos octais (0,1,2,3.. ..,6,7), não ultrapassando a 177777 octal. O prefixo "O" poderá ser omitido, assim &Oxxxxx=&xxxxx.

As constantes representam números inteiros, positivos ou negativos. Números hexadecimais maiores que &H7FFF e octais maiores que &O77777 são interpretados como valores negativos.

EXEMPLOS:

Decimal	Hexadecimal	Octal
1	&H1	&O1
10	&HA	&O12
32767	&H7FFF	&O77777
-32768	&H8000	&O100000
-32767	&H8001	&O100001
-10	&HFFF6	&O177766
-1	&HFFFF	&O177777

NOTA:
Constantes hexadecimais e octais não poderão ser utilizadas em resposta a um INPUT, nem listados em DATA. Na maioria das vezes estas constantes deverão estar entre parênteses para se evitar erros de sintaxe.

EXEMPLOS:

- PRINT &H7FFF,&O77777

Imprime os decimais correspondentes às constantes referidas (ambas iguais a 32767).

- POKE &H3C00,&H40

Coloca o hexadecimal 40 (código ASCII para @) no endereço &H3C00 da memória, que corresponde ao primeiro byte da memória de vídeo.

3- COMANDOS DE EDIÇÃO DO DISK BASIC

O DISK-BASIC oferece uma série de comandos que facilitam bastante a programação e edição de linhas.

COMANDOS DIRETOS

. (ponto) - Lista a linha atual.

↓ - Lista a próxima linha do texto.Caso não exista, a atual será novamente listada.

↑ - Lista a linha anterior. Caso não exista, a atual será novamente listada.

: - Lista a primeira linha do programa. Pode-se usar também SHIFT ↑ .

/ - Lista a última linha do programa. Pode-se usar também SHIFT ↓ .

: - Lista a página anterior, sendo a linha atual a última mostrada na tela.

@ - Lista uma página na tela a começar pela linha atual. Quando feito a linha atual passa a ser a última da tela.

, - (vírgula)- Edita a linha atual.

COMANDOS ABREVIADOS

Com o DIGDOS DISK BASIC, os comandos AUTO, LIST, DELETE e EDIT poderão ser abreviados utilizando somente a primeira letra.

EXEMPLOS:

1 - AUTO 10,5 = A10,5
2 - DELETE 200-400 = D200-400
3 - EDIT 100 = E100

4- COMANDOS DE EDIÇÃO ADICIONAIS

Com o DISK BASIC, foram implementados mais dois comandos de edição, muito úteis, ao DIGBASIC II. São eles :

1 - DI xxxxx,yyyyy 2 - DU xxxxx,yyyyy

Onde:

xxxxx - número da linha a ser movida ou copiada;
yyyyy - número da linha duplicada ou movida;
DI xxxxx,yyyyy - Deleta a linha xxxxx colocando-a em yyyyyy;
DU xxxxx,yyyyy - Duplica a linha xxxxx em yyyyy não deletando xxxxx.

5- RENUM

Renumera as linhas do programa na memória e permite também mover um grupo de linhas, renumerando todos os endereçamentos referidos às linhas. Possui os seguintes formatos:

1- RENUM sssss,iiiii,ppppp,qqqqq <,U> 3- RENUM,
2- RENUM sssss,iiiii 4- RENUM U

Onde:

sssss - número da primeira linha renumerada;
iiiii - incremento;
ppppp,qqqqq - números da primeira e última linhas a serem renumeradas, se omitidos será renumerado todo o programa.

EXEMPLOS:

- RENUM U - O texto é checado e se houver algum erro de endereçamento estes serão indicados;

- RENUM , - O texto é renumerado iniciando em 10 com o incremento de 10;

- RENUM 200,100 - O texto é renumerado iniciando em 200, com o incremento de 100;

- RENUM 2000,,5500,6170 - Todas as linhas de 5500 a 6170 inclusive são renumeradas iniciando por 2000 com incremento de 10;

- RENUM 10000,5,15374,17140 - todas as linhas de 15374 a 17140 são renumeradas iniciando por 10000 com incremento de 5.

NOTA:
O maior número de linha é 65529.

6- REF

O comando REF fornece uma listagem referente às linhas onde se encontram as variáveis, strings, funções do BASIC ou seqüências de caracteres desejados.

O comando REF tem os seguintes formatos:

1 - REF* - Lista de todos os números inteiros e variáveis com as respectivas linhas onde se encontram.

2 - REF$ - O mesmo que REF* listado na impressora.

3 - REFnn - Lista de todas as variáveis iniciadas por nn com as respectivas linhas onde se encontram.

4 - REFsssss- Lista todas as linhas que contêm o número inteiro sssss(entre 0 e 99999). Constantes octais e hexadecimais não poderão ser referidas neste formato.

5 - REF - Lista a próxima linha que contém a variável ou o número especificado pelo último REFnn ou REFsssss executado. Digitado novamente, listará a próxima linha e, caso não haja próxima linha, a mensagem "TEXT END" aparecerá no vídeo.

6 - REF@sssss - Similar ao formato 5, exceto que iniciará pela primeira linha igual ou superior a sssss.

7 - REF=xxx - Lista as linhas que contenham a palavra xxx. Este comando localiza funções do BASIC dentro do texto,tais como PRINT,GOTO, LINEINPUT, etc.

8 - REF"xxx" - Similar ao formato 7 mas xxx é considerado como uma string.

NOTA:

A tecla BREAK, paralisa a execução, RETURN prossegue a execução e SHIFT↑ termina a função.

7- COMANDOS UTILITÁRIOS

CMD"C"

Este comando faz com que todos os espaços do programa sejam elimi - nados, assim como comentários.

CMD"C",S permite, que somente os espaços sejam eliminados, permanecendo os comentários. Deverá ser tomado muito cuidado quando existirem linhas, que sejam de comentários, e que estejam mencionadas em GOSUB e GOTO, pois estas linhas serão eliminadas, causando erro de "linha indefinida" na execução do programa. Deve-se então, após a execução de CMD"C", executar um RENUM U, para verificar se existem linhas indefinidas.

Outro cuidado, que se deve tomar, é no caso de strings, que deverão estar todas entre " (aspas), pois se houver qualquer espaço na string, será deletado. Por exemplo:

```
10 REM EXEMPLO 1
20 DIM A$(2)
30 FOR I% = 0 TO 2
40 READ A$(I%): PRINT A$(I%)
50 NEXT I%
60 DATA JOSE MARIA,JOAO DA SILVA,CARLOS MANUEL
70 END
```

Executando o comando CMD"C":

```
20 DIMA$(2)
30 FORI%=0TO2
40 READA$(I%):PRINTA$(I%)
50 NEXTI%
60 DATAJOSEMARIA,JOAODASILVA,CARLOSMANUEL
70 END
```

Note que na linha 60, todos os espaços desapareceram, para evitar isto, a linha deveria estar da seguinte forma:

```
60 DATA "JOSE MARIA","JOAO DA SILVA","CARLOS MANUEL"
```

CMD"E"

Retorna com a última mensagem de erro.

CMD"doscmd"

doscmd = Expressão string associada a um comando do DIGDOS.
Após executado o comando, volta para Ready (BASIC).

EXEMPLOS:

- CMD"DIR 1 A"
Lista o diretório do drive 1.

- CMD"COPY XXX:0,YYY:1"
Cópia de arquivos.

- CMD"COPY,0,1,12/15/82,DGT"
Copia o disquete do drive 0 para 1 com o nome de DGT e a data de 12/15/82 (mm/dd/aa).

CMD"J",data_1,data_2

Converte a expressão data1 para o formato data2.

Formatos: f1 = mm/dd/aa
f2 = -aa/ddd

Se data1 estiver no formato f1, então data2 terá o formato ddd, e se data1 estiver no formato f2, então data2 terá o formato mm/dd/aa.

mm - Dois dígitos que indicam do mês 01 a 12.
dd - Dois dígitos que indicam o dia do mês entre 01 a 31.
ddd- Três dígitos que indicam o número de dias do ano de 001 a 366.
aa - Dois dígitos que indicam o número do ano de 00 a 99, i.e., 1900 a 1999.

EXEMPLOS:

- CMD "J","12/07/59",A$

Calcula o dia do ano e armazena-o em A$.

- CMD "J","-82/340",B$

Calcula a data correspondente em mm/dd/aa armazenando-a em B$ (data do 340º. dia de 1982).

CMD"O"

Coloca em ordem alfabética (ou numérica) um ou mais vetores strings (ou numéricos).

```
CMD "O",n,dst1$ <,dst2$,...>
```

n - número de ítens a serem ordenados.
d - sinal que indicará se o vetor deverá ser ordenado em ordem crescente ou decrescente.
stm$ - elemento inicial, a ser ordenado do vetor.

EXEMPLOS:

O exemplo abaixo ordena 10 nomes a sua escolha.

```
5 REM ORDENAR O VETOR A$
10 CLEAR 500
20 DIM A$(9)
30 FOR I%=1 TO 10
40 PRINT I%" ";:LINE INPUT A$(I%-1)
50 NEXT I%
60 CMD"O",10,A$(0)
70 CLS
80 FOR I%=1 TO 10
90 PRINT I%" "A$(I%-1)
100 NEXT I%
110 END
```

Agora troque a linha 60 por:

```
60 CMD"O",10,-A$(0)
```

NOTA:

Uma das grandes vantagens deste comando é poder ordenar o vetor levando em consideração somente alguns caracteres mencionados. Desta forma deverão ser indicados quais são estes caracteres.

EXEMPLO:

```
- CMD"O",100,-B$(1)(5,10)
```

Os 100 primeiros elementos do vetor B$ serão ordenados em ordem decrescente levando em consideração somente os caracteres das posições de 5 a 10.

CMD"S"

Retorno ao DIGBASIC. Poderá ser usado também o formato:

CMD "S=doscmd" - Faz com que logo após executado o comando do DOS o controle passe para o DIGDOS READY.

EXEMPLOS:

- CMD"S=DIR 0" - Saída do BASIC, será dado o diretório do drive 0 e o controle passará ao DIGDOS.

- CMD"S=BASIC,5V" - Saída do BASIC, carrega-o novamente na memória, reserva 5 buffers de campos variáveis e o controle passará ao BASIC.

8- FUNÇÕES ADICIONAIS DO DISK BASIC

INSTR

INSTR (<n,> st1$,st2$)

Esta funçao faz com que o resultado seja um valor, que indique a posição da primeira ocorrência da string st2$ em st1$, começando a procura pelo caracter da posição n. Caso n seja omitido, a busca iniciará pelo primeiro caracter da string st1$. Caso st1$ não contenha st2$, retornará o valor 0 (zero).

EXEMPLOS:

- A$="RUA PARAIBA,2261478"
PRINT INSTR(A$,",") fornecerá a posição da vírgula no string A$.

- O programa a seguir separa as variáveis, delimitadas por um espaço.

```
10 REM EXEMPLO INSTR
20 CLEAR 50:CLS:DEFINT I,J
30 DIM A$(2)
40 B$="A2.44 B3.14 C2.71 "
50 I=2
60 FOR J=0 TO 2
70 A(J)=VAL(MID$(B$,I,INSTR(I,B$,CHR$(32))-I))
80 I=INSTR(I,B$,CHR$(32))+2
90 NEXT J
100 PRINT "A= "A(0)
110 PRINT "B= "A(1)
120 PRINT "C= "A(2)
130 END
```

MID$=

```
MID$ (var$,n1<,n2>)= string$
```

var$ - Nome da variável string a ser alterada.
n1 - Posição inicial que deverá ser alterada.
n2 - Número de caracteres a serem alterados, se n2 for omitido, este assumirá o menor dos valores LEN(string$) ou LEN(var$)-n1+1.

Este comando serve para alterar parte de uma string, sendo que o tamanho da string a ser alterada var$ não é modificado. Note, que este comando MID$ está à esquerda do sinal de igual, e serve somente para alterar uma string antiga, não confundido com MID$(string,c,n), que lhe fornece uma substring de comprimento n e começando na posição c da string.

EXEMPLO:

```
B$="ABCD1234"
```

```
1 - MID$(B$,1)="****"     fornecerá B$="****1234"
2 - MID$(B$,1,2)=""       fornecerá B$="****1234"
3 - MID$(B$,5)="567890"   fornecerá B$="****5678"
4 - MID$(B$,4,2)="##"     fornecerá B$="*****678"
```

LINEINPUT

```
LINE INPUT <"mensagem";> st$
```

Este comando faz com que o computador paralise a execução do programa, e espere que uma linha seja introduzida pelo teclado terminando com RETURN, que será atribuída à variável st$ especificada.
Este comando é similar ao INPUT, com as seguintes diferenças:

- Não aparecerá no vídeo o ponto de interrogação;
- Vírgulas e aspas poderão ser aceitos como parte da variável string;
- Cada instrução LINEINPUT poderá determinar somente uma variável;
- Espaços em branco à esquerda não são ignorados, fazem parte da variável string.

EXEMPLOS:

```
- LINEINPUT B$
- LINE INPUT "NOME,TELEFONE ?";N$
```

A mensagem será impressa no vídeo e a vírgula não será delimitador do string, fazendo parte dele.

DEF_FN

```
DEF FN nome ( arg1 <,arg2,...,argn> ) = expressão
```

Onde:

nome : nome da função. É uma variável e indicará o tipo da variável de retorno.
arg : variáveis usadas na definição da função.
expressão : expressão, que geralmente envolve as variáveis definidas pela função.

Este comando serve para você criar sua própria função, que quando chamada será automaticamente executada, devolvendo como resultado um valor, do tipo determinado pelo nome da função.

EXEMPLOS:

- `DEF FN SQ#(X#) = SQR(1/X#)`

Função SQ# de um argumento X#, que retornará um valor em precisão dupla. Neste caso a função SQ# será a raiz quadrada do inverso do argumento.

- `DEF FN C(A,B) = (A-B)*(A+B)`

Função C de dois argumentos A e B, que retornará um valor em precisão simples.

- `DEF FN M3$(X$) = MID$(X$,3,2)`

Função M3$ de um argumento X$, que retornará uma string composta pelo terceiro e o quarto caracteres do argumento.

- Seja uma função VL(X), definida por:

  a) VL(X) = 5 se X <= 0
  b) VL(X) = X+10 se X > 10

Então podemos definir uma função utilizando operadores lógicos da maneira como se segue:

```
DEF FN VL(X) = -5*(X<=0)-(X+10)*(X>0)
```

Caso X<=0 o valor de (X<=0) será -1 e (X>0) será 0, portanto o valor a ser devolvido será -5*(-1)+0=5. Caso X>10 o valor de (X<=0) será 0 e de (X>0) será -1, portanto o valor a ser devolvido será 0-(X+10)*(-1) = X+10.

Vejamos agora num programa como funciona :

```
10 REM EXEMPLO DEF FN
20 DEF FN VL(A) = -5*(A<=0)-(A+10)*(A>0)
30 CLS:PRINT@512,"";
40 INPUT "ENTRE COM O VALOR DE X ";X
50 PRINT "VALOR DA FUNCAO : ";FN VL(X)
60 PRINT "TECLE <C> PARA CONTINUAR"
70 Z$=INKEY$:IF Z$="" THEN 70
   ELSE IF Z$<>"C" THEN 70
        ELSE 30
```

Note que no exemplo acima foi utilizado para a definição da função VL o argumento A, e na linha 50 quando foi chamada o argumento era X. Portanto, não é necessário que a variável a ser passada para a função seja a mesma definida por ela.

DEF USR

DEFUSR n = endereço

Onde:

n - É um dígito de 0 a 9, n=0 poderá ser omitido.
endereço - Especifica o endereço de entrada de rotinas em linguagem máquina.

Este comando especifica o endereço de entrada de até 10 rotinas em linguagem máquina. Para chamá-las use USRn(arg).

EXEMPLO:

Do DIGDOS READY carregue o BASIC reservando a memória de 49136 em diante:

Digite: BASIC,49136

```
10 REM EXEMPLO DEFUSR
20 DEFUSR0 = &HBFF0
30 DATA 33,0,60,17,1,60,1,255,3,237,176,201
40 FOR I%=0 TO 11
50 READ P%: POKE &HBFF0+I%,P%
60 NEXT I%
70 PRINT@0,CHR$(191)
80 J=USR0(0)
90 Z$=INKEY$: IF Z$="" THEN 90
100 END
```

Rode o programa e a tela ficará instantaneamente branca. Para parar, tecle <RETURN>. Agora troque na linha 70,191 por 64 e rode o programa.

CAPÍTULO 4

MANIPULAÇÃO DE ARQUIVOS

1- SAVE

SAVE"arquivo1"<,A>

Este comando permite que seja gravado nos disquetes, programas em BASIC. O parâmetro opcional "A", faz com que o programa seja gravado em formato ASCII, ao invés do formato comprimido.

EXEMPLO:

- SAVE"MENU/BAS:1"

Grava no disquete montado no drive 1, o programa em BASIC atualmente na memória principal, com o nome de "MENU" e extensão "/BAS".

- SAVE"PROG1/BAS.SECRETO:0",A

Grava no disquete montado no drive 0, com o nome de "PROG1", extensão "/BAS", chave "SECRETO", o programa em BASIC da memória principal, em formato ASCII.

- SAVE"PROG2/BAS"
Grava no primeiro drive disponível.

2- LOAD

LOAD"arquivo1"<,R><,V>

Carrega na memória principal do computador o programa especificado por "arquivo1", sendo a opção R, para que o sistema não feche os arquivos e inicie a execução do programa e, a opção V, para que seja iniciado sem zerar as variáveis.

EXEMPLOS:

- LOAD"PROG1/BAS:0",R

Carrega o programa "PROG1/BAS" do disquete montado no drive 0, com a opção R,ou seja, inicializando-o logo a seguir sem que os arquivos, abertos até a execução do comando, sejam fechados, Corresponde ao RUN"PROG/BAS:0",R.

- LOAD"TESTE/BAS",V

Carrega na memória principal o programa "TESTE/BAS" com opção V, indicando ao sistema para iniciar a execução do programa preservando todas as variáveis, que estejam na memória, intactas.

- LOAD"TESTE1/BAS"

Carrega o programa referido, sem qualquer opção, deixando-o na memória principal.A ordem de execução deverá ser dada pelo operador (RUN).

NOTA:
As opções R e V não podem ser utilizadas simultaneamente.

3- RUN

RUN"arquivo1"<,R><,V>

Carrega e inicia a execução de um programa, sendo que as opções R e V têm o mesmo significado anterior, não podendo assim, serem utilizadas simultaneamente.

4- KILL

KILL"arquivo1"

Deleta um arquivo especificado por "arquivo1".

EXEMPLOS:

- KILL "PROG1/BAS:1"
- KILL "ESTOQUE/DAD:0"

5- MERGE

MERGE "programa"

programa - especificação de um programa em BASIC

Faz com que o programa especificado seja incluído no programa da memória. MERGE poderá ser executado tanto com o programa gravado em ASCII quanto no formato comprimido. Se MERGE for executado como uma instrução de um programa, após ser executada, o programa continuará na próxima linha, por exemplo:

```
200 MERGE "MENU/BAS"
210 IF A=50 THEN 250
```

As variáveis não são alteradas após a execução do comando MERGE.

NOTAS:

1- Caso o comando MERGE seja executado dentro de um programa, o programa a ser combinado não poderá ter qualquer linha que tenha mesma numeração do programa residente.

2- Não havendo espaço para ser executado o MERGE, ocorrerá erro de falta de memória.

6- ACESSO A ARQUIVOS DE DADOS

Existem duas técnicas de manipulação de arquivos e portanto dois tipos de arquivos distintos podem ser criados e manipulados. São estes:

ARQUIVO DE ACESSO SEQÜENCIAL
Neste tipo de arquivo, os dados são armazenados seqüencialmente, um após o outro até o fim do arquivo. Assim, a posição física de um dado é referente à posição do dado anterior e posiciona o dado seguinte, fazendo com que os dados só possam ser acessados seqüencialmente. É semelhante ao armazenamento de dados em cassete.

ARQUIVO DE ACESSO DIRETO (OU ALEATÓRIO)
Neste tipo de arquivo, a posição física de um dado é referida pelo próprio dado, já que estes ocupam campos fixos e pré-estabelecidos. Desta forma o arquivo aleatório permite que o dado seja acessado diretamente.

Cria-se os arquivos como segue:

- Através do comando CREATE do DIGDOS.
- Através do comando OPEN do DISK BASIC.

NOTA
Ao entrar no BASIC, deve-se especificar o número de BUFFERS a serem reservados para a manipulação de arquivos. Maiores detalhes, veja capítulo 3.

OPEN

Para acessar dados gravados em um arquivo, este deverá ser obrigatoriamente "aberto", o que se faz através do comando OPEN. Este comando permite também que através dele, o arquivo seja criado, não sendo necessário usar o comando CREATE do DIGDOS, em casos mais imediatos.

Existem 4 formatos para o comando OPEN:

1 - OPEN"I",n,"arquivo1"
2 - OPEN"O",n,"arquivo1"
3 - OPEN"E",n,"arquivo1"
4 - OPEN"R",n,"arquivo1",campo

Sendo:

n - O número do BUFFER usado para o acesso.
arquivo1 - especificação do arquivo.
campo - Número de bytes total referente ao campo do registro, comprimento lógico do registro.

EXEMPLO:

- OPEN"I",2,"XXX/DAD"

Abre o arquivo "XXX/DAD" no BUFFER 2, de acesso seqüencial, somente para leitura, iniciando no primeiro registro.

- OPEN"O",3,"XXX/DAD"

Abre o arquivo "XXX/DAD" de acesso seqüencial, somente para gravação, no BUFFER 3. Caso o arquivo já exista, as gravações anteriores serão destruídas.

- OPEN"E",11,"XXX/DAD"

Abre o arquivo para gravação de modo seqüencial, no BUFFER 11, posicionando o registro a ser gravado no final do arquivo. Esta opção é utilizada para "extender" um arquivo seqüencial já existente.

- OPEN"R",1,"XXX/DAD"

Abre o arquivo "XXX/DAD", e caso não exista, ele será criado, para acesso aleatório, no BUFFER 1, com o comprimento lógico do registro de 256 bytes e campo do registro conforme a especificação do comando FIELD.

- OPEN"R",6,"XXX/DAD",133

Para o mesmo acesso aleatório, o arquivo será aberto ou criado, com o comprimento lógico do registro de 133 bytes. Este arquivo, somente poderá ser manipulado se ao ser acionado o DISK BASIC, for especificada a opção "V", permitindo os BUFFERS terem comprimentos variáveis.

NOTA
Os arquivos diretos são abertos da mesma forma, para leitura ou gravação, já os seqüenciais serão abertos ou para leitura ou para gravação, nunca simultaneamente.

CLOSE

CLOSE<n<,n.....>>

Fecha os arquivos abertos via OPEN.

n - É um parâmetro opcional que se refere ao BUFFER que deve ser fechado. Se omitido, todos os BUFFERS serão fechados. Seu valor é naturalmente entre 1 e 15.

EXEMPLO:

- CLOSE

Fecha todos os arquivos abertos.

- CLOSE 1,2,5

Fecha os arquivos nos BUFFERS 1,2,5.

- CLOSE N%

Fecha o arquivo que está no BUFFER cujo número é o valor da variável N%.

Em algumas circunstâncias, os arquivos serão fechados automaticamente pelo sistema, afim de evitar qualquer alteração em seus conteúdos. Quando se executa um NEW, RUN, EDIT, ou adicionando e deletando linhas em programas BASIC, executando CLEAR, erros nos discos, os arquivos serão automaticamente "fechados".

NOTA

Nunca retire o disquete do drive com algum arquivo aberto. Isto poderá danificar o conteúdo dos últimos 256 bytes. Desta forma, certifique-se que todos os arquivos estejam fechados para depois retirar os disquetes.

CAPÍTULO 5

ARQUIVOS

A) COMANDOS DE MANIPULAÇÃO DE ARQUIVOS SEQÜENCIAIS

1 — INPUT#

INPUT#n,var1<,var2,var3,.....,varN>

n - Especifica o número do BUFFER usado para a transferência dos dados.
var - Variável a ser lida no disco.

Esta instrução e' semelhante à instrução INPUT#-1 usada na leitura de dados gravados em cassete. Os dados serão lidos seqüencialmente, e a leitura iniciará no primeiro registro do arquivo, pois quando aberto, um apontador indicará o primeiro dado, que após lido faz que o apontador passe ao segundo e assim sucessivamente até o fim do arquivo (EOF). Para reiniciar a leitura do arquivo, ele deverá ser fechado e reaberto para leitura (OPEN "I").

O importante em uma instrução INPUT# está em como os "caracteres delimitadores" estão posicionados, e se foi alcançado o final do arquivo (EOF).

CARACTERES DELIMITADORES

VARIÁVEIS NUMÉRICAS:

EOF - Fim de arquivo encontrado
255g. caracter encontrado
, - vírgula (CHR$(44))
RETURN <↓> - CHR$(13) CHR$(10)
espaço <espaço ...> <RETURN> <↓>

VARIÁVEIS STRINGS ENTRE ASPAS:

EOF - Fim de arquivo encontrado
255g. caracter encontrado
" - aspas
" <espaço ...> <,>
" <espaço ...> <RETURN> <↓>

VARIÁVEIS STRINGS SEM ASPAS:

EOF - Fim de arquivo encontrado
255g. caracter encontrado
, - vírgula (CHR$(44))
RETURN <↓>

NOTA:
Quando o caracter <RETURN> (CHR$(13)) for precedido por <↓> (CHR$(10)), este conjunto não será considerado como delimitador.

EXEMPLOS:

- LEITURA NUMÉRICA

a) Suponha que a imagem do disco seja:
<espaço>2.718<espaço>-3.1416<espaço>A10.3<RETURN>

Uma instrução INPUT#1,A,B,C equivalerá a 3 instruções INPUT#1,A: INPUT#1,B: INPUT#1,C e fará que as variáveis A,B,C contenham os valores 2.718, -3.1416, 0, respectivamente. Note que o valor atribuído a "C" foi 0(zero), pois ao ser lido um valor do disco, o computador executa um VAL, e como o primeiro caracter lido foi um não numérico, o resultado será zero.

b) Suponha a imagem do disco:
<espaço>3<↓><RETURN>14<espaço>16.3<RETURN>32.6<EOF>

Uma instrução INPUT#1,A,B,C, fará que as variáveis A,B,C tenham os valores 314,16.3,32.6 respectivamente. Neste caso o caracter RETURN (CHR$(13)) vem precedido de ↓ (CHR$(10)), não considerando este conjunto como delimitadores.
Os delimitadores neste exemplo são:

- Espaço após 14
- RETURN após 16.3
- EOF após 32.6.

- LEITURA DE STRINGS

Suponha as imagens do disco:

a) "DIGITUS"DGT-1000<EOF>
b) <espaço>DIGITUS<RETURN><↓>DGT-1000<EOF>
c) <espaço>DIGITUS<,>DGT-1000<EOF>

A instrução INPUT#1,A$,B$ atribuirá às variáveis A$ e B$ os mesmos valores, i.e. DIGITUS e DGT-1000, utilizando caracteres delimitadores diferentes.

2- LINE INPUT#

LINEINPUT# n,vstr$

n - especifica o número do BUFFER usado para a transferência de dados.
vstr$ - nome da variável string.

Esta instrução lê uma linha de texto gravada em disco. Esta instrução é semelhante ao LINEINPUT introduzido pelo teclado. A leitura é feita até que se encontre um dos seguintes caracteres finalizadores:

1 - <RETURN> não precedido por <↓> (CHR$(10))
2 - EOF - fim de arquivo encontrado
3 - 255o. caracter encontrado.

Outros caracteres encontrados como aspas, vírgulas, espaços, ou <RETURN> farão parte da string.

EXEMPLOS:

- "DIGITUS"DGT-1000<EOF>

LINEINPUT#1,A$ faz que A$ assuma o valor de "DIGITUS"DGT-1000, inclusive as aspas.

- Programas gravados em ASCII poderão ser lidos linha a linha através de instruções LINEINPUT#. As linhas serão lidas até que se encontre o caracter finalizador da linha, no caso de programa o <RETURN>.

3- PRINT#

PRINT #n,<USING "formato$";> var <d var>

n - especifica o número do BUFFER usado para transferência dos dados.
formato$ - argumento que indica o formato de gravação, utilizando a opção USING do DIGBASIC II.
d - carater delimitador colocado entre duas expressões a serem gravadas no disco, podendo ser usados ponto e vírgula (;) ou vírgula (,) (ponto e vírgula é mais usado).
var - variável ou expressão que deverá ser gravada no disco.

Esta instrução funciona como PRINT#-1 utilizado na gravação em cassetes. A instrução PRINT#n gravará no formato ASCII a imagem do dado, portanto -1.457 ocupará 6 bytes no disco. Deve-se sempre tomar o cuidado de posicionar corretamente todos os caracteres.

O arquivo deverá estar aberto com opção "E" ou "O" para gravação.

EXEMPLOS:

- A=-2.7182
  PRINT#1,A <RETURN>
Isto gravará no disco 10 caracteres, sendo 7 do número, um espaço à frente, outro atrás e o caracter <RETURN>

- A=31416 B=2.71828
  PRINT #1,A,B <RETURN>
Gravará no disco:
<espaço>31416<10 espaços>2.71828<espaço>
A vírgula entre A e B fará que se gaste 10 espaços extras no disco, por isto em lugar de se usar a vírgula deve-se usar o ponto e vírgula, como:
  PRINT #1,A;B <RETURN>
Produzirá no disco a imagem:
  31416 2.71828 <RETURN>
No caso de variáveis strings deverão ser usados delimitadores como no exemplo seguinte.

- A$="DIGBASIC II"
  B$="DIGDOS"
  PRINT#1,A$;B$ <RETURN>
Produzirá uma imagem de:
DIGBASIC IIDIGDOS <RETURN>

Desta maneira as duas variáveis não poderiam ser lidas separadamente pois não há nenhum delimitador entre as variáveis, veja relação dos "caracteres delimitadores". A instrução acima deverá então ter o seguinte formato:
PRINT#1,A$;",";B$ <RETURN>
Produzindo uma imagem de:
DIGBASIC II,DIGDOS <RETURN>
Agora as duas variáveis poderão ser recuperadas separadamente.

B) COMANDOS DE MANIPULAÇÃO DE ARQUIVOS DE AC DIRETO

4- FIELD

FIELD n,v1 AS str1$ <,v2 AS str2$ ...>

n - especifica o número do BUFFER usado pelo arquivo de acesso direto (ou aleatório), para transferência de dados.
v1 - comprimento do primeiro campo do buffer.
str1$ - nome do primeiro campo.
Os próximos pares especificarão os comprimentos e nomes dos respectivos campos usados no buffer.

Esta instrução é usada para organizar os campos no buffer de um arquivo de acesso aleatório. A soma de todos os comprimentos de campos relacionados no FIELD, deverá ser igual ao comprimento do registro especificado no OPEN respectivo. O FIELD poderá ser usado quantas vezes for necessário para reorganizar o buffer.

Neste tipo de arquivo todos os dados são gravados na forma de strings e existem 3 pares de funções que servirão para transformar variáveis numéricas em strings e vice-versa. Estas funções (MKI$/CVI, MKD$/CVD e MKS$/CVS) serão vistas posteriormente.

NOTA:
Nenhuma variável relacionada com o FIELD, poderá ser utilizada em um comando INPUT, LINEINPUT e nem em instrução comum de atribuição, pois deixarão de fazer parte do campo do BUFFER tornando-se variáveis comuns.

EXEMPLO:

```
20 OPEN "R",1,"ARQUIVO1/DAD",36
30 FIELD 1,2 AS X$,4 AS Y$,30 AS NM$
```

NOTA:
Na linha 30
2 AS X$ - 2 bytes reservados para um número inteiro.
4 AS Y$ - 4 bytes reservados para um número de precisão simples.
30 AS NM$ - 30 bytes reservados para uma variável string.
Os espaços serão alocados seguindo as posições indicadas pelo FIELD, no BUFFER 1.

5- GET - PUT

GET n,n1 - PUT n,n1

n - especifica o número do buffer usado para a transferência dos dados.
n1 - número do registro a ser lido ou gravado.

A instrução GET n,n1 lê o registro número n1 especificado no buffer n.

A instrução PUT n,n1 grava o conteúdo do buffer n no registro n1.

Se n1 for omitido, este assumirá o valor do registro corrente. O registro corrente corresponde ao número do último registro lido ou gravado +1. Logo após ser aberto o arquivo, o registro corrente é 1.

EXEMPLOS:

- GET 1,N%
Lê o registro N% e o transfere para o buffer 1.

- GET A%
Lê o registro corrente e o transfere para o buffer A%.

- PUT N1%,N2%
Grava o conteúdo do buffer N1% no registro N2%.

6- LSET - RSET

LSET str$=exp$ - RSET str$=exp$

str$ - é o nome do campo
exp$ - dado que deverá ser colocado no campo especificado.

Estes comandos posicionam as variáveis nos campos especificados no FIELD.

LSET - Posiciona a variável à esquerda do campo.
RSET - Posiciona a variável à direita do campo.

NOTA:
Caso a string seja maior que o campo especificado, esta será sempre truncada à direita e os caracteres excedentes serão ignorados.

7- CVD - CVI - CVS

CVD(exp$) - Transforma 8 caracteres strings em um número de precisão dupla. Se LEN(exp$)<8 ocorrerá erro, se LEN(exp$)>8 então somente os oito primeiros caracteres serão usados.

CVI(exp$) - Transforma 2 caracteres strings em um número inteiro. Se LEN(exp$)<2 ocorrerá erro, se LEN(exp$)>2 então somente os dois primeiros caracteres serão usados.

CVS(exp$) - Transforma 4 caracteres strings em um número de precisão simples. Se LEN(exp$)<4 ocorrerá erro, se LEN(exp$)>4 então somente os quatro primeiros caracteres serão usados.

Estas funções servem para recuperar variáveis numéricas gravadas em arquivos de acesso direto.

8- MKD$ - MKI$ - MKS$

MKD$(nd) - Converte um valor de precisão dupla em um string de 8 caracteres.

MKI$(ni) - Converte um valor inteiro em um string de 2 caracteres. Se ABS(ni)>32768 ocorrerá erro. Todas as decimais, caso existam serão abandonadas.

MKS$(ns) - Converte um valor de precisão simples em um string de 4 caracteres.

Estas funções executam o inverso das funções vistas anteriormente, transformando as variáveis numéricas em strings, afim de posicioná-las nos respectivos campos relacionados no FIELD.

EXEMPLOS:

```
90 ...
100 OPEN "R",1,"EXEMPLO/DAD",32
110 FIELD 1,4 AS E1$,8 AS E2$,20 AS NM$
...
...
200 A#=3.141592654
210 E!=2.71828
220 D$="DIGITUS"
230 LSET E1$=MKS$(E!)
240 RSET E2$=MKD$(A#)
250 LSET NM$=D$
260 PUT 1,1
...
```

Nas linhas 230 a 250 as variáveis numéricas são transformadas em strings e posicionadas no buffer. Na linha 260 o conteúdo do buffer 1 é gravado no registro número 1.

9- EOF

EOF(n)

n - especifica o número do buffer.

Esta função verifica se o EOF (fim de arquivo) foi atingido, evitando ultrapassagem do número máximo de dados gravados no arquivo. Caso seja verdadeiro, i.e. fim de arquivo atingido, o valor de EOF(n) será -1, caso contrário será 0(zero).

EXEMPLOS:

- IF EOF(4) THEN PRINT "FIM DO ARQUIVO "N4$
- IF EOF(I%) THEN CLOSE I%:END

10- LOF

LOF(n)

n - especifica o número do buffer.

Esta função fornece o número do maior registro existente no arquivo.

Quando se deseja estender um arquivo de acesso direto, o primeiro registro a gravar será o número LOF(n)+1.

EXEMPLOS:

- Ler um arquivo de 100 ítens e armazená-lo na memória do computador.

```
10 CLEAR 25500
20 OPEN "R",1,"ARQ23/TXT",255
30 FIELD 1,255 AS AR$
40 DIM B$(LOF(1))
50 FOR I%=1 TO LOF(1)
60 GET 1
70 B$(I%)=AR$
80 NEXT
```

- Acrescentar um registro após o último do arquivo.

```
...
100 R%=LOF(1)+1
110 PUT 1,R%
...
```

C) CARACTERÍSTICAS DOS ARQUIVOS DE ACESSOS DIRETOS E SEQÜENCIAIS

1- ARQUIVOS DE ACESSO SEQÜENCIAL

1 - O arquivo é gravado em ASCII.

2 - Todos os dados são acessados seqüencialmente.

3 - Quando o arquivo é aberto, somente poderá ser feito gravação ou leitura, uma de cada vez.

4 - Para atualização de qualquer dado do arquivo, este deverá ser totalmente reescrito.

5 - Os comprimentos dos registros são sempre iguais a 256 bytes.

2- ARQUIVOS DE ACESSO DIRETO (OU ALEATÓRIO)

1 - O arquivo é gravado no formato comprimido.

2 - Os dados podem ser acessados aleatoriamente.

3 - O arquivo é aberto tanto para gravação como para leitura.

4 - Qualquer dado poderá ser atualizado desde que seu conteúdo nao exceda os campos anteriormente previstos.

5 - Os dados são gravados em blocos de comprimentos fixos.

Os arquivos de acesso direto são de mais rápido acesso e ocupam menos espaço físico no disco, enquanto que os de acesso seqüencial são mais facilmente manipulados.

APÊNDICE 1

CHAINBLD/BAS - PROGRAMA EDITOR

CHAINBLD/BAS é um programa em BASIC, para que o usuário possa criar e modificar arquivos, usados pelo o comando CHAIN (ou DO).

Carregue o programa normalmente, como se faz com um programa em BASIC. Ao iniciar sua execução virá no vídeo o "menu principal", não confundir com o "menu de edição", e deverá ser escolhida uma das opções apresentadas.

1 - DELETAR TODAS AS LINHAS DO TEXTO

Deletar todas as linhas do texto existente na memória. Ao ser perguntado se é este o desejo, e for respondido com a opção afirmativa, o texto será deletado e será apresentado no vídeo o "menu de edição". Caso não exista algum texto, o "menu de edição" será apresentado logo após a entrada da opção.

2 - CARREGAR UM TEXTO DO DISCO

Carrega um texto já existente para a edição. Ao ser introduzida a opção, deverá ser dada a especificação do arquivo desejado. Caso o arquivo exista, o "menu de edição" será apresentado.

3 - GRAVAR UM TEXTO NO DISCO

Esta opção grava um arquivo, que será utilizado pelo CHAIN, existente na memória. Se o arquivo especificado não contiver uma extensão, esta será "/JCL". Caso não exista texto na memória, será perguntado se deverá ser gravado um arquivo vazio.

4 - EDITAR TEXTO

Caso não exista um texto na memória, este deverá ser carregado pela opção 2. Ao ser escolhida esta opção o "menu de edição" será apresentado.

SINTAXE:

MM, NN e TT são números de linhas.
Para algumas linhas, poderão ser usadas algumas abreviaturas como:

- ↑ = Primeira linha.
- / = Última linha.
- Ø = Linha anterior a primeira, se existir a 1ª.

L MM - Listar linhas a começar pela linha MM. Será mostrado uma página, e se desejar use o comando ";" (ponto e vírgula) para listar a próxima página.

EXEMPLO:
- L 10 lista a partir da décima linha.

R MM - Recolocar a linha MM. A linha MM deve existir, caso contrário dará um erro.

EXEMPLO:
- R 03 A linha 3, caso exista será reposta e o CHAINBLD perguntará por ela.

D MM NN - Deletar da linha MM até NN, inclusive. Ao serem deletadas estas linhas, as de numeração superior, caso existam, ocuparão os lugares das outras. As linhas MM e NN deverão existir.

EXEMPLO:
- D 10 20 As linhas de 10 a 20, inclusive, serão deletadas.

I MM - Inserir novas linhas após a linha MM. A linha MM deverá existir.

EXEMPLO:
I 1 Acrescentar após a 1ª. linha.
I 0 Acrescentar antes da 1ª. linha.
I / Acrescentar após a última linha.

X MM - Extender a linha MM.

EXEMPLO:
- X 10 Extender a linha 10.

C MM NN TT - Inserir cópias das linhas MM até NN, inclusive, após a linha TT. As linhas de MM a NN não serão deletadas.

EXEMPLO:
- C 1 10 / Copia as linhas de 1 a 10 após a última linha do texto.

M MM NN TT - Executa o mesmo comando que "C", porém as linhas de MM a NN serão deletadas.

EXEMPLO:
- M 10 20 / As linhas de 10 a 20 serão copiadas após a última linha do texto, e serão deletadas as linhas originais.

P - Este comando fará que o "menu principal" seja mostrado.

E - Este comando fará que o "menu de edição" seja mostrado.

5 - FINALIZAR

Esta opção termina a execução do programa. Se houver algum novo texto ou texto editado na memória sem gravar, será perguntado se este deverá ou não ser preservado.

A melhor maneira de se aprender o CHAINBLD é utilizando-o. Carregue o exemplo CHAINTST/JCL, que vai em cada disco de sistema do DIGDOS, e veja como ele foi elaborado, editando-o. Naturalmente, ao retorná-lo para o disco, use outro nome para não perdê-lo.

APÊNDICE 2

CONVERTE/JCL

O programa utilitário CONVERTE/JCL, acompanha o seu sistema para tornar possível a conversão de arquivos de disquetes cujo formato e sistema operacional sejam estranhos ao DIGDOS. Seu acionamento se faz através dos comandos:

DO CONVERTE/JCL ou
CHAIN CONVERTE/JCL

acionados quando em DIGDOS READY.

É necessário porém que o usuário assegure-se antes de iniciar sua execução que o disquete DESTINO tenha o formato de acordo com o sistema em que será utilizado. Este programa não é capaz de formatar os disquetes envolvidos na conversão, mesmo sendo o formato do próprio DIGDOS.

Os processos de conversão são detalhados pelo próprio programa de forma que não é necessário qualquer estudo prévio para sua utilização. O usuário deve garantir que o disquete de sistema montado no drive Ø durante a conversão, possua a chave de acesso DIGITUS, qualquer outra chave atribuida ao sistema tornará a conversão impossível.

APÊNDICE 3

DEBUG

O DEBUG é um utilitário, que faz parte do DIGDOS e, que lhe facilitará interromper a execução do programa corrente, afim de inspecionar a memória, ou disco, alterando-os se for o caso.

O DEBUG poderá ser chamado pressionando as teclas 123 simultaneamente.

No vídeo aparecerão 3 conjuntos de 64 bytes com seus respectivos endereços e os caracteres correspondentes em ASCII ao lado direito da tela. Códigos que não correspondam a nenhum caracter ASCII serão representados como um "."(ponto).

Os comandos do DEBUG são geralmente de um caracter e todos terminados por <RETURN>. Todos os endereços mencionados serão em HEXADECIMAL, não necessitando do sufixo H.

COMANDOS:

Q <RETURN> - retorno ao DIGDOS READY.

X <RETURN> - tela "parcial". O vídeo conterá 15 linhas, que são:

1 a 4 - primeiro conjunto de 64 bytes.
5 - conteúdo dos registradores AF, BC, DE, HL.
6 a 9 - segundo conjunto de 64 bytes.
1Ø - conteúdo dos registradores AF', BC', DE', HL'.
11 a 14- terceiro conjunto de 64 bytes.
15 - conteúdo dos registradores PC, SP, IX, IY.

S <RETURN> - tela "total". Apresenta os 256 bytes do primeiro conjunto da tela X.

EXEMPLO:
Supondo que no primeiro conjunto da tela X o endereço inicial seja AE28, ao executar o comando S <RETURN>, a tela apresentará um conjunto de 256 bytes desde AEØØ até AEFF com seus caracteres ASCII correspondentes à direita.

<n>Dxxxx - Mostrará os primeiros 64 bytes no conjunto especificado por n, a partir do endereço xxxx. Se a tela estiver no tipo "total", e n não for especificado, serão apresentados 256 bytes na tela a partir de xx00 ate xxFF.

EXEMPLO:
D5250 - se a tela estiver no tipo "total", serão listados os bytes de 5200 até 52FF, caso contrário, serão mostrados no primeiro conjunto da tela, os bytes de 5250 a 528F.

Mxxxx - Indica que os bytes a partir de xxxx poderão ser modificados. Um cursor intermitente indicará o byte do endereço especificado, podendo agora serem feitas as modificações desejadas. O vídeo irá sempre para a tela do tipo "total".

EXEMPLO:
M 843C - O vídeo irá para o tipo "total", caso não esteja, e serão mostrados os bytes de 8400 a 84FF com o cursor acima do byte 843C. Terminadas as modificações, tecle <RETURN>.

F<XXXX><,bt1><,bt2><,bt3><,bt4>

O computador iniciará uma busca, a partir do endereço XXXX, do conjunto de até 4 bytes especificados por bt1, bt2, bt3, bt4. Se nenhum dos bytes bt1, bt2, bt3, bt4 são especificados, seus valores assumirão os mesmos valores anteriores.

EXEMPLO:
F4000,C3,00,FF

Deverão ser procurados a seqüência de 3 bytes C3, 00, FF a partir do endereço 4000.
Se digitar F <RETURN>, a próxima ocorrência do byte será mostrada e assim por diante.

O debug guarda os bytes a serem procurados na região de memória S1xxH. Portanto se esta região é procurada, haverá uma ocorrência neste local.

DIGITUS
Ind. Com. e Serv. de Eletrônica Ltda.

RUA GÁVEA, 150 - FONE: 332-8300
BELO HORIZONTE - MG