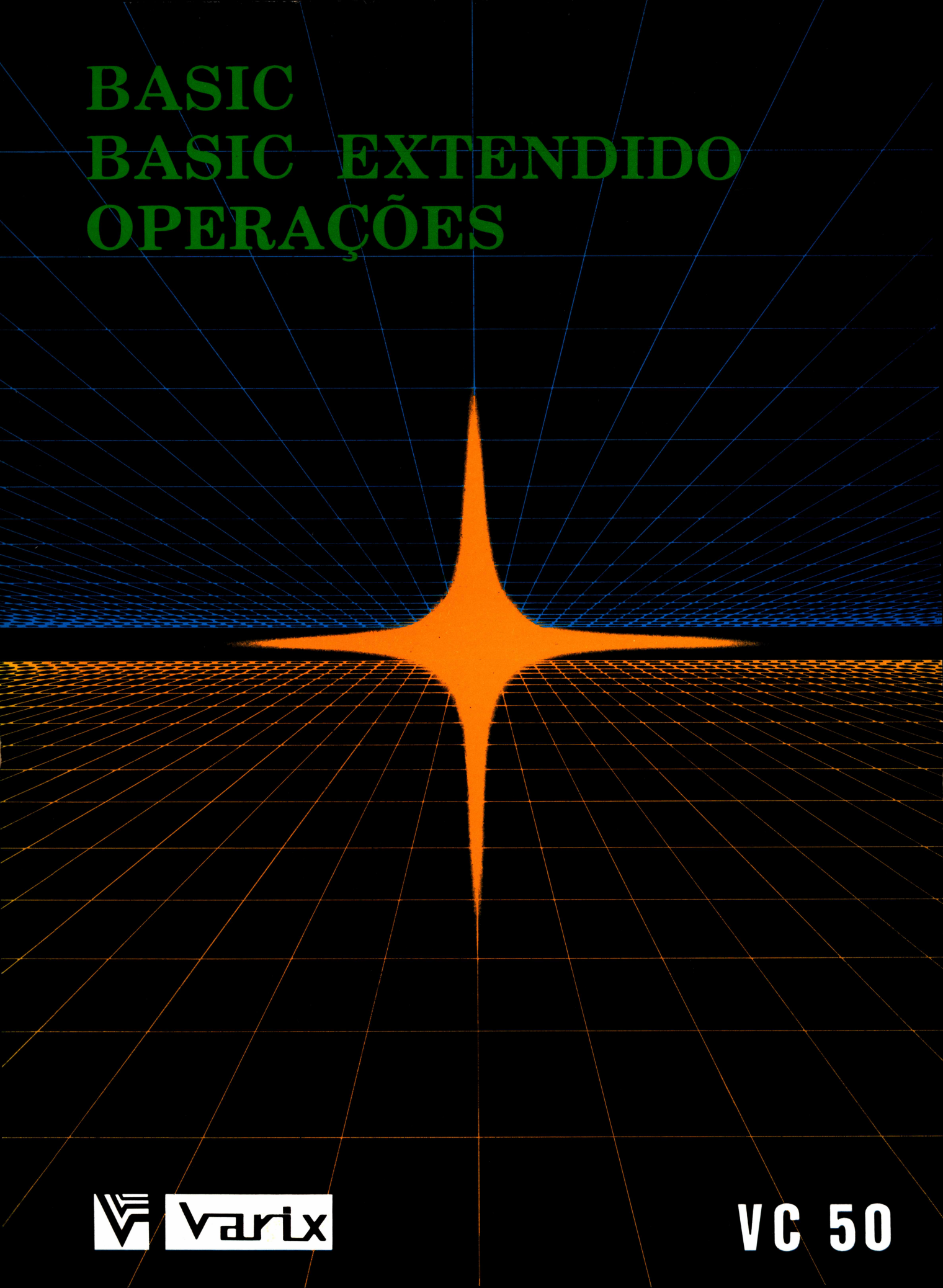
BASIC
BASIC EXTENDIDO
OPERAÇÕES



Varix

VC 50

BASIC
BASIC EXTENDIDO
MANUAL DE OPERAÇÕES

MICROCOMPUTADOR VARIX VC50

1.ª EDIÇÃO

Engetécnica Varix Eng. Com. e Ind. Ltda.
1984 - Piracicaba - SP - Brasil

Fica proibida a reprodução total ou parcial deste livro sem expressa permissão por escrito. Os infratores estarão sujeitos às penas previstas na lei.

Composição, Arte e Fotolitos:
Engetécnica Varix Eng. Com. e Ind. Ltda.

Direitos autorais reservados ao editor:

Engetécnica Varix Eng. Com. Ind. Ltda.
Rua Phelippe Zaidan Maluf, 1501
Distrito Industrial Unileste
Caixa Postal 642 - Telex 019 2454 VARX-BR
Fone (DDD 0194) 33-7000 (tronco)
13400 Piracicaba (SP)

ENGETÉCNICA e VARIX são marcas registradas

EQUIPE TÉCNICA DA ENGETÉCNICA VARIX

BASIC
BASIC EXTENDIDO
MANUAL DE OPERAÇÕES

MICROCOMPUTADOR VARIX VC50

1.ª EDIÇÃO

© 1984 BRASIL

AO LEITOR

Este livro foi escrito e editado não só para servir de manual para o usuário do microcomputador VARIX VC 50, mas também, como ferramenta de aprendizado da linguagem Basic.

Como o Basic aqui tratado é um dos mais completos existentes no mercado mundial, sendo um dos mais utilizados nos Estados Unidos, possuindo comandos gráficos poderosos, comandos de som, manipulação de strings (letras) , etc., será uma ótima opção para o estudante.

Após o aprendizado com este livro, qualquer pessoa que vá trabalhar com computadores, de qualquer modelo, terá pouca ou nenhuma dificuldade para se adaptar às pequenas diferenças existentes entre as diversas versões de Basic.

Como qualquer matéria, o leitor poderá achar o início do aprendizado, um tanto insípido, porém com o domínio do assunto, irá adquirindo gosto. O importante é perseverar.

O EDITOR

ÍNDICE

Página

MANUAL DE OPERAÇÕES XVII
INTRODUÇÃO 1

BASIC

CAPÍTULO I - PRIMEIROS CONTACTOS 2
1. Ligando 2
2. Teclado 2
3. Cursor 3

CAPÍTULO II - ESCREVENDO NA TELA 4

CAPÍTULO III - LIMPANDO A TELA 6

CAPÍTULO IV - SOM 7

CAPÍTULO V - VARIÁVEIS 8

CAPÍTULO VI - PROGRAMA 10

CAPÍTULO VII - UM POUCO DE MATEMÁTICA 12
1. Operações elementares 12
2. Exponenciação 14
3. Notação científica 15

CAPÍTULO VIII - MAIS PROGRAMAÇÃO 16
1. INPUT 16
2. READ e DATA 18
3. RESTORE 20

Página

CAPÍTULO IX - DESVIOS 21

1. GOTO 21
2. IF ... THEN 23
3. ELSE 25
4. AND, OR e NOT 25
5. Ordenando alfabeticamente 26

CAPÍTULO X - FOR/NEXT 26

CAPÍTULO XI - TEMPO 29

CAPÍTULO XII - NÚMEROS SURPRESA 32

CAPÍTULO XIII - TELA 33

1. PRINT @ 34
2. SET e RESET 35
3. JOYSTICK 37
4. PRINT TAB 40
5. O uso da vírgula e do ponto e vírgula em comandos PRINT 41
6. CHR$(A) e ASC(A$) 41
7. POINT (X,Y) 45

CAPÍTULO XIV - ORDENANDO ALFABETICAMENTE 45

CAPÍTULO XV - O USO DO GRAVADOR K-7 46

1. Salvando o programa em fita 47
2. Carregando o programa da fita 47
3. Posicionando a fita 48

CAPÍTULO XVI - SUB-ROTINAS 49

CAPÍTULO XVII - COMENTÁRIO NA LISTAGEM 51

Página

CAPÍTULO XVIII - MANIPULANDO STRINGS 51

1. LEN(A$) 51
2. STR$(X) 52
3. VAL(A$) 52
4. LEFT$(A$,N) 52
5. RIGHT$(A$,N) 53
6. MID$(A$,M,N) 53
7. Exemplos 53
8. INKEY$ 54

CAPÍTULO XIX - COMANDOS ÚTEIS 55

1. STOP e CONT 56
2. MEM 57

CAPÍTULO XX - MAIS DESVIOS 57

1. ON...GOTO 57
2. ON...GOSUB 59

CAPÍTULO XXI - FAZENDO O COMPUTADOR FALAR 60

CAPÍTULO XXII - VARIÁVEIS INDEXADAS 61

1. Variáveis unidimensionais (índice simples) 62
2. Variáveis bidimensionais (índice duplo) .. 63
3. Variáveis tridimensionais (índice triplo). 63
4. Instrução DIM 63
5. Exemplo 1 64
6. Exemplo 2 65

CAPÍTULO XXIII - ARQUIVO MAGNÉTICO EM FITA 69

CAPÍTULO XXIV - IMPRESSORA 72

1. LLIST 73
2. PRINT #-2 73

Página

BASIC EXTENDIDO

CAPÍTULO I - ALTA RESOLUÇÃO GRÁFICA 76

CAPÍTULO II - TRAÇANDO LINHAS 79

CAPÍTULO III - CONTROLANDO AS CORES 84

CAPÍTULO IV - MEMÓRIA E VÍDEO 85

CAPÍTULO V - PCLS 91

CAPÍTULO VI - PCLEAR 91

CAPÍTULO VII - PCOPY 92

CAPÍTULO VIII - PPOINT 94

CAPÍTULO IX - SCREEN 96

CAPÍTULO X - CÍRCULOS 97

CAPÍTULO XI - PAINT 102

CAPÍTULO XII - SIMPLIFICANDO AINDA MAIS 104

CAPÍTULO XIII - GET E PUT 117

CAPÍTULO XIV - TOQUE NOVAMENTE, VC50 122

CAPÍTULO XV - ALTERANDO LINHAS DE UM PROGRAMA. 131

CAPÍTULO XVI - UM POUCO MAIS DE MATEMÁTICA 143

1. Raiz Quadrada (SQR) 143
2. Funções Trigonométricas 144
3. LOG(X) 145
4. EXP(X) 146
5. INT(X) 146
6. FIX(X) 147

Página

7. ABS(X) 147
8. SGN(X) 148
9. DEF FNA(X) 148

CAPÍTULO XVII - MANIPULANDO CADEIAS 149

1. STRING $ 149
2. INSTR 151
3. MID$ 153

CAPÍTULO XVIII - INPUT/OUTPUT 156

1. LINE INPUT 156
2. PRINT USING 157
3. POS 163
4. IMPRESSÃO 164
5. TRON/TROFF 165
6. TIMER 166

CAPÍTULO XIX - CONSTANTES HEXADECIMAIS E OCTAIS. 167

CAPÍTULO XX - USANDO SUB-ROTINAS EM LINGUAGEM DE MÁQUINA 169

CAPÍTULO XXI - SUB-ROTINAS DA ROM DISPONÍVEIS PARA USO DO BASIC 180

APÊNDICE

EXERCÍCIOS RESOLVIDOS 187
PROGRAMAS SIMPLES 201
SUB-ROTINAS CIENTÍFICAS 224
FÓRMULAS TRIGONOMÉTRICAS 230

Página

FUNÇÕES DERIVADAS 233
RELAÇÃO DE CÓDIGOS ASCII 235
CARACTERES DO TECLADO 237
SÍMBOLOS BASIC 237
OPERADORES BASIC 237
CONVERSÃO DE BASE 239
PRINT @ 242
64 x 32 243
128 x 96 244
128 x 192 245
256 x 192 246
CORES E CARACTERES GRÁFICOS 247
CORES COM BASIC EXTENDIDO 249
TABELA DE CONJUNTO DE CORES 249
TONS MUSICAIS 250
NOTAS MUSICAIS 253
COMANDOS DO VC-50 254
MENSAGENS DE ERRO 269
MAPA DE MEMÓRIA 273
COMANDOS POKE, PEEK e EXEC 292
MAXITELA 300
VIDEO PACK 301

MANUAL DE OPERAÇÕES

MANUAL DE OPERAÇÕES

1. INTRODUÇÃO

O Manual de Operações destina-se a familiarizar o usuário, com a parte física do equipamento, descrevendo as ligações dos periféricos e comandos necessários para inicializá-los.

Por outro lado, o usuário encontrará também algumas informações úteis, como comandos de alteração, de operação, etc.

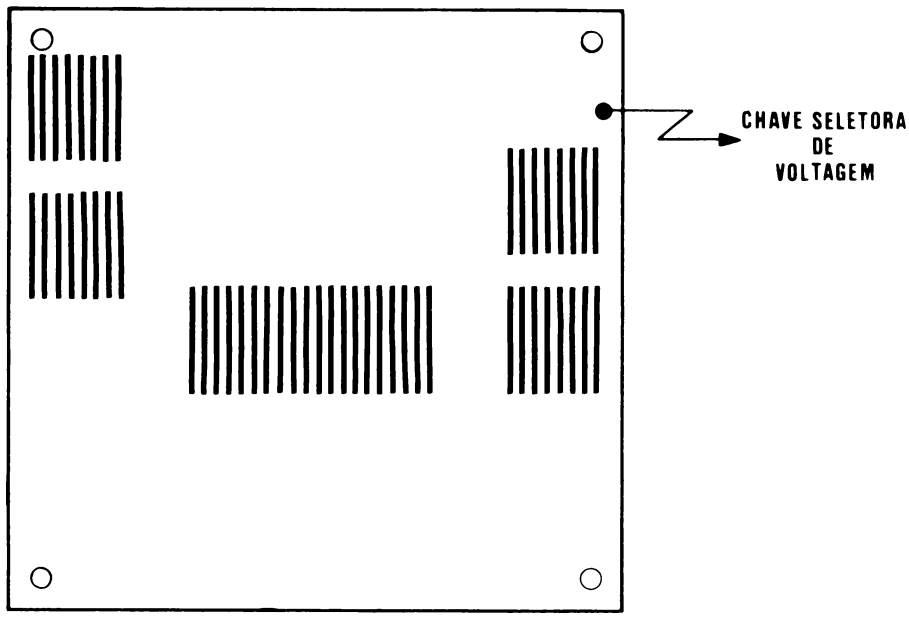
Leia com atenção o tópico correspondente, antes de conectar um periférico ou acessório.

Se você não entender alguma informação técnica mais avançada (POKE por ex.) não se preocupe, mais tarde você terá melhores detalhes e com o tempo, o domínio do sistema. Muitas das informações que se obtém nesse manual, não são necessárias para que se opere o sistema normalmente, porém para o programador avançado, que se utiliza do assembler por exemplo, serão de grande utilidade.

2. INSTALAÇÃO

2.1. CONECÇÃO À REDE DE FORÇA

Após abrir a caixa, verifique a voltagem na chave posicionada embaixo do computador (furo no canto posterior esquerdo), com auxílio de uma chave de fenda, mude a tensão para 110 ou 220 V., conforme a sua instalação.



2.2. CONEXÃO DE VIDEO

Pode-se usar 3 tipos de vídeo a saber:

1 - T.V. modificada ou monitor de vídeo
2 - T.V. Normal (opcional)
3 - Monitor de vídeo interno incorporado (opcional)

2.2.1. T.V. Modificada ou monitor de vídeo

Este é o modo normal de operação do computador e mesmo que se tenha opcionalmente os outros dois modos, a saída de vídeo composto, estará presente para ligação do monitor.

A modificação de uma T.V. para receber vídeo composto é muito simples, podendo ser feita por qualquer técnico de televisão.

A modificação consta de uma tomada com chave normalmente fechada e um cabo coaxial ligado diretamente na entrada do amplificador de vídeo da T.V., conseguindo assim melhor qualidade em relação a que se consegue ligando-se à antena (através do modulador de RF opcional).

Basta desconectar o cabo, para que a T.V. seja usada normalmente.

A T.V. pode ser colorida (ideal) ou branco e preto.

No caso de se utilizar um **monitor de vídeo**,(existem vários no mercado) este já possue entrada de vídeo composto, estando pronto para uso. O monitor poderá ser colorido ou não. A grande vantagem do monitor é a alta resolução ou seja "nitidez" e "detalhes" do vídeo. O custo de um monitor monocromático pode ser mais barato que uma T.V. colorida, enquanto que, o monitor colorido é mais caro ou equivalente.

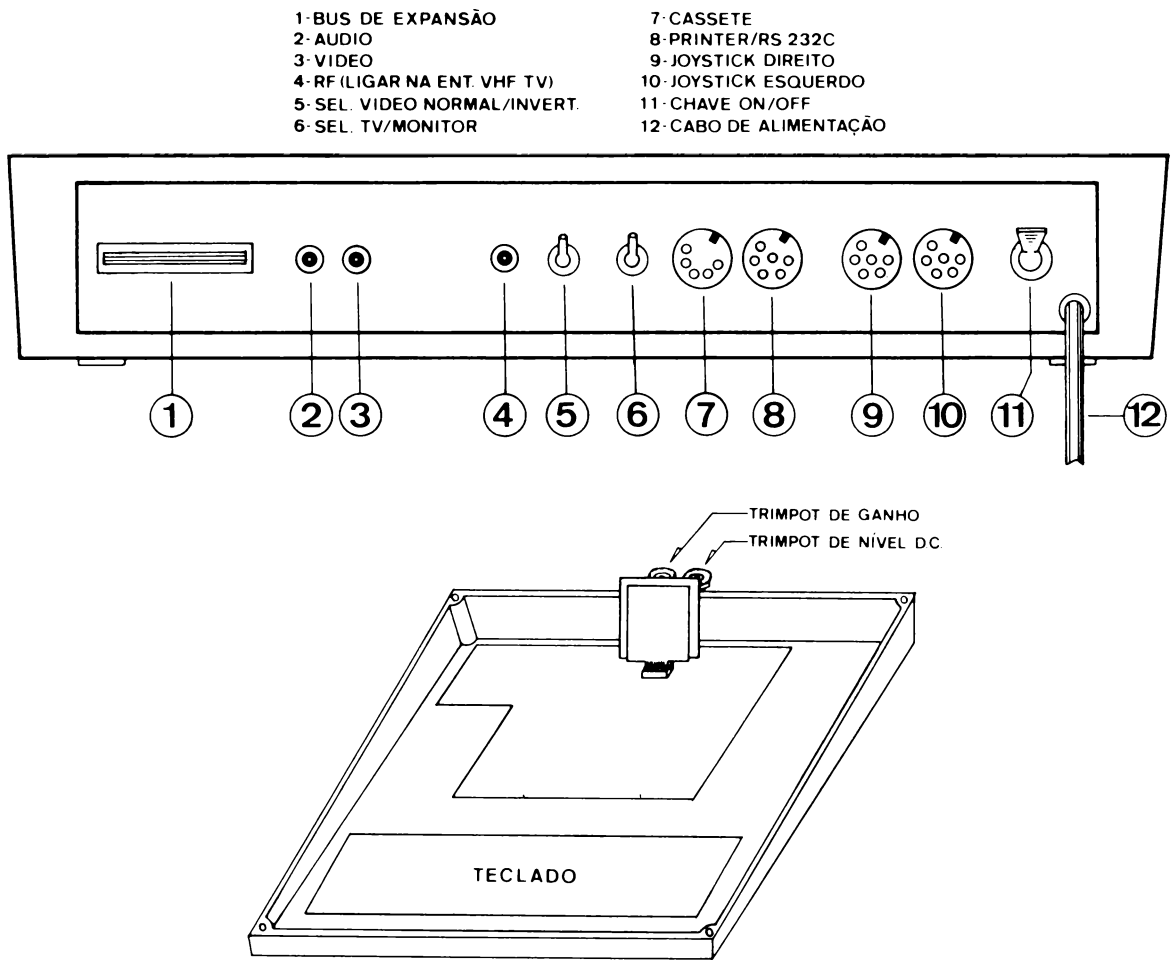
SINAL DISPONÍVEL: O sinal disponível na saída do vídeo composto é ajustado de fábrica para 3,5 V.pp sobreposto a um nível D.C. de 4,5 V.cc, porém estes dois níveis podem ser ajustados independentemente para se adaptar ao seu monitor ou T.V. modificada.

Para fazer isto é necessário abrir o computador. Localize os dois trimpots (veja diagrama).

Ajuste o trimpot de nível D.C. para que a tela fique estável.

Ajuste o trimpot de ganho para o maior ganho que não apresente distorções e que resulte boa definição e contraste.

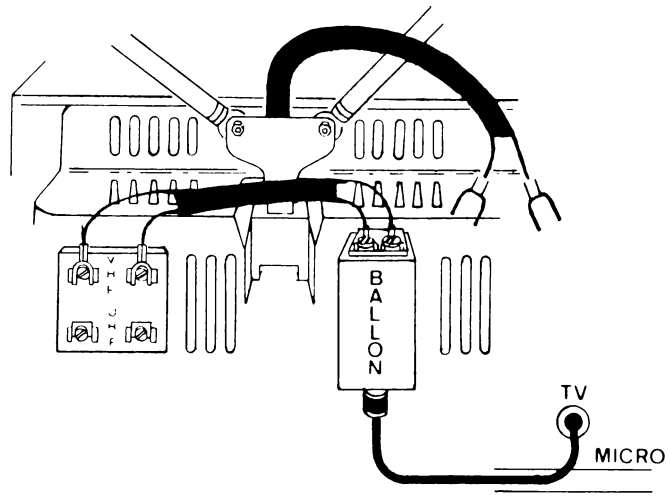
CONEXÃO: Conecte o cabo (fornecido) ao conector RCA marcado "VÍDEO".



2.2.2. T.V. Normal

Para se ligar o computador a uma T.V. Normal é necessário ter-se um modulador de RF interno (opcional) para se ligar diretamente a antena (entrada de VHF) da T.V. Como desvantagem a qualidade do vídeo não é tão boa como em vídeo composto.

No caso de se utilizar a T.V. normal, ligue o cabo com um ballon ao conector RCA marcado T.V. e outra extremidade (cabo paralelo) à entrada de VHF da T.V., deixando desconectado o cabo da antena.



2.2.3. **Monitor de vídeo interno** (opcional)

No caso de se ter um monitor interno de vídeo (fornecido opcionalmente pela Varix) o microcomputador Varix VC50 está pronto para uso.

O monitor fornecido é de 5" phosforo verde e de alta resolução.

Neste modo de operação, pode-se entretanto, ligar um monitor ou T.V. colorido externamente (já que o monitor interno é monocromático), para programas que se utilizam de cores.

Neste caso, o display é apresentado nos dois monitores ao mesmo tempo.

Ajuste o "brilho" e "contraste" nos knobs correspondentes para se ter uma boa definição.

No caso de microcomputadores fornecidos com monitor interno, é certo que também haverá amplificador de audio e alto falante interno. Ajuste o "volume" no knob marcado "volume".

2.2.4. **Áudio**

O microcomputador VARIX VC50 permite usar 3 fontes

de áudio a saber: interna (comandos SOUND e PLAY), do gravador cassete e da ROM PACK ou "BUS de EXPANSÃO".

Para ouvir estes sons, é necessário amplificá-los e enviá-los a um alto - falante. Pode-se usar 3 tipos de amplificadores a saber:

A) AMPLIFICADOR DA PRÓPRIA T.V. Neste caso, se estiver sendo usado um modulador de RF (opcional) o som já estará presente na T.V. Basta ajustar o volume. Se a T.V. for modificada para entrada de "vídeo composto", também será necessário uma entrada de "AUDIO" (facilmente adaptada). Ligue com o cabo coaxial no plug RCA marcado "AUDIO".

B) AMPLIFICADOR EXTERNO. Ligue à entrada "AUX" ou "INPUT" do mesmo plug RCA marcado "AUDIO".

C) AMPLIFICADOR INTERNO. (Opcional). Opcionalmente o microcomputador VARIX VC50 é fornecido com amplificador e alto falante internos. Neste caso não é necessária nenhuma conexão. Basta ajustar o "volume" no knob correspondente do microcomputador. Neste caso as outras duas opções continuam disponíveis ou seja, o sinal de áudio continua disponível tanto no conector "AUDIO" como no caso de se usar um modulador de R.F.

2.3. Conexão do gravador cassete

Você poderá ligar um gravador cassete ao VARIX VC-50, caso pretenda gravar programas ou dados permanentemen-

te a uma taxa de aproximadamente 11.000 caracteres por minuto (1500 BAUDS).

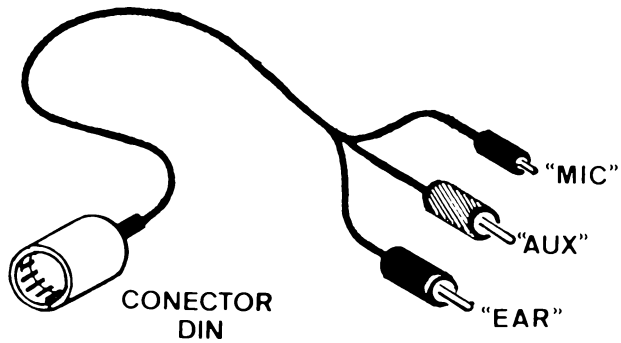
1 - Você precisará de um cabo especial, opcional, que possue em uma das extremidades um conector Din e na outra extremidade 3 plugs para o gravador a ser utilizado. No revendedor em que você adquiriu seu VC-50, será fornecido o cabo para o gravador que você pretenda usar (aconselhamos Philips, National ou qualquer outro com as extremidades e saídas "EAR" ou "FONE", "AUX" e chave de MIC).

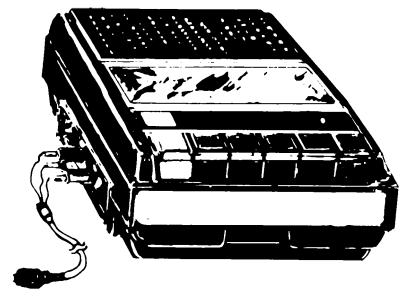
2 - O plug preto grande, deve ser ligado ao JACK "EAR" do gravador cassete.

3 - O plug vermelho grande, deve ser ligado ao JACK "AUX" do gravador cassete.

4 - O plug preto pequeno, deve ser ligado ao JACK pequeno do "MIC" (que é o controle remoto do motor pelo microfone) para permitir que o micro comande o motor do gravador.

VER DIAGRAMA





2.4. Conexão de joysticks

Os joysticks são utilizados para videogames , para comandar o cursor na tela em programas específicos ou para geração de desenhos em programas gráficos.

1 - Podem ser utilizados joysticks tipo ON-OFF (comum) ou tipo linear, com potenciômetros internos. Alguns programas gráficos exigem joysticks lineares. A maioria dos programas, inclusive videogames podem ser operados com joysticks normais ou lineares.

2 - Ligue um joystick no JACK "JOYSTICK ESQUERDO" e o outro opcionalmente no JACK "JOYSTICK DIREITO".

Os programas ou videogames que usam somente um joystick, podem necessitar o esquerdo ou o direito. Verifique no manual do Videogame e mude o joystick para outro JACK se necessário.

2.5. Conexão da ROM PACK ou PROGRAM PACK (CARTUCHOS)

ROM PACK ou PROGRAM PACK são cartuchos contendo uma

ou mais memórias EPROM, com programas utilitários, jogos ou programas aplicativos. Ex.: Varixcalc, Editor assembler, ZAKSUND, PACC BUG, etc. (existem dezenas de títulos).

A ROM PACK deve ser inserida na gaveta apropriada, marcada "ROM PACK".

IMPORTANTE: Sempre desligue o computador quando for inserir ou retirar a ROM PACK.

Insira a ROM PACK com o rótulo para cima e verifique se está bem conectada.

Para usar a ROM PACK leia "UTILIZANDO A ROM PACK".

2.6. CONEXÃO DE DISK DRIVER

Deve ser feita somente por pessoal especializado, de preferência na fábrica, pois são instalados internamente (até 2) e externamente (mais 2).

2.7. CONEXÃO DA IMPRESSORA

Esta deve ser conectada ao plug "RS232C/PRINTER" na trazeira do micro.

Veja diagrama e sinais na parte de especificações.

Pode ser utilizada qualquer impressora com entrada serial de 300 a 9600 BAUDS.

Quando o micro é ligado, fica configurado com 600 BAUDS, porém, você pode alterar a configuração no início do programa utilizando-se de POKES conforme tabela a seguir.

Por exemplo: Para alterar para 1200 BAUDS digite POKE 150,41 (ENTER) .

VARIÁVEIS DA IMPRESSORA

VARIÁVEL	Endereço Hexadecimal	Endereço Decimal	Valor(HEX) Inicial	Valor(DEC) inicial
Baud RATE				
MSB	0095	149	00	0
LSB	0096	150	57	87
LINE Delay				
MSB	0097	151	00	0
LSB	0098	152	01	1
Tamanho Campo vírgula	0099	153	10	16
Último Campo vírgula	0099A	154	70	112
Largura Impressão	009B	155	84	132
Posição	009C	156	00	00

Como se pode observar os valores iniciais são:

- Comprimento: 600 Bauds
- Largura impressão: 132 colunas

EXEMPLOS DE VALORES ALTERNATIVOS

Baud RATE	Decimal MSB	Decimal LSB	Hexa MSB	Hexa LSB
300 Bauds	0	180	0	BE
600 Bauds	0	87	0	57
1200 Bauds	0	41	0	29
2400 Bauds	0	18	0	12

LINE Delay	Decimal MSB	Decimal LSB	Hexa MSB	Hexa LSB
228 seg.	64	0	40	00
576 seg.	128	0	80	00
1.15 seg.	255	255	FF	FF

Largura	Decimal	Hexadecimal
16 caract/linha	16	10
32 caract/linha	32	20
64 caract/linha	64	40
80 caract/linha	80	50
255 caract/linha	255	FF

2.8. Conexão de modem ou terminal

Pode-se conectar um modem ou um terminal com impressora acoplada ao VC 50, para isso, utilize o conector "RS-232C/PRINTER" e um software adequado. Os sistemas operacionais FLEX ou OS9 já comportam um terminal com impressora. No caso do modem você necessitará de um software de comunicação.

2.9. conexão de outros periféricos

Estes dependendo do caso, podem ser conectados ao BUS de expansão trazeiro, ao RS 232C ou ao conector de ROM PACK.

O BUS de expansão trazeiro, marcado "BUS EXP", contém todos os sinais necessários para implementar sistemas de controle, automação, aquisição de dados, interface centronics, módulo de sintetização de voz, etc. (ver especificação).

3. OPERAÇÃO

3.1 Ligando e desligando

1. Ligue a T.V. e ajuste o volume
2. Selecione o canal 3 da T.V. se estiver usando RF
3. Insira a ROM PACK se for utilizar uma
4. Ligue o microcomputador e ajuste a T.V.
5. Ligue os acessórios (impressora, etc.)
   Deverá aparecer a mensagem de "start up" seguida de OK.

6. Se você deseja usar ROM PACK acione o botão"EXEC" momentaneamente.
7. Se você acionar a chave "RESET" o microcomputador reinicia o Basic sem perder o conteúdo da memória, como programas e dados.
8. Para desligar o conjunto, desligue primeiro o microcomputador e depois os acessórios.
9. Se for religar, espere aproximadamente 15 segundos.

3.2. USANDO O TECLADO

1. Como em uma máquina de escrever, use (SHIFT) para entrar com os caracteres marcados acima das teclas.
2. Maiúsculas: Sempre que você ligar o computador, as teclas estarão, automaticamente em modo maiúsculo, que é usual em programação.

Para passar para o modo maiúsculo/minúsculo, pressione "SHIFT ZERO" ou seja (SHIFT) (0)

Dessa maneira as teclas produzirão caracteres minúsculos normalmente, e maiúsculos se acionadas juntamente com (SHIFT) .

Para voltar ao modo normal, pressionar (SHIFT) (0) novamente.

Obs.: As minúsculas, serão representadas na tela por vídeo invertido.

Na impressora as letras serão minúsculas reais. A saída RS232C, também apresenta minúsculas reais para comunicação com periféricos ou outros computadores.

OPERAÇÕES

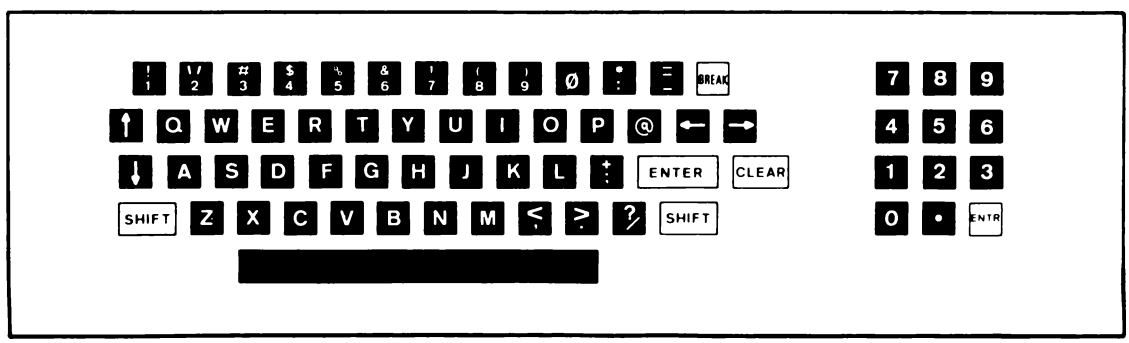
Obs.: Pode-se conectar placas de 80x24 caracteres com minúsculas reais no vídeo.

Com MAXITELA você também terá minúsculas reais (veja Apêndice).

Quando escrever comandos Basic, você deve utilizar o modo maiúsculo.

Teclas especiais

Tecla	Função
⊖	Volta um espaço e apaga o último caracter
SHIFT 0	Muda modo maiúsculo/minúsculo
SHIFT →	Indica um colchete direito]
SHIFT ↓	Indica um colchete esquerdo [
SHIFT @	Pausa para programa. Acionando qualquer tecla o programa continua. Pausa para listagem de programa.
SHIFT ↑	Indica Flexa ↑
SHIFT ←	Apaga toda a linha
CLEAR	Limpa toda a tela e posiciona cursor no início (HOME)
ENTER	Entra a linha de programação e comanda a sequência de um programa ou entrada de dados, durante a execução.
BREAK	Interrompe o programa, quando o mesmo está rodando.



3.3. USANDO O GRAVADOR CASSETE

- Conecte o gravador conforme explicado anteriormente. A taxa de transferência é de aproximadamente 11.000 caracteres por minuto ou 1500 BAUDS.

- Ajuste o volume (volume médio) e tonalidade, se houver, para mais aguda.

- Com a experiência, você determinará qual é o melhor ponto do volume para seu gravador, que permite ler sem erro de leitura.

O microcomputador "checa" a leitura constantemente, e se houver erro ele avisará. Neste caso, altere o volume ou limpe a cabeça de leitura/gravação.

- Em condições normais, não haverá erro de leitura, sendo a interface extremamente confiável. Se houver erros de leitura repetidos, afaste o gravador do microcomputador, pois o gravador pode estar sofrendo influência dos transformadores internos, limpe a cabeça e use somente fitas cassete de ótima qualidade.

3.3.1. **Salvando um programa**

1. Coloque uma fita virgem no gravador (nunca na parte inicial, sem óxido).
2. Pressione PLAY e RECORD
3. Tecle: CSAVE "PROGRAMA"
   Obs.: "PROGRAMA" é o nome do programa e deve ter no máximo 8 caracteres
4. Tecle **ENTER**

Quando o programa estiver salvo aparecerá OK, e o motor desligará automaticamente.

3.3.2. **Lendo (carregando) um programa**

1. Para apagar um programa existente na memória do computador tecle: NEW (ENTER)
2. Pressione PLAY no gravador
3. Tecle: CLOAD "PROGRAMA" ou simplesmente CLOAD para carregar o 1º programa
4. Tecle (ENTER)
5. Aparecerá a letra "S" no canto superior direito, enquanto o computador procura o programa
6. Quando o programa é localizado a letra "F" e o nome do programa aparecerão na tela.

Quando o programa estiver carregado, aparecerá na tela: OK.

3.3.3. **Procurando um programa**

Se você tiver vários programas no mesmo lado da fita, você pode procurá-los, até achar o programa com o comando SKIPF (SKIP UNTIL YOU FIND).

1. Pressione "REWIND"
2. Pressione: "PLAY"
3. Tecle: SKIPF "PROGRAMA"
4. Tecle: (ENTER)
5. Quando o programa for localizado, aparecerá OK
6. Se você esqueceu o nome do programa use um nome improvável como: SKIPF "X"

Os nomes dos programas aparecerão na tela.

Obs.: 1. Alguns gravadores não retornam a fita "REWIND" sem o comando do motor. Neste caso o comando (ENTER) deve ser dado antes de pressionar REWIND e PLAY.

2. Se for necessário, reutilizar uma fita deve-se apagá-la inteiramente com o gravador em RECORD sem sinal ou com um BULK ERASER (apagador magnético). Isto evita gravações entre um programa e outro ocasionando erros.
3. Caso a leitura se inicie no meio de um programa, ocorrerá erro.
4. Se a fita estiver toda virgem, o computador tentará lê-la sem parar, até o final da mesma. Você terá que acionar "RESET" para interromper o processo.
5. Para uma explanação das mensagens de erro, consulte o apêndice.

4. USANDO A T.V. OU MONITOR

1. Ajuste o brilho e contraste para uma boa imagem.
2. Selecione vídeo normal ou invertido na chave traseira marcada "SEL VIDEO" e "NORMAL/INVER" conforme sua preferência. O modo video invertido ficará com letras verdes sobre fundo negro em uma T.V. colorida, sendo o modo mais confortável para a vista (em modo gráfico não altera a tela).
3. Conforme o modulador de RF utilizado, poderá ser necessário selecionar o modo de saída do vídeo na chave marcada "T.V./MONIT", para uso de T.V. normal ou modificada (monitor) respectivamente. Se o computador não tiver modulador de RF, também não possuirá a chave "T.V./MONIT".

Se estiver usando modulador de R.F., ligue a T.V. no canal 3 (VHF) e ajuste na sintonia fina o melhor ponto.

4. Ajuste o volume da T.V. para médio (se estiver utilizando modulador de RF ou T.V. com entrada de áudio).
   Tecle: SOUND 10,100 (ENTER) para testar

   Obs.: Se o microcomputador possuir alto falante interno ajuste o volume no Varix VC 50.

5. No modo normal de operação você terá 32x16 caracteres na tela com minúsculas representadas por vídeo invertido. Como opcional você poderá ter o utilitário maxitela em cassete ou disco, que ficará transparente para os seus programas em Basic. Veja detalhes no apêndice.
   Para cassete:
   Tecle: CLOADM "MAXITELA" (ENTER) EXEC (ENTER)
   Para diskette:
   Tecle: LOADM "MAXITELA" (ENTER) EXEC (ENTER)

   Você terá um vídeo de 51x24 caracteres com minúsculas reais e teclas com repetição automática. Neste modo tire a cor da T.V. ou monitor. Você pode mixar gráficos e textos facilmente neste modo de vídeo (Veja apêndice).
6. Existe a possibilidade de se usar uma placa de 80x24 caracteres (VIDEO PACK) (veja Apêndice).

5. USANDO A "ROM PACK"

Para utilizar o cartucho, estando este inserido e o micro ligado, acione momentaneamente a chave "EXEC" do VC 50 ou tecle EXEC 49152 (ENTER) . Após alguns instantes o pro-

grama deve rodar. Para voltar ao Basic, acione a chave RESET. Cada cartucho possui seu manual de instrução próprio.

6. ALTERANDO MODO DE OPERAÇÃO

Existem centenas de maneiras de se alterar o modo de operação do microcomputador, utilizando-se do comando POKE ou de programas em ASSEMBLER.

Os programas em assembler, para esta finalidade, são programas utilitários, que residem em áreas protegidas da memória e rodam juntamente com os programas em Basic. Exemplos: Alteração do video (Display) para 21 ou 24 linhas com 51, 64, 80 ou 85 caracteres, inclusive minúsculas, redefinição do teclado com teclas de controle e um toque para o comando completo, como "G" para GOSUB, "L" para LIST, etc.

O comando POKE permite alterar posições da memória RAM utilizadas pelo BASIC e desse modo alterar o modo de funcionamento.

O comando PEEK é utilizado para "ler" posições de memória a fim de se obter informações úteis ou para se usar em conjunto com "POKE".

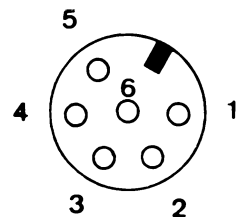
7. MAPA DA MEMÓRIA

Veja apêndice.

8. ESPECIFICAÇÕES

ALIMENTAÇÃO: 110-130 VAC ou 220-230 VAC
MICROPROCESSADOR: 6809 E - 16 BITS internos
CLOCK RATE: 0,89 e 1,78 MHZ

INTERFACE SERIAL RS232C

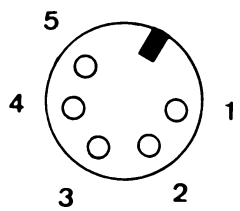


Vista por tráz do computador

Pino 1 - CD-Carrier Detect
(Status input line)
Pino 2 - RD-Receive Data
Pino 3 - GROUND-Terra de sinal
Pino 4 - TD-Transmit Data out
Pinos 5 e 6 - Não conectados

PRINTER INTERFACE (mesmo conector RS 232C)
Pino 1 - N.C.-Não conectado
Pino 2 - BUSY-Sinal enviado pela impressora
(ocupado) BUFFER cheio.
Pino 3 - GROUND - Terra de sinal
Pino 4 - TD - Transmit Data
Pinos 5 e 6 - Não conectados

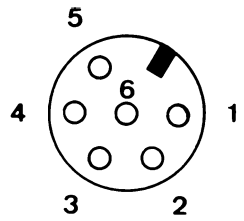
CASSETE INTERFACE
Nível sugerido de entrada - 1 a 5 volts pico a pico
sobre 220 ohms
Nível de saída típico - 800 mV sobre 1K ohm
Chave do motor - 0,5 ampere



Vista por tráz do computador

1 - Controle remoto do motor
2 - GROUND (terra)
3 - Controle remoto do motor
4 - Entrada de sinal (EAR)
5 - Saída de sinal (AUX ou MIC)

JOYSTICK



vista por traz do computador

1 - Entrada do comparador (direito/esquerdo)
2 - Entrada do comparador (acima/abaixo)
3 - GROUND - Terra
4 - Botão de tiro ou ação
5 - VCC + 5 VDC
6 - Não conectado

BUS EXPANÇÃO

PINO	NOME	DESCRIÇÃO
1	-12V	-12V (100 MA)
2	+12V	+12V (300 MA)
3	HALT*	Entrada de Halt (interrupção) para CPU
4	NMI*	Entrada de interrupção não mascarável
5	RESET*	RESET principal do sistema
6	E	Clock principal da CPU
7	Q	Clock defasado
8	CART*	Entrada de interrupção para detecção de ROM PACK
9	+5V	+5V (300 MA)
10	D0	Linha de dados BIT 0
11	D1	Linha de dados BIT 1
12	D2	Linha de dados BIT 2
13	D3	Linha de dados BIT 3
14	D4	Linha de dados BIT 4
15	D5	Linha de dados BIT 5
16	D6	Linha de dados BIT 6

PINO	NOME	DESCRIÇÃO
17	D7	Linha de dados BIT 7
18	R/W*	Leitura/Escrita - (Read/Write)
19	A0	Linha de endereço BIT 0
20	A1	Linha de endereço BIT 1
21	A2	Linha de endereço BIT 2
22	A3	Linha de endereço BIT 3
23	A4	Linha de endereço BIT 4
24	A5	Linha de endereço BIT 5
25	A6	Linha de endereço BIT 6
26	A7	Linha de endereço BIT 7
27	A8	Linha de endereço BIT 8
28	A9	Linha de endereço BIT 9
29	A10	Linha de endereço BIT 10
30	A11	Linha de endereço BIT 11
31	A12	Linha de endereço BIT 12
32	CTS*	Sinal de select para Disco e ROM PACK
33	GND	Terra de sinal
34	GND	Terra de sinal
35	SND	Entrada de som (audio)
36	SCS*	Sinal de seleção para controle (Spare Select Signal)
37	A13	Linha de endereço BIT 13
38	A14	Linha de endereço BIT 14
39	A15	Linha de endereço BIT 15
40	SLENB*	Sinal para ativar uso de memória diretamente.

*significa "ativo em nível baixo (LOW)"

BASIC

INTRODUÇÃO

A linguagem de programação BASIC (Beginners All - purpose Symbolic Instruction Code, ou seja, Código de Instrução Simbólico de Uso Geral para Principiantes), foi desenvolvida por John G. Kemeny e Thomas E. Kurtz no início da década de 60, EUA.

O BASIC, é a linguagem de alto nível de mais fácil aprendizado, podendo ser imediatamente usada a medida que o conhecimento da linguagem vai se aprimorando. Assim você poderá preparar, logo inicie seu estudo nesse manual, pequenos programas, os quais serão, paulatinamente sofisticados. Isto faz com que o BASIC, atualmente, seja a linguagem mais usada para os microcomputadores.

Uma das formas de se aprender uma linguagem de programação, é a de estudar programas prontos. Por esse motivo apresentamos alguns exemplos nos capítulos finais.

O contacto direto com seu equipamento será também, a melhor forma de aperfeiçoar o conhecimento. Então, mãos a obra.

I. PRIMEIROS CONTACTOS

1. LIGANDO

Tão logo você ligue seu VC50, conforme orientação do manual de operações, você deverá observar a seguinte mensagem na tela:

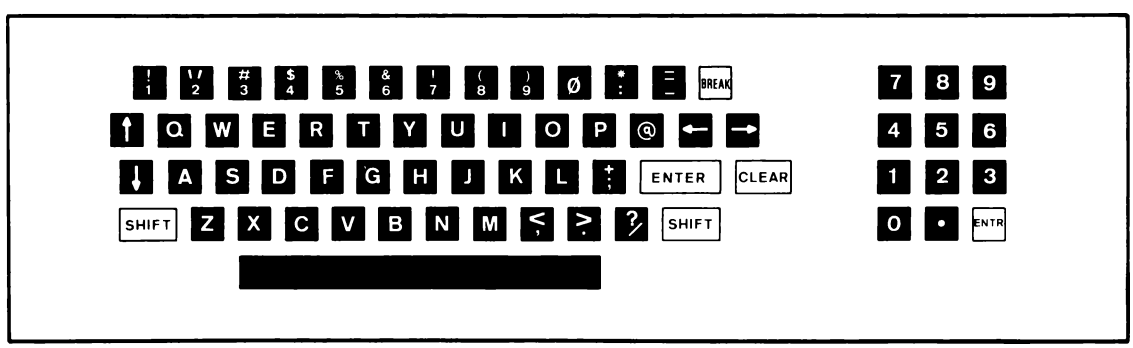
```
VARIX BASIC L.0:EXT.
(C) 1983 VARIX (MR)
DIREITOS DE VARIX SOFTWARE
OK
```

Caso não tenha ocorrido, verifique as conexões na TV novamente.

A mensagem OK significa que seu computador está pronto para receber as ordens de seu mestre, o Homem.

2. TECLADO

O teclado usado é o denominado ASCII de 53 teclas, cujo aspecto é mostrado abaixo:



As teclas numéricas poderão ser obtidas no teclado principal, como também no teclado número que você tem a sua direita. Esse teclado, facilitará a digitação de dados numéricos, pois se assemelha ao de uma máquina de calcular.

Quanto aos caracteres do tipo !,", # ,$, etc., você os obterá digitando (SHIFT) juntamente com a tecla que os contém.

Outras teclas como ⊖ , ⊖ , (CLEAR) , (BREAK) etc, veremos suas funções a medida que precisarmos do seu uso.

Obs.: Por convenção, para facilidade de entendimento programador-digitador, a letra o é representada por ⓞ e o número zero por (0)

Pressionando-se (SHIFT) e (0) (zero) simultaneamente, as letras digitadas após esse comando, aparecerão em vídeo inverso. Para voltar ao normal digite (SHIFT) e (0) novamente. Isto será útil para títulos onde precisamos dar um destaque do resto do texto, ou para fazermos um nome piscar na tela. Este último recurso se obtém escrevendo-se no mesmo local da tela, ora em vídeo normal, ora em vídeo inverso. Essas utilizações você terá condições de fazer tão logo que adquira os conhecimentos fornecidos por esse manual.

Obs.: Quando estivermos no modo minúsculo, digitando (SHIFT) e a letra desejada, esta se tornará maiúscula, como numa máquina de escrever. Os comandos BASIC serão sempre com maiúsculas.

3. CURSOR

Ao ligar seu VC50 você observou a mensagem descrita e a palavra OK, certo? Mas vemos ainda um quadrado, inter-

mitente na tela, significando que nesta posição o computador receberá as instruções a ele fornecidas. Este é o cursor.

II. ESCREVENDO NA TELA

A palavra em BASIC que faz com que o computador exiba algo na tela é PRINT.

Digite:

```
PRINT 2
```

(ENTER)

Tela:

```
2
OK
```

Isto é, o computador exibiu o número 2 solicitado pelo comando PRINT 2.

Digite:

```
PRINT 2+2
```

(ENTER)

Tela:

```
4
OK
```

Desta vez foi mostrado o resultado da operação 2+2.

E agora se quisermos escrever um nome na Tela? Para isso precisamos entender o conceito de String.

String é um conjunto de letras e números que o computador manipulará tal como foram escritos, não tendo significado para o computador. Em outras palavras, string é uma cadeia de caracteres que o computador não processa, ou se-

ja, ele exibe ou guarda na memória, da forma com que lhe foi indicada. Todo string é colocado entre aspas ("string").

Digite:

```
PRINT "2+2"
```

(ENTER)

Tela:

```
2+2
OK
```

Isso mesmo, ele não processou a soma, e sim mostrou nos exatamente tudo o que está dentro das aspas.

Digite:

```
PRINT "VARIX VC50"
```

(ENTER)

Tela:

```
VARIX VC50
OK
```

Que tal mostrarmos o resultado de uma operação, acompanhada pela operação que será calculada?

Digite:

```
PRINT "2+2="2+2
```

(ENTER)

Tela:

```
2+2=4
```

Você deve ter observado que em todo final de comando, devemos digitar (ENTER) . Isto diz ao computador que os dados de um determinado comando já acabou, e que pode ser processado. Nos próximos exemplos não escreveremos mais a instrução (ENTER) , mas você deverá usá-la.

Sempre que existir um erro de sintaxe ou uma operação ilógica ou impossível de executar, será exibida uma mensagem de êrro.

Tente:

```
PRNT "EINSTEIN"
```

Tela:

```
?SN *ERRO
```

onde SN ERRO significa que houve um erro de sintaxe. O erro ocorreu pois foi escrito PRNT ao invés de PRINT.

Tente:

```
PRINT 5/0
```

Tela:

```
?/0 *ERRO
```

onde /0 ERRO significa que houve um erro de lógica, pois o computador não executa divisão por zero.

Caso você cometa um erro durante a digitação, mas antes de digitar (ENTER) , use a tecla ⊖ para voltar o cursor. Com isso você apaga o caracter errado, podendo assim digitar o caracter correto. Se for mais de um caracter , digite quantas vezes forem necessárias a tecla.

Para facilidade de digitação você poderá ao invés de digitar 5 caracteres para obter a palavra PRINT, digitar apenas o sinal de interrogação (?).

Digite:

```
? "VARIX  - VC50"
```

III. LIMPANDO A TELA

A função responsável para a limpeza de uma tela é CLS.

Digite:

```
CLS(0)
```

Isso mesmo, a tela ficou negra.

Digite:

```
CLS(2)
```

Agora temos a tela branca. O que indica a cor da

tela, é o número indicado após o comando CLS. A seguir mostraremos todas as operações disponíveis no VC50.

CLS normal	CLS(4) - vermelho
CLS(0) - preto	CLS(5) - bege
CLS(1) - verde	CLS(6) - azul piscina
CLS(2) - amarelo	CLS(7) - roxo
CLS(3) - azul	CLS(8) - laranja

Se for usado um nº fora do intervalo de 0 a 8, será exibida na tela a mensagem VARIXSOFT.

IV. SOM

A primeira operação a fazer, é aumentar o volume da T.V..

A palavra em BASIC responsável para obtenção de som em seu VC50 é SOUND.

SOUND TOM, TEMPO

A tonalidade pode variar desde o número 1 indicando o tom mais grave até 255 indicando o mais agudo.

Quanto ao tempo, temos por exemplo:

100 -> 6 segundos
10 -> 6/10 segundos
1 -> 6/100 segundos.

Por exemplo:

SOUND 1,100

Temos assim, um som grave durante 6 segundos.

Outros exemplos:

SOUND 255,100 -> som agudo durante, 6 segundos.

SOUND 128,50 -> som intermediário durante, 3 segundos.

Exemplos:

1. SOUND 1,19
2. SOUND 10,10
3. SOUND 100,15
4. SOUND 200,120
5. SOUND 255,13

Se for usado um número para a tonalidade fora do intervalo de 1 a 255, será exibida a mensagem de erro ?FC*ERRO.

Obs.: Mais tarde você vai conhecer outros comandos para som mais poderosos.

V. VARIÁVEIS

Variáveis são letras ou combinações de letras e números que em um programa, serão atribuídos valores numéricos ou alfanuméricos.

Os nomes de variáveis em BASIC podem ser formados por um ou dois caracteres. O primeiro obrigatoriamente tem que ser uma letra, e o segundo uma letra ou número.

Exemplo: A, X, A1, A2, AB, XY.

No caso de ser atribuído a variável um valor alfanumérico (string), o nome desta deverá ser acrescido do sím

bolo especial $. Portanto, nomes de variáveis alfanuméricas, podem ser constituídos de dois ou três caracteres.

Exemplo: A$, X$, A1$, A2$, AB$, XY$.

A palavra em BASIC que associa um valor numérico ou alfanumérico a uma variável é o LET. Mas no BASIC do seu VC50 você poderá omitir essa palavra.

Exemplo:

```
LET A=2          ou  A=2
LET A$="VC50"    ou  A$="VC50"
```

Para que você visualize na tela essas duas variáveis, digite:

```
?A
?A$
```

Quando você liga o computador, todas as variáveis numéricas são iguais a zero e as alfanuméricas (strings) estão em branco.

Digite:

```
?X
?X$
```

Você também poderá redefinir um valor string assumido por uma variável, durante o programa. A variável terá como conteúdo o último valor ou string associado a ela.

Exemplo:

```
A=10              A$="VC50"
A=50              A$="COLOR"
?A                ?A$
```

Condições de erro:

1. A="JOSE"
   nesse caso a variável A não foi acompanhada do caracter $.

2. A$=10

   nesse caso como o número 10 está sem aspas, trata-se de um valor numérico, e portanto deve ser armazenado em uma variável numérica.

A mensagem de erro exibida para os dois casos é:

```
?TM *ERRO
```

VI. PROGRAMA

Um programa é uma sequência de instruções que o computador deverá executar uma a uma.

As instruções são colocadas nas linhas do programa, que deverão receber um número. É conveniente que essa numeração seja de dez em dez, deixando assim espaços para a inserção de linhas suplementares, caso haja necessidade.

As linhas podem ser numeradas de 1 a 65529, sempre inteiras.

As instruções serão executadas na ordem do número das linhas e não na ordem de entrada. Assim, as instruções podem ser fornecidas ao computador em qualquer ordem, mas numeradas na ordem correta de execução.

Exemplo:

```
10 A=1
20 B=2
30 PRINT "A+B="A+B
```

Para que o computador execute o programa acima, você deverá digitar o comando RUN, que em BASIC significa rode o programa. E ainda o comando ENTER . Feito isso, temos:

A+B=3

OK

Caso você queira novamente a listagem digite LIST e (ENTER) . Daí, temos:

```
10 A=1
20 B=2
30 PRINT"A+B="A+B
OK
```

Vamos agora inserir uma linha entre a 20 e a 30 para definir uma variável C. Digite:

```
25 C=3
(ENTER)
LIST
(ENTER)
```

Daí, temos:

```
10 A=1
20 B=2
25 C=3
30 PRINT"A+B ="A+B
```

Digite novamente a linha 30:

```
30 PRINT"A+B+C="A+B+C.
```

Pronto, tudo OK, digite LIST para ter uma idéia de como ficou o programa. Agora você pode rodá-lo novamente.

Caso você queira apagar uma linha do programa digite o número da linha a ser apagada e (ENTER)

Neste ponto, talvez você queira limpar a memória do computador para iniciar um novo programa. É conveniente eliminarmos o anterior evitando-se assim que, misturem-se instruções e dados dos dois programas. Para isso devemos digitar:

```
NEW
(ENTER)
```

Para certificar-se que estamos com o computador pronto para outro programa, digite:

LIST

ENTER

Antes de começarmos a digitação de um novo programa, vejamos um conceito que facilitará a sua programação no VC50. Esse equipamento aceita mais de uma instrução por linha de programa, sendo essas separadas por dois pontos (:).

No caso do programa anterior, temos:

```
10 A=1:B=2:C=3:?"A+B+C="A+B+C
```

Ótimo não? sim, por economia de memória e facilidade de digitação.

VII. UM POUCO DE MATEMÁTICA

Mostraremos nesse capítulo, como você poderá executar em seu VC50, uma operação matemática simples.

Mais adiante você ficará familiarizado com operações mais complexas.

1. OPERAÇÕES ELEMENTARES

SOMA(+) SUBTRAÇÃO(-) MULTIPLICAÇÃO(*) DIVISÃO(/)

Para se usar o computador na execução de uma dessas operações, basta usar o sinal correspondente no teclado. A execução das operações é imediata, por comando direto ou por programa. É necessário dar a instrução PRINT quando queremos ver o resultado; mas em certos programas, onde existem

vários cálculos intermediários, somente mandamos executar a operação.

Exemplos:

1. 10 A=100
   20 B=200
   30 C=B-A
   40 PRINT C
2. PRINT 10+10
3. PRINT 2*10
4. PRINT 100/2
5. PRINT 40*6/4+36/4

Observe que os quatro primeiros exemplos mostram as operações executadas uma a uma. Mas, é possível executar de uma só vez um conjunto de operações, como no exemplo 5.

Analisemos melhor o exemplo 5. O resultado da operação vale 69, mas poderiamos ser levados a 24 como resultado.

As operações aritméticas são executadas pelo computador da esquerda para a direita, seguindo a uma hierarquia existente entre as operações. A hierarquia é a seguinte:

1. OPERAÇÕES ENTRE PARÊNTESES
2. FUNÇÕES MATEMÁTICAS (SIN,COS,ABS,SQR,...)
3. EXPONENCIAÇÃO
4. MULTIPLICAÇÃO E DIVISÃO
5. ADIÇÃO E SUBTRAÇÃO.

Obs.: Se houver parênteses dentro de parênteses, as operações serão começadas pelos parênteses interiores, seguindo-se os parênteses mais exteriores, e assim por diante.

Vejamos as regras hierárquicas sobre o exemplo 5.

1. operação - 40*6=240
2. operação - 240/4= 60
3. operação - 36/4= 9
4. operação - 60+9= 69

- **Exercício 1:** coloque parênteses na operação abaixo, de modo que seu resultado seja 28:

```
PRINT   30 - 9 - 8 - 7 - 6
```

FAÇA VOCÊ MESMO O SEU PROGRAMA

Obs.: Os exercícios resolvidos estão no final deste manual.

2. EXPONENCIAÇÃO (↑)

Permite elevar um número ou expressão numérica a determinada potência que por sua vez também poderá ser um número ou uma expressão numérica.

```
EX.: PRINT 2↑3
     8
     PRINT -2↑2
     -4
     PRINT (-2)↑2
     4
```

3. NOTAÇÃO CIENTÍFICA

Quando o computador processa um cálculo cujo resultado é maior que 9 algarismos, esse resultado será mostrado em notação científica, isto é, uma mantissa e um fator multiplicador em base de potência de 10.

```
EX.: PRINT 1234
     1234
     PRINT 123456789
     123456789
     PRINT 1234567890
     1.23456789E+09
```

A letra E indica "vezes 10 elevado a" e o número que se segue é o expoente (o sinal + indica expoente positivo). No caso de números fracionários muito menores que 1, o expoente de 10 será negativo. É importante lembrar que no sistema numérico inglês, a nossa vírgula corresponde a um ponto; e os números fracionários menores que 1, não tem o zero inicial.

EX.: 3,1416 (português) equivale a 3.1416 (inglês)
0,31416 (português) equivale a .31416 (inglês).

Tente agora 100 elevado a centésima potência. Você deve ter conseguido uma mensagem de erro ?0V *ERRO ao invés de um número. Isso ocorreu devido ao número gerado sermaior que 10E+38. Isso mesmo, o número está fora de faixa. A faixa permitida é de 10E-38 a 10E+38.

VIII. MAIS PROGRAMAÇÃO

Veremos neste capítulo algumas instruções em BASIC que farão seus programas mais eficientes.

1. INPUT

Essa instrução faz com que o programa pare e fique aguardando a entrada, pelo teclado, de algum dado. Nesse momento, aparece na tela do vídeo uma interrogação, que indica ao operador que o programa está aguardando a entrada de algum dado. Esse dado pode ser numérico ou alfanumérico. Logo que o operador digite o dado e (ENTER) , o programa o atribui a uma variável.

Exemplo:

```
10 INPUT A
20 X=A+2
30 ?X
```

Agora digitando RUN e (ENTER) , temos:

```
?
```

Muito bem, o programa parou na linha 10 e a interrogação pergunta ao operador qual o valor de A. Após teclado um valor, A passará a ter esse valor durante o programa, a menos que o mudemos.

Tecle o número 10, por exemplo:

```
?10
```

E (ENTER) , o programa prosseguirá:

```
12
OK (A=10,X=A+2=10+2=12)
```

Vamos agora, usar o comando para variáveis alfanuméricas.

Digite o seguinte programa:

```
10 ?"EU SOU O VC50"
20 ?"QUAL E O SEU NOME?"
30 INPUT A$
40 ?"OI,"A$
RUN
(ENTER)
EU SOU O VC50
QUAL E O SEU NOME? (está esperando você digitar
                    seu nome: digite-o e (ENTER)
```

É sempre aconselhável, como visto no programa acima que, antes de pedirmos o valor de uma variável, seja ela numérica ou alfanumérica, imprimirmos alguma instrução sobre a variável que está sendo pedida (linha 20). Caso contrário teríamos que lembrar o que o computador está pedindo (isso torna-se útil quando temos em um programa muitas variáveis).

Exemplo:

```
 5 PRINT "SOMA DE TRES NUMEROS"
10 ?"DIGITE VALOR PARA A"
20 INPUT A
30 ?"DIGITE VALOR PARA B"
40 INPUT B
50 ?"DIGITE VALOR PARA C"
60 INPUT C
70 X=A+B+C
80 ?"X="X
```

Para sua felicidade e facilidade, o BASIC, do seu VC50, aceita na própria instrução INPUT, a STRING que indica o que o operador deverá digitar. O programa acima ficaria:

```
 5 PRINT "SOMA DE TRES NUMEROS"
10 INPUT "DIGITE VALOR PARA A";A
20 INPUT "DIGITE VALOR PARA B";B
30 INPUT "DIGITE VALOR PARA C";C
40 X=A+B+C
50 ?"X="X
```

Outra variante do comando INPUT, que poderá facilitar mais ainda seus programas, é a de aceitar a entrada de diversos dados em um único comando INPUT.

Exemplo:

```
 5 PRINT "SOMA DE TRES NUMEROS"
10 INPUT"DIGITE A,B,C";A,B,C
20 X=A+B+C
30 ?"X="X
```

Obs.: Você deve digitar os três números separados por vírgula e após (ENTER)

- **Exercício 2:** Faça um programa para somar dois números subsequentes, digitando somente o primeiro.

FAÇA VOCÊ MESMO O SEU PROGRAMA

Obs.: Os exercícios resolvidos estão no final deste manual.

2. READ E DATA

Essas instruções são, também, usadas para atribuir

valores a variáveis. Ao invés de darmos entrada nas variáveis uma a uma, entramos com uma tabela de dados (DATA), e o programa vai, progressivamente, dando os valores dessa tabela as variáveis através do comando READ.

Exemplo:

```
10 DATA 50
20 READ A
30 ?A
```

A instrução READ (linha 10) leu o dado para a variável A na linha 30, associando o valor 50 a variável A.

Colocando-se mais dados para serem lidos, observe que os dados serão lidos seqüencialmente.

```
10 DATA 50,10,-18,0,5
20 READ A,B,C,D,E
30 ?A;B;C;D;E
```

Ou ainda:

```
10 DATA 50,10,-18,0,5
20 FOR X= 1 TO 5
30 READ A
40 ? A
50 NEXT X
```

Obs.: As instruções FOR e NEXT serão vistas mais adiante; elas têm como função, fazer com que o computador execute as instruções entre o FOR e o NEXT um número especificado de vezes.

As instruções READ e DATA podem, também, ser associadas a variáveis alfanuméricas.

Exemplo:

```
10 DATA VC 50
20 READ A$
30 ?A$
```

Uma característica interessante do comando DATA é que ele pode ser colocado em qualquer lugar do programa.

Exemplos:

1.
```
10 DATA SOL,LUA,TERRA
20 FOR X = 1 TO 3
30 READ A$
40 ? A$
50 NEXT X
```

2.
```
10 FOR X = 1 TO 3
20 READ A$
30 ? A$
40 NEXT X
50 DATA SOL,LUA,TERRA
```

3. RESTORE

Esta instrução permite restabelecer a leitura dos mesmos dados, dentro do programa, desde o início.

Exemplo:

```
10 READ A$
20 ?A$
30 READ B$
40 ?B$
50 DATA SISTEMA SOLAR
```

Você observou que foi mostrado na tela "Sistema Solar" e logo após a mensagem de erro **?OD *ERRO NA 30**. Isso porque a linha 10 diz ao computador para procurar uma linha com o comando DATA, ler o primeiro item da lista ou seja, sistema solar, dar a "sistema solar" o nome A$ e riscar "Sistema Solar" da linha DATA. Consequentemente, quando o programa atinge a linha 30, não encontra mais dados para associar a B$, dando então a mensagem de erro **?OD** que significa "Sem Dados" (OUT OF DATA).

Agora tente rodar o programa após ter inserido a linha:

```
25 RESTORE
```

Agora é como se o computador não riscasse nada. A linha foi restaurada, e o computador pode lê-la novamente.

Para o exemplo do item 2 fica:

```
10 DATA SOL,LUA,TERRA
20 FOR X = 1 TO 3
30 READ A$
40 ? A$
50 NEXT X
60 RESTORE
70 GOTO 20
```

IX. DESVIOS

Até o momento, nos exemplos que foram mostrados, as instruções eram executadas sequencialmente, das linhas de menor para as linhas de maior número de comando.

1. GOTO

Esta instrução permite encaminhar para uma determinada linha, saindo da sequência normal de linhas no sentido da numeração crescente. Deve-se digitar o número da linha para o qual se deseja "saltar".

Exemplo:

```
10 INPUT"EM QUE ANO ESTAMOS";A
20 INPUT"QUAL SUA IDADE";B
30 X=2000-A+B
40 ?"NO ANO 2000, VOCE TERA";X;"ANOS DE IDADE"
50 GOTO 10
```

Dessa forma o programa será executado infinitas vezes. Para sairmos do chamado looping não controlado, digite (BREAK) .

O programa interrompido por (BREAK) , nos mostra na tela do vídeo a seguinte mensagem:

BREAK NA XX

onde XX é o número da linha que seria executada no instante em que acionamos a tecla (BREAK).

Para evitarmos loopings infinitos, devemos utilizar de contadores para controlarmos assim, o número de "giros" que o programa deverá executar, e ainda uma condição temque ser satisfeita, ou seja, definirmos um número limite para o contador. Nesse ponto para-se a execução do programa através do comando <END>. Mas antes, vejamos o comando responsável pela condição a ser satisfeita:

- **Exercício** 3: Faça um programa para mudar as cores da tela da T.V.. Dica: a linha 40 pode ser: 40 CLS(T)

FAÇA VOCÊ MESMO O SEU PROGRAMA:

2. IF...THEN

Esse par de instruções sempre é empregado em conjunto. Se a condição entre o IF e o THEN for verdadeira, será executada a instrução indicada após o THEN. Se for falsa, o programa prossegue para a linha seguinte.

Exemplo:

1. vamos escrever 5 vezes VC 50 na tela.

```
10 ? "VC 50"
20 ? "VC 50"
30 ? "VC 50"
40 ? "VC 50"
50 ? "VC 50"
```

Nota-se que esse método não seria eficiente caso quiséssemos escrever muitas vezes na tela a mesma mensagem.

A forma correta de programarmos será a de atribuir um contador, e controlá-lo até um valor final, que será o número de vezes que desejamos que seja escrito a mensagem na tela. O programa acima fica:

```
10 ? "VC 50"
20 C=C+1
30 IF C=5 THEN END
40 GOTO 10
```

Assim o programa gira 5 vezes, escrevendo a mensagem toda vez que passar pela linha 10. Se quisermos outro número de mensagens exibidas na tela diferente de 5, é só mudarmos o valor a ser comparado na linha 30.

O comando END finaliza o programa, embora o processamento não esteja na última linha do programa.

2.
```
10 INPUT"QUANTO VALE 12x12";A
20 IF A=144 THEN PRINT"OK":END
30 PRINT"ERRADO"
```

Analisando o programa: na linha 10 o computador testa as suas habilidades matemáticas esperando você digitar o resultado da operação; na linha 20 é feita a comparação do valor digitado com o valor correto, e se for verdadeira a comparação, o computador segue para a instrução após o THEN, ou seja, escreve na tela OK e finaliza o programa através do comando END; e no caso de ser falsa, o programa segue para a linha 30 escrevendo ERRADO.

A seguir temos as condições que poderão ser usadas, além da igualdade:

```
= igual a
< menor do que
> maior do que
<= menor ou igual a
>= maior ou igual a
<> diferente de
```

Exemplo:

```
 5 CLS
10 INPUT"ESCOLHA UM NUMERO";A
20 INPUT"ESCOLHA OUTRO";B
30 IF A=B THEN  PRINT"OS NUMEROS SAO IGUAIS"
40 IF A<B THEN PRINT "O NUMERO";A;"E MAIOR QUE";B
50 IF A>B THEN PRINT "O NUMERO";B;"E MAIOR QUE";A
60 INPUT"QUER CONTINUAR";A$
70 IF A$="SIM" THEN GOTO 5
80 PRINT"FIM"
```

Obs.: o comando de desvio GOTO após o THEN não precisa ser colocado, ganhando-se assim memória e tempo. A linha 70 ficará:

```
70 IF A$="SIM"THEN 5
```

A linha 70 serve para que o programa seja executado outra vez sem digitarmos RUN e (ENTER)

A seguir algumas instruções que agilizam os comandos IF...THEN de seu VC50.

3. ELSE

Em inglês significa, em caso contrário. Este comando faz com que você após a condição, tenha duas alternativas, uma para condição verdadeira e outra para falsa.

Exemplo: Substitua a linha 70 do programa anterior por:

```
70 IF A$="SIM" THEN 5 ELSE PRINT"FIM"
```

4. AND, OR E NOT

Essas instruções servem para fazer comparações múltiplas.

AND - mais de uma condição tem que ser satisfeita.

OR - uma ou outra condição tem que ser satisfeita.

NOT - essa instrução equivale a "não é igual a" (NOT A=B equivale a A<>B).

Exemplos:

a.
```
10 INPUT"DIGITE UM NUMERO";A
20 INPUT"OUTRO";B
30 INPUT"MAIS UM";C
40 IF A=B AND B=C THEN PRINT"OS 3 NUMEROS SAO
   IGUAIS"ELSE PRINT" OS NUMEROS SAO DIFERENTES"
```

b.
```
10 INPUT"DIGITE UM NOME";A$
20 INPUT"OUTRO";B$
30 IF A$="JOSE" OR B$="JOSE" THEN END
40 GOTO 10
```

(qualquer uma das variáveis tendo como conteúdo o string "Jose", o programa pára. Caso contrário o programa volta a pedir nomes).

c.
```
10 INPUT"DIGITE UM NUMERO";A
20 IF NOT A=0 THEN PRINT"A E DIFERENTE DE ZERO"
30 PRINT"A E IGUAL A ZERO"
```

5. ORDENANDO ALFABETICAMENTE

Os operadores relacionais para expressões STRING são os mesmos, das numéricas. Ao invés de comparar magnitudes numéricas, estes comparam, dentro das expressões STRING, a sequência ASCII.

Vamos escolher algumas palavras e mandar o programa colocá-las em ordem alfabética.

```
10 INPUT"DIGITE 2 PALAVRAS";A$,B$
20 IF A$<B$ THEN PRINT A$,B$
30 IF A$>B$ THEN PRINT B$,A$
40 IF A$=B$ THEN PRINT A$,B$
50 GOTO 10
```

X. FOR/NEXT

Essas instruções vêm sempre juntas, executando um loop programado.

```
FOR variável de  =  valor  TO  valor
  .   controle      inicial     final
  :
NEXT variável de controle
```

```
10 FOR X=1 TO 10
20 PRINT"X="X
30 NEXT X
40 PRINT"FIM DO LOOPING CONTROLADO"
```

Com essas instruções, você evita fazer um contador do tipo:

```
C=C+1
```

E ainda de compará-lo com um valor final:

```
IF C>10 THEN END
```

Façamos o programa, usado no capítulo IX para escrevermos 5 vezes na tela a mesma mensagem, usando-se os comandos FOR e NEXT.

```
10 FOR C = 1 TO 5
20 ? "VC 50"
30 NEXT C
```

- **Exercício 4:** Faça um programa para executar a tabuada do número 9.

FAÇA VOCÊ MESMO O SEU PROGRAMA

Um recurso extra, é que a variável de controle pode ser incrementada com valores diferentes de 1, ou ainda decrementada. Para isso, usa-se o comando STEP complementando a instrução FOR.

FOR variável de controle = valor inicial **TO** valor final **STEP** incremento ou decremento

Exemplos:

a.
```
10 FOR X=0 TO 10 STEP 2
20 PRINT"X="X
30 NEXT X
```

b.
```
10 FOR X=100 TO 10 STEP-10
20 PRINT "X="X
30 NEXT X
```

c.
```
10 FOR X=5.8 TO 40.85 STEP 4.5
20 PRINT"X="X
30 NEXT X
```

Note que, os valores iniciais, finais, incremento e decremento podem ser valores não inteiros.

Obs.: se a variável de controle for decrementada, o valor inicial deverá ser maior que o final.

Experimente, agora esse programa sonoro:

```
10 FOR A=1 TO 255
20 SOUND A,1
30 NEXT A
40 FOR A=255 TO 1 STEP-1
50 SOUND A,1
60 NEXT A
```

XI. TEMPO

Podemos paralisar temporariamente a execução de um programa, externamente ou internamente no programa.

Externamente, pressione (SHIFT) e (@) simultaneamente. Para continuar, pressione qualquer tecla.

Obs.: a tecla (BREAK) também interrompe o programa; para continuar digite CONT.

Tente com o programa anterior e note o resultado.

Internamente, é possível retardar a apresentação de dados na tela do vídeo, exibir os dados na tela com um certo intervalo de tempo entre eles, ou simular movimentos na tela. Este último recurso, é causado pela apresentação de um dado em sucessivas posições da tela.

Para que tenhamos um tempo definido nestas pausas, temos:

```
FOR A=1 TO 460*S
```

onde S é o tempo em segundos.

Exemplo:

```
10 FOR A=1 TO 920
20 NEXT A
```

Com isso paramos o processamento durante 2 segundos.

O programa a seguir faz com que o computador gere tons musicais dando um espaço de tempo de um segundo entre um tom e outro:

```
10 FOR X=1 TO 255
20 SOUND X,1
30 FOR A=1 TO 460
40 NEXT A
50 NEXT X
```

Você deverá tomar cuidado, quando utilizar mais de um looping. Estes deverão estar um dentro do outro, ou seja, embutidos. Assim a execução começará pelos loops mais internos até os mais externos.

CERTO

```
10 FOR X = 1 TO 10
20 FOR Y = 1 TO 10
30 ? X,Y
40 NEXT Y
50 NEXT X
```

ERRADO

```
10 FOR X = 1 TO 10
20 FOR Y = 1 TO 10
30 ? X,Y
40 NEXT X
50 NEXT Y
```

A vírgula usada no comando da linha 30, será explicada mais adiante.

Observe que o programa certo, quando executado, faz com que para cada valor de X, tenhamos 10 valores de Y. O programa terminará quando X atingir o valor 10.

Exemplo:

```
 10 FOR H=0 TO 23
 20 FOR M=0 TO 59
 30 FOR S=0 TO 59
 40 CLS
 50 PRINT H":"M":"S
 60 SOUND 150,2
 70 FOR T=1 TO 375
 80 NEXT T
 90 NEXT S
100 NEXT M
110 NEXT H
```

Com esse programa você tem um verdadeiro relógio digital em seu VC50.

Obs.:

1. Não é necessário colocar o nome da variável, depois do comando NEXT, exceto quanto há mais de um looping sendo executado no programa.

Exemplo:

```
10 FOR A=10 TO 50 STEP 10
20 PRINT"A="A
30 NEXT
```

2. No caso de mais de um looping, ou seja loopings embutidos, podemos usar um mesmo NEXT para várias variáveis O programa do relógio ficará:

```
10 FOR H=0 TO 23
 :
80 NEXT T,S,M,H
```

- **Exercício 5:** Faça um programa para o computador mostrar uma de suas 9 cores a cada segundo.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XII. NÚMEROS SURPRESA

Usando-se a função RND, o seu VC50, dará um número aleatório, entre vários. RND(2) pode dar 1 ou 2, como resultado. RND(6) pode dar um número entre 1 e 6. Logo:

```
RND(X)
```

onde o número gerado deverá estar compreendido entre o intervalo 0 e X, ou seja, 1 <= No. <=X.

Exemplo:

```
10 FOR X=1 TO 5
20 PRINT RND(100)
30 NEXT
```

O programa acima fará com que o computador gere 5 números entre 1 e 100, aleatoriamente. Aproveite e jogue na LOTO.

Outro exemplo:

```
10 T=RND(255)
20 C=RND(8)
30 SOUND T,1
40 CLS(C)
50 GOTO 10
```

Obs.: RND é pseudo randomico. Você pode obter um número verdadeiramente randomico utilizando-se do TIMER interno RND (-TIMER) que gera um número aleatório menor que 1.

Exemplo: Para conseguir um número aleatório entre 1 e 10:

```
10 X = RND (-TIMER)
20 B = INT (10*X)+1
30 PRINT B : GOTO 10
```

- Exercício 6 - Faça um programa que simule um jogo de "Roleta Russa".

Se você achar difícil, não tem problema, vá até o final do manual e veja o programa resolvido e se divirta.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XIII. TELA

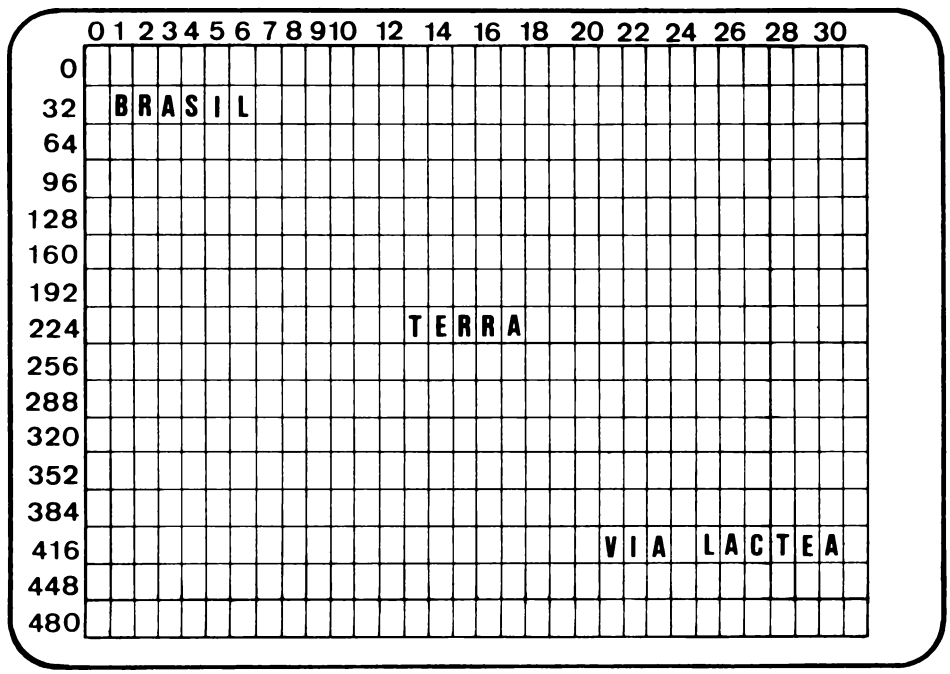
A tela do seu VC50 é formada por 32 colunas e 16 linhas, formando assim uma malha de 512 pontos.

Obs.: Com o utilitário "MAXITELA", oferecido pela VARIX em fita ou disco você terá 51 colunas x 24 linhas, com maiúsculas e minúsculas para usar em qualquer programa. Veja Apêndice.

1. PRINT @

Esta função @ (arroba) define na tela qual a coluna que o primeiro caracter de uma cadeia de caracteresou o primeiro algarismo de um número deverá ser exibido.

Quando for utilizada a função @ , usaremos o gabarito de espacejamento de tela, a seguir:



@ x+y, onde x é o número da linha em que se quer escrever e y é a coluna.

Exemplo:

1. Vamos escrever VC50 bem no centro da tela:

```
@=(224+13)
```

com isso definimos a posição que a letra V irá ocupar, sendo que as demais ocuparão as posições seguintes.

```
PRINT @237, "VC50"
```

Para escrevermos da forma com que foi mostrado no gabarito acima, tem-se o seguinte programa;

```
10 CLS
20 PRINT @33,  "BRASIL"      →  (32+1)
30 PRINT @237, "TERRA"       →  (224+13)
40 PRINT @437, "VIA LACTEA"  →  (416+21)
```

2. SET E RESET

2.1. SET

Essa instrução produz o desenho de um pequeno retângulo para representar um ponto de coordenadas x e y, na tela do vídeo. As coordenadas podem ser definidas por variáveis, constantes ou expressões.

O eixo X é numerado de 0 a 63 da esquerda para a direita do vídeo, visto de frente.

O eixo Y é numerado de 0 a 31 de cima para baixo.

O argumento da função SET deverá ser colocado entre parênteses da seguinte forma:

```
SET(X,Y,C)
```

onde: X -> coordenada do eixo X (0 <= X <= 63)
Y -> coordenada do eixo Y (0 <= Y <= 31)
C -> cor do pequeno retângulo (1 <= C <= 8)

Exemplos:

1.
```
10 CLS(0)
20 SET(0,0,3)
30 SET (63,0,3)
40 SET (0,31,3)
50 SET(63,31,3)
60 GOTO 60
```

com isso colocamos um ponto igual nosquatro cantos da tela.

```
2. 10 CLS(5)
   20 FOR X=2 TO 61
   30 SET(X,2,7)
   40 SET(X,29,7)
   50 NEXT X
   60 FOR Y=2 TO 29
   70 SET(2,Y,7)
   80 SET(61,Y,7)
   90 NEXT Y
  100 GOTO 100
```

Rodando-se o programa tem-se uma moldura na tela. O comando FOR/NEXT foi responsável pela variação de todos os pontos horizontais e verticais. Observe que, quando X variou, Y ficou constante; e quando Y variou X ficou constante.

2.2. RESET

Esta instrução apaga o ponto na coordenada (X,Y), colocado pela instrução SET.

```
RESET (X,Y)
```

Associando-se as instruções SET e RESET, podemos simular movimentos na tela:

```
10 CLS(0)
20 FOR X=0 TO 63
30 SET(X,15,2)
40 RESET (X,15)
50 NEXT X
60 FOR X=63 TO 0 STEP-1
70 SET (X,15,2)
80 RESET (X,15)
90 NEXT X
100 GOTO 10
```

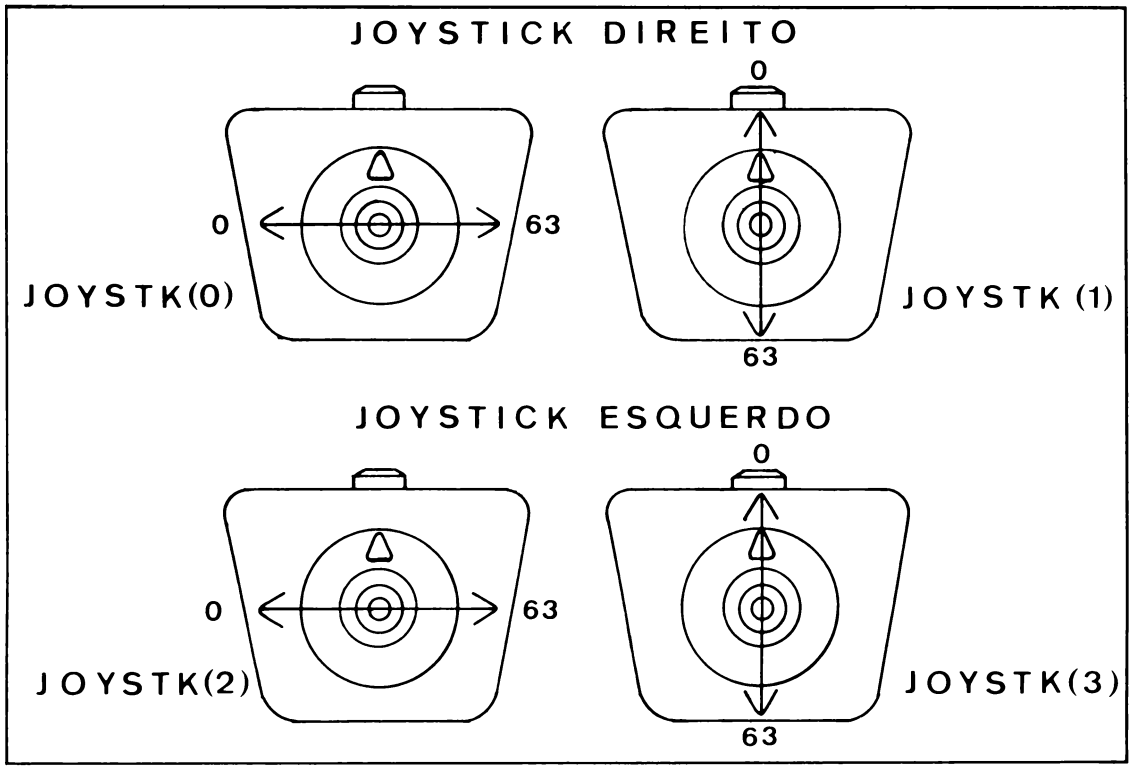
Dessa forma, logo após ser plotado o ponto numa posição da tela ele será apagado, e a posição seguinte será ocupada. Assim, temos o ponto deslocando-se na tela. A instrução da linha 60 faz com que o ponto volte da direita para a esquerda e na linha 100, temos novamente o início, ou seja, o ponto deslocando-se da esquerda para a direita. Como você já sabe, para interrompermos esse programa, digitamos a tecla (BREAK).

- **Exercício 7** - Use as instruções SET e RESET para desenhar qualquer coisa na tela. Nossa solução se encontra no final do manual.

FAÇA VOCÊ MESMO O SEU PROGRAMA

3. JOYSTICK

Joystick, é um dispositivo em seu VC50 que faz com que você tenha as coordenadas da tela, ou seja:



Digite:

```
10 CLS
20 print @0,JOYSTK(0);     J.direita (COORDENADA HORIZONTAL)
30 print @5,JOYSTK(1);               (COORDENADA VERTICAL)
40 print @10,JOYSTK(2);    J.esquerda(COORDENADA HORIZONTAL)
50 print @15,JOYSTK(3);              (COORDENADA VERTICAL)
60 GOTO 20
```

Esse pequeno programa nos dá uma idéia de como esses joysticks simulam as coordenadas da tela. Rodando-se o programa nota-se 4 números na tela. Eles fornecem ao computador as coordenadas horizontais e verticais do "manche" dos dois joysticks.

- segurando o joystick direito, coloque o manche no centro; agora mova-o da esquerda para direita. Observe que o primeiro número da tela mudará de 0 a 63.

- idem para o joystick esquerdo; a variação será do terceiro número na tela.

- agora mova-os de cima para baixo; movendo-se o da direita, você fará o segundo número mudar de 0 a 63; e o da esquerda, fará com que o quarto número mude de 0 a 63.

Para usarmos os botões dos joysticks, use a instrução PEEK(65280). PEEK diz ao computador para ler o conteúdo de certa posição da memória, para ver que número ele contém.

Digite:

```
10 X=PEEK(65280)
20 ? X
30 GOTO 10
```

Rodando-se o programa tem-se o número 255 exibido na tela continuamente. Aperte o botão do joystick e observe que o número mudou para 254. Dessa forma com o auxílio da instrução IF/THEN podemos simular situações de disparo, por exemplo. Quando X for diferente de 255, alguma coisa deverá acontecer, como por exemplo, a partida de um motor controlado por computador, ou um tiro de um canhão em um jogo.

O joystick é bastante interessante quando da utilização do computador para jogos. Isso mesmo, um video-game que você poderá programar o jogo de seu interesse e criatividade. Os jogos, poderão ser jogados por duas pessoas ao mesmo tempo, isto porque, há a possibilidade da conexão de um par de joysticks. A denominação joystick da direita e da esquerda, é dada pela ordem de conexão na parte traseira do VC50.

Exemplos onde você poderá, efetivamente, usar seus joysticks, estão no apêndice referente a programas para rodar no VC50.

4. PRINT TAB

Essa é uma função de tabulação que determina a posição do cursor, para impressão de um número ou STRING (cadeia de caracteres), seja na tela do vídeo, seja em uma impressora.

TAB(X)

O argumento X representa a posição, em que, o primeiro caracter de um string ou o primeiro algarismo de um número, será exibido ou imprimido.

Exemplos:

1. PRINT TAB(5); "COMPUTADOR"
   (observe o ponto e vírgula entre TAB(5) e o string).
2. PRINT TAB(3);"NOME";TAB(10);"ENDEREÇO";TAB(22); "FONE"
3. PRINT TAB(30);"COMPUTADOR"
   (observe que, como a tela possui 32 colunas, ou seja, da posição 0 até 31, a palavra COMPUTADOR foi truncada , pois a letra C ocupou a posição 30, a letra O a posição 31, e as demais tiveram que ser escritas na linha seguinte, pois não havia mais colunas).

A aplicação desse comando é bem usada na formação de tabelas, onde somente definimos as colunas e os dados vão sendo imprimidos um abaixo do outro:

Exemplo:

```
 5 CLS
10 PRINT TAB(2);"NUMERO";TAB(10);"QUADRADO";
   TAB(20);"CUBO"
20 FOR A=1 T0 5
30 PRINT TAB(5);A;TAB(14);A↑2;TAB(21);A↑3
40 NEXT A
```

(os comandos A↑2 e A↑3 serão explicados mais adiante).

5. O USO DA VÍRGULA E DO PONTO E VÍRGULA EM COMANDOS PRINT

A vírgula divide cada linha da tela em duas zonas. Tente:

```
? "SISTEMA","SOLAR"
```

O ponto e vírgula exibe sem espaço os dados a serem exibidos por PRINT, quando tivermos 2 strings.

Tente:

```
? "SISTEMA";"SOLAR"
```

Quando da utilização do ponto e vírgula entre uma variável e um string, o computador exibirá um espaço em branco entre ambos.

Tente:

```
10 INPUT "DIGITE VALOR P/L";L
20 A=L*L
30 ? "AREA=";A
```

Assim o valor da área calculada foi colocada com um espaço, logo após o sinal de igual.

6. CHR$(A) E ASC(A$)

6.1. CHR$(A)

Essa função efetiva a conversão de um código numérico (A), para o símbolo representativo desse código.

Os sistemas de computação, as letras, sinais, caracteres especiais, tem um código numérico equivalente. A tabela de códigos mais usada é a denominada ASCII (American Standard Code for Information Interchange). Vide apêndice.

Assim, verificamos que a letra A, por exemplo, é representada pelo número decimal 65, B é 66, e assim por diante.

A armazenagem no computador é feita através de códigos, ou seja, quando for armazenado na memória, por exemplo, um STRING, ele está, na realidade, armazenando os códigos numéricos correspondentes às letras, inclusive espaços.

Exemplo:

```
A$="VC50"
```

no computador fica:

```
65 36 61 34 86 67 53 48 34
```

O procedimento contrário também é viável:

```
PRINT CHR$(65)
```

O computador exibe:

```
A
```

Consultando-se o quadro a seguir, podemos observar que os códigos do número 128 a 143 são referentes a blocos gráficos. Esta característica é muito interessante, pois você poderá criar seus próprios desenhos animados na tela, ou utilizar-se desse recurso para execução de diagramas, histogramas, etc.





Na codificação do seu VC50 os blocos gráficos estão definidos da posição 128 a 255.

Exemplo:

```
PRINT CHR$(128)
```

A partir da posição 143 os blocos são repetitivos mu dando somente a cor do caracter.

As cores são obtidas da seguinte forma:

```
CHR$(A+C)
```

onde:

A - código do bloco gráfico

C - cor desejada.

A relação das cores, ou seja, os valores que C pode assumir, são:

16 - amarelo
32 - azul
48 - vermelho
64 - bege
80 - azul piscina
96 - roxo
112 - laranja

Exemplo:

```
PRINT CHR$(129+16)
```

Observe que o computador exibe o caracter correspondente ao código 129, exceto a área que é ocupada pela cor verde, será agora ocupada pela cor amarela (16).

Digite:

```
PRINT CHR$(129+32)
PRINT CHR$(129+48)
PRINT CRH$(129+64)
PRINT CHR$(129+80)
PRINT CHR$(129+96)
PRINT CHR$(129+112)
```

6.2. **ASC(A$)**

Esta função é oposta a CHR$(A). Ela dá o valor do código ASCII do caracter A$.

Exemplo:

```
PRINT ASC("A")
```

O computador exibe:

```
65
```

Se colocarmos dentro dos parênteses da função ASC um conjunto de caracteres, será impresso o código ASCII do primeiro caracter:

Exemplo:

```
PRINT ASC("CODIGO")
```

O computador exibe:

```
67
```

Contornando a situação, digite o programa abaixo:

```
10 DATA C,O,D,I,G,O
20 FOR A=1 TO 6
30 READ X$(A)
40 PRINT ASC(X$(A));
50 NEXT
```

O computador exibe:

```
67 79 68 73 71 79
```

7. POINT (X,Y)

Essa instrução verifica se o ponto está ativado ou não. Se estiver ativado é porque a instrução SET foi utilizada . Normalmente a instrução POINT é escrita dentro da instrução IF/THEN.

Exemplo:

```
 5 CLS(0)
10 SET(32,15,8)
20 IF POINT(32,15)<> 0 THEN PRINT "ON" ELSE
   PRINT "OFF"
```

XIV. ORDENANDO ALFABETICAMENTE

Neste capítulo mostraremos com mais detalhes como o computador efetua a ordenação de STRINGS.

O BASIC compara as expressões STRING, caracter por caracter. Quando um caracter não é coincidente, é analisado aquele que tem o menor valor no código ASCII, ou seja, o caracter de menor prioridade.

Relações STRING:

< menor prioridade que
> maior prioridade que
<> não tem a mesma prioridade que
<= prioridade menor ou igual que
>= prioridade maior ou igual que

Exemplos:

"A" < "B"

O caracter A no código ASCII é o decimal 65 e B é 66. Portanto A<B, ou seja, 65<66.

"MARIA" > "MARCIA".

O caracter I tem seu código igual a 73 enquanto C vale 67. Portanto I (73) é maior que C (67).

"SOL" < "SOLAR".

Os espaços são significativos. O código ASCII para o caracter espaço é 32 e A é 65.

XV. O USO DO GRAVADOR K-7

Quando o VC50, é desligado, perde-se todo o programa e as variáveis que se encontram armazenadas na memória RAM. Uma das formas de preservá-los é gravá-los numa fita cassete, caso você não tenha uma ou mais unidades de disco. Assim sendo, você poderá carregá-los de volta a memória, posteriormente quando necessário.

No revendedor onde você adquiriu seu "VC50", você poderá adquirir também, um cabo para conectá-lo a um gravador K7. Vide Manual de Operação.

Digite, então, um programa no computador.

Por exemplo:

```
10 ?"VOU PENSAR UM NUMERO DE 1 A 100"
20 N=RND(100)
30 INPUT"SEU PALPITE";A
40 IF N>A THEN 70
50 IF N<A THEN 90
60 ?"VOCE ACERTOU":END
70 ?"PALPITE BAIXO, TENTE OUTRA VEZ"
80 GOTO 30
90 ?"PALPITE ALTO, TENTE OUTRA VEZ"
100 GOTO 30
```

1. SALVANDO O PROGRAMA EM FITA

O comando CSAVE "nome do programa" tem como função armazenar os programas na fita cassete.

Para o programa acima, digite:

```
CSAVE "JOGO"
```

Com o gravador conectado apropriadamente, coloque-o para gravar (PLAY+RECORD). Agora digite (ENTER) e aguarde a transferência dos dados da memória do computador para a fita.

Acabando a transferência, o computador "para" o funcionamento do gravador, exibindo na tela a mensagem OK.

2. CARREGANDO O PROGRAMA DA FITA

O comando CLOAD "nome do programa" possibilita o carregamento de um programa armazenado em fita. Coloque o gravador no modo "PLAY", digite CLOAD "nome do programa" e (ENTER).

Nesse momento o computador exibe o caracter **S** na tela. Encontrado o programa solicitado, exibirá o caracter **F** e o nome do programa. Quando for exibido OK, o gravador "para" e o programa está na memória do computador.

Obs.: o comando CLOAD, pode ser usado sem especificar-se o nome do programa. Nesse caso, o carregamento será feito do primeiro programa que estiver na sequência da fita.

Portanto:

CLOAD "nome do programa" ou CLOAD.

3. POSICIONANDO A FITA

A seguir apresentaremos uma maneira fácil de posicionar a fita no fim do último programa gravado. Isto é muito útil, quando quisermos gravar mais de um programa em uma fita, e não sabemos onde acaba o último programa.

Rebobine a fita, pressione PLAY no gravador e SKIPF "nome do último programa gravado". Quando for alcançado o final do último programa o gravador pára e aparece na tela a mensagem OK. Nesse ponto é só procedermos como no item 1, para gravarmos o novo programa.

Obs.: caso você não saiba o nome do último programa, use SKIPF "X". Durante a procura, o computador indicará o nome de cada programa que encontrar e quando chegar ao final exibirá a mensagem **I/O *ERRO.** Não se preocupe com esta mensagem.

XVI. SUB-ROTINAS

A instrução GOSUB X, determina que o programa deve pular para a sub-rotina que começa na linha de número X.

Uma sub-rotina tem que ter um fim, caso contrário, seria difícil separar a sub-rotina do resto do programa. A instrução que termina uma sub-rotina, em BASIC, é RETURN. Dessa forma toda a sub-rotina terá um retorno ao ponto onde estava o programa, ou seja, para a linha seguinte aquela em que estava a instrução GOSUB. A mesma sub-rotina pode ser chamada diversas vezes no programa e voltará sempre para a linha seguinte aquela que contém a instrução GOSUB que a chamou.

Exemplo:

```
  5 CLS
 10 INPUT "QUER UMA LISTA DE NUMEROS PARES";A$
 15 IF A$="S" THEN GOSUB 500
 20 INPUT "QUER OS IMPARES";A$
 25 IF A$="S" THEN GOSUB 600
 30 END
500 FOR N=0 TO 10 STEP 2
510 PRINT N
520 NEXT N
530 RETURN
600 FOR N=1 TO 9 STEP 2
610 PRINT N
620 NEXT N
630 RETURN
```

É necessário tomar cuidado, não permitindo que o programa passe pela sub-rotina sem ter sido chamado pelo comando GOSUB. Portanto, é bom que a sub-rotina fique de pre

ferência no final do programa, e no final das instruções do programa que antecedem as sub-rotinas usar o comando END, finalizando o programa.

As sub-rotinas tornam-se úteis em um programa quando, as instruções contidas na sub-rotina terão que ser executadas no mínimo duas vezes e o número delas seja grande. O exemplo acima tem como objetivo, somente mostrar a técnica do uso de sub-rotinas.

- **Exercício 8:** Faça um programa para elevar um número a uma potência qualquer, usando sub-rotinas.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XVII. COMENTÁRIO NA LISTAGEM

A instrução REM permite incluir um comentário na listagem do programa, não produzindo efeito sobre o programa. É muito útil, para o entendimento dos objetivos de um programa, métodos utilizados, unidades, data, programador, etc.

Porém, não se deve abusar da instrução REM, pois os comentários ocupam memória do computador. Use-as apenas em lugares estratégicos do programa, para facilitar o estudo posterior.

Exemplo:

```
 10 REM PROG. CALCULO FINANCEIRO
 20 CLS
 30 REM ENTRADA DE DADOS
 40 PRINT @74, "DIGITE"
 50 INPUT "CAPITAL A SER AMORTIZADO";C
 60 INPUT "TAXA DE JUROS";I
 70 INPUT "NUMERO DE PRESTACOES";N
 80 REM CALCULO DA PRESTACAO
 90 I=I/100:P=C*I*(1+I)↑(N-1)/((1+I)↑N-1)
100 REM MOSTRANDO O RESULTADO
110 PRINT @384, "PRESTACAO IGUAL A"P
120 REM FIM
```

Obs.: O apóstrofe ('), também significa REM

Ex.: 120 'FIM

XVIII. MANIPULANDO STRINGS

1. LEN(A$)

Esta instrução fornece o número de caracteres de um string, incluídos os espaços, sinais de pontuação, etc.

Exemplo:

```
10 INPUT"DIGITE UMA SENTENCA";A$
20 ?"ESTA SENTENCA TEM"LEN(A$)"CARACTERES"
30 INPUT"QUER OUTRA SENTENCA";S$
40 IF S$="S" THEN 10
```

2. STR$(X)

Essa instrução transforma um número X em um STRING.

```
X=12345
A$=STR$(X)
```

Uma vez efetuada a conversão, o STRING só poderá ser manipulado como STRING e não como número.

Exemplo:

```
10 INPUT"DIGITE UM NUMERO";N
20 A$=STR$(N)
30 PRINT A$;"AGORA EU SOU UM STRING"
```

3. VAL(A$)

Essa função faz voltar a dar característica de número a um STRING, obtido por STR$, ou seja, transforma-se um STRING que contenha um número em número.

```
A$="1234"
VAL(A$)
```

4. LEFT$(A$,N)

Essa função conta os caracteres de um STRING, onde

(A$) é o STRING e (N) são os primeiros N caracteres a partir da esquerda.

```
A$="MAQUINA A VAPOR"
LEFT$(A$,3) -> MAQ
```

5. RIGHT$(A$,N)

É uma função simétrica a anterior, com a diferença de contar os caracteres da direita para a esquerda.

```
A$="MAQUINA A VAPOR"
RIGHT(A$,5) -> VAPOR
```

6. MID$(A$,M,N)

Essa função nos dá um sub-string, onde (M) é o início e (N) é o comprimento.

```
A$="MAQUINA A VAPOR"
MID$(A$,9,3) => A V
```

7. EXEMPLOS

7.1.
```
10 A$="COMPUTADOR"
20 FOR N=1 TO LEN(A$)
30 ? LEFT$(A$,N):NEXT
```

7.2.
```
10 A$="COMPUTADOR"
20 FOR N=LEN(A$) TO 1 STEP - 1
30 ? RIGHT$(A$,N)
40 NEXT
```

7.3.
```
10 A$="0194337000"
20 PRINT"O NUMERO" MID$(A$,5,2)" E O PREFIXO
   DO TELEFONE" RIGHTS(A$,6)"E O "LEFT$(A$,4)"
   E O DDD"
```

8. INKEY$

Essa função possibilita a introdução de valores enquanto o computador se encontrar em execução de um programa, sem a utilização da tecla (ENTER) . Essa é a grande vantagem de INKEY$ sobre INPUT, pois quando se quer velocidade de resposta (jogos, por exemplo) é melhor digitarmos uma única tecla do que a tecla e (ENTER) .

Os caracteres digitados para uma função INKEY$ não são visualizados na tela.

Exemplo:

```
 10 REM PIANO ELETRONICO
 20 A$=INKEY$
 30 IF A$=""THEN 20
 40 IF A$="A" THEN T=89
 50 IF A$="S" THEN T=108
 60 IF A$="D" THEN T=125
 70 IF A$="F" THEN T=133
 80 IF A$="G" THEN T=147
 90 IF A$="H" THEN T=159
100 IF A$="J" THEN T=170
110 IF A$="K" THEN T=176
120 IF T=0 THEN 20
130 SOUND T,5
140 T=0
150 GOTO 20
```

Obs.: 1. para tocar, pressione as teclas de A até K.
2. "" significa um string vazio.

- Exercício 9: Faça um programa usando a instrução INKEY$, para que o mesmo seja um vigia de seu teclado, ou seja, faça com que ele exiba uma mensagem na tela quando você pressionar alguma tecla.

Dica: a linha 20 pode ser

```
20 IF A$ <>"" GOTO ...
```

FAÇA VOCÊ MESMO O SEU PROGRAMA

XIX. COMANDOS ÚTEIS

A seguir mostraremos alguns comandos em BASIC, que o ajudarão a controlar seus programas.

1. STOP E CONT

O comando STOP "pára" o processamento mas não finaliza. O comando para finalizar um programa, antes de chegarmos a última linha é END.

Digite:

```
10 X=1
20 X=X+1
30 STOP
40 GOTO 20
```

Rodando-se o programa, o computador exibe:

```
BREAK NA 30
OK
```

O computador parou o processamento na linha 30. Neste ponto podemos digitar uma linha de comando para termos uma idéia do que o programa fez até aqui. Por exemplo:

```
PRINT X
```

O computador exibe o valor 2, ou seja, o valor de X quando o processamento foi interrompido.

O comando CONT fará com que a execução retorne do ponto de interrupção. Caso o programa tenha sido alterado quando da parada por STOP (ou **BREAK**), o comando CONT não poderá ser usado.

Digite:

CONT e **ENTER**

O computador continua a execução.

Os comandos STOP e CONT são úteis para analisar um programa que não estiver funcionando corretamente. Uma vez localizado o erro, você poderá tirar as linhas STOP.

2. MEM

O comando MEM nos fornece a quantidade de memória disponível. Usando-se o comando NEW para limparmos a memória, tecle:

NEW (ENTER)
PRINT MEM (ENTER)

E o computador nos fornecerá toda a memória disponível, ou seja, o valor 24871 bytes ou 22823 com DISK DRIVER.

Quando você estiver digitando um programa longo, você poderá solicitar ao computador quanto de memória ainda há disponível.

Obs.: O computador, quando ligado, reserva 4 páginas de gráficos (PCLEAR 4). Você pode reservar até 8. Se não for usá-las digite:

PCLEAR 3	26407 bytes
PCLEAR 2	27943 bytes
PCLEAR 1	29479 bytes
POKE 25,6 : NEW	31015 bytes

Quando estamos usando DISK DRIVER temos:

PCLEAR 3	24359 bytes
PCLEAR 2	25897 bytes
PCLEAR 1	27431 bytes
POKE 25,14 : NEW	28967 bytes

XX. MAIS DESVIOS

1. ON.....GOTO

Produz desvio da execução do programa para uma das linhas indicadas adiante do GOTO. A instrução é escrita da seguinte forma:

ON X GOTO XX,YY,ZZ,.....

onde X é uma variável que pode assumir valores, 1, 2, 3,... .. e XX,YY,ZZ são números de linha.

Quando X=1, a execução do programa salta para a linha XX, que é a primeira indicada após o GOTO, quando X=2, salta para a linha YY; e assim por diante.

Exemplo:

```
10 CLS
20 ?@134,"(1) OPCAO 1"
30 ?@166,"(2) OPCAO 2"
40 ?@198,"(3) OPCAO 3"
50 ?@354,"QUAL OPCAO VOCE QUER?"
60 INPUT X
65 CLS
70 ON X GOTO 1000,2000,3000
1000 ?@230,"OK OPCAO 1"
1010 END
2000 ?@230,"OK OPCAO 2"
2010 END
3000 ?@230,"OK OPCAO 3"
3010 END
```

O programa acima, caso não tivéssemos usado a instrução ON.....GOTO na linha 70, ficaria:

```
70 IF X=1 THEN 1000
80 IF X=2 THEN 2000
90 IF X=3 THEN 3000
```

Nota-se que a instrução ON.....GOTO simplifica bastante a programação. Imagine um programa onde você tem muitas opções.

2. ON.....GOSUB

O funcionamento dessa instrução é muito semelhante a instrução ON.....GOTO, com a única diferença que, para ON.... GOSUB, há um retorno através da instrução RETURN.

ON.....GOSUB produz um desvio para uma das sub-rotinas, cujo número de linha inicial esteja indicado adiante de GOSUB.

A instrução é escrita da seguinte forma:

```
ON X GOSUB XX,YY,ZZ,.....
```

onde X é uma variável ou expressão que pode assumir os valores inteiros 1,2,3,... . Para X=1, a execução salta para subrotina que inicia pela linha de número XX, e assim por diante.

Ao final de cada sub-rotina tem-se uma instrução RETURN, a qual fará com que a execução volte a linha seguinte aquela onde foi encontrada a instrução ON.....GOSUB.

Exemplo:

```
 5 FOR P=1 TO 600:NEXT P
10 CLS:X=RND(100):Y=RND(100)
20 ?"(1) ADICAO"
30 ?"(2) SUBTRACAO"
40 ?"(3) MULTIPLICACAO"
50 ?"(4) DIVISAO"
60 INPUT"ESCOLHA UMA OPCAO";R
70 CLS
80 ON R GOSUB 1000, 2000, 3000, 4000
90 GOSUB 5
1000 ?"QUANTO E"X"+"Y
1010 INPUT A
1020 IF A=X+Y THEN PRINT "CERTO" ELSE PRINT"ERRADO"
1030 RETURN
2000 ?"QUANTO E"X"-"Y
2010 INPUT A
2020 IF A=X-Y THEN PRINT "CERTO" ELSE PRINT "ERRADO"
2030 RETURN
3000 ?"QUANTO E "X"*"Y
3010 INPUT A
3020 IF A=X*Y THEN PRINT "CERTO" ELSE PRINT "ERRADO"
3030 RETURN
4000 ?"QUANTO E"X"/"Y
4010 INPUT A
4020 IF A=X/Y THEN PRINT "CERTO" ELSE PRINT "ERRADO"
4030 RETURN
```

XXI. FAZENDO O COMPUTADOR FALAR

Quem disse que o computador não pode falar? A sua voz será exatamente como a sua, isto porque ele se utilizará da sua própria voz gravada.

Fazendo isto, você irá ampliar muito a qualidade de seus programas, principalmente para jogos e programas de ensino.

Procedimento:

1. desligue os cabos que ligam seu gravador ao computador.
2. coloque o gravador para gravar (PLAY+RECORD).
3. diga alguma coisa no microfone.
4. desligue o gravador e volte a fita.
5. ligue os cabos.
6. digite o seguinte programa:

```
 5  CLS
10  INPUT"DIGITE  <ENTER> P/OUVIR";A$
20  MOTOR ON
30  AUDIO ON
```

7. rode o programa, ou seja RUN e (ENTER) .
8. coloque o gravador para tocar (PLAY).
9. aumente o volume da T.V.
10. digite (ENTER) .

Caso você queira ouvir música no seu computador proceda da mesma forma, ou seja, siga as instruções a partir do item 5.

A instrução MOTOR ON, liga o seu gravador e a AUDIO ON conecta o som do gravador com o auto-falante da T.V.

Você deve ter percebido que o gravador não pára de "tocar" mesmo depois de finalizado o som. Para contornar essa situação, acrescente ao seu programa:

```
35 CLS
40 A$=INKEY$
50 ?@225,"DIGITE<X>"
60 IF A$ <> "X"THEN 40
70 AUDIO OFF
80 MOTOR OFF
```

Assim, se você digitar a tecla X, o computador executará as linhas 70 e 80 parando a execução do gravador.

XXII. VARIÁVEIS INDEXADAS

Até o momento, vimos as variáveis numéricas e alfanuméricas (STRING) simples, isto é, cada variável assume um valor de cada vez.

Assim uma variável A pode assumir, por exemplo, valores 1, 10, -5, etc., um de cada vez.

Trataremos neste capítulo das variáveis indexadas que poderão assumir um valor diferente para cada valor do índice.

O conjunto de valores das variáveis indexadas é denominado, em inglês, "array", que traduzindo literalmente seria, matriz.

```
A=1          A(1)=1
A=10         A(2)=10
A=-5         A(3)=-5
```

Dessa forma podemos ter todos os valores assumidos pela variável A durante o programa.

As variáveis indexadas podem ser de índice simples, de índice duplo e triplo.

Uma variável de índice simples pode ser indicada da seguinte maneira:

```
A(N)
```

onde A é o nome da variável e N é o seu índice.

Uma variável de índice duplo pode ser indicada da seguinte forma:

```
A(N,M)
```

onde A é o nome da variável e N e M são os índices.

Uma variável de índice triplo fica:

```
A(M,N,L)
```

1. VARIÁVEIS UNIDIMENSIONAIS (ÍNDICE SIMPLES)

A principal maneira de manipular as variáveis indexadas é com o uso de instruções de repetição controlada, FOR e NEXT.

No caso das variáveis indexadas unidimensionais, basta um único par de instruções FOR e NEXT.

Exemplo:

```
FOR N-1 TO 10
INPUT A(N)
NEXT N
```

2. VARIÁVEIS BIDIMENSIONAIS (ÍNDICE DUPLO)

Neste caso precisaremos de dois pares de instrução, um embutido no outro.

Exemplo:

```
FOR N=1 TO 10
FOR M=1 TO 10
INPUT A$(N,M)
NEXT M
NEXT N
```

3. VARIÁVEIS TRIDIMENSIONAIS (ÍNDICE TRIPLO)

Para variáveis de 3 índices tem-se:

```
FOR M=1 TO 3
FOR N=1 TO 2
FOR L=1 TO 2
INPUT A$(M,N,L)
NEXT L,N,M
```

4. INSTRUÇÃO DIM

Quando trabalhamos com variáveis indexadas, a linguagem BASIC estabelece uma quantidade de memórias para armazenar os valores dessas variáveis. Assim sendo, antes de começar o programa, as variáveis indexadas precisam estar dimensionadas.

Na versão BASIC do seu VC50, não é necessário declarar tamanhos das variáveis com menos de 11 elementos.

A instrução DIM quer dizer DIMENSION, ou seja, dimensão. É usada da seguinte maneira:

```
DIM A(N)
```

onde poderemos ter N variáveis A.

```
DIM A(N,M)
```

onde a variável A poderá assumir índices de A(0,0) até A(N,M).

O uso de variáveis indexadas é muito útil, mas ocupa muita memória. Por isso é bom se calcular com razoável aproximação a dimensão, evitando-se assim dimensões maiores do que realmente necessário.

5. Exemplo 1

Considerando a seguinte tabela de dados:

CHEQUE	DATA	QUANTIA
1	1.01.84	1.000,00
2	2.02.84	5.000,00
3	5.02.84	8.000,00
4	4.03.84	2.000,00
5	4.03.84	60.000,00

Observe que cada item da tabela pode ser especificado por 2 coordenadas, uma de linha e outra de coluna. Por exemplo: a linha 4 coluna 3 refere-se a quantia de 2.000,00.

Digite:

```
 10 DATA 1, 1.0184,  1000.00
 20 DATA 2, 2.0284,  5000.00
 30 DATA 3, 5.0284,  8000.00
 40 DATA 4, 4.0384,  2000.00
 50 DATA 5, 4.0384, 60000.00
 60 FOR I=1 TO 5
 70 FOR J=1 TO 3
 80 READ A(I,J)
 90 PRINT A(I,J);"     ";
100 NEXT J,I
```

Obs.: A linha 90 exibe na tela os valores armazenados na instrução DATA, separados por 5 espaços. Esse artifício de se colocar os espaços e ainda deixar o próximo item a ser exibido na mesma linha (ponto e vírgula no final da instrução PRINT) nos dá um melhor aproveitamento da tela. Experimente mudar a linha 90 para:

```
 90 PRINT A(I,J)
```

e observe o resultado.

Note que não foi necessário a instrução DIM, pois para variáveis com menos de **11** índices você não precisará dimensioná-la.

6. Exemplo 2

Imagine se fossemos fazer uma agenda telefônica de nossos amigos, usando variáveis simples. Para que não perdessemos nenhum telefone teríamos que armazenar em variáveis com nomes diferentes.

```
Amigo 1    A$    NOME
           B$    FONE

Amigo 2    C$    NOME
           D$    FONE

    :
    :
    :
```

Já imaginou quantos INPUTs não teríamos? E não é só isso, pois usando-se esse método, poderíamos cadastrar pouco mais de uma dúzia de amigos com seus respectivos telefones.

Usando-se variáveis indexadas, poderemos cadastrar N amigos e N telefones (desde que N caiba na memória) e usar somente 2 comandos INPUT (um para o nome e outro para o telefone) com o auxílio dos comandos FOR/NEXT.

```
Amigo 1    N$(1)    NOME
           F$(1)    FONE

Amigo 2    N$(2)    NOME
           F$(2)    FONE
  :
  :
  :

Amigo N    N$(N)    NOME
           F$(N)    FONE
```

Digite:

```
 10 CLS
 20 INPUT "DIGITE NUMERO DE AMIGOS";N
 30 DIM N$(N)
 40 DIM F$(N)
 50 FOR I = 1 TO N
 60 INPUT "NOME";N$(I)
 70 INPUT "FONE";F$(I)
 80 NEXT
 90 INPUT "DIGITE <L> PARA LISTAR";A$
100 IF A$ <> "L" THEN END
110 CLS
120 FOR I = 1 TO N
130 ? N$(I),F$(I)
140 NEXT
```

Respire fundo e prepare-se; este programa testa sua memória e a do computador. Digite NEW para apagar o programa anterior.

Digite:

```
  5 DIM A(7)
 10 PRINT "DECORE ESTES NUMEROS"
 15 PRINT "VOCE TEM 10 SEGUNDOS"
 20 FOR X=1 TO 7
 30 A(X)=RND(100)
 40 PRINT A(X)
 50 NEXT X
 60 FOR X=1 TO 460*10: NEXT X
 70 CLS
 80 FOR X=1 TO 7
 90 PRINT"QUAL ERA O NUMERO"X
100 INPUT R
110 IF A(X)=R THEN PRINT "CERTO" ELSE PRINT "ERRADO-ERA"A(X)
120 NEXT X
```

A linha 5 reserva área para uma variável indexada de nome A com 7 ítens.

As linhas 20 a 50 armazenam 7 números aleatórios na variável indexada A.

A linha 60 introduz uma pausa de 10 segundos no programa, e a linha 70 limpa a tela.

As linhas 80 a 120 verificam se você ainda se lembra dos números que estão armazenados em A.

Obs.: Na verdade, nesse programa, não é necessário usar o comando DIM, pois ele possue somente 7 elementos. Entretanto, é sempre aconselhável colocar esse comando para reservar o tamanho exato de área na memória.

- Exercício 10: Faça um programa para controlar um estoque que apresenta os seguintes elementos:

ITEM Nº	QUANTIDADE
1	32
2	12
3	42
4	13
5	15
6	23
7	25
8	30
9	33
10	27
11	14
12	8

FAÇA VOCÊ MESMO O SEU PROGRAMA

XXIII. ARQUIVO MAGNÉTICO EM FITA

Além de armazenar programas em fita, você pode também usar seu gravador para armazenar dados. Isso é útil pois à medida que vamos "carregando" as variáveis podemos jogá-las diretamente na fita, proporcionando espaço na memória. A agenda telefônica feita anteriormente tem um limite para o índice N, pois a memória tem um limite. Portanto se tivermos muitos amigos, teremos que arquivá-los por partes. Fazendo um arquivo magnético em fita, esse problema é solucionado.

A instrução PRINT # -1 tem como função gravar todos os valores de variáveis em fita cassete. Quando esta instrução for executada, o gravador deverá estar na modalidade RECORD + PLAY.

O comando OPEN faz com que o computador abra as linhas de comunicação para que possamos ter acesso a fita cassete. Já o comando CLOSE fecha as linhas.

Exemplo:

```
 1 CLS:PRINT"PREPARE A FITA E PLAY+RECORD"
 5 INPUT"DIGITE<ENTER>QUANDO PRONTO";A$
10 OPEN"O",#-1, "NOMES"
15 CLS:PRINT"DIGITE<XX>QUANDO ACABAR"
20 INPUT"NOME";N$
25 IF N$="XX"THEN 50
30 PRINT #-1,N$
40 GOTO 15
50 CLOSE#-1
```

Assim sendo, armazenaremos quantos nomes quisermos em fita, até digitarmos <XX>.

O "O" localiza a saída de dados, ou seja, OUTPUT.

Agora veremos como trazer dados do arquivo magnético em fita.

Digite:

```
60 CLS:PRINT"VOLTE A FITA E PLAY"
70 INPUT"DIGITE<ENTER>QUANTO PRONTO";A$
80 OPEN"I",#-1,"NOMES"
```

onde "I", indica a entrada de dados, ou seja, INPUT.

A instrução INPUT #-1 tem como função trazer para a memória do computador, todos os valores de variáveis que estão no arquivo magnético. Adicione ainda:

```
 90 INPUT #-1,N$
100 PRINT N$  (exibe na tela os valores lidos
              na fita).
```

Para que o computador saiba onde acaba o arquivo "NOMES" e feche as linhas de comunicação, precisamos colocar a instrução EOF(-1). Essa instrução END OFF FILE, ou seja, fim do arquivo, vem junto com as instruções IF/THEN.

Digite:

```
 85 IF EOF(-1) THEN 120
110 GOTO 85
120 CLOSE#-1
```

Quando estivermos no fim do arquivo, a execução do programa salta para a linha de número 120, onde serão fechadas as linhas de comunicação.

Para que você teste somente a recuperação do arquivo em fita, levando-se em conta que você já armazenou-o nas primeiras instruções desse capítulo, digite RUN 60, ou seja, o programa será executado a partir da linha de número 60.

Vejamos a seguir o programa completo, como ficou:

Digite:

```
CLEAR
LIST
```
ENTER

daí temos:

```
 1 CLS:PRINT"PREPARE A FITA E PLAY+RECORD"
 5 INPUT"DIGITE<ENTER> QUANDO PRONTO";A$
10 OPEN "O",#-1,"NOMES"
15 CLS:PRINT"DIGITE<XX> QUANDO ACABAR"
20 INPUT"NOME";N$
25 IF N$="XX" THEN 50
30 PRINT#-1,N$
40 GOTO 15
50 CLOSE#-1
```
OUTPUT

(Cont.)

```
          60 CLS:PRINT"VOLTE A FITA E PLAY"
          70 INPUT"DIGITE <ENTER> QUANDO PRONTO";A$
          80 OPEN"I",#-1,"NOMES"
          85 IF EOF(-1) THEN 120
INPUT     90 INPUT #-1, N$
         100 PRINT N$
         110 GOTO 85
         120 CLOSE#-1
```

Para incluirmos em nosso arquivo magnético, endereços e telefones aos nomes arquivados acima, digite as seguintes linhas:

```
          26 INPUT "END.";E$
          28 INPUT "FONE";F$
          29 IF E$="XX" OR F$="XX" THEN 50
```

mude a linha 30 e a 90 para:

```
          30 PRINT #-1,N$,E$,F$
          90 INPUT #-1,N$,E$,F$
```

adicione ainda:

```
         101 PRINT E$
         102 PRINT F$
```

XXIV. IMPRESSORA

Detalhes de como ligar sua impressora ao VC50 estão no manual de operações, no início deste.

Leia também o manual da impressora utilizada.

1. LLIST

Esse comando faz com que o programa seja listado na impressora. Isto certamente vai facilitar a correção de longos programas.

2. PRINT #-2

Esse comando faz o computador imprimir não na tela, mas na impressora.

Experimente:

```
10 INPUT "DIGITE ALGO";A$
20 PRINT #-2,A$
```

Na tela não é possível escrever-se com letras minúsculas, mas na impressora sim.*

Pressione a tecla (SHIFT) e o zero (0) simultaneamente de modo que as linhas que forem digitadas aparecem na tela em vídeo inverso. Nesse modo você estará escrevendo com letras minúsculas. Caso você queira letras maiúsculas é só digitar (SHIFT) enquanto digita a letra.

Obs.: Para voltar a utilização do computador ao normal, devemos voltar a escrever em caracteres normais. Para isso digite (SHIFT) e o (0) simultaneamente.

***Obs.:** Com o uso do MAXITELA (ver apêndice), OS 9 ou Flex você terá 51 x 24 caracteres na tela, inclusive minúsculas.

BASIC EXTENDIDO

BASIC EXTENDIDO

Partindo-se do princípio de que você já tomou contato com a primeira parte desse manual, o BASIC EXTENDIDO fará com que você amplie as possibilidades de programação de seu VC50.

Com o BASIC EXTENDIDO, você poderá utilizar-se da alta resolução do VC50, traçar círculos, linhas, quadrados, com muito mais facilidade e precisão; e muitos outros recursos.

Então, sente-se e mãos a obra. Estamos certos de que você concordará que o computador é uma excelente aquisição para seu uso pessoal, sua família ou empresa.

I. ALTA RESOLUÇÃO GRÁFICA

Uma das mais interessantes características do BASIC EXTENDIDO é a sua habilidade em apresentar gráficos (tais como linhas e círculos) que são mais precisos, variados e fáceis do que os gráficos apresentados na primeira parte. Chamamos isso de alta resolução gráfica.

Começaremos com o elemento gráfico mais simples, que é um simples ponto.

Digite:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 PSET (10,20,8)
40 GOTO 40
```

A tela estará "limpa" e olhando-se cuidadosamente, veremos um pequeno ponto laranja no canto superior esquerdo da tela. Este ponto foi colocado lá pela instrução PSET da linha 30 do programa anterior.

Essa instrução possibilita a você colocar um ponto de determinada cor em qualquer posição da tela.

Os comandos PMODE e SCREEN, determinam o grau de detalhe e a faixa de cor. Estes serão estudados, posteriormente, detalhadamente.

PSET tem o seguinte formato:

```
PSET(X,Y,C)
```

onde:

X - indica a posição do eixo X (horizontal) de 0 a 255.

Y - indica a posição do eixo Y (vertical) de 0 a 191.

C - especifica a cor do ponto, podendo assumir valores de 0 a 8.

Obs.: a coordenada X deve vir sempre antes da coordenada y.

Embora não seja visualizado, o computador divide a tela em 49152 pontos. Cada um desses pontos pode ser localizado em uma malha, se visualizarmos a tela como um retângulo com 256 pontos horizontais e 192 verticais. Por isso é necessário se referir as coordenadas X e Y para especificar exatamente a posição desejada.

O VC50 é capaz de gerar nove cores distintas:preto, verde, amarelo, azul, vermelho, bege, azul piscina, roxo e laranja. Cada cor, como visto na primeira parte do manual, tem um código numérico. No programa anterior especificamos a cor laranja,pois usamos como valor para C o número 8.

Para apagarmos um ponto na tela usamos a instrução PRESET.

PRESET(X,Y)

onde:

X - especifica a posição no eixo X de 0 a 255

Y - especifica a posição no eixo Y de 0 a 191.

Observe que não é necessário especificar a cor na instrução PRESET, pois, o computador usa a cor do fundo automaticamente.

Para se familiarizar com as várias posições dos pontos na tela utilize-se dos gabaritos de espacejamento de tela, que estão no apêndice desse manual.

Exercício 11: Usando o exemplo anterior, faça com que o ponto fique agora "piscando", usando o comando PRESET.

FAÇA VOCÊ MESMO O SEU PROGRAMA

Exercício 12: Usando a função RND (Randômica), faça um pequeno programa que preencha a tela com pontos aleatórios de cores aleatórias.

FAÇA VOCÊ MESMO O SEU PROGRAMA

II. TRAÇANDO LINHAS

Com a instrução SET vista na primeira parte, as linhas retas traçadas eram grossas e para muitos casos isso poderá não ser desejado. Um caso muito desagradável do uso da instrução SET, é com uma linha diagonal, onde esta fica semelhante a uma escada e não com uma linha reta. Com a ótima resolução gráfica do VC50 poderemos facilmente contornar esta situação. Além disso, você poderá variar a largura da linha.

A instrução LINE é a responsável por esses recursos. Digite:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 GOTO 40
```

Rodando-se o programa nota-se uma diagonal laranja de fora a fora na tela.

Você poderá colocar na tela quantas linhas quiser. Essas linhas são perfeitamente definidas por dois pontos. Esses pontos são definidos através das coordenadas X e Y.

```
LINE (X1,Y1)-(X2,Y2),a,b
```

onde:

X1,Y1 - especifica o ponto de início da linha
X1 varia de 0 a 255
Y1 varia de 0 a 191

X2,Y2 - especifica o ponto final da linha
X2 varia de 0 a 255
Y2 varia de 0 a 191

a - pode ser PSET ou PRESET

b - pode ser B ou BF (veremos mais adiante).

Lembre-se que o primeiro conjunto de coordenadas indica o ponto de início da linha, e o segundo conjunto após o hifen indica o ponto de término. Caso não seja especificado o ponto de início de uma linha, o computador automaticamente iniciará no último ponto de término, da última linha. Se no programa ainda não foi utilizada nenhuma instrução LINE, o computador iniciará a linha em (128,96).

Observe, trocando-se a linha 30 do programa anterior por:

```
30 LINE -(255,191),PSET
```

Caso você não especifique o ponto de início, o ponto de término deverá ser precedido de um hifen (-).

A especificação da cor pode ser indicada em outra parte do programa, não na instrução LINE, como veremos mais adiante.

Na instrução LINE, a posição (a) pode ser PSET ou PRESET, certo? Antes de explicarmos a diferença entre PSET e PRESET, tente alterar a linha 30 do programa anterior para:

```
30 LINE(0,0)-(255,191),PRESET
```

E agora? sua tela ficou vazia.

Não realmente; a linha ainda está lá, só que não podemos vê-la. PSET faz a linha ficar na cor pré-especificada em primeiro plano, enquanto PRESET faz a linha ficar na cor pré-especificada do fundo da tela, ou seja, segundo plano.

Nesse programa estamos usando laranja como a cor de primeiro plano (cor da linha) e bege como a cor de segundo plano (cor da tela).

Para distinguirmos entre cores de primeiro e segundo plano vamos pensar em várias cores de giz para escrever num quadro. Alguns quadros são verdes, outros pretos, etc. A cor do quadro seria a cor de segundo plano (ou cor de fundo); o giz, a cor de primeiro plano. Se tentarmos escrever em um quadro verde com giz verde, não será possível distinguir nada. É exatamente isso o que acontece quando você substitui PSET por PRESET.

A posição (b) na instrução LINE, pode ser B ou BF, quando usada.

B - BOX (CAIXA)

Com essa posição B na instrução LINE, torna-se simples a execução de retângulos ou quadrados sem ter que incluir 4 linhas separadas no seu programa, para formar os 4 lados. O que temos que fazer é especificar dois vértices opostos do quadrado ou retângulo e acrescentar B na posição (b).

Mude novamente, a linha 30 do programa para:

```
30 LINE(0,0)-(255,191),PSET,B
```

Acrescentando-se F ao sufixo B, temos BF para a posição (b).

Essa opção permite que você preencha o retângulo ou o quadrado com a cor do primeiro plano.

Mude a linha 30 para:

```
30 LINE .....,PSET,BF
```

Com isso, você terá um retângulo laranja de 256x192.

Tenha sempre em mente que tudo o que foi visto nesse capítulo pode ser realizado com um simples comando LINE.

Exemplo:

```
10 PMODE 3,1:SCREEN 1,1:PCLS
20 X=RND(126):Y=RND(96):X2=255-X:Y2=191-Y:W=RND
   (X):H=RND(Y)
30 N=RND(255):POKE178,N
40 LINE(X,Y)-(X-W,Y+H),PSET,BF
50 LINE(X2,Y)-(X2+W,Y+H),PSET,BF
60 LINE(X,Y2)-(X-W,Y2-H),PSET,BF
70 LINE(X2,Y2)-(X2+W,Y2-H),PSET,BF
80 GOTO 20
90 PMODE 1,1
100 PCLS
110 SCREEN 1,1
120 A=RND(255)
130 X1=RND(255):Y1=RND(191):X2=RND(255):X3= RND
    (191)
140 POKE 178,A
150 LINE(X1,Y1)-(X2,Y2),PSET,BF
170 GOTO 120
```

Obs.: Os comandos que você ainda não conhece, serão vistos mais adiante, com detalhes.

Exercício 13: Faça um programa que crie uma caixa com 2 linhas se cruzando no centro da tela.

FAÇA VOCÊ MESMO O SEU PROGRAMA

Exercício 14: Faça um programa para desenhar uma casa na tela de sua T.V. Ela deve conter pelo menos:

1. uma porta na frente, claro;
2. uma janela;
3. uma chaminé.

Obs.: Será fácil de fazer este programa se você marcar previamente os pontos no gabarito da tela de gráficos.

FAÇA VOCÊ MESMO O SEU PROGRAMA

III. CONTROLANDO AS CORES

Falamos em cores de primeiro e segundo plano, mas nesse capítulo vamos determinar as cores desses planos.

COLOR lo. PLANO,2o. PLANO

onde:

lo. plano - é um número de 0 a 8 representado pelo código de cores já visto.

2o. plano - idem.

Quando não for especificado as cores de lo. e 2o. plano, o computador automaticamente escolhe o número mais alto do código de cores disponível (número 8) para o lo. plano. E o menor número para a cor de fundo (número 0).

Com a instrução COLOR, entretanto você poderá definir as cores do lo. e 2o. plano.

Acrescente no programa usado no início do capítulo II a linha:

```
6 COLOR 7,5
```

IV. MEMÓRIA E VÍDEO

Quando é enviado algum tipo de informação para a TV, esta é armazenada em uma parte da memória RAM reservada para o vídeo. Chamaremos essa memória de VÍDEO RAM, OK?

A memória de vídeo normal é usada para caracteres como letras e números, mas não para gráficos, ou seja, para alta resolução gráfica. Consequentemente, o computador utiliza outra parte do VÍDEO RAM para armazenar gráficos. Essa região do VÍDEO RAM pode apresentar 8 níveis de memória e dependendo da solicitação, pode-se por de lado um desses 8 níveis com PCLEAR. Um nível contém 1536 locações de memória. PCLEAR determina o tamanho do VÍDEO RAM utilizado para gráficos. Quando não especificado, é automaticamente assumido 4 níveis. A necessidade de níveis depende da quantidade de detalhes ou faixa de cores que deseja-se utilizarno programa. Quanto mais detalhes e cores o programa utilizar, mais memória será usada.

A instrução PMODE, possibilita a escolha da resolução (quantidade de detalhes) e selecionar o nível de memória a ser utilizada.

PMODE modo, nível inicial

onde:

modo - é um número de 0 a 4, sendo opcional; se omitido o valor atual é usado; e se não foi ainda especificado desde o início, 2 é usado.

nível inicial - é um número de 1 a 8 e especifica qual dos 1,5k de memória de vídeo será utilizado inicialmente; é opcional, e sendo omitido será usado o anterior; e se não foi ainda especificado desde o início, 1 é usado.

Portanto quando não especificado, temos:

```
PMODE 2,1
```

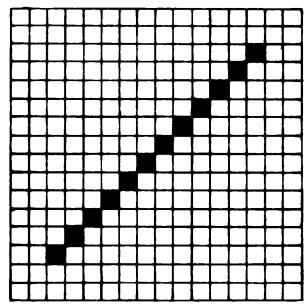
Como já foi visto, quando trabalha-se com gráficos, o computador divide a tela da T.V. em 256 pontos no eixo horizontal (X) e 192 no eixo vertical (Y). O tamanho da malha é considerado normal quando utilizada a alta resolução gráfica.

Uma malha de 256x192 também proporciona um altíssimo nível de resolução. Chamaremos isso de PMODE 4.

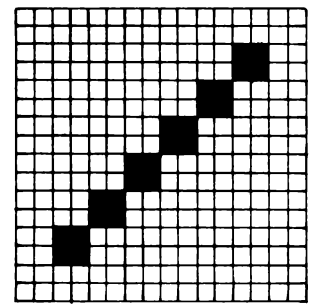
Podemos ter 5 PMODEs de 0 a 4, cada um com diferentes capacidades. Você já sabe que a única diferença entre alta resolução e baixa resolução (PMODE 4 e PMODE 0 respectivamente) é o tamanho do ponto. Alta resolução usa um único ponto, enquanto que baixa usa 4 desses pontos para formar um.

TABELA

PMODE	MALHA	CORES	NÍVEL	PONTO	RESOLUÇÃO
4	256x192	2	4		alta
3	128x192	4	4		média
2	128x192	2	2		média
1	128x96	4	2		baixa
0	128x96	2	1		baixa



ALTA RESOLUÇÃO



BAIXA RESOLUÇÃO

Obs.: embora tenhamos chamado o PMODE 0 E 1 de baixa resolução, ainda assim tem mais detalhes do que os gráficos 64x32 obtidos por SET em texto normal.

Digite:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE(0,0)-(255,191),PSET
30 LINE(0,191)-(255,0),PSET
40 GOTO 40
```

Depois de rodar o programa, observando-o mude a linha 5 para PMODE 4,1. Você deverá perceber duas diferenças:

1. a linha está mais nítida em detalhes devido a alta resolução.
2. a mudança de cor se deve ao fato de que nesse modo, temos somente duas cores.

PMODE 4 lhe dá uma alta resolução mas limita a escolha da cor. As combinações possíveis nesse modo e nos outros estão na tabela abaixo:

TABELA DE CORES/MODO

PMODE	CORES-SET (SCREEN)	2 CORES COMBINADAS	4 CORES COMBINADAS
4	0	preto e verde	
	1	preto e bege	
3	0		verde/amarelo/azul/ vermelho
	1		bege/azul piscina/ roxo/laranja
2	0	preto e verde	
	1	preto e bege	
1	0		verde/amarelo/azul/ vermelho
	1		bege/azul piscina/ roxo/laranja
0	0	preto e verde	
	1	preto e bege.	

A seguir temos um programa que mostra um quadrado e como seu desenho varia ao se variar os diferentes modos. Note a mudança das linhas (grossa e fina) e a mudança de cor, quando o programa varre os diferentes PMODEs.

```
 5 FOR MODO=0 TO 4
10 PMODE MODO,1
20 PCLS
30 SCREEN 1,1
40 LINE(75,50)-(125,100),PSET,B
50 FOR Y=0 TO 500:NEXT Y
60 NEXT MODO
70 GOTO 5
```

Como já mencionado, a memória gráfica de vídeo é dividida em oito níveis de memória.

O segundo parâmetro de PMODE, nível inicial, permite selecionar qual destes níveis você quer utilizar para apresentar seus gráficos. Se não for especificado o nível, automaticamente o computador utiliza o nível 1 ou o último nível especificado.

Antes de irmos mais além, observe a tabela abaixo:

n = 1 a 5	PMODE 4,n	PCLEAR 4+(n-1)
	PMODE 3,n	PCLEAR 4+(n-1)
n = 1 a 7	PMODE 2,n	PCLEAR 2+(n-1)
	PMODE 1,n	PCLEAR 2+(n-1)
n = 1 a 8	PMODE 0,n	PCLEAR 1+(n-1)

Essa tabela detalha o número de níveis que cada PMODE requer. Note que, com o aumento da resolução e das cores disponíveis, mais memória é usada. Cada nível reservado possui 1536 bytes de memória.

Digite:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE(110,20)-(120,30),PSET,B
40 GOTO 40
```

Rodando-se o programa, tem-se um pequeno quadrado, próximo ao alto da tela. Agora altere a linha 5 para:

```
5 PMODE 3,2
```

e rode o programa. Obtem-se, então, a condição de erro "FC ERRO?". É que mudando-se o nível inicial de 1 para 2 a instrução PMODE 3 requer 4 níveis de memória gráfica. Quando você começou no nível 2, são requeridos os níveis 2,3,4 e5. Mas como somente foi "limpado" os níveis de 1 a 4, logo o nível 5 não é utilizado.

Mais adiante será mostrado como solucionar essa questão.

Exercício 15: Você já sabe que os PMODEs 4 e 3 exigem quatro níveis de memória e portanto, você só pode selecionar o nível 1 como inicial. Mas o que aconteceria se você passasse para PMODE 2,1 ou 0?. Em que nível de memória você poderia começar nesses PMODEs sem gerar um **?FC ERRO** ?. Mude os parâmetros do PMODE para descobrir isto e compare suas respostas com as nossas.

FAÇA VOCÊ MESMO O SEU PROGRAMA

V. PCLS

Digite novamente o programa:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE(0,0)-(255,191),PSET
30 LINE(0,191)-(255,0),PSET
40 GOTO 40
```

A instrução PCLS limpa a tela gráfica, da mesma forma com que CLS limpa a tela de textos.

A cor de fundo (2o. plano) utilizada é usada para limpar a tela, a não ser que seja especificado outra cor que, pode ser de 0 a 8 representando os códigos de cores disponíveis.

No programa acima a cor usada foi a bege que é a cor de fundo da tela. Agora tente, mudando a linha 10 do programa para:

```
10 PCLS 6
```

e rode o programa. Assim temos linhas laranjas sobre um fundo azul piscina.

VI. PCLEAR

Quando for usado uma característica de nível de memória, você precisará (algumas vezes) usar PCLEAR para reservar 1,5k níveis de memória, ou seja, determinar o tamanho da VÍDEO RAM.

Obs.: desde que PCLEAR determina o tamanho da VÍDEO RAM, ele deverá ser a primeira ou segunda instrução no seu programa, ou seja, depois da instrução CLEAR, se usada.

PCLEAR n

onde **n** é o número de 1 a 8 que, especifica o número de níveis gráficos a serem reservados.

Para PCLEAR 4 não há necessidade de reserva através de PCLEAR, pois o seu VC50 faz isso automaticamente.

Para exemplificar a instrução PCLEAR, usaremos aquele programa que apresentou uma mensagem de erro "FC ERRO":

```
 5 PMODE 3,2
10 PCLS
20 SCREEN 1,1
30 LINE(110,20)-(120,30),PSET,B
40 GOTO 40
```

Acrescente ao programa a linha:

```
4 PCLEAR 8
```

e execute-o. De fato, agora você pode iniciar o programa utilizando qualquer nível de 1 a 5.

Você pode estar perguntando, porque então o computador não utiliza PCLEAR 8 todo o tempo?

A razão é: PCLEAR 8 diminui a quantidade de memória disponível.

VII. PCOPY

Você sabia que para se fazer um desenho animado de 7 minutos de duração, são necessários 12.000 desenhos indi

viduais? Mas com o auxílio do seu VC50, você poderá realizar desenhos animados com muita facilidade.

A instrução PCOPY, permite a reprodução dos dados de um nível de memória para outro.

PCOPY origem **TO** destino

onde origem e destino são números entre 1 e 8 especificando os níveis de memória.

Caso você queira reproduzir gráficos do nível 3 para o nível 8 use PCOPY 3 TO 8.

Uma vantagem de PCOPY é encurtar programas, eliminando-se repetições.

Exercício 16: O programa a seguir mostra quatro quadrados (que estão em quatro níveis de memória diferentes) na tela ao mesmo tempo. Execute e diminua o programa usando PCOPY.

```
 4 PCLEAR 8
 5 PMODE 3,4
10 PCLS
11 SCREEN 1,1
12 LINE (110,20)-(120,30),PSET,B
20 PMODE 3,3
21 SCREEN 1,1
22 LINE (110,20)-(120,30),PSET,B
30 PMODE 3,2
31 SCREEN 1,1
32 LINE (110,20)-(120,30),PSET,B
40 PMODE 3,1
41 SCREEN 1,1
42 LINE (110,20)-(120,30),PSET,B
50 GOTO 50
```

FAÇA VOCÊ MESMO O SEU PROGRAMA

VIII. PPOINT

A instrução PPOINT testa a cor de um ponto gráfico especificado.

PPOINT(X,Y)

onde:
X - coordenada X do ponto de 0 a 255
Y - coordenada Y do ponto de 0 a 191.

Procure usar sempre, PPOINT no mesmo PMODE utilizado para marcar os pontos gráficos; do contrário, poderá causar muita confusão.

O programa seguinte mostra como PPOINT pode ser uma instrução conveniente incluída num programa.

```
  5 PMODE 3,1
 10 PCLS
 20 SCREEN 1,1
 30 X=RND(10)
 40 Y=RND(10)
 50 C=RND(8)
 60 PSET(X,Y,C)
 70 IF PPOINT(5,5) = 8 THEN GOTO 90
 80 GOTO 30
 90 CLS
100 PRINT@100,"POSICAO(5,5)AGORA E LARANJA"
```

O que acontece é que o computador preenche um quadrado de 10x10 com pontos coloridos aleatórios. Entretanto, quando o ponto (5,5) for laranja (código de cor 8), o computador pára de gerar pontos, limpa a tela e exibe a mensagem.

Na linha 70 o programa testa a cor do ponto (5,5).

Exercício 17: Faça um programa para simular uma tempestade, pondo linhas desordenadas em posições aleatórias, em diferentes níveis de memória.

FAÇA VOCÊ MESMO O SEU PROGRAMA

IX. SCREEN

A instrução SCREEN fornece ao computador duas informações da seguinte forma:

SCREEN tipo, tipo de cor

onde:

tipo - será 0 para textos e 1 para gráficos.

tipo de cor - para o valor 0 tem-se a combinação preto e verde; para 1 tem-se duas combinações: verde, amarela, azul e vermelha ou bege, azul piscina, roxo e laranja; desde que em modo de 4 cores.

Note que nos programas anteriores usamos a instrução SCREEN 1,1.

Toda vez que seu programa apresentar textos (PRINT, INPUT), o computador automaticamente realizará o COMANDO SCREEN 0,0. Mas podemos, mudar a cor de um texto especificando:

```
SCREEN 0,1
```

Utilizando SCREEN desse modo, pode-se obter a tela de preto à laranja. Entretanto o computador volta automaticamente para preto e verde logo após a instrução PRINT ou INPUT selecionado para preto e laranja.

Digite o seguinte programa:

```
 5 CLS
10 PRINT@195,"VERMELHO E LARANJA"
20 SCREEN 0,1
30 FOR A=0 TO 1000:NEXT A
40 CLS
50 PRINT@195,"PRETO E VERDE"
60 FOR X=0 TO 1000:NEXT A
70 GOTO 5
```

Como já visto com (BREAK) paramos a execução do programa acima, certo? Note que após a instrução (BREAK) o computador volta ao display preto e verde, mesmo que a mensagem estivesse em preto e laranja.

Exercício 18: Faça um programa que mude a tela de textos para a tela de gráficos. Você talvez queira colocar um "loop" no programa de maneira que ele mude a combinação de cores a cada passagem completa pelos comandos do "loop". Desse modo, você pode ver todas as características da instrução SCREEN em ação.

FAÇA VOCÊ MESMO O SEU PROGRAMA

X. CÍRCULOS

Com a instrução CIRCLE pode-se criar um círculo cheio ou parcial, ou então uma elipse.

CIRCLE(X,Y),R,C,hw,início, fim

onde:

X - coordenada X do ponto central; é um número de 0 a 255

Y - coordenada Y do ponto central; é um número de 0 a 191.

R - raio do círculo

C - código de cor disponível de 0 a 8; se omitido a cor de lo. plano é utilizada.

hw - especifica a proporção entre altura e largura e é um número de 0 a 255; se omitido 1 é usado.

início - especifica o ponto de partida do círculo e é um número de 0 a 1; se omitido 0 é usado.

final - especifica o ponto de término do círculo e é um número de 0 a 1; se omitido 1 é usado.

Somente o ponto central e o raio são necessários para desenhar um círculo.

O maior círculo que pode-se traçar na tela tem o centro em (128,96) e um raio de 95. Se o raio for maior que 95, o círculo ficará achatado contra o perímetro da tela.

Digite o programa:

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE(128,96),95
40 GOTO 40
```

Rodando-se o programa, aparecerá na tela um círculo laranja em um fundo bege. Nota-se também, que o círculo não é realmente redondo. Isto porque na linha 5, usamos PMODE 1 (média resolução).

Altere o PMODE na linha 5, para PMODE 4:

```
5 PMODE 4,1
```

Nota-se agora um círculo realmente redondo de cor bege em fundo preto.

Para PMODE 1 ou 3, temos opções de 4 cores, perdendo-se na resolução gráfica. Mas com o parâmetro C, podemos escolher uma cor para o círculo a ser exibido.

Altere agora a linha 30 para:

```
30 CIRCLE(128,96),95,6
```

Rodando-se o programa não veremos nada, pois o PMODE 4 não nos possibilita essa opção de cor. Mude também a linha 5 para PMODE 1,1. Assim teremos um círculo de cor azul piscina.

Exercício 19: Usando o programa acima, desenhe um alvo. Você pode fazer isso de 2 maneiras:

1. Acrescente uma linha no programa para cada círculo concêntrico, mas use as coordenadas X e Y iguais.
2. Use um "loop" FOR...NEXT com STEP 10 para que o computador faça isso por você.

FAÇA VOCÊ MESMO O SEU PROGRAMA

ELIPSE

Para gerarmos uma elipse, utilizaremos a relação altura/largura (hw). Esta relação é a proporção entre a largura do círculo e sua altura.

A largura do círculo sempre será igual ao raio especificado. A altura é determinada por (hw). Se (hw) é maior que 1, o círculo será mais "alto" do que "largo"; se (hw) é menor que 1, o círculo será mais "largo" do que "alto". Por exemplo:

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
30 CIRCLE(128,96),30,,1
40 GOTO 40
```

Como foi usado (hw) igual a 1, temos um círculo real como se tivéssemos omitido a relação (hw).

Note que foi omitido o código de cor. Isso diz ao computador para utilizar a cor de 1º plano. Todavia é necessário incluir a vírgula.

Para um círculo mais "alto" do que "largo", mude a linha 30 para:

```
30 CIRCLE(128,96),30,,3
```

Para um círculo mais "largo" do que "alto":

```
30 CIRCLE(128,96),30,,.25
```

Caso (hw) seja igual a zero, o círculo será tão infinitamente largo como alto, ou seja, se tornará uma linha horizontal. Se (hw) for muito além do valor 1, o círculo se aproximará de uma linha vertical. Experimente:

```
30 CIRCLE(128,96),30,,0
```

e

```
30 CIRCLE(128,96),30,,100
```

INÍCIO E FIM

Os dois últimos parâmetros da instrução CIRCLE, permitem que se exiba apenas uma parte do círculo (um arco).

Para isso, especifique o ponto onde o arco inicia (0 a 1), coloque uma vírgula e indique onde é o final (0 a 1).

Uma vez iniciado o círculo, o seu traçado se processa no sentido dos ponteiros do relógio.

AÇÃO CIRCULAR	INTERVALO INICIAL/FINAL
1/4 de círculo	0.25
1/2 de círculo	0.50
3/4 de círculo	0.75
círculo	1

Por exemplo, se você quiser apenas meio círculo, digamos de 6 horas (0.25) à 12 horas (0.75), mude a linha 30 para:

```
30 CIRCLE(128,96),30,1,1,.25,.75
```

Obs.: sempre que o ponto de início for igual ou maior que o ponto final, ou quando início/fim for omitido, o computador desenhará um círculo completo.

Exercício 20: Faça um programa para simular uma lua crescente em um céu escuro. Isso vai exigir dois arcos intercruzados e muitas tentativas e erros de sua parte.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XI. PAINT

A função gráfica PAINT permite exibir-se na tela qualquer forma com qualquer cor disponível.

PAINT (x,y),c,b

onde:
x - coordenada x de 0 a 255.
y - coordenada y de 0 a 191.
c - código de cor de 0 a 8.
b - indica a cor do contorno; é um número de 0 a 8.

Os números entre parênteses indicam onde as cores começam. O primeiro parâmetro (c), indica a cor e o segundo parâmetro (b) indica a cor da borda que delimita o desenho. Se o computador atinge uma borda de uma outra cor que não a especificada, ele continuará pintando além dela.

Digite:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 LINE (0,191)-(255,0),PSET
50 CIRCLE (128,96),90
60 PAINT (135,125),8,8
70 GOTO 70
```

Rodando-se o programa temos um círculo cujo centro está no ponto onde se cruzam as duas linhas definidas pelas linhas 30 e 40 do programa. A linha 60 de seu programa, faz com que o computador vá até a posição (135,125) da tela e use a cor laranja para pintar, até que a pintura atinja uma borda laranja.

Apague a linha 30 e rode o programa. Veja como o computador "pinta" metade do círculo após os contornos serem definidos.

Tente agora:

PAINT (1,1),8,8 e note o resultado. Assim temos infinitas formas de obtermos desenhos na tela. Use a imaginação.

Exercício 21: Você está lembrado daquele programa da casa visto no capítulo II, use-o como base para desenhar uma casa, agora com garagem. Use a instrução PAINT para levantar e abaixar a porta. Isso exigirá algum esforço de sua parte. Acrescente um período de espera antes e depois de abrir a porta da garagem. Aproveite e coloque um sol na tela usando CIRCLE.

Obs.: não se esqueça de pintar a sua casa.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XII. SIMPLIFICANDO AINDA MAIS

Mostramos como criar linhas, círculos, retângulos e quadrados, até mesmo como "pintá-los", se necessário. Mas há um modo mais fácil para fazer tudo isso.

A instrução DRAW permite-nos desenhar uma linha ou uma série delas, só especificando sua direção, ângulo e cor.

DRAW LINHA é uma expressão string que contém várias variáveis, e inclue:

- **comandos de movimento:**

M - mover a posição do desenho

U - subir

D - descer

L - esquerda

R - direita

E - ângulo de 45 graus

F - ângulo de 135 graus

G - ângulo de 225 graus
H - ângulo de 315 graus
X - executa um sub-string e retorna

- **modos:**

C - cor
A - ângulo
S - escala

- **opções:**

N - nenhuma nova posição de desenho
B - espaço em branco (nenhum desenho, apenas mover)

Obs.: sempre coloque a opção B antes do comando de movimento M; de outro modo, linhas indesejáveis aparecerão.

Anteriormente, mostramos como criar um retângulo ou quadrado usando a instrução LINE. Tinha-se que localizar os pontos de início e o de término, sendo que o computador os conectava. Isso ocupa apenas uma linha de programa, mas tem-se muito trabalho ao tentar plotar nos gabaritos de espacejamento de tela de alta resolução gráfica, o quadrado ou retângulo desejado. Com DRAW, simplesmente você localiza o ponto inicial e diz ao computador qual direção e até onde a linha vai. Caso seja omitido o ponto inicial, o computador começará na última posição de DRAW; ou no centro da tela se DRAW não tiver sido usado no programa anteriormente.

Digite:

```
 5 PMODE 4,1
10 PCLS
15 SCREEN 1,1
20 DRAW "BM128,96;U25;R25;D25;L25"
30 GOTO 30
```

As cotas (128,96) nos dá o canto inferior esquerdo do quadrado apresentado na tela, isto é, o comando de movimento M, posiciona o primeiro movimento a ser traçado na tela.

M,x,y

onde: x - é a coordenada x de 0 a 255.
y - é a coordenada y de 0 a 191.

Obs.: novamente, destacamos a necessidade do comando M ser colocado diretamente após a opção B; do contrário, linhas indesejáveis aparecerão.

No programa acima, instruimos o computador a começar a desenhar no ponto de coordenada (128,96), subir (U) 25 pontos, mais 25 para a direita (R), descer (D) mais 25 pontos, e finalmente ir para a esquerda (L) 25.

Caso não seja especificado o comprimento, o computador assume 1 como comprimento.

Cada comando de movimento foi separado com um ponto e vírgula (;), mas não é necessário fazer isso. Apenas torna o programa mais fácil de se ler. Entretanto, você sempre deve separar as coordenadas x e y do comando M com uma vírgula (,).

Vamos agora girar o quadrado, apoiando-o num dos cantos. Altere a linha 20 para:

```
20 DRAW "BM128,96;E25;F25;G25;H25"
```

Este DRAW também começa na coordenada (128,96) da tela. Ao invés de subir reto, a primeira linha faz um ângulo de 45 graus e as próximas três linhas são feitas com seus ângulos designados.

Caso usarmos PMODE 0 ou 1 e E,F,G, ou H para gerar linhas as quais tem um comprimento ímpar e no mínimo um número ímpar nas coordenadas (x,y), linhas F e H darão uma leve "escorregada" para o ponto central; se ambas as coordenadas (x,y) são números pares, linhas E e G "escorregarão". Isto é normal.

Exercício 22: Agora que você já aprendeu o comando de movimento DRAW, use-o para desenhar uma estrela, tanto para as linhas perpendiculares como para as diagonais.

FAÇA VOCÊ MESMO O SEU PROGRAMA

MOVIMENTO ABSOLUTO x MOVIMENTO RELATIVO

Suponhamos que você desenhou um quadrado mas quer desenhar outro próximo. O problema é, você saber exatamente a distância em que quer o segundo quadrado, sem ter que localizar as coordenadas (x,y).

Há uma outra forma do comando M que permite a você especificar um movimento relativo ao invés de absoluto. Até

agora, temos usado movimento absoluto, isto é, pontos especificados em termos de suas coordenadas atuais (x,y). Com movimento relativo, especificamos pontos em relação ao ponto atual (último desenho).

Aqui está a instrução M para movimento relativo:

Mx-offset,y-offset

onde:

x-offset - é um número que especifica a distância da posição x atual. Se x-offset é precedida por um sinal "+", ela é incrementada de uma quantidade especificada. Se for um sinal "-", ela é decrementada de uma quantidade especificada.

y-offset - é um número que especifica a distância da posição y atual. Quanto ao incremento ou decremento, em y funciona como em x. A única diferença está em que para offsets positivos, somente o sinal do y-offset pode ser omitido.

Digite:

```
 5 PMODE 4,1
10 PCLS
15 SCREEN 1,1
20 DRAW "BM128,96;U25;R25;D25;L25"
30 DRAW "BM+15,15;U25;R25;D25;L25"
40 GOTO 40
```

Quando o computador executar a linha 30, a posição atual será (128,96). Então o canto inferior esquerdo do novo quadrado está localizado em:

```
x = 128+15=143
y =  96+15=111
```

Altere a linha 30 para:

```
30 DRAW "BM+15,-15;U25;R25;D25;L25"
```

Ao rodar o programa com essa modificação, o ponto inicial do novo quadrado, ou seja, o seu canto inferior esquerdo se localizará em:

```
x = 128+15=143
y =  96-15= 81
```

Exercício 23: Tente desenhar um cubo em sua tela. Isto pode ser feito somente com a instrução DRAW, mas procure incorporar em seu programa o comando LINE, para que você utilize movimento relativo e absoluto.

FAÇA VOCÊ MESMO O SEU PROGRAMA

ESCALA

Para definirmos a escala de uma figura qualquer gerada por DRAW, tudo o que temos que fazer é incluir o seguinte:

Sx

onde x é um número de 1 a 62 e indica o fator de escala em unidades de 1/4.

```
 1 ----- 1/4
 2 ----- 2/4
 3 ----- 3/4
 4 ----- 4/4 (completa)
 5 ----- 5/4
 '
 '
 8 ----- 8/4 (dupla)
 '
 '
12 -----12/4 (tripla)
etc.
```

Obs.: se não for especificado um número para S, o computador usa S4 (4/4=1).

Depois de um comando Sx, todo comando de movimento absoluto e relativo será gerado ou escalado de acordo com o próximo comando Sx.

Temos condições de melhorar o nosso programa que gera um quadrado na tela através da função DRAW. Utilizando-se do último programa, apague a linha 30 e mude 20 para:

```
20 DRAW "S2;BM128,96;U25;R25;D25;L25"
```

Rodando-se o programa, obteremos um quadrado no canto inferior esquerdo em (128,96), mas apenas com metade do tamanho que você especificou.

Digite o seguinte programa para ver como um pequeno quadrado pode ser reduzido e aumentado até preencher toda a tela.

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
25 FOR S=1 TO 62
30 S$="S"+STR$(S)+";"
35 DRAW S$+"BM10,100U20R20D20L20"
40 NEXT S
50 GOTO 50
```

Obs.: quando usamos uma escala cuja linha resultante não é um número inteiro, o computador arredondará o comprimento da linha para o número inteiro mais próximo. Por exemplo: "S2U25R25D25L25" resultaria em um quadrado de 12 1/2 x 12 1/2 de escala; o computador desenhará um quadrado de 13x13.

COR

Com o comando C de DRAW temos a possibilidade de especificar a cor de uma linha em particular.

Cx

onde x é um número de 0 a 8 e especifica um dos 9 códigos de cor. Se omitido, a cor de primeiro plano é usada.

Digite:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 DRAW "C6;BM128,96U25R25D25L25"
40 GOTO 40
```

Rodando-se o programa, obtem-se um quadrado azul piscina em fundo bege.

Você pode colocar Cx em qualquer ponto da instrução DRAW, e todos os comandos utilizarão a cor especificada.Por exemplo, mude a linha 30 para:

```
30 DRAW"C8;BM128,96U25R25;C6;D25L25"
```

Rodando o programa você deverá ter um quadrado de 2 cores na tela. As primeiras duas linhas devem ser laranja, e as duas seguintes azul piscina.

ÂNGULO

Outro parâmetro que existe em DRAW é (A), que permite especificar o ângulo no qual a linha deve ser desenhada. Depois que esse parâmetro é colocado na instrução DRAW, todas as linhas subsequentes deverão ser desenhadas com a disposição de ângulo especificado por Ax.

Ax

onde x é um número de 0 a 3 e especifica um dos seguintes ângulos:

```
0 ---   0 graus
1 ---  90 graus (sentido horário)
2 --- 180 graus
3 --- 270 graus
```

Caso não seja especificado o parâmetro Ax, o computador assume como A0.

Para ilustrar isso, altere a linha 30 para:

```
30 DRAW"A0;BM128,96;U25"
```

Rodando-se o programa, teremos na tela uma linha de 25 pontos de comprimento, reta e subindo.

Agora mude para:

```
30 DRAW"A1;BM128,96;U25"
```

Rodando-se, agora tem-se uma linha "apontando" para a direita.

Obs.: a diferença no comprimento da linha é porque os "pontos" do PMODE 3 são retângulos que tem a altura maior que a largura. Portanto, linhas horizontais são ligeiramente mais longas do que as verticais.

CRIANDO VAZIOS

Suponha que queiramos desenhar um "espaço vazio" ou uma linha invisível. Pode-se fazer isto com o comando B. Usando-se ainda o programa anterior, façamos o lado direito ficar um espaço vazio. Para isso mude a linha 30 para:

```
30 DRAW"BM128,96;U25;R25;B;D25;L25"
```

Exercício 24: Imprima seu nome na tela usando DRAW. Isso significa que você deverá permanecer na tela de gráficos. Claro, seria mais fácil escrever seu nome na tela de texto, mas você pode ter texto e gráfico ao mesmo tempo.

Obs.: com o "MAXITELA" você poderá fazer isso futuramente com maior facilidade. Veja Apêndice.

FAÇA VOCÊ MESMO O SEU PROGRAMA

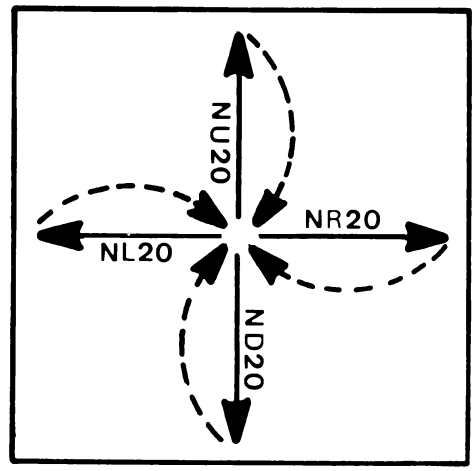
MAIS COMANDOS

Um outro parâmetro de DRAW é N. Este faz com que, após ter sido desenhada qualquer linha, o processamento retorne a posição atual definida por M. Usando novamente o programa anterior que gera um quadrado, altere a linha 30 para:

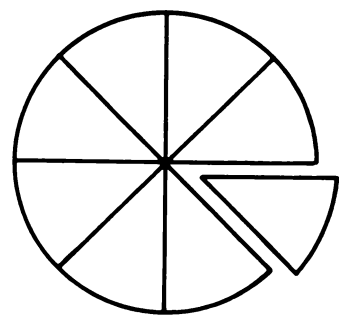
```
30 DRAW"M128,96;N;U25;N;R25;N;D25;N;L25"
```

Rodando-se o programa, deverá haver 4 linhas radiais partindo do centro da tela (128,96), cada uma em 4 diferentes direções, ou seja, para cima, para a direita, para baixo e para a esquerda. O que aconteceu foi que, ao desenhar uma linha para cima (U25) começando em (128,96) com 25 pontos de comprimento, o computador retornou para a posição inicial antes de iniciar nova linha. Depois foi desenhada uma linha de 25 pontos para a direita e a volta para a posição inicial. A seguir, uma linha de 25 pontos para baixo, o re

torno; e finalmente, uma linha de 25 pontos para a esquerda e mais uma vez, o retorno a posição inicial.



Exercício 25: Usando a opção N do DRAW (e CIRCLE) faça o computador desenhar uma torta com oito pedaços. Após fazer isso, corte um pedaço da torta e separe-o para um lado.



FAÇA VOCÊ MESMO O SEU PROGRAMA

CONSTANTES STRING x VARIÁVEIS STRING

Pode-se também, usar uma variável string com a instrução DRAW. A variável string deverá ser identificada na linha de programa que precede DRAW. Por exemplo, mude o programa anterior para:

```
25 A$="BM128,96;U25R25D25L25"
30 DRAW A$
```

Com isso, o BASIC EXTENDIDO de seu VC50, pode oferecer uma variação a que chamamos de comando (X). Enquanto você está executando uma rotina DRAW, o comando (X) permite realizar outra string DRAW, a seguir, voltar e completar a primeira operação. Altere a linha 30 para:

```
30 DRAW"BM95,50;U25;R25;XA$;D25;L25"
```

Rode-o. O computador iniciará uma linha em (95,50) que sobe e vira para a direita. Nesse ponto, A$ é executada e um quadrado de 25x25 é desenhado, começando em (128,96). Depois da execução de A$, o computador volta a string original (atual) e completa a sua execução.

Obs.: a posição de desenho não retorna a sua posição anterior, permanecendo onde foi terminada **A$**.

Sempre é necessário colocar um ponto e vírgula (;) após o "dolar" (**$**) no comando DRAW. Os outros (;) não são necessários.

Pode-se ter mais de uma string DRAW, sendo executadas ao mesmo tempo.

```
.... XA$;XB$;XC$ ....
```

Exercício 26: Você ainda tem aquele programa para desenhar uma casa? Tome-o por base e agora usando a maioria das instruções vistas até aqui, faça com que a porta de sua casa, abra e feche. Se você está achando difícil, dê uma olhada em nossa solução, no final deste manual.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XIII. GET E PUT

Essas instruções permitem que o computador armazene em um array (matriz) uma área retangular que contenha um display gráfico, e colocá-lo na tela mais tarde, quando quisermos.

Quando da simulação de movimento, GET e PUT podem "mover" objetos mais rapidamente. Os formatos de GET e PUT são:

GET ponto inicial-ponto de término, destino, G

onde:

ponto inicial - é a coordenada (X1,Y1) do canto superior esquerdo de um retângulo do display gráfico.

ponto de término - é a coordenada (X2,Y2) do canto inferior direito do mesmo retângulo.

destino - é o nome de um array pré-definido que armazenará o conteúdo dos retângulos.

G - faz com que o conteúdo dos retângulos seja armazenado com completo detalhe gráfico. Em alguns casos, é opcional.

PUT ponto inicial-ponto de término, fonte, processo

onde:

ponto inicial e final estão explicados acima.

fonte - é o nome do array pré-definido.

processo - determina como os dados serão escritos dentro do retângulo.

E ainda:

PSET - ativa cada ponto no array de origem ou fonte.

PRESET - apaga cada ponto que está colocado no array de origem.

AND, OR, NOT

AND - operador lógico; compara os pontos armazenados no retângulo de origem com o retângulo destino. Se ambos os pontos estão posicionados, então o ponto da tela será localizado.

OR - operador lógico; compara os pontos citados acima. Se pelo menos um estiver posicionado, o ponto da tela será localizado.

NOT - operador lógico; inverte o estado de cada ponto no retângulo de destino indiferente ao conteúdo da instrução PUT.

O programa seguinte nos dará uma idéia do que GET e PUT podem fazer.

```
 5 PCLEAR 4
10 PMODE 3,1
15 PCLS
20 SCREEN 1,1
25 DIM V(20,20)
30 CIRCLE (20,20),10
35 GET (10,10)-(30,30),V
40 PCLS
45 FOR D=1 TO 300:NEXT D
50 PUT (110,110)-(130,130),V
55 FOR D=1 TO 300:NEXT D
60 GOTO 60
```

Um círculo laranja deve ter aparecido no alto esquerdo da tela (posição (10,10)-linha 30). Depois de um segundo ou dois, ele desaparecerá subitamente, e reaparecerá em outro lugar da tela (posição (110,110)-linha 45).

O que o computador fez foi:

- criar um array (linha 25)
- criar um círculo (linha 30)
- retirar o círculo que estava no retângulo fonte (especificado pela instrução GET) e armazená-lo em array (linha 35).
- limpar a tela (linha 40)
- colocar o círculo do array no retângulo de destino (especificado pela instrução PUT-linha 50).

Vamos estudar um pouco mais a respeito desses retângulos fonte e destino. A posição e o tamanho do retângulo do display é determinado por 2 coordenadas (x,y) usadas em GET/PUT:

POSIÇÃO canto superior esquerdo = ponto inicial=(X1,Y1)
canto inferior direito = ponto de término=(X2,Y2)

TAMANHO largura = X2-X1
comprimento = Y2-Y1

Agora vejamos, como escolher o tamanho de cada dimensão do array da linha 25 do programa. O tamanho do array deve ser compatível com o retângulo:

- primeira dimensão do array = largura do retângulo
- segunda dimensão do array = comprimento do retângulo

Por exemplo, um retângulo de 40x20 requer um array de 40x20. Desde que um array tenha um elemento zero, o seguinte dimensionamento é adequado:

```
DIM V(39,19)
```

O tamanho do retângulo é limitado pela quantidade de memória disponível para ser utilizada pelo seu programa. Cada elemento do array ocupa 5 bytes da memória RAM disponível.

Caso você use o parâmetro G da instrução GET, você deve usar um dos parâmetros disponíveis na instrução PUT (PSET, PRESET, AND, OR, NOT), por outro lado, "resíduos" aparecerão quando da recolocação do retângulo na tela. Mas nem sempre isso é necessário. Em PMODE 0,1 ou 3 não há necessidade. Para PMODE 2 ou 4, use as opções.

Observe como o próximo programa utiliza GET e PUT para simular movimento. Para acelerar a ação, você pode au

mentar o incremento STEP. Preste particular atenção na maneira em que o programa utiliza as funções já vistas como CIRCLE, LINE, PAINT, etc.

```
 5 PCLEAR 4
10 DIM V(30,30)
15 PMODE 3,1
20 PCLS
25 SCREEN 1,1
30 CIRCLE (128,96),30
35 PAINT (128,95),2,4
40 PAINT (128,97),3,4
45 GET (98,81)-(128,111),V,G
50 PCLS
55 FOR I=150 TO 1 STEP-1
60 PUT (I,81-I/5)(I+60,111-I/5),V,PSET
65 NEXT I
70 GOTO 70
```

O único artifício real que fizemos, foi mudar a posição do retângulo PUT para o valor de I.

Exercício 27: Use GET e PUT para fazer uma nave espacial, atravessar os limites da tela de sua T.V.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XIV. TOQUE NOVAMENTE, VC50

A instrução PLAY permite que você não só toque, mas também componha. Antes de irmos mais além, devemos ressaltar que PLAY não é uma instrução gráfica. Consequentemente, você não terá que iniciar seu programa com PMODE, PCLS e SCREEN.

O formato de PLAY:

PLAY música

onde música é uma string que especifica a:

NOTA - uma letra de "A" até "G" ou um número de 1 a 12.

OITAVA - a letra "O" seguida de um número de 1 a 5. Se omitido, a oitava 2 é usada.

DURAÇÃO DA NOTA - a letra "L" seguida de um número de 1 a 255. Se omitido, o comprimento atual é usado.

ANDAMENTO ou TEMPO - a letra "T" seguida de um número de 1 a 255. Se omitido, o tempo 2 é usado.

VOLUME - a letra "V" seguida de um número de 1 a 31. Se omitido, volume 15 é usado.

DURAÇÃO DA PAUSA - a letra "P" seguida de um número de 1 a 255.

EXECUÇÃO DE SUB-STRING - substrings devem ser precedidos pelo prefixo X e seguidos de um ponto e vírgula (XA$;) como no comando DRAW, visto anteriormente.

Obviamente, você não têm musica sem notas musicais. PLAY permite não um, mas dois modos de você especificar a nota musical.

O primeiro (e provavelmente o mais fácil) modo de tocar a nota, é utilizando uma das notas musicais "A", "B",

"C",...."F" ou "G". Para indicar uma nota sustenida ou bemol, use "+" ou "#" para sustenidos e "-" para bemois.

Por exemplo, "A" representa A natural; "A#", A sustenido; "A-", A bemol. Entre com o seguinte comando para ver (ouvir) o que queremos dizer:

```
PLAY"A"  ENTER
```

Para ouvir a mudança que um sustenido e um bemol podem fazer, tecle:

```
PLAY"A;A#"  ENTER
PLAY"A-;A;A#;A;A-"  ENTER
```

Você pode fazer o mesmo com todas as sete notas (A-G) da escala.

Obs.: há somente um semi-tom separando B e C e E e F; então B#=C-=C e E#=F-=F

O segundo modo de especificar uma nota musical é pelo uso de um número entre 1 e 12 precedido pela letra N (N é opcional; se omitido, o número sozinho indicará a nota).

Os números de 1 a 12 representam cada nota de escala musical, incluindo todos os sustenidos e bemois. Esta é uma conotação mais resumida, mas é também mais difícil de ler.

Obs.: PLAY não reconhece a notação "B#" ou "C-". Use os números 1 e 12, respectivamente, ou substitua "B#" por C e "C-" por B. Você conseguirá a nota que quiser, já que B#=C e C-=B.

Digite o programa seguinte, que toca as notas depois que você entra com uma letra ou número e ENTER :

```
 5 CLS
10 INPUT "DIGITE UMA NOTA (A-G,#,-) ou (1-12) E
   <ENTER>";A$
20 PLAY A$
30 GOTO 5
```

Para ouvir a escala completa de 12 tons, digite o seguinte programa:

```
 5 CLS
10 FOR N=1 TO 12
15 PRINT "NOTA#";N
20 PLAY STR$(N)
30 NEXT N
```

Acrescente um atraso no programa, dessa forma você poderá comparar os números com as notas da escala (1 a 12, C a B).

```
25 FOR I= 1 TO 500:NEXT I
```

NÚMERO	NOTA
1	C
2	C#/D-
3	D/E-
4	E-/D#
5	E/F-
6	F/E#
7	F#/G-
8	G
9	G#/A-
10	A
11	A#/B-
12	B

Exercício 28: Modifique o programa anterior para que ele execute a escala descendente, em vez de ascendente.

FAÇA VOCÊ MESMO O SEU PROGRAMA

As notas do programa anterior eram semínimas, já que não especificamos a duração da nota (semi-breve, mínima e semínima, etc.).

Não é necessário cronometrar as notas para se ter certeza de que elas possuem 4 tempos, pois o computador possui um clock interno que controla isso.

Para escolher a duração da nota, use "L" seguido de um número entre 1 e 255. O número 1, por exemplo, indica uma semi-breve, 2 uma mínima, 4 uma semínima, 8 uma oitava, 16 uma 1/16, etc.

Realmente, você pode usar qualquer número de 1 a 255, mas quem já ouviu 1/15 de nota?

Vamos variar a duração das notas para produzir um rufar de tambor. Tecle:

```
PLAY "L2;A;L4;A;A;L2;A;A"  ENTER
```

TABELA DE DURAÇÃO DAS NOTAS

NÚMERO L	L1	L2	L3	L4 .. L8	L16...	L32 ...	L255
DURAÇÃO DA NOTA	semi-breve	mínima	3/4	semi mínima	1/8	1/16	1/32	1/255

O parâmetro "L" não precisa ser repetido para toda e cada nota que for tocada. PLAY usará o valor atual da nota até você lhe dar outro N. Foi por isso que não foi repetido o L4 no último exemplo.

MARCADA

Se você "lê" música, já sabe o que são "notas marcadas". O ponto ou marca, indica-lhe um aumento da duração da nota em metade do seu valor normal. Existem outros tipos de duração das notas, além de semi-breve, mínima, etc. Por exemplo, uma semínima marcada, equivale a uma nota 3/8. Você pode representar tais notas pela adição de um ponto (.) ou uma série deles (....) ao número L. Cada ponto, aumenta a duração da nota em metade do valor normal. Você poderá acrescentar quantos pontos desejar. Por exemplo:

L4.=1/4+1/8=3/8

Tente isso:

```
PLAY"L4.;A;L8;C;L4.;E;L8;C;E;C;E;C;L4;A"
```

OITAVA

Nossa única oitava (#2) soa agradável mas, variando o tempo da música tanto quanto o de um guisado irlandes, ela se tornará um pouco monótona quando tocada e tocada (como um piano com somente 12 teclas).

"O" (oitava) é o parâmetro de PLAY que permite selecionar outras oitavas. Para mudar a oitava, use "O" seguido por um número de 1 a 5 (qualquer número fora dessa faixa, resultará em uma mensagem de erro).

Seu VC50, automaticamente, usa a segunda oitava, a qual inclui C-central, se não for especificada a oitava. Vamos tocar uma simples escala de C (do):

```
PLAY"CDEFGABAGFEDCBA"
```

O que aconteceu? G é a mais alta nota na segunda oitava, então o computador alcança A, e começa novamente no início da oitava. Temos que tirar a segunda oitava. Tente isso:

```
PLAY"CDEFG;O3;ABAO2;FEDCBA"
```

VOLUME

Você poderá ajustar o volume de uma música, por exemplo, pelo ajuste da TV., mas o computador pode fazer isso, sem que você tenha que ficar diante da TV.

O parâmetro "V" seguido de um número entre 0 e 31 é responsável pelo ajuste de volume na instrução PLAY. Se omitido, é usado V15.

Obs.: o computador usará o valor atual de V até que seja mudado.

Ajuste normalmente o volume da TV e rode o seguinte programa:

```
 5 CLS
10 PLAY"V5;A;V10;A;V15;A;V20;A;V25;A;V30;A"
20 GOTO 10
```

Lembre-se, com (BREAK) você poderá parar o looping acima.

PAUSA

O último programa, talvez fosse mais fácil ouví-lo, se todas as notas não fossem tocadas juntas. Usaremos o parâmetro "P" seguido de um número entre 1 e 255, para ter alguns espaços entre as notas. Esses números correspondem aos usados com "L" (duração da nota) com uma importante diferença: os pontos não podem ser usados com "P". Para compensar, apenas tecle uma série de pausas. Por exemplo, para obter-se uma pausa 3/8 da duração de uma semibreve, digite P4P8.

Mude a linha 10 do programa anterior para:

```
10 PLAY"V5;A;P2;V10;A;P2;V15;A;P2;V2O;A;P2;V25;A;P2;V30;A;P2"
```

TEMPO (ANDAMENTO)

Você poderá aumentar ou diminuir o tempo com "T" seguido de um número entre 1 e 255. Se não especificado, o computador assume T2.

Comece aumentando o tempo do programa anterior:

```
10 PLAY"T1;V5;A;P2;V10;A;P2;V15;A;P2;V20;A;P2;V25;A;P2;V30;A;P2"
```

Agora acelere, alterando T1 para T15.

EXECUTANDO UMA SUB-STRING (X)

Da mesma forma que usamos o parâmetro "X" em DRAW, poderemos usá-lo em PLAY.

O parâmetro "X" tem a seguinte forma:

XA$;

onde A$ contém uma sequência normal de comandos e funções de PLAY. Lembre-se o ponto e vírgula após A$ é necessário.

Digite:

```
 5 CLS
10 A$="A;A#;A-;"
20 B$="O5;XA$;"
30 C$="Ol;XA$;XB$;"
40 PLAY C$
```

MAIS UMA NOTA (+,-,<,>)

Com "O" (oitava),"V" (volume), "T" (tempo) e "L"(duração da nota), você pode usar um dos seguintes sufixos ao invés de um número.(+,-,<,>).

SUFIXO	OPERAÇÃO
+	soma 1 ao valor atual
-	subtrai 1 ao valor atual
>	multiplica o valor atual por 2
<	divide o valor atual por 2

Ilustremos essa característica com um dos programas exemplos:

```
 5 CLS
10 PLAY"T2"
20 PLAY"A;A#;A-"
30 GOTO 20
```

Observe que foi estabelecido o valor T na linha 10. Rode o programa uma vez para ouví-lo. Nada mudou. Agora, coloque T+ na linha 20:

```
20 PLAY"T+;A;A#;A-"
```

O sinal (+) aumenta o valor de T cada vez que a li-

nha 20 é tocada. Agora altere para T- e rode o programa.

```
 5 CLS
10 PLAY"T255"
20 PLAY"T-;A;A#;A-"
30 GOTO 20
```

A multiplicação não é mais "rápida" que a soma?

Retorne na linha 10, o tempo para 2, e na linha 20 mude T- por T> e deixe rodar.

```
10 PLAY"T2"
20 PLAY"T>;A;A#;A-"
```

O valor de T inicial foi 2. Depois o computador multiplicou esse valor por 2, deu 4, 4x2 deu 8, 8x2 deu 16, e assim até chegar a 255.

Você poderá diminuir o tempo rapidamente também. Para isso use:

```
10 PLAY"T255"
20 PLAY"T<;A;A#;A-"
```

Lembre-se: você poderá fazer a mesma coisa com "L" para aumentar ou diminuir a duração da nota, "V" para aumentar ou diminuir o volume, e "O" para subir ou baixar a oitava.

A seguir mostraremos uma pequena música:

```
 5 CLS
10 A$="T5;C;E;F;L1;G;P4;L4;C;E;D;L1;G"
20 B$="P4;L4;C;E;F;L2;G;E;C;E;L1;D"
30 C$="P8;L4;E;E;D;L2.;C;L4;C;L2;E"
40 D$="L4;G;G;G;L1;F;L4;E;F"
50 E$="L2;G;E;L4;C;L8;D;D+;E;G;L4;A;L1;D3;C"
60 X$="XA$;XB$;XC$;XD$;XE$;"
70 PLAY X$
```

Agora você já pode tentar alguns arranjos musicais próprios.

XV. ALTERANDO LINHAS DE UM PROGRAMA

Através desse manual, você provavelmente gastou muito tempo re-escrevendo linhas de programa quando da alteração de alguma coisa.

Para contornarmos esta situação e agilizar mais as alterações, seu VC50 possui uma característica, a qual chamamos de "edição", onde você não somente poderá alterar uma linha do programa, mas também apagar, inserir, procurar, analisar e expandir ou cortar linhas de programa.

EDIT

A forma geral é EDIT número de linha (ENTER)

Vamos supor que você cometeu um erro ao digitar a linha 30 de um programa:

```
30 LINF (0,191)-(2550,0),PET
```

Para editar a linha 30, digite:

```
EDIT 30 (ENTER)
```

Na tela aparecerá:

```
30    <CURSOR>
```

Agora você poderá alterar a linha 30. Para posicionar o cursor para edição, pressione a tecla de espaços para mover para frente, e ⊖ para mover para trás.

Primeiro precisamos trocar a letra F em LINF por E. Para fazer isso deve-se:

- posicionar o cursor sobre a letra incorreta (F)
- pressionar C para indicar troca (change)
- teclar a letra correta (E)

e a mudança será feita. A linha então ficará:

30 LINE <CURSOR>

Para verificar toda a linha, tecle L, e a linha será listada na forma atual, pronta para ser editada.

Para alterar mais que um caracter no modo de edição:

- posicionar o cursor sobre a primeira linha a ser mudada.
- tecle o número especificando quantos caracteres serão alterados.
- pressione C para a mudança.
- tecle os novos caracteres.

Obs.: Uma vez teclado o número e pressionado C, você deve pressionar N teclas antes de sair do sub-comando. Não adianta pressionar (ENTER), (BREAK) ou (SHIFT) (↑) para sair do sub-comando. É preciso completar toda a mudança.

DELETE

Cuidamos do primeiro erro, mas ainda há um zero extra no segundo conjunto de parênteses. Ele precisa ser apagado e a primeira forma de fazê-lo é similar ao primeiro método de troca:

- posicione o cursor sobre o caracter errado.

- pressione D para apagar o caracter (DELETE).

e a operação está completa.

Por exemplo:

```
30 LINE(0,191)-(255 <CURSOR>
```

pressione D para apagar o zero. Nada parece ter ocorrido. Para confirmar se o caracter foi apagado, pressione L para listar a linha e continuar editando, ou pressione (ENTER) para finalizar a operação de edição.

Há uma outra maneira de se apagar um caracter que envolve a eliminação de um número especificado de caracteres:

- posicione o cursor sobre o primeiro caracter a ser apagado.
- tecle um número especificando quantos caracteres serão apagados.
- pressione D.

INSERIR

Até agora, a linha 30 está assim:

```
30 LINE(0,191)-(255,0),PET
```

Está quase correta, exceto pela falta do S no parâmetro final para que se torne PSET. Você deve:

- posicionar o cursor sobre o caracter onde a inserção é necessária.
- pressionar I para inserir.
- tecle os caracteres que deverão ser inseridos (no caso somente a letra S).

Há uma grande diferença entre inserir e as características de edição descritas anteriormente.

Depois que você fizer a inserção, você deverá "sair" do modo de inserção. Por exemplo, se você avançar o cursor (pressionando a tecla de espaços), depois de feita a inserção, você inserirá espaços. Para sair do modo de inserção, pressione (ENTER) para completar toda a operação de edição, ou (SHIFT) (↑) para abandonar a inserção, porém continuar a editar.

Imaginemos uma situação onde precisamos inserir algo no final de uma linha. Uma forma de fazê-lo é entrarmos em edição, mover o cursor até o final da linha, e então usar I para inserir. Mas existe no VC50 uma forma mais fácil de fazê-lo:

- pressione X
- tecle os caracteres adicionais
- (ENTER) para finalizar, ou (SHIFT) (↑) para continuar.

Por exemplo, entre no modo de edição para linha 30 que estamos usando:

```
30 <CURSOR>
```

Pressione X e o cursor imediatamente "correrá" para o final da linha, colocando-o no modo de inserção.

```
30 LINE(0,191)-(255,0),PSET <CURSOR>
```

Você agora está no modo de inserção, e poderá começar a acrescentar caracteres no final da linha:

```
30 LINE(0,191)-(255,0),PSET,B  <CURSOR>
```

e pressione (ENTER) para sair da edição, ou (SHIFT) (↑) para confirmar, caso não queira mais inserir.

HACK

A função denominada HACK (corte) permite a combinação de uma inserção e uma deleção. Assim, podemos alterar uma linha cortando seu final e inserindo novos caracteres. Para isso:

- posicione o cursor sobre o primeiro caracter que deve ser cortado.
- pressione H.
- tecle os novos caracteres.
- pressione (ENTER) para finalizar, ou (SHIFT) (↓) para continuar editando.

Suponha que na linha 30 se lê:

```
30 LINE(0,191)-PRESET,(255,0)
```

e você quer alterar tudo depois do hifen (-). Você deverá posicionar o cursor no P de PRESET:

```
30 LINE (0,191)-<CURSOR>
```

pressione H e tecle a nova informação:

```
30 LINE(0,191)-(255,0),PSET  <CURSOR>
```

pressione (ENTER) para completar a edição, ou (SHIFT) (↓) para continuar a editar.

PROCURA

Outra maneira de avançar o cursor rapidamente, é fazer o computador procurar na linha de programa, por um caracter especificado.

Imagine a linha 30 abaixo, se tivéssemos que apagar o caracter B sem ter que pressionar várias vezes a tecla de espaços.

```
30 LINE(0,191)-(255,0),PSET,B
```

Proceda:

- pressione S para a busca (SEARCH).
- digite o caracter a ser procurado (no caso B).

O cursor buscará na linha até encontrar o caracter especificado. Você então poderá apagá-lo, ou utilizar qualquer outra das características de edição.

Este procedimento deu certo com uma linha como a 30 a qual estudamos. Essa possui apenas um único erro de caracter , mas e se uma linha possui mais de um erro? Por exemplo: se você tivesse colocado indevidamente um parênteses extra no segundo grupo de parênteses:

```
30 LINE (0,191)-((255,0),PSET
```

Se você diz simplesmente ao computador para procurar um parênteses, ele posicionará o cursor no primeiro parênteses que encontrar.

Para evitarmos esse inconveniente faça:

- edite a linha 30.
- pressione o número 2 (significando o segundo) seguido por S
- pressione "(" (o caracter a ser procurado).

Teremos na tela o seguinte:

```
30 LINE(0,191) -  <CURSOR>
```

Você pode efetuar qualquer operação de edição. Neste caso, queremos eliminar um caracter . Se lembra como fazer isso? Ótimo, então faça!

KILL

KILL é quase o oposto de HACK. Ele lhe permite anular tudo até a n-ésima ocorrência de um caracter especificado. No modo de edição:

- tecle o número de ocorrência do primeiro caracter a ser preservado e não anulado.
- pressione K
- pressione a tecla do primeiro caracter a ser preservado.

Tendo-se a linha 30 como exemplo:

```
30 LINE (0,191)-(255,0),PSET
```

Suponhamos que, para exemplificar, queiramos anular dessa linha (0,191).

Primeiro tecle EDIT 30, leve o cursor até o primeiro parênteses, pressione 1 (para a primeira ocorrência), a seguir K. Então pressione (-) (o primeiro caracter a ser preservado), e finalmente (ENTER). A linha 30 ficará:

```
30 LINE - (255,0),PSET
```

MOVIMENTO DO CURSOR

Já foi mostrado como mover o CURSOR para frente e para trás numa velocidade relativamente lenta. Também como avançar rapidamente durante uma operação particular de edição, tal como procura S.

Algumas vezes, entretanto, você precisa mover o cursor rapidamente, mas não quer buscar, anular, apagar, ou ampliar a linha. Para fazer isso:

- tecle o número dos espaços que o cursor deve percorrer.

- pressione a tecla de espaços.

Tendo-se novamente a linha 30 como exemplo:

```
30 LINE (0,191)-(255,0),PSET
```

Edite a linha 30, e:

- tecle 5
- pressione a tecla de espaços.

Daí, tem-se:

```
30 LINE <CURSOR>
```

Então você poderá realizar as funções de edição.

Para retornar rapidamente, siga o mesmo processo, mas substitua a tecla de espaços por ⊖ .

```
30 LINE (0,191)-(255,0),PSET  <CURSOR>
```

Pressione 6 (o número de espaços para retornar) seguido de ⊖ e a tela mostrará:

```
30 LINE (0,191)-(255,0    <CURSOR>
```

Você pode, então, realizar qualquer das funções de edição.

Veja tabela a seguir.

CARACTERÍSTICAS DO COMANDO EDIT	
Sintaxe de comando	Operação desenvolvida
EDIT número da linha	Edita a linha especificada.
Ⓛ	Lista a linha que está sendo editada
Ⓒ novo caracter	Muda o caracter atual.
n Ⓒ novos caracteres	Muda os próximos n caracteres.
Ⓘ	Insere caracteres.
Ⓓ	Cancela o caracter atual.
n Ⓓ	Cancela n caracteres.
Ⓗ	Elimina a linha a partir da posição atual e permite inserção de novos caracteres.
Ⓧ	Extende a linha (o cursor é movido para o final da linha e a modalidade de inserção é ativada).
Ⓢ caracter	Pesquisa a primeira ocorrência do caracter especificado.
n Ⓢ caracter	Procura a enésima ocorrência do caracter especificado.
Ⓚ	Elimina (cancela) o resto da linha.
n Ⓚ	Elimina (cancela) até a enésima ocorrência do caracter especificado.
n ESPAÇO	Avança o cursor n espaços. Se n for omitido, o computador usará 1.
n ⊖	Retrocede o cursor n espaços. Se n for omitido, será usado 1.

DEL

Uma forma de você apagar uma linha de um programa é digitar o número da linha que se quer apagar e (ENTER).

Isso resolve para uma ou duas linhas, mas para 50 ou 60 linhas? Não, porque você terá que digitar todos os números de linhas e (ENTER) várias vezes.

O comando DEL permite apagar várias linhas de um programa de uma forma fácil e rápida.

Apagando todas as linhas do programa:

DEL- (ENTER) ou NEW (ENTER)

Obs.: não esqueça de incluir um hifen (-) depois do comando.

Apagando uma linha do programa:

DEL Número da linha (ENTER) ou número da linha (ENTER)

Apagando algumas linhas do programa:

DEL A-B

onde A é o número da primeira linha a ser apagada e B é a última.

DEL A-

onde A é o número da primeira linha a ser apagada. Como não foi especificada a última linha, o programa é apagado até o final.

DEL -B

onde B é a última linha a ser apagada. Como não foi especificada a primeira linha, o programa será apagado do início até B.

RENUM

Até aqui, pudemos alterar tudo numa linha de programa, exceto o número da linha.

RENUM permite mudar os números da linha em um limite especificado, como todos os números da linha que aparecem após GOTO, GOSUB, THEN, ON.....GOTO, e ON.....GOSUB.

RENUM nova linha, linha inicial, incremento

onde:

nova linha - especifica o novo número de linha da primeira linha a ser renumerada; se omitido, o número 10 é usado.

linha inicial - especifica o número de linha no programa original onde você quer começar a renumerar; se omitido, todo o programa será renumerado.

incremento - especifica o incremento a ser usado entre cada linha sucessiva renumerada; se omitido, 10 é usado.

Obs.: RENUM não re-ordena as linhas de um programa, mas somente renumera-as.

Como exemplo, usaremos um programa bastante conhecido:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 LINE (0,191)-(255,0),PSET
50 GOTO 50
```

Vamos fazer com que o programa comece na linha 100, com linhas "espaçadas" de 5 em 5? Tecle:

```
RENUM 100,5,5
```

A primeira nova linha do programa será de número 100. O processo de renumeração começará com a linha 5 PMODE 3,1 e cada número da linha seguinte será incrementada de 5. Liste o programa renumerado.

```
100 PMODE 3,1
105 PCLS
110 SCREEN 1,1
115 LINE (0,0)-(255,191),PSET
120 LINE (0,191)-(255,0),PSET
125 GOTO 125
```

Digite, agora, RENUM e (ENTER) . O computador renumerará o programa todo, com 10 como primeiro número de linha, e um incremento de 10.

Vejamos ainda, algumas variações para RENUM:

RENUM 1000,100 (ENTER) - começará pela linha 100, sendo a primeira linha 1000, com um incremento de 10.

RENUM 100,,100 (ENTER) - todo o programa será renumerado, começando por 100 e com um incremento de 100.

RENUM,,5 (ENTER) - todo o programa será renumerado, iniciando com 10 e um incremento de 5.

CONDIÇÕES DE ERRO PARA RENUM

Imagine um programa que tem as linhas numeradas 10, 20,30,... o comando RENUM 15,30 é ilegal, já que o resultado da linha 30 com a nova numeração seria 15. Desta forma, a linha 30 entraria antes da linha 20 no programa original. Nesse caso, um ? **FC*ERRO** será mostrado e o programa original deixará de ser renumerado.

RENUM não criará novas linhas de programa maiores que 63999.

Se for usado um número de linha não definido no desvio do programa original, RENUM apresentará:

ULxxxxINyyyy

onde xxxx é o número da linha referência original e yyyy é o novo número da linha contendo xxxx. Nesse caso, RENUM renumerará o programa, apesar dessa mensagem, e embora a linha de referência indefinida não seja numerada.

Exemplo:

```
10 REM MENSAGEM DE ERRO
20 GOTO 40
30 REM FIM

RENUM 100,10,10
UL 40 NA 110
OK
```

Liste o programa para ter uma melhor noção.
Observe que na linha 110 (nova) ou 20 (velha) pede-se um desvio para a linha 40 que não existe.

XVI. UM POUCO MAIS DE MATEMÁTICA

1. Raiz Quadrada (SQR)

SQR número

onde o número é uma expressão numérica nunca menor que zero, pois a linguagem BASIC não trabalha com números complexos.

```
Ex.: PRINT SQR(100)
     10
```

Exercício 29: Faça um pequeno programa que dê a raiz quadrada dos números de 10 em 10, de 100 a zero.

FAÇA VOCÊ MESMO O SEU PROGRAMA

2. Funções Trigonométricas

2.1. **SIN(X)** - fornece o seno de X, sendo X expresso em radianos.

2.2. **COS(X)** - fornece o cosseno de X, sendo X expresso em radianos.

2.3. **TAN(X)** - fornece a tangente de X, sendo X em radianos.

2.4. **ATN(X)** - fornece o valor do ângulo em radianos, cuja tangente é X, ou seja, fornece o arco tangente de X.

O computador trabalha com todos os ângulos em radianos. Uma conversão simples de graus em radianos e vice-versa, pode ser deste modo:

graus para radianos → graus/57.29577951
radianos para graus → radianos*57.29577951

Exemplos:

```
PRINT SIN(90/57.29577951)
1

PRINT COS(0)
1

PRINT ATN(1)*57.29577951
45
```

3. LOG(X)

Fornece o valor do logarítmo neperiano de base "e", de X.(e=2.718281828).

Exemplo:

```
PRINT LOG(2.718281828)
1
```

Para se ter o logarítmo de base 10, divida LOG(X) por LOG(10):

```
PRINT LOG(100)/LOG(10)     (log.10 de 100)
2
```

Obs.: O LOG de 1 em qualquer base é zero. A resposta que o computador fornece, deve-se a erros de precisão, que ocorrem em todos os computadores.

Exercício 30: Faça um programa para calcular o logarítmo na base 10 dos seguintes números:

a) 1 b) 10 c)100 d) 500 e)0.1 f) 1001

Sugestão: $\log_{10} X = \frac{\text{Log}_e X}{\text{Log}_e 10}$

FAÇA VOCÊ MESMO O SEU PROGRAMA

4. EXP(X)

Fornece o valor constante "e" elevada a X.

Exemplo:

```
PRINT EXP(2)
7.3890561  ("e" elevado ao quadrado)
```

Obs.: Esta função é o inverso do LOG, portanto, X=EXP(LOG(X)).

5. INT(X)

Fornece o maior valor inteiro contido em X.

Exemplo:

```
PRINT INT(3.1416)
3

PRINT INT(3.50)
3

PRINT INT(3.99)
3

PRINT INT(-2.58)
-3
```

Essa função pode ser muito útil quando trabalhando com números em notação científica, não se quer os algarismos a direita do ponto decimal.

6. FIX(X)

Fornece uma representação truncada de X. Todos os dígitos a direita do ponto decimal são cortados e o valor resultante será um número inteiro. Para X positivo, FIX(X) = INT(X); para X negativo, FIX(X)=INT(X)+1.

Exemplo:

```
PRINT FIX(-2.58)
-2
```

7. ABS(X)

Fornece o valor absoluto de X.

Exemplo:

```
PRINT ABS(1)
1
```

```
PRINT ABS(-1)
1
```

8. SGN(X)

É uma função que indica se X é maior, igual ou menor que zero. Quando X é maior que zero, SGN(X)=1; se X é igual a zero, SGN(X)=0; e se X é menor que zero, SGN(X)=-1.

Exemplo:

```
PRINT SGN(5)
1

PRINT SGN(0)
0

PRINT SGN(-5)
-1
```

9. DEF FNA(X)

A função DEF, em conjunto com FNA(X), serve para definir, em um programa, a equação de uma função A, de variável X. Daí, poderemos somente fazer referência à FNA(X), e o computador saberá qual a equação correspondente. Isso leva você a criar suas próprias funções matemáticas.

Por exemplo, uma operação matemática que tem que ser usada muitas vezes em um programa é a conversão de graus para radianos.

```
DEF FNR(X)=X/57.29577951
```

Com isso você pode economizar linhas de programação, sendo que 57.29577951 aparecerá somente uma vez.

Obs.: DEF FN somente pode ser usada em um programa, e nunca por comando direto.

Exercício 31: Faça um programa usando DEF FN, que eleve números ao cubo.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XVII. MANIPULANDO CADEIAS

1. STRING $

Não queremos cansá-lo com repetições, pois strings foram muito discutidos na primeira parte desse manual. Mas agora acrescentaremos a instrução STRING$ na função de facilitar a criação de gráficos, tabelas, ou qualquer display semelhante que crie uma cadeia de caracteres.

STRING$(comprimento,caracter)

onde comprimento é um número de 0 a 255. E caracter é cada expressão string para um caracter ou número no código ASCII, colocado entre aspas.

O número de caracteres mostrado depende do número que você especifica em comprimento. Para exemplificar, digite:

```
 5 CLS
10 X$=STRING$(13,"*")
15 PRINT@96,X$;"LINHAS";X$
20 FOR X=1 TO 1000:NEXT X
25 PMODE 3,1
30 PCLS
35 SCREEN 1,1
40 LINE(0,0)-(255,191),PSET
45 LINE(0,191)-(255,0),PSET
50 GOTO 50
```

Na linha 10 indicamos em X$ o valor STRING$(13,"*"), isto é, uma string de 13 asteríscos. Na linha 15 a palavra "linhas" será exibida com 13 asteríscos antes e 13 depois. Observe a economia que deixaríamos de fazer, se tivéssemos que digitar 13 asteríscos antes e 13 depois da palavra "linha".

Mude a linha 10 para:

```
10 X$=STRING$(13,42)
```

Rodando-se o programa, percebe-se o mesmo resultado. Desta vez, você não colocou (*), mas ao invés disso, você disse ao computador para mostrar o caracter que é representado no código ASCII, por 42. Consultando-se o apêndice temos que o código 42 representa um asterístico.

Exercício 32: Você já fez listas para controlar os trabalhos que você ou outras pessoas tem que fazer? Faça um programa que crie uma lista ou controle usando STRING$.

FAÇA VOCÊ MESMO O SEU PROGRAMA

2. INSTR

A função INSTR permite "olharmos" através de uma string, outra string.

A forma de INSTR é:

INSTR (posição,exame,alvo)

onde:

posição - especifica a posição no "exame", onde a busca deve começar; é um número de 0 a 255. Se omitido, a busca começa no primeiro caracter de "exame".

exame - é a string a ser analisada.

alvo - é a string "alvo" que você está procurando.

Vamos ver como INSTR funciona no seguinte programa:

```
 5 CLEAR 500
10 CLS
15 INPUT "TEXTO A SER EXAMINADO";E$
20 INPUT "PARTE DO TEXTO A SER PROCURADA";A$
25 C=0:P=1
30 F=INSTR(P,E$,A$)
35 IF F=0 THEN 60
40 C=C+1
45 PRINT LEFT$(E$,F-1)+STRING$(LEN(A$),CHR$(128)
   +RIGHT$(E$,LEN(E$)-F-LEN(A$)+1)
50 P=F+LEN(A$)
55 IF P <= LEN(E$)-LEN(A$)+1 THEN 30
60 PRINT "OCORREU ";C;" VEZES"
```

O seguinte texto é um exemplo. Você poderá colocar qualquer texto que desejar.

```
TEXTO A SER EXAMINADO? SAO MEGAB
YTE, VÓS QUE PROTEGEIS A TODOS O
S COMPUTADORES E  SEUS  MISEROS
USUARIOS,ESCUTAI A NOSSA PRECE E A
TENDEI AOS NOSSOS PEDIDOS.

PARTE DO TEXTO A SER PROCURADA? PR ENTER

SAO MEGABYTE, VOS QUE --OTEGEIS A
 TODOS OS.........AOS NOSSOS PED
IDOS.

SAO MEGABYTE, VOS QUE PROTEGEIS A
 TODOS OS .....ESCUTAI A NOSSA
 --ECE E ATENDEI AOS NOSSOS PEDI
DOS.

     OCORREU 2 VEZES
     OK.
```

O que aconteceu foi que quando a instrução INSTR localizou A$, o computador exibiu um A$(CHR$(128)). Então foi a busca da próxima ocorrência de A$ e fez o mesmo quando encontrou. Finalmente o computador mostrou quantas ocorrências de A$ foram encontradas em E$.

Esta forma de programação, poderá lhe ser muito útil quando do armazenamento de dados que contenham, por exemplo, uma lista de clientes com nomes e endereços, e queiramos fazer uma busca nesse arquivo.

3. MID$

A instrução MID$ lhe dá uma poderosa capacidade de editar strings, permitindo-lhe trocar uma parte de uma string com outra.

Sua forma é a seguinte:

MID$(string antiga, posição, **comprimento)=substituta**

onde:

string antiga - é a variável especificada da string que você quer mudar.

posição - é um número especificando a posição do primeiro caracter a ser mudado.

comprimento - é um número que indica a quantidade de caracteres a serem substituídos. Se omitido, todo o conteúdo da string substituta é utilizado.

substituta - é uma expressão string que troca a porção especificada da string antiga.

Obs.: se a "substituta" é mais curta do que a antiga, toda a string será usada e o resultado será sempre do mesmo comprimento da original.

Para entendermos melhor, rode o programa abaixo:

```
 5 CLS
10 A$="SISTEMA LINEAR"
20 MID$(A$,9)="SOLAR"
30 PRINT A$
```

Na linha 10, A$ estava indicando o valor SISTEMA LINEAR. A seguir, na linha 20, você disse ao computador para usar MID$ para trocar parte de A$ por SOLAR iniciando na posição 9. Observe a letra "R" repetida. Para evitar isso, agora acrescente a opção "comprimento" a linha 20.

```
20 MID$(A$,9,5)="SOLAR"
```

e observe que nada afetou o resultado. Agora altere o comprimento para 1:

```
20 MID$(A$,9,1)="LINEAR"
```

somente um caracter na string antiga foi substituído.

Para resolvermos o problema anterior faça:

```
20 MID$(A$,9)="SOLAR "
```

colocando o espaço em branco depois da palavra SOLAR, faz com que a nova string fique com 6 caracteres. Assim não teremos a última letra repetida (no caso a letra R).

Vamos agora rodar um programa que mostra como usar MID$ e INSTR juntas. Assim, INSTR busca os textos, e MID$ altera ou destrói.

Digite:

```
 5 CLS
10 INPUT "ENTRE COM MES E DIA (MM/DD)";X$
20 P=INSTR(X$,"/")
30 IF P=0 THEN 10
40 MID$(X$,P,1)="-"
50 PRINT X$
```

Obs.: Neste programa INSTR busca uma barra "/". Quando encontra uma, MID$ a substitui por um hifen "-".

Exercício 33: Faça de conta que você trabalha para a CIA. TELEFÔNICA nos velhos tempos, quando os centros telefônicos, estavam sendo trocados de caracteres "alpha" para numéricos. Faça um programa que use MID$ para substituir todas as letras para números, exceto as letras Q e Z.

Obs.: MID$ pode ser utilizado também para pesquisa.
Ex.:

```
 5 CLS
10 X$="EU QUERO SOMBRA E AGUA FRESCA"
20 A$=MID$ (X$,10,6)
30 PRINT A$
```

A$ será igual a sombra.

Se você quiser procurar uma palavra (de 4 letras por exemplo) num texto, você deve usar MID$ comprimento igual a 4 e início no começo do texto e em seguida ir incrementando a posição de 1 em 1 e comparando com a palavra. Isto é muito útil em programas conversacionais processamento de textos, etc.

FAÇA VOCÊ MESMO O SEU PROGRAMA

XVIII. INPUT/OUTPUT

As instruções INPUT/OUTPUT lhe permitem enviar dados do teclado para o computador, e do computador para a tela e impressora.

1. LINE INPUT

Sua forma é:

LINE INPUT "mensagem",variável string

onde:

mensagem - é a mensagem sugerida; é opcional, mas se usada deve ser colocada entre aspas.

variável string - é o nome designado para a linha que será colocada através do teclado.

A instrução LINE INPUT é semelhante a INPUT, exceto por:

- quando a instrução é executada e o computador está a espera dos dados, nenhuma interrogação é colocada na tela.
- cada LINE INPUT pode fixar um valor para apenas uma variável.
- vírgula e aspas serão aceitas como partes da string de entrada.
- espaços vazios no início não são ignorados, tornando-se parte da variável string.

LINE INPUT é uma maneira conveniente de colocar uma string sem se preocupar com entradas acidentais de delimitadores (vírgula, aspas, ponto e vírgula, etc.). Tudo é acei

to. De fato, algumas situações requerem que você coloque vírgulas, aspas, e espaços vazios como parte dos dados. A instrução LINE INPUT também é ótima nesses casos. Por exemplo:

```
LINE INPUT X$
```

lhe permite colocar X$ sem apresentar nenhuma sugestão.

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

apresenta uma mensagem sugerida e dados de entrada. Vírgulas não encerrarão a string de entrada. Observe que nós tivemos que colocar o ponto de interrogação e os espaços.

Exemplo:

```
10 INPUT "DIGITE SEU ENDERECO";E$
20 PRINT E$
```

rodando-se o programa, tem-se:

```
DIGITE SEU ENDERECO?   (CURSOR A ESPERA DE DADOS)
```

Digite:

```
Av. ESTADOS UNIDOS, 983.  (ENTER)
```

você obterá a mensagem ?EXTRA IGNORADO. Tente agora:

```
10 LINE INPUT "DIGITE SEU ENDERECO?";E$
20 PRINT E$
```

rodando-se o programa e entrando com os dados acima, você conseguirá obter o efeito desejado.

2. PRINT USING

A instrução PRINT USING permite que o computador exiba na tela ou impressora, valores strings e numéricos em um

formato escolhido. Isto torna-se de extrema utilidade, quando você está trabalhando com cheques, tabelas, gráficos ou qualquer situação que requer um formato de impressão específico.

A forma de PRINT USING é:

PRINT USING formato;itens

onde:

formato - é uma string que indica ao computador que formato usar ao imprimir cada um dos itens. Consiste em "especificadores de campo" e outros caracteres.

itens - são dados numéricos ou strings.

Obs.: PRINT USING não imprime automaticamente espaços vazios antes e depois dos números, exceto se indicado no formato.

Os seguintes especificadores de campo podem ser usados como parte de formato:

(#) este sinal especifica a posição de cada dígito localizado no número que você entrar. O número de # estabelece o campo numérico. Se o campo numérico for maior do que o número de dígitos do número, então as posições não utilizadas a esquerda do número serão preenchidas com espaços e aqueles a direita do ponto decimal com zeros. Se for muito pequeno para manter o número, esse será precedido do sinal "%".

(.) o ponto colocado no campo numérico, especificado pelo sinal #, fará o arredondamento dos dígitos da direita do ponto decimal, quando suprimidos.

Exemplos:

```
PRINT USING "######";66.2
   66
```

```
PRINT USING "#";66.2
   %66

PRINT USING "#.#";66.25
   %66.3

PRINT USING "##.#";58.76
   58.8

PRINT USING "##.##  "10.2,5.3,66.789,.234
   10.20      5.30    66.79   0.23
```

Observe que no último exemplo, 2 espaços foram colocados no formato da string após o último #, para separar os números quando esses forem impressos. Tente sem os espaços e observe a bagunça.

(,) a vírgula, quando colocada em qualquer posição entre o primeiro dígito e o ponto decimal, apresentará na tela uma vírgula a esquerda de cada terceiro dígito. A vírgula estabelece uma posição adicional no campo.

```
PRINT USING "#######,";123456
   123,456

PRINT USING "###,###,###";123456789
   123,456,789
```

(**) dois asteríscos colocados no início do campo farão com que, todas as posições não utilizadas a esquerda do ponto decimal, sejam preenchidas com asteríscos. Os 2 asteríscos estabelecerão mais duas posições no campo numérico.

```
PRINT USING "**#####";44.0
   ****44
```

($) um cifrão colocado no início do campo será colocado diante do número na impressão.

```
PRINT USING "$###.##";18.6735
   $ 18.67
```

($$) dois cifrões colocados no início do campo, atuarão como um $ flutuante, isto é, ocupará a primeira posição anterior ao número.

```
PRINT USING "$$###.##";18.6735
   $18.67
```

(**$) se estes 3 símbolos são usados no início do campo, então as posições livres a esquerda do número serão preenchidas pelo sinal *, e o símbolo $ será colocado na primeira posição, precedendo ao número.

```
PRINT USING "**$.##";8.333
   *$8.33
```

(+) quando um sinal (+) for colocado no início ou no final de um campo, será impresso um (+) para números positivos e um (-) para números negativos.

```
PRINT USING"+**#######";75200
   **+75200

PRINT USING"+###";-216
   -216
```

(-) quando um sinal (-) é colocado no final do campo, acarretará no aparecimento de um sinal (-) após os números negativos e um espaço após os positivos.

```
PRINT USING"####.#-";-8124.420
   8124.4-
```

(↑↑↑↑) faz com que o número seja impresso na forma exponencial.

```
PRINT USING"##.####↑↑↑↑";123456
   1.2346E+05
```

(!) faz com que somente o primeiro caracter de uma string escolhida seja impresso.

```
PRINT USING"!";"UNIVERSO"
   U
```

(% espaços %) os espaços são usados para especificações de um campo de string maior do que um caracter . O comprimento de um campo string será 2, mais o número de espaços entre os sinais (%).

```
PRINT USING"% %";"ABCDEFG"
ABCD

PRINT USING"%%";"ABCDEFG"
AB
```

O programa seguinte o ajudará a verificar os especificadores de campo:

```
 5 CLS
10 INPUT"DIGITE FORMATO, DEPOIS OS DADOS";C$,N
20 PRINT USING C$;N
30 FOR A=1 TO 1000:NEXT
40 GOTO 5
```

Caso você queira trabalhar com dados string, altere N na linha 10 e 20 para N$.

Rodando o programa:

```
DIGITE FORMATO, DEPOIS OS DADOS?
###.##
?? 25.192
   25.19
```

Execute o programa abaixo, para ver como os especificadores de campo funcionam:

```
  5 CLS
 10 A$="**$##,######.## CRUZEIROS"
 20 INPUT"QUAL E O SEU PRIMEIRO NOME";P$
 30 INPUT"QUAL E O SEU SEGUNDO NOME";S$
 40 INPUT"QUAL E O SEU ULTIMO NOME";U$
 50 INPUT"ESCREVA A QUANTIA A SER PAGA";P
 60 CLS
 70 PRINT "PAGUE A ORDEM DE ";
 80 PRINT USING"!";P$;".";S$;".";
 90 PRINT U$
100 PRINT:PRINT USING A$;P
110 GOTO 110
```

Exercício 34: Faça um programa que cria uma tabela mostrando sua receita e despesas mensais. Encontre apenas o total em dinheiro que entra e sai e faça o computador calcular o lucro líquido (mais ou menos).

Use STRING$ para organizar a tabela, mas certifique-se de que há flexibilidade suficiente para poder usar a tabela por vários meses, sem ter que mudar o programa inteiro.

FAÇA VOCÊ MESMO O SEU PROGRAMA

3. POS

Esta instrução lhe permite testar a posição do cursor na tela ou na impressora.

A forma de POS é:

POS (disposição do número)

onde disposição do número vale 0(tela) ou -2 (impressora).

POS "volta" um número, indicando a atual posição do cursor na tela ou na impressora.

```
PRINT TAB(8) POS(0)
```

"retorna" o número 8 para a coluna 8 na linha atual.

Uma maneira de usar POS é inutilizar a característica de retorno da tela ou da impressora. Isso será útil porque, pode-se completar palavras que foram quebradas ao meio. Por outro lado encurta o comprimento de linha. Rode o seguinte programa:

```
 5 CLS
10 A$=INKEY$
20 IF A$=""THEN 10
30 IF POS(0)> 22 THEN IF A$=CHR$(32) THEN
   A$=CHR$(13)
40 PRINT A$;
50 GOTO 10
```

Esse programa lhe permite usar o teclado como uma máquina de escrever. POS controlará o final da linha, assim nenhuma palavra será dividida. Na linha 30 do programa, o computador analisa se a posição atual do cursor é maior do que a posição de coluna 22 (a tela possui 32 colunas). Se o cursor ultrapassar a posição 22, o computador inicia uma nova linha na próxima vez que você digitar a tecla de espa-

ços (CHR$ (32)). Quando o computador decidir iniciar uma nova linha, ele o faz ordenando um retorno no carro da impressão (CHR$(13)). Deixando-se 10 espaços tem-se tamanho suficiente para palavras longas.

Exercício 35: Escreva um programa que use POS para espaçar as palavras uniformemente numa única linha.

FAÇA VOCÊ MESMO O SEU PROGRAMA

4. IMPRESSÃO

A instrução PRINT #-2 permite a saída de dados através de uma impressora. Pode ser usada como PRINT, exceto PRINT@.

```
PRINT #-2, USING"####.##";123.456
```

a tela não exibirá o valor, mas a impressora nos mostra:

```
123.46
```

Você poderá usar qualquer um dos especificadores de campo com PRINT #-2

5. TRON/TROFF

TRON (trace on) e TROFF (trace off) são importantes funções, que poderão ajudá-lo a seguir a execução das instruções de um programa, para análise de execução, localização e correção de erros.

TRON imprime cada número de linha do programa que é executado. O número aparece entre colchetes.

TROFF desativa.

Para exemplificar, usamos o programa que gera linhas na tela:

```
 5 PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 LINE (0,0)-(255,191),PSET
40 GOTO 40
```

Agora digite TRON (ENTER) . A seguir rode o programa. Na tela aparecerá:

```
5   10   20   30   40   40   40 ...
OK
```

Isso significa que o computador executou primeiro a linha 5, a seguir a 10, a 20 e assim por diante. Digite TROFF (ENTER) para desativar o comando.

TRON e TROFF podem ser usados em programas, a fim de sabermos quando uma linha específica foi executada.

Por exemplo:

```
100 TRON
110 X=X+1
120 TROFF
```

Assim poderemos saber se realmente a linha 110 está sendo executada. Depois que o programa foi corrigido, os comandos TRON e TROFF podem ser eliminados.

6. TIMER

Seu VC50 possui um temporizador e através da função TIMER, temos a possibilidade de marcarmos tempo em um programa. Esta mede tempo em 1/60 centésimos de segundo aproximadamente. No momento em que você liga o computador, o temporizador começa a contar de zero, indo até 65535. Daí retorna ao zero, e assim por diante. Isso leva cerca de 18 minutos.

Obs.: o contador "pára" durante operações com cassete e impressora.

Experimente com:

PRINT TIMER (ENTER)

e a tela mostrará um número de 0 a 65535.

Você também poderá iniciar o timer em qualquer tempo que especificar, por:

TIMER=número (ENTER)

onde número é uma expressão numérica entre 0 e 65535.

Para observarmos melhor TIMER, rode o seguinte programa. Ele testará sua habilidade matemática com uma simples tabuada. Será pedido uma operação, e tão logo você te-

nha acertado, o computador lhe dará o tempo em segundos que você levou para calcular e digitar (reflexos também serão testados).

```
 10 CLS
 20 TIMER=0
 30 A=RND(10)
 40 B=RND(10)
 50 PRINT"QUANTO VALE ";A;"X";B
 60 TIMER=0
 70 INPUT X
 80 IF X< >(A*B) THEN 70
 90 PRINT">>>  TEMPO <<<"
100 PRINT USING"######.##";TIMER/60
110 FOR C=1 TO 1000:NEXT
120 GOTO 10
```

XIX. CONSTANTES HEXADECIMAIS E OCTAIS

Números hexadecimais são quantidades representadas na base 16, compostas dos números de 0 a 9, e as letras de A a F. Constantes hexadecimais devem estar na faixa de 0 a FFFF, ou seja de 0 a 65535 na base 10.

Qualquer número precedido pelo símbolo &H é interpretado como um número hexadecimal. Por exemplo:

```
&HA010   &HFE   &HD1   &HC   &H5000
```

Números octais são quantidades representadas na base 8, compostas dos números de 0 a 7. As constantes octais devem estar na faixa de 0 a 177777. Elas serão armazenadas

como 2 bytes inteiros, correspondendo a faixa decimal de 0 a 65535.

Qualquer número precedido pelo símbolo &0 ou & é interpretado como uma constante octal. Por exemplo:

```
&070    &044    &025    &55    &1800
```

As constantes hexadecimais ou octais são convenientes em programas que indicam locação de memória e índices.

HEX$

Sempre que você trabalhar com programas em Linguagem de Máquina, achará conveniente usar números hexadecimais.

A função HEX$ torna fácil a conversão de números decimais em hexadecimais.

Sua forma é:

HEX$(número)

onde número é um número decimal de 0 a 65535.

O programa seguinte converterá em hexadecimal qualquer número que você solicitar, desde que menor que 65536.

```
 5 CLS
10 INPUT"DIGITE NUMERO NA BASE 10";N
20 PRINT"NUMERO NA BASE 16 VALE "HEX$(N)
30 GOTO 10
```

XX. USANDO SUB-ROTINAS EM LINGUAGEM DE MÁQUINA

Linguagem de máquina é a linguagem simples e limitada entendida pelos microprocessadores. Ela consiste de um conjunto de instruções básicas do microprocessador (no caso do Varix VC 50 o 6809 E).

Rotinas em linguagem de máquina são úteis por serem muito rápidas (1000 vezes mais que o basic) e podem fazer virtualmente qualquer coisa, inclusive controle direto do hardware do micro, controle de processos, etc.

São entretanto mais difíceis de serem escritas e requerem familiaridade com programação assembler.

Assembler é uma linguagem que, a partir dos mnemônicos (comandos) monta ou "assembler" a rotina ou programa em linguagem de máquina (em hexadecimal).

A Varix oferece um ótimo editor Assembler com ZBUG em ROM PACK e em disco, que permite a montagem e DEBUG de rotinas e programas em assembler.

Para mais informações, aconselhamos o livro:

6809 - ASSEMBLY LANGUAGE PROGRAMMING
Autor: LANCE A. LEVENTHAL
Editora: OSBORNE / McGRAW-HILL

Este livro esgota o assunto e pode ser encontrado na Varix.

PASSOS NECESSÁRIOS

1 - Protegendo a memória
2 - Armazenando a sub-rotina na RAM
3 - Dizendo ao basic onde está a sub-rotina

4 - Chamando a sub-rotina
5 - Retornando ao basic

Nós daremos um exemplo de programa com sub-rotina em L.M., mas não rode o mesmo até ler o capítulo inteiro.

O programa é dos mais simples. Ele retorna um ASC II code para as teclas do teclado ao contrário de uma string.

A sub-rotina possue algumas características não disponíveis em INKEY$ ou INPUT, pois esta retorna qualquer código, inclusive o da tecla (BREAK) e códigos de controle CTRL-A até CTRL-Z. Para obter um controle tecle (↑) e depois uma decla de (A) à (Z) , que gerará códigos ASCII de 1 à 26. Chamaremos esta sub-rotina de GETKEY.

PASSO I - PROTEGENDO A MEMÓRIA

Reserve espaço na memória com o comando CLEAR para a sub-rotina. O basic não ocupará este espaço.

```
10 CLEAR 25,4050
```

Reserva 25 bytes a partir de 4051 para armazenagem segura da sub-rotina.

PASSO II - ARMAZENANDO A SUB-ROTINA

Rotinas em L.M. devem ser carregadas da RAM via CLOADM ou LOADM, para cassete e disco respectivamente ou POKED na RAM. No nosso caso ela será parte integrante do programa basic e será POKED na memória com auxílio do comando DATA. Cada instrução é um código individual no comando DATA. Os números nas linhas DATA são os mesmos do programa listado posteriormente neste capítulo.

Observe o programa seguinte:

```
20 FOR I = 1 TO 28
30 READ B: POKE 4050 + I,B
40 NEXT I
50 DATA 173,159,160,0
60 DATA 39,250,129,10,38,12
70 DATA 173,159,160,0,39,250
75 DATA 129,65,45,2
80 DATA 128,64,31,137,79
90 DATA 126,180,244
```

PASSO III - DIZENDO AO BASIC ONDE ESTÁ A SUB-ROTINA

Nossa L.M. iniciou em 4051 decimal, que é igual a 0FD3

OF = 15 (decimal) mais significativo (MSB)
D3 = 211 (decimal) menos significativo (LSB)

Aqui uma linha de programação para realizar isto:

```
100 POKE 275,15: POKE 276,211
```

Os locais 275 e 276 são fixados pelo basic e formam o ponteiro da sub-rotina.

PASSO IV - CHAMANDO A SUB-ROTINA

Isto é feito com a função USR

```
110 A = USR(0)
```

No exemplo o zero (0) é um "argumento".

SINTAXE

USRn (argumento)

n = um dos USR disponíveis (0 à 9), se não especificado, será usado "0".

Argumento é um valor numérico ou expressão string que pode ser passado à sub-rotina pelo basic. No nosso caso "0" não será usado pela sub-rotina.

Obs.: Este argumento poderia ser obtido, chamando-se a sub-rotina da ROM, INTCNV, que retorna o valor inteiro no registro D do microprocessador. O endereço da INTCNV é o hexadecimal B3ED.

PASSO V - RETORNANDO AO BASIC

Se você não precisa retornar um valor ao basic, termine a sub-rotina com a instrução RTS. Se você quer retornar um valor inteiro com 2 BYTES, carregue o valor no registro D em MSB - LSB na sequência (mais significativa - menos significativa) e chame uma rotina especial da ROM que é a GIVABF, no endereço B4F4.

Após RTS, a sua referência será igual ao argumento nulo usado (no nosso caso o zero) ou ao valor carregado no registro D, se você chamar GIVABF.

No exemplo, A = 0, o que não afeta em nada a execução do programa.

Programa completo:

(Ver página seguinte)

```
10 CLEAR 25, 4050 'RESERVA MEMORIA
15 CLS
20 FOR I=1 TO 28 'ARMAZENA CADA BYTE DO CODIGO
   OBJETO
30 READ B: POKE 4050 + I, B
40 NEXT I
45 'AQUI ESTA O CODIGO OBJETO
50 DATA 173,159,160,0
60 DATA 39,250,129,10,38,12
70 DATA 173,159,160,0,39,250
75 DATA 129,65,45,2
80 DATA 128,64,31,137,79
90 DATA 126,180,244
99 'FALANDO AO BASIC ONDE ESTA A SUB-ROTINA
100 POKE 275,15: POKE 276,211
110 A = USR(0)'CHAMANDO A SUB-ROTINA
115 IF A = 13 THEN END
120 PRINT "CODIGO = ";A
130 GOTO 110
```

Rode o programa e experimente.

Quando você tecla (BREAK) você obtém o código 3. Para terminar tecle (ENTER) ou (↑) (M).

Para obter os códigos de 1 à 26 tecle (↑), solte e tecle de (A) à (Z).

Para uma variação no programa, mude a linha 120 para:

```
120 PRINT CHR$(A); REM DISPLAY O CARACTER
```

A maioria dos controles não são usados pelo micro. São controles para o sistema, terminais de vídeo, etc.

Tente o controle H (retorno de um espaço).

LISTA DA SUB-ROTINA

HEXADECIMAL CÓDIGO OBJETO		FONTE	COMENTÁRIOS
AD 9F A0 00	LOOP1	JSR [POLCAT]	Varre teclado
27 FA		BEQ LOOP1	LOOP (continua varrendo)
81 OA		CMPA =# 10	Controle ?
26 OC		BNE OUT	Saída se não
AD 9F A0 OO	LOOP2	JSR [POLCAT]	Obtém próxima tecla
27 FA		BEQ LOOP2	LOOP (se não continua esperando)
81 20		CMPA =# 65	é A - Z ?
2D 02		BLT OUT	Se < A,saia
80 40		SUBA =# 64	Converte p/controle A/Z
1F 89	OUT	TRF A,B	Apronta Byte de retorno
4F		CLRA	Zera MSB
7E B4 F4		JMP GIVABF	Retorna valor
	POLCAT	EQU 40960	
	GIVABF	EQU 46324	

O campo fonte são os mnemônicos que o "assembler" ou o programador, convertem em código hexadecimal (objeto). Para o Basic esses códigos (objeto) devem ser convertidos para decimal e usados no comando DATA.

COMPLEMENTAÇÕES:

1 - O comando CLEAR deve ser usado no início do programa.
2 - O comando DLOAD pode ser usado para transferir funções USR para outro computador.
3 - Uso do STACK (pilha). Uma função USR que requeira mais de 30 BYTES de área, deve prover sua própria área de STACK. Isto é feito salvando o ponteiro de STACK do basic antes de entrar

na função USR, setando um novo ponteiro e restaurando-o antes de retornar ao basic.

4 - DEFININDO FUNÇÕES USR

DEF USR

O endereço de entrada pode ser definido mais facilmente por esta função (basic extendido).

Sintaxe:

DEF USRn = endereço

n é um dígito de 0 à 9, se omitido zero é usado.

endereço - Deve ser entre 0 e 65535 (é usado aqui como se fosse o "argumento", já visto anteriormente).

5 - Retornando ao basic da função USR, deve-se restaurar o valor inicial do ponteiro de STACK e os valores dos registros A,B,X e CC (do micro 6809) devem ser preservados pela função USR.

6 - ARGUMENTOS

O registro A do microprocessador indica se o argumento usado é numérico ou não.

A = zero (numérico)

A ≠ zero (string)

Se é numérico o registro X contém um ponteiro para o "Acumulador de Ponto Flutuante" (FAC) que contém o argumento. Pode-se forçar o argumento para inteiro chamando - se INTCNV pela USR (INTCNV = X'B3ED'). Se o argumento é STRING ou está fora da faixa - 32768 a + 32768 a INTCNV causará um erro e o controle retorna ao basic.

INTCNV retorna o complemento a dois de um inteiro de 16 BITS no registro D.

Para argumentos numéricos ((A) = 0), FAC contém o expoente, FAC + 1 contém o MSB ... FAC+4 contém o LSB e FAC+5 contém o sinal de mantissa.

O expoente é um inteiro de oito bits, com 128 decimal adicionado a ele.

Um expoente zero significa que o número é zero e a mantissa é insignificante.

O BIT mais significativo da MSB indica o sinal: "0" para positiva e "1" para negativa.

Para argumentos não numéricos "strings" o registro X aponta para um arquivo descritivo de 5 BYTES. O primeiro BYTE indica o tamanho em caracteres da string, o segundo e o quinto são reservados e não disponíveis para uso. O terceiro e o quarto contém o endereço do 1º BYTE da string.

Um ponteiro para uma variável basic pode ser passado por uma função USR usando-se a função VARPTR .

Sintaxe:

VARPTR (nome da variável)

Exemplo:

```
X = USR 0 (VARPTR (A))
```

Passa um ponteiro para a variável "A".INTCNV pode ser chamada para obter o ponteiro.

O tipo de variável deve ser determinada pela função USR.

Para variáveis numéricas, o ponteiro aponta para um campo de 5 bytes em ponto flutuante, sendo que o primeiro byte é o expoente e os 4 bytes restantes formam a mantissa.

Valores em ponto flutuante são armazenados na tabela de variáveis num formato ligeiramente diferente dos que são armazenados em FAC. O bit mais significativo da mantissa é assumido como sendo um (1) (já que mantissas de pontos

flutuantes são sempre normalizadas), e a posição deste bit é usada para armazenar o sinal da mantissa. O número é positivo se o bit de sinal for um 0 e negativo se o bit de sinal for um 1.

Para variáveis string, o ponteiro aponta para um descritor de string de 5 bytes. O primeiro byte contém o comprimento do string e o terceiro e o quarto bytes contém o endereço do primeiro caracter no string. O segundo e o quinto bytes são reservados para o computador e não estão disponíveis para seu uso.

É possível passar um ponteiro de um vetor como o argumento de uma função USR. Deste modo a função USR pode ter acesso a qualquer dos elementos do vetor. Os valores dos elementos são armazenados na memória da seguinte maneira (ordenados da memória inferior à superior):

. **Valor** do primeiro elemento da última dimensão

. **Valor** do último elemento da última dimensão

. **Valor** do primeiro elemento da primeira dimensão

. **Valor** do último elemento da primeira dimensão

O comprimento de cada elemento é 5 bytes. **Valor** é o valor designado para as variáveis dentro do string.

Outro método de passar valores para uma função USR é colocar (usando POKE) os valores nas posições de memória reservadas para o uso da função USR.

RETORNO DE VALORES PARA BASIC

Uma função USR retorna sempre pelo menos um valor para BASIC: o valor da função. Se a rotina de USR não retorna explicitamente este valor, o valor retornado é o valor do argumento passado para a função USR. Geralmente, o tipo de valor retornado é o mesmo do argumento da função USR. Neste caso, a função USR retorna seu valor para o FAC ou para o descritor de string apontado por X, no mesmo formato do argumento.

Independentemente do tipo do argumento, um valor inteiro pode ser retornado através do carregamento de D com o complemento a dois de um inteiro de 16 bits e da chamada da rotina GIVABF do BASIC (GIVABF = X'B4F4') para retornar para BASIC.

Valores adicionais podem ser retornados para o BASIC através da modificação de valores de variáveis do BASIC. Deve-se ter extrema cautela ao se retornar valores string para BASIC. A advertência a seguir se aplica quando se retorna um string como o valor da função, bem como quando se modifica uma variável string do BASIC.

O comprimento de um string pode ser modificado com a mudança do byte de comprimento no descritor do string. Os strings podem ser encurtados desta forma, mas não se deve **nunca** tentar aumentar o comprimento de um string com a função USR. Se o comprimento do string a ser retornado para BASIC não for conhecido no momento da chamada, o string deve ser forçado a ter o comprimento máximo de 255 caracteres.

Por exemplo:

```
A$ = USR0 (STRING$(" ",255))
```

passa um string de 255 espaços em branco para a função USR. Esta função pode então colocar um string de até 255 caracteres dentro da memória apontada pelo descritor de string e encurtar esse string caso seja necessário.

O endereço inicial de um string pode ser modificado mudando-se o ponteiro de 2 bytes no descritor de string. Entretanto, o novo endereço inicial deve ser, geralmente, o de alguma locação de memória incluída no string original. Isto é, o endereço inicial pode ser mudado para qualquer um dos endereços OSA até OSA+OSL -1, onde OSA é o endereço inicial original e OSL é o comprimento do string original. Também é aceitável se permutar os endereços iniciais de dois strings. Deve-se tomar cuidado para que dois strings não se interceptem.

É possível para um descritor de string indicar o dado em string que está dentro do programa BASIC. Isto só pode ocorrer quando um string é definido como um string literal. Por exemplo, A$ = "ABC" e USR0("DEF") causam ambos a criação de descritores que apontam para dentro do texto do programa em BASIC.

Se uma função USR fosse usada para modificar tal string, o programa BASIC seria na verdade mudado. Este problema pode ser evitado acrescentando-se o string nulo("") a quaisquer literais string que venham a ser modificados por uma função USR. Por exemplo:

```
A$ = "ABC" + ""
```

irá fazer com que o string seja copiado para dentro do espaço de string, onde ele poderá ser modificado por uma função USR com toda a segurança.

Finalmente, é possível retornar valores adicionais para BASIC, através do armazenamento dos mesmos dentro das posições de memória alocadas para uso pela função USR. O programa BASIC pode então ter acesso a esses valores usando a função PEEK.

XXI. SUB-ROTINAS DA ROM DISPONÍVEIS PARA USO DO BASIC

Muitas rotinas do basic podem ser utilizadas pelo programa em basic do usuário ou como partes de programas e rotinas em linguagem de máquina. Estas rotinas normalmente são utilizadas pelos comandos normais do basic sem que o usuário se dê conta disto.

APRESENTAÇÃO:

NOME: Endereço de entrada (Nome só para referência)
Operação executada:
Condições de entrada (para L.M.)
Condições de saída (para L.M.)

BLKIN = (A006)

Lê um bloco do gravador cassete

Condições de entrada:

O cassete tem que estar ligado e em "bit sync" (veja CSRDON). CBMFAD contém o endereço do buffer.

Condições de saída:

BLKTYP, que está na posição 7C, contém o tipo de bloco que pode ser:

0 = início de arquivo (Header)

1 = dado

FF = fim de arquivo

BLKLEN, que está na posição 7D, contém o número de bytes que existem no bloco (0-255).

Z = 1,A = CSRERR = 0 (se não houve erro)

Z = 0,A = CSRERR = 1 (se ocorrer um erro de paridade (checksum))

Z = 0,A = CSRERR = 2 (se ocorrer um erro de memória)

(NOTA: CSREER = 81)

A menos que ocorra um erro de memória, X = CBUFAD + BLKEN. Se ocorrer um erro de memória, X aponta para o endereço onde ocorreu o erro.

As interrupções são mascaradas e os registradores U e Y são preservados enquanto que os demais são modificados.

BLKOUT = (A008)

Grava um bloco no gravador cassete.

Condições de entrada:

A fita cassete tem que estar girando na sua velocidade normal e o código hexadecimal 55 tem que estar gravado se este for o primeiro bloco a ser gravado.

CBUFAD, que está na posição 7E contém o endereço do buffer

BLKTYP, que está na posição 7C contém o tipo de bloco

BLKLEN, que está na posição 7D contém o número de bytes que existem no bloco (0-255)

Condições de saída:

As interrupções são mascaradas.

X =CBUFAD +BLKLEN e todos os registradores são modificados.

WRTLDR = (A00C)

Liga o gravador cassete e grava uma marca (LEADER).

Condições de entrada:

Não há.

Condições de saída:

Não há.

CHROUT = (A002)

Grava um caracter num periférico

CHROUT grava um caracter no periférico indicado pelo conteúdo do endereço hexadecimal 6F (DEVNUM).

DEVNUM = -2 (impressora)

DEVNUM = 0 (tela)

Condições de entrada:

Na entrada, o caracter a ser transferido está no registro A.

Condições de saída:

Todos os registradores com excessão do CC são mantidos.

CSRDON = (A004)

Liga o cassete.

CSRDON liga o cassete e vai ao "bit sync" para leitura.

Condições de entrada:

Não há.

Condições de saída:

FIRQ e IRO são mascarados e os registradores U e Y são preservados. Os demais são modificados.

JOYIN = (A00A)

Testa a porta do Joystick

JOYIN testa as portas dos quatro joysticks e armazena os seus valores de POTVAL até POTVAL+3.

JOYSTIK ESQUERDO	
Cima/baixo	15A
Direita/Esquerda	15B

JOYSTICK DIREITO	
Cima/Baixo	15C
Direita/Esquerda	15D

Para CIMA/BAIXO, o menor valor é CIMA.

Para DIREITA/ESQUERDA, o menor valor é ESQUERDA.

Condições de entrada:

Não há.

Condições de saída:

O registrador Y é mantido. Os demais são alterados.

POLCAT = (A000)

Pesquisa o teclado

Condições de entrada:

Não há.

Condições de saída:

Z = 1,A = 0 (nenhuma tecla foi pressionada)

Z = 0,A = Código da tecla (se foi pressionada uma tecla).

O conteúdo dos registradores B e X são preservados. Os demais são modificados.

APÊNDICE

EXERCÍCIOS RESOLVIDOS

- EXERCÍCIO 1

```
PRINT 30-(9-(8-(7-6)))
```

- EXERCÍCIO 2

```
 5 PRINT"SOMA DE DOIS NUMEROS SUBSEQUENTES"
10 INPUT"DIGITE UM NUMERO";X
20 Y=X+1
30 PRINT"SOMA="X+Y
```

- EXERCÍCIO 3

```
10 PRINT "PARA MUDAR MINHA COR"
20 PRINT "ESCOLHA UM NUMERO DE 0 A 8"
30 INPUT T
40 CLS(T)
50 GOTO 10
```

- EXERCÍCIO 4

```
10 FOR X=1 TO 10
20 PRINT "9*"X"="9*X
30 NEXT X
```

- EXERCÍCIO 5

```
10 FOR C=0 TO 8
20 CLS(C)
30 FOR X=1 TO 460
40 NEXT X
50 NEXT C
```

APÊNDICE

- EXERCÍCIO 6

```
10 PRINT"ESCOLHA O TIRO DE 1 A 10"
20 INPUT X
30 IF X=RND(10) THEN 70
40 SOUND 200,1
50 PRINT"--CLICK--"
60 GOTO 10
70 SOUND 100,1
80 PRINT "--BANG--VOCE ESTA MORTO"
```

- EXERCÍCIO 7

```
  5 CLS(0)
 10 FOR H=15 TO 48
 20 SET(H,5,5)
 30 SET(H,20,5)
 40 NEXT H
 50 FOR V=5 TO 20
 60 SET (15,V,5)
 70 SET(48,V,5)
 80 NEXT V
 90 SET(32,13,8)
100 FOR H=28 TO 36
110 SET(H,16,4)
120 NEXT H
130 SET(25,10,3)
140 SET(38,10,3)
150 RESET (38,10)
160 GOTO 140
```

- EXERCÍCIO 8

```
  10 INPUT"DIGITE UM NUMERO";N
  20 INPUT"DIGITE A POTENCIA QUE VOCE QUER ELEVA-LO";P
  30 GOSUB 2000
  40 PRINT N"ELEVADO A POTENCIA"P"E"E
  50 GOTO 10
2000 E=1
2010 FOR X=1 TO P
2020 E=E*N
2030 NEXT X
2040 IF P=0 THEN E=1
2050 RETURN
```

- EXERCÍCIO 9

```
10 A$=INKEY$
20 IF A$<>""GOTO 50
40 GOTO 10
50 PRINT"A TECLA QUE VOCE PRESSIONOU  E "A$
60 GOTO 10
```

- EXERCÍCIO 10

```
 5 CLS
10 DATA 33,12,42,13,15,23
20 DATA 25,30,33,27,14,8
30 DIM E(12)
40 FOR X=1 TO 12
50 READ E(X)
60 NEXT X
70 INPUT "ITEM NO.";N
75 IF N>12 THEN 70
80 PRINT "O ESTOQUE DO ITEM"N"E"E(N)
90 GOTO 70
```

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

- EXERCÍCIO 11

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 PSET(10,20,8)
40 PRESET(10,20,8)
50 GOTO 5
```

- EXERCÍCIO 12

```
10 PMODE 1,1
20 PCLS
30 SCREEN 1,1
40 X=RND(256)-1
50 Y=RND(196)-1
60 C=RND(9)-1
70 PSET(X,Y,C)
80 GOTO 40
```

- EXERCÍCIO 13

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
25 LINE(0,0)-(255,191),PSET
30 LINE(0,191)-(255,0),PSET
35 LINE(0,0)-(255,191),PSET,B
40 GOTO 40
```

- EXERCÍCIO 14

```
 5 PMODE 1,1
10 PCLS
20 SCREEN 1,1
30 LINE (72,168)-(200,72),PSET,B
40 LINE (72,72)-(136,36),PSET
```

(Continua)

EXERCÍCIO 14 (Cont.)

```
45 LINE (200,72)-(136,36),PSET
50 LINE (120,168)-(152,100),PSET,B
55 LINE (152,60)-(168,36),PSET,BF
60 LINE (165,128)-(191,100),PSET
65 LINE (178,128)-(178,100),PSET
70 LINE (165,114)-(191,114),PSET
75 LINE (85,128)-(111,100),PSET,B
80 LINE (85,114)-(111,114),PSET
85 LINE (98,100)-(98,128),PSET
90 GOTO 90
```

- EXERCÍCIO 15

```
PMODE 2,(1-3)
PMODE 1,(1-3)
PMODE 0,(1-4)
```

- EXERCÍCIO 16

```
22 PCOPY 4 TO 3
32 PCOPY 3 TO 2
42 PCOPY 2 TO 1
```

E elimine as linhas 11,21 e 31.

- EXERCÍCIO 17

```
 10 PCLEAR 8
 20 PMODE 4,1
 25 PCLS
 30 SCREEN 1,1
 40 LINE (0,0)-(255,191),PSET
 45 FOR Y=1 TO 20 :NEXT Y
 50 PMODE 4,2
 55 SCREEN 1,0
 60 LINE (0,0)-(255,191),PSET
```

(Continua)

EXERCÍCIO 17 (Cont.)

```
 65 FOR Z=1 TO 20 :NEXT Z
 70 PMODE 0,3
 75 SCREEN 1,1
 80 LINE (0,0)-(255,191),PSET
 85 FOR A=1 TO 20 :NEXT A
 90 PMODE 1,4
 95 SCREEN 1,1
 96 PCLS
100 LINE (0,0)-(255,191),PSET
105 FOR R=1 TO 20:NEXT R
110 GOTO 20
```

- EXERCÍCIO 18

```
  1 Y=-1
  5 CLS
 10 PRINT @194,"VOCE QUER VER UM QUADRADO?"
 20 FOR X=1 TO 1000:NEXT X
 30 PMODE 1,1
 35 PCLS
 40 SCREEN 1,Y+1
 60 LINE (75,150)-(150,75),PSET,B
 70 FOR X=1 TO 1000:NEXT X
 75 Y=-Y
 80 GOTO 5
```

- EXERCÍCIO 19

```
 10 PMODE 4,1
 20 PCLS
 30 SCREEN 1,0
 40 FOR RAIO =1 TO 100 STEP 10
 50 CIRCLE (128,96),RAIO
 60 NEXT RAIO
 70 GOTO 70
```

- EXERCÍCIO 20

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,0
30 CIRCLE(200,40),30,,1,.13,.63
40 CIRCLE(230,10),52,,1,.29,.48
50 GOTO 50
```

- EXERCÍCIO 21

```
 5 PMODE 1,1
10 PCLS
15 SCREEN 1,0
20 PCLS 3
25 COLOR 1,0
30 CIRCLE (200,30),15
35 PAINT (200,30),2,1
40 LINE (100,185)-(180,125),PSET,B
45 LINE -(140,90),PSET
50 LINE -(100,125),PSET
55 PAINT (135,115),4,1
60 LINE (110,160)-(125,130),PSET,B
65 LINE (155,160)-(170,130),PSET,B
70 PSET (134,157,1)
75 PAINT (120,180),0,1
80 LINE (130,130)-(149,185),PSET,B
85 LINE (101,135)-(41,185),PSET,B
90 LINE (91,140)-(51,185),PSET,B
100 PAINT (89,183),4,1
105 FOR X = 1 TO 500: NEXT X
110 PAINT (89,183),2,1
115 FOR X = 1 TO 500: NEXT X
120 PAINT (89,155),4,1
140 GOTO 110
```

- EXERCÍCIO 22

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,0
30 DRAW"BM68,116;E20;BE20;E20;F20;BF20;F20;L40;BL40;L40;
   BU40;R40;BR40;R40;G20;BG20;G20;H20;BH20;H20;BM128,96;
   NU40;ND40;NE20;NF20;NG20;NH20;NL40;R40;"
40 GOTO 40
```

Obs.: A estrela que você criou não é tão bonita quanto esta, já que você ainda não sabe usar os parâmetros B e N. Logo você vai aprender como utilizá-los.

- EXERCÍCIO 23

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
25 DRAW"BM40,80;U40;R40;D40;L40"
30 DRAW"BM+20,20;U40;R40;D40;L40"
40 LINE(60,100)-(40,80),PSET
50 LINE(60,60)-(40,40),PSET
60 LINE(100,60)-(80,40),PSET
70 LINE(100,100)-(80,80),PSET
80 GOTO 80
```

- EXERCÍCIO 24

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,1
25 DRAW"BM50,50L30D30R30D30L30"
30 DRAW"BM90,50D60R30U60"
40 DRAW"BM160,50D60R30BU60L30D30R30"
50 GOTO 50
```

- EXERCÍCIO 25

```
 5 PMODE 4,1
10 PCLS
20 SCREEN 1,0
30 DRAW"BM98,96;NU80;NE56;NR80;NF56;ND80;NG56;NL80;NH56"
40 CIRCLE(98,96),80,1,1,.125,1
50 CIRCLE(135,110),80,1,1,1,.125
60 LINE(135,110)-(190,167),PSET
70 LINE(135,110)-(213,110),PSET
80 GOTO 80
```

- EXERCÍCIO 26

```
 5 PMODE 3,1
10 PCLS
15 SCREEN 1,0
20 DRAW"BM59,170;U80;NG30;E80;F80;NF30;D80;L50;U70;L50;
   D70;L60"
25 LINE (50,170)-(170,170),PSET
30 LINE (110,170)-(160,170),PSET
35 FOR X = 1 TO 500: NEXT X
40 LINE(110,170)-(160,170),PRESET
45 LINE (120,180)-(120,110),PSET
50 LINE (160,100)-(125,110),PSET
55 LINE (160,170)-(125,180),PSET
60 LINE (120,180)-(120,110),PRESET
65 LINE (160,100)-(125,110),PRESET
70 LINE (160,170)-(125,180),PRESET
75 DRAW "BM110,170;BU70;BR50;G25;D70;E25"
80 CIRCLE (130,125),10,,1,.135,.9
85 DRAW"BM130,130;D15;D15;G10;E10;U15;L10"
90 LINE (120,145)-(120,135),PSET
91 FOR X = 1 TO 60: NEXT X
95 LINE (120,145)-(120,135),PRESET
```

(Continua)

EXERCÍCIO 26 (Cont.)

```
 96 FOR X = 1 TO 120: NEXT X
100 LINE (120,145)-(110,145),PSET
101 FOR X = 1 TO 60: NEXT X
105 LINE (120,145)-(110,145),PRESET
106 FOR X = 1 TO 60: NEXT X
110 LINE (120,145)-(120,135),PSET
120 FOR X = 1 TO 120: NEXT X
121 CIRCLE (130,125),10,1
122 DRAW "BM130,130;C1;D30;G10;E10;U15;L10"
125 DRAW "BM110,170;BU70;BR50;C1;G25;D70;E25;"
130 COLOR 4,1
135 LINE (120,180)-(120,110),PSET
140 LINE (160,100)-(125,110),PSET
145 LINE (160,170)-(125,180),PSET
150 LINE (120,180)-(120,110),PRESET
155 LINE (160,100)-(125,110),PRESET
160 LINE (160,170)-(125,180),PRESET
165 LINE (110,170)-(160,170),PSET
170 FOR X = 1 TO 500: NEXT X
175 GOTO 20
```

- EXERCÍCIO 27

```
 5 PCLEAR 4
10 PMODE 4,1
15 PCLS
20 SCREEN 1,1
25 DIM V(35,35)
30 X=10:Y=10
35 DRAW"BM10,10;S2;H10;R15;F10;R20;F10;G10;L20;G10;L15;
   E10;U20;D4;NL8;D4;NL12;D4NL16;D4;NL12;D4;NL8"
40 GET (X-X,Y-Y)-(X*3.5,Y*3.5),V,G
45 A$=INKEY$:IF A$="" THEN 45'PRESSIONE QUALQUER TECLA
   P/COMEÇAR
```

(Continua)

EXERCÍCIO 27 (Cont.)

```
50 PCLS
55 FOR A=10 TO 200 STEP 5
60 PUT (X+A,Y)-(X+A+35,Y+35),V,PSET
65 NEXT A
70 PCLS:GOTO 55
```

- EXERCÍCIO 28

```
 5 CLS
10 FOR N=12 TO 1 STEP-1
15 PRINT "NOTA#";N
20 PLAY STR$(N)
25 FOR I=1 TO 500:NEXT I
30 NEXT N
```

- EXERCÍCIO 29

```
 5 CLS
10 FOR NUM=100 TO 0 STEP -10
20 PRINT SQR(NUM)
30 NEXT NUM
```

- EXERCÍCIO 30

```
 5 CLS
10 INPUT "QUAL NUMERO ";NUM
15 X=LOG(NUM)/LOG(10)
20 PRINT "O LOG NA BASE 10 DE" NUM "E" X
30 GOTO 10
```

a) 7.00890077 E -11
b) 1
c) 2
d) 2.69897001
e) -1
f) 3.00043408

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

- EXERCÍCIO 31

```
 5 CLS
10 DEF FNC(X)=X↑3
20 INPUT" QUAL O NUMERO QUE VOCE QUER ELEVAR AO CUBO";X
30 X=FNC(X)
40 PRINT X
50 FOR A=1 TO 300 : NEXT A
60 GOTO 5
```

- EXERCÍCIO 32

```
 5 CLS
10 X$=STRING$(30,"-")
20 FOR X=64 TO 416 STEP 64
30 PRINT @X,X$
40 PRINT @97,"JOAO"
41 PRINT @161,"MARIA"
42 PRINT @225,"PEDRO"
43 PRINT @289,"MARTA"
51 PRINT @38,"MAT."
52 PRINT @45,"PORT."
53 PRINT @53,"INGLES"
60 PRINT @103,"X"
61 PRINT @175,"X"
62 PRINT @231,"X"
63 PRINT @311,"X"
70 NEXT X
80 GOTO 80
```

- EXERCÍCIO 33

```
 5 CLS
10 DIM TBL$(26)
20 FOR I=0 TO 25
30 READ TBL$(I):NEXT I
```

(Continua)

EXERCÍCIO 33 (Cont.)

```
 40 PRINT"FORNECA ONUMERO DO TELEFONE NA FORMA ANTIGA"
 50 INPUT N$
 60 IF N$="" THEN 40
 70 FOR I=1 TO LEN(N$)
 80 C$=MID$(N$,I,1)
 90 IF C$<"A" OR C$>"Z" THEN 120
100 C$=TB$(ASC(C$)-65)
110 MID$(N$,I)=C$
120 NEXT I
130 PRINT"ESTA E A NOVA FORMA=";N$
140 REM A B C D E F
150 DATA"2","2","2","3","3","3"
160 REM G H I J K L
170 DATA"4,"4","4","5","5","5"
180 REM M N O P Q R
190 DATA"6","6","6","7","Q","7"
200 REM S T U V W X
210 DATA"7","8","8","8","9","9"
220 REM Y Z
230 DATA"9","Z"
```

Obs.: Repare que as letras Q e Z não foram transformadas para números

- EXERCÍCIO 34

```
  5 CLS
 10 INPUT "RECEITA"; R
 15 INPUT "DESPESA"; D
 20 N = R-D ' LUCRO OU PERDA
 25 A$ = "$$#####.##"
 30 B$ = "$$#####.##"
 35 C$ = "+$$#####.##"
```

(Continua)

EXERCÍCIO 34 (Cont.)

```
40 CLS: PRINT @33, "RELATORIO ECONOMICO MENSAL"
45 PRINT @96, STRING$ (32,"-")
50 PRINT @160, "RECEITA"
55 PRINT @256, "DESPESAS"
60 PRINT @352, "TOTAL (+) OU (-)"
65 PRINT @340, STRING$(10,"-")
70 PRINT @180, USING A$; R
75 PRINT @276, USING B$; D
80 PRINT @371, USING C$; N
90 GOTO 90
```

- EXERCÍCIO 35

```
 5 CLS
10 PRINT "ISTO" TAB(POS(0)+4) "E";
20 PRINT TAB(POS(0)+4) "ESPACAMENTO" TAB(POS(0)+4)
   "UNIFORME"
```

PROGRAMAS SIMPLES

- PROGRAMA Nº 1

```
  1' **ARMA ESPACIAL**
  2'
  10 CLEAR 1000
  20 FOR Y=0 TO 1
  30 C=(Y+1)*16
  40 S$(Y)=CHR$(131+C)+CHR$(139+C)+CHR$(130+C)
  50 S2$(Y)=CHR$(128+C)+CHR$(136+C)
  60 NEXT Y
 100 FOR Y=0 TO 1
 105 C=JOYSTK (0)
 110 A(Y)=JOYSTK(0+Y*2)
 120 B(Y)=JOYSTK(1+(Y*2))
 130 IF A(Y)>59 THEN A(Y) = 59
 140 B(Y)=INT(B(Y)/4)*4
 150 L(Y)=B(Y)*8+INT(A(Y)/2)
 160 IF L(Y)>=480 THEN L(Y)=L(Y)-32
 170 NEXT Y
 180 CLS(0)
 190 FOR Y=0 TO 1
 200 PRINT @L(Y), S$(Y);
 210 PRINT @L(Y)+32, S2$(Y);
 220 NEXT Y
 500 P=PEEK(65280)
 510 IF P=125 OR P=253 THEN GOSUB 1000
 530 GOTO 100
 800 REM
 900 REM ROTINA DE TIRO
1000 V1=INT(B(1)/2)+1
```

(Cont.)

```
1010 H1=A(1)+2
1020 IF A(1) > A(0) THEN 1100
1030 FOR H=H1+3 TO 63
1040 IF POINT (H,V1)=2 THEN SOUND 100,2
1050 SET(H,V1,4)
1060 IF H <= H1+4 THEN 1080
1070 RESET (H-2,V1)
1080 NEXT H
1090 RETURN
1100 FOR H=H1 TO 4 STEP -1
1110 IF H=H1 THEN 1160
1120 IF POINT (H-4,V1)=2 THEN SOUND 100,2
1130 SET(H-4,V1,4)
1140 IF H >= H1-2 THEN 1160
1150 RESET(H-2,V1)
1160 NEXT H
1170 RETURN
```

- PROGRAMA Nº 2

```
  1' **BOLA SALTITANTE**
  2'
   5 CLEAR 12
   8 INPUT "COR DO FUNDO (1-8)";C
   9 CLS(C)
  10 X=13: Y=13
  15 XM=20: YM=15
 400 F=0
 410 XT=X: YT=Y
 420 X=X+XM: Y=Y+YM
 430 TX=X: TY=Y: T1=XM: T2=YM
 440 GOSUB 1000
```

(Cont.)

```
 450 X=TX: Y=TY: XM=T1: YM=T2
 455 H=INT (XT/2)*2: V=INT(YT/2)*2
 460 SET (H,V,C): SET(H+1,V,C)
 462 SET(H,V+1,C): SET(H+1,V+1,C)
 470 RESET(X,Y)
 480 GOTO 400
 499 REM
1000 REM VERIFICACAO DOS LIMITES
1010 IF TX>63 THEN TX=63: T1=-T1
1020 IF TX< 0 THEN TX= 0: T1=-T1
1030 IF TY>31 THEN TY=31: T2=-T2
1040 IF TY< 0 THEN TY= 0: T2=-T2
1099 RETURN
```

- PROGRAMA Nº 3

```
 1' **BLACKJACK**
 2'
 5  REM CONSTRUCAO DOS VETORES
 7  DIM S$(5), N$(13), D(52), P(5), C(5)
 10 DATA 16, 32, 48, 96, 1
 20 DATA **AS**, *DOIS*, *TRES*, QUATRO, CINCO*,
    *SEIS*, *SETE*, *OITO*, *NOVE*, *DEZ**, VALETE,
    *DAMA*, *REI**
 30 FOR X=1 TO 5: READ S:S$(X)=CHR$(143+S): NEXT X
 40 FOR X=1 TO 13: READ N$: N$(X)=N$: NEXT X
 45 CLS(6)
 46 PT=0: CT=0
 47 FOR X=1 TO 5: P(X)=0: C(X)=0: NEXT
 50 FOR X=1 TO 52: D(X)=X: NEXT X
 60 FOR X=1 TO  5: GOSUB 1000: P(X)=Z: NEXT X
 70 FOR X=1 TO 3: GOSUB 1000: C(X)=Z: NEXT X
```

(Cont.)

```
 72 REM
 75 REM IMPRESSAO DA 'MAO' DO JOGADOR
 80 L=257
 90 FOR M=1 TO 2: C=P(M): GOSUB 500: PT=PT+T: NEXT
100 FOR M=1 TO 3: S=5: GOSUB 2000: NEXT
102 REM
105 REM IMPRESSAO DA 'MAO' DO COMPUTADOR
110 L=10
120 S=5: GOSUB 2000
130 C=C(2): GOSUB 500: CT=CT+T
150 PRINT @6, "'MAO' DO COMPUTADOR"
160 PRINT @267, "SUA 'MAO'"
200 L=269: K=3
205 PRINT @231, "OUTRA CARTA(S/N)?";
210 R$=INKEY$: IF R$="" THEN 210
220 IF R$="N" THEN 255
230 C=P(K): GOSUB 500
240 PT=PT+T
242 FOR X=1 TO K
244 IF PT>21 AND (P(X)-1)/13=INT((P(X)-1)/13) THEN
    PT=PT-10
246 NEXT X
247 IF PT>21 THEN PRINT  488, "VOCE ESTOUROU!!!";:
    GOTO 400
250 K=K+1: IF K<6 THEN 205
255 L=10
260 C=C(1): GOSUB 500: CT=CT+T
360 IF PT <= CT THEN 380
370 PRINT @484, "PARABENS, VENCEDOR!";
375 GOTO 390
380 PRINT @487, "VOCE DEU AZAR!"
390 REM
400 PRINT @231, "OUTRO JOGO (S/N)?";
410 R$=INKEY$: IF R$="" THEN 410        (Cont.)
```

```
 420 IF R$="S" THEN 45 ELSE END
 430 IF N=1 THEN T=11
 500 GOSUB 4000: GOSUB 2000
 510 GOSUB 3000: RETURN
 900 REM
1000 REM DANDO AS CARTAS
1005 Z=RND(52)
1010 IF D(Z)=0 THEN 1000
1020 D(Z)=0
1030 RETURN
1900 REM
2000 REM IMPRESSAO DOS NAIPES
2005 L1=L
2010 FOR X=1 TO 6
2015 L1=L1+32
2020 FOR Y=1 TO 5
2030 PRINT @L1+(Y-1), S$(S);
2040 NEXT Y,X
2045 L1=0: L=L+6
2050 RETURN
2900 REM
3000 REM IMPRESSAO DOS VALORES DAS CARTAS
3005 L1=L-6
3010 FOR X=1 TO 6
3020 L1=L1+32
3030 PRINT @L1+2,MID$(N$(N),X,1);
3040 NEXT X
3045 L1=0
3050 RETURN
3900 REM
4000 REM CALCULO DO VALOR E DO NAIPE
4005 S=INT((C-1)/13)+1
4010 N=C-(S*13-13)
```

(Cont.)

```
4015 REM CALCULO DOS PONTOS
4020 IF N=11 OR N=12 OR N=13 THEN  T=10 ELSE T=N
4030 IF N=1 THEN T=11
4040 RETURN
```

- PROGRAMA Nº 4

```
 1' **CALEIDOSCOPIO**
 2'
 10 CLS0
 20 X=RND(32)-1
 30 Y=RND(16)-1
 40 Z=RND(9)-1
 50 GOSUB 90
 60 GOTO 20
 90 IF Z=0 OR RND(7)=3 THEN 150
100 SET (31-X,16+Y,Z)
110 SET (31-X,15-Y,Z)
120 SET (32+X,16+Y,Z)
130 SET (32+X,15-Y,Z)
140 RETURN
150 RESET(31-X,16+Y)
160 RESET (31-X,15-Y)
170 RESET (32+X,16+Y)
180 RESET (32+X,15-Y)
190 RETURN
```

- PROGRAMA Nº 5

```
 1' **DADO ELETRONICO**
 2'
  4 CLEAR 2000
  5 CLS(3)
```

(Cont.)

```
  6 DIM DB$(6)
  8 DIM DF(21), P(6), D$(6)
 10 REM
 20 FOR X=1 TO 21
 30 READ DF(X)
 40 NEXT X
 50 DATA 39
 60 DATA 14, 64
 70 DATA 14, 39, 64
 80 DATA 14, 20, 58, 64
 90 DATA 14, 20, 39, 58, 64
 95 DATA 14, 20, 36, 42, 58, 64
100 REM
105 FOR X=1 TO 7
110 REM
120 REM POSICAO NO VETOR DF
130 FOR X=1 TO 6
140 READ P(X)
150 NEXT X
160 DATA 1, 2, 4, 7, 11, 16
165 REM
170 REM CONSTRUCAO DO STRING (DADO)
175 FOR X=1 TO 6
180 M=P(X)
185 FOR Y=1 TO 7
190 FOR Z=1 TO 11
192 IF (Y-1)*11+Z <> DF(M) THEN 200
194 D$(X)=D$(X) + CHR$(128)
196 M=M+1
197 IF M=22 THEN M=0
198 IF M=X THEN M=0
199 GOTO 230
200 D$(X)=D$(X)+CHR$(143+96)
```

(Cont.)

```
230 NEXT Z
240 FOR Z=0 TO 31-11
250 D$(X)=D$(X)+CHR$(143+32)
260 NEXT Z
270 NEXT Y,X
480 REM
490 REM JOGANDO OS DADOS
500 FOR T=1 TO 10
510 A=RND(6): B=RND(6)
520 PRINT @35, D$(A);
530 PRINT @273, D$(B);
540 NEXT T
550 PRINT @80, "PRESSIONE QUAL-";
555 PRINT @112, "QUER TECLA PARA";
560 PRINT @144, "A PROXIMA JOGADA";
570 K$=INKEY$: IF K$="" THEN 570
580 GOTO 500
```

- PROGRAMA Nº 6

```
 1' **GRAFICOS DE BARRA**
 2'
 10 DIM A(5,3,2), A$(5)
 20 DATA UTILIDADES, PESSOAL, FORNECEDORES, ALUGUEL,
    VIAGENS
 30 FOR X=1 TO 5
 40 READ A$(X)
 50 CLS
 60 PRINT @139, "DESPESAS"
 70 PRINT @175 - INT(LEN(A$(X))/2), A$(X)
 80 PRINT
 90 FOR Y=1 TO 3
100 PRINT "DEPTO." Y                    (Cont.)
```

```
110 INPUT "PREVISTAS"; A(X,Y,1)
120 INPUT "REAIS"; A(X,Y,2)
130 NEXT Y
140 NEXT X
150 CLS
160 PRINT @130, "O QUE VOCE GOSTARIA DE VER"
170 L=203
180 FOR X=1 TO 5
190 PRINT @L, X; A$(X)
200 L=L+32
210 NEXT X
220 PRINT @460, "(1-5)"
230 INPUT X
235 C(1)=0: C(2)=0: LC(1)=0: LC(2)=0
240 FOR Y=1 TO 3
250 C(1) = A(X,Y,1) + C(1)
260 C(2) = A(X,Y,2) + C(2)
270 NEXT Y
280 IF C(2) > C(1) THEN 310
290 LC(1)=30: LC(2)=INT(C(2)/C(1)*30)
300 GOTO 320
310 LC(2)=30: LC(1)=INT(C(1)/C(2)*30)
320 P=129
330 CLS(0)
340 PRINT @11, "DESPESAS";
350 PRINT @47 - INT(LEN(A$(X))/2), A$(X);
360 PRINT @161, "PREVISTAS";
370 PRINT @321, "REAIS";
380 PRINT @448, CHR$(159)+CHR$(159);
390 PRINT @451, "DEPTO.1";
400 PRINT @459, CHR$(175)+CHR$(175);
410 PRINT @462, "DEPTO.2";
420 PRINT @470, CHR$(191)+CHR$(191);
```

(Cont.)

```
430 PRINT @ 473, "DEPTO.3";
440 PRINT @ 480, "PRESSIONE QQ TECLA P/CONTINUAR"
450 FOR M=1 TO 2
460 FOR N=1 TO 2
470 P1=P+32
480 FOR Y=1 TO 3
490 D(Y) = INT(A(X,Y,M)/C(1)*LC(1))
500 FOR O=1 TO D(Y)
510 PRINT @P1, CHR$(143+16*Y);
520 P1=P1+1
530 NEXT O
540 NEXT Y
550 P=P+32
560 NEXT N
570 P=289
580 NEXT M
590 K$=INKEY$: IF K$="" THEN 590
600 GOTO 150
```

- PROGRAMA Nº 7

```
1' **TONS**
2'
10 DIM M(50), T(8)
20 FOR B=1 TO 8
30 READ T(B)
40 NEXT B
50 X=1
60 M(X)=RND(8)
70 FOR Y=1 TO X
80 CLS (M(Y))
90 PRINT @239, M(Y);
```

(Cont.)

```
100 SOUND T(M(Y)), 8
110 NEXT Y
120 CLS
130 PRINT@ 234, "REPITA A NOTA"
140 FOR Y=1 TO X
150 T=1
160 K$=INKEY$
170 T= T+1
180 IF T>150 THEN 310
190 IF K$="" THEN 160
200 K= VAL(K$)
210 IF K<>  M(Y) THEN 310
220 CLS(K)
230 PRINT @239, K;
240 SOUND T(K), 3
250 NEXT Y
260 X=X+1
270 CLS: PRINT@ 230, "OUCA A PROXIMA NOTA"
280 FOR T=1 TO 500: NEXT T
290 CLS: PRINT @230, "OUCA A PROXIMA NOTA"
310 GOTO 60
310 CLS(0)
320 PRINT @235, "VOCE PERDEU";
330 SOUND 1, 25
340 DATA 89, 108, 125, 133, 147, 159, 170, 176
```

- PROGRAMA Nº 8

```
1' **COMPOSITOR**
2'
5  DIM A(25), S$(13), B(200): Y=1
9 FOR X=1 TO 25: READ A(X): NEXT X
```

(Cont.)

```
 20 DATA 89, 99, 108, 117, 125
 30 DATA 133, 140, 147, 153, 159
 40 DATA 165, 170, 176, 180, 185
 50 DATA 189, 193, 197, 200, 204
 60 DATA 207, 210, 213, 216, 218
 70 FOR X=1 TO 13:READ S$(X): NEXT X
 80 DATA A,W,S,E,D,F,T,G,Y,H,U,J,K
 90 CLS: PRINT @165, "COMPONHA A SUA MUSICA"
 94 PRINT @224, "USE AS TECLAS DAS  2' E 3' FILAS"
 96 PRINT @289, "PRESSIONE < X > QUANDO TERMINAR"
100 P$=INKEY$
110 IF P$="" THEN 100
115 FOR X=1 TO 13
120 IF P$ <> S$(X) THEN 150
130 SOUND A(X), 5
140 B(Y)=X: Y=Y+1
150 NEXT X
160 IF P$ <> "X" THEN 100
165 CLS
170 PRINT @198, "REPRODUCAO DA MUSICA"
174 PRINT @262, "QUE TECLA (1-11)";
176 INPUT K
180 FOR X=1 TO Y-1:SOUND A(B(X)+K),5
190 NEXT X : GOTO 165
```

- PROGRAMA Nº 9

```
 1'***DENTRO-FORA***
 2'
 5  PMODE 3,1
 10 PCLS3
 15 SCREEN 1,0
```

(Cont.)

```
20 FOR I=3 TO 7
25 FOR J=2 TO 6
30 FOR S=0 TO 3
35 FOR R=0 TO 3
40 COLOR R,S
45 A=0:B=255:C=0:D=191
50 LINE (A,C)-(B,D),PSET,B
55 A=A+J:B=B-J:C=C+I:D=D-I
60 IF A<255 AND C<191 THEN 50
65 NEXT R
70 NEXT S
75 NEXT J,I
80 GOTO 30
```

- PROGRAMA Nº 10

```
1' ***ESTUDOS DE PROJECAO***
2'
5  PMODE 4,1
10 PCLS
15 SCREEN 1,0
20 DRAW "BM50,50R60D10NL20D20L20NU20L20NU20L20U20NR20
   U10" 'VISTA DE TOPO
25 DRAW "BM50,100R20ND20R20ND20R20D20NL20D10L60U10
   NR20U20" 'VISTA FRONTAL
30 DRAW "BM150,100R30D30L30U10NE20U20" 'VISTA
   LATERAL
35 'VISTA OBLIQUA--LINHAS 40-60
40 DRAW "BM150,50U5E15R10BF20BD30NR5L20H25U10
45 DRAW "BM150,50U5F8U15R15H8F8L15F8NR15D15F8ND10E15
   NR10H8
50 LINE (175,30)-(200,55),PSET
55 LINE -(200,80),PSET
60 LINE (167,60)-(183,46),PSET
65 GOTO 65
```

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

- PROGRAMA Nº 11

```
 1' ***ABRINDO CAIXAS***
 2'
 5  PCLEAR 8
 10 PMODE 3,1
 15 PCLS
 20 COLOR 6,5
 25 DRAW "BM100,100U30NR30E15R30NG15D30G15NU30L30"
 30 PAINT (105,95),8,6
 35 PAINT (135,80),8,6
 40 PAINT (110,65),8,6
 45 SCREEN 1,1
 50 FOR X=1 TO 600: NEXT X
110 PMODE 3,5
112 PCLS
115 COLOR 6,5
120 DRAW "BM100,100U30NR30E20R30G20D30NL30F20L30H20
125 LINE (100,100)-(70,95),PSET
130 LINE -(70,65),PSET
135 LINE -(100,70),PSET
140 LINE (70,95)-(40,65),PSET,B
145 LINE (130,100)-(160,95),PSET
150 LINE -(160,65),PSET
155 LINE -(130,70),PSET
160 PAINT (95,95),8,6
165 PAINT (105,95),8,6
170 PAINT (135,85),8,6
175 PAINT (45,85),8,6
180 PAINT (115,65),8,6
185 PAINT (125,114),8,6
190 SCREEN 1,1
195 FOR X=1 TO 600:NEXT X
200 GOTO 10
```

- PROGRAMA Nº 12

```
1' ***CURVA DO SENO***
2'
5  PMODE 4,1
10 PCLS
15 SCREEN 1,1
20 LINE (0,86)-(255,86),PSET
25 PI=3,14159
30 A1=-4*PI
35 A2=4*PI
40 N=180
45 R=50
50 X=(A2-A1)/N
55 F=255/(A2-A1)
60 FOR I=A1 TO A2 STEP X
65 X=I*F
70 Y=R*SIN(I)
75 PSET ((X+140),(80+Y),1)
80 NEXT I
90 GOTO 90
```

- PROGRAMA Nº 13

```
1' ***SENO/COSENO***
2'
10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 LINE (127,5)-(127,185),PSET
50 LINE (7,95)-(247,95),PSET
60 FOR XSCALE=7 TO 247 STEP 20
```

(Cont.)

```
 70 PRESET(XSCALE,95)
 80 NEXT XSCALE
 90 FOR YSCALE=5 TO 185 STEP 10
100 PRESET (127,YSCALE)
110 NEXT YSCALE
130 FOR X=-180 TO 180 STEP 1.5
140 AX=X/57.29578
145 XP=X/1.5+127
150 F1=-(SIN(AX)*90)+95
160 F2=-(COS(AX)*90)+95
170 PSET (XP,F1,1):PSET (XP,F2,1)
180 NEXT X
190 GOTO 190
```

- PROGRAMA Nº 14

```
1' ***GRÁFICOS RANDOMICOS***
2'
10 PMODE 3,1
15 PCLS
16 Z=(RND(2)-1)
20 SCREEN 1,Z
25 F=RND(4):B=RND(8):IF B=F OR (B-4=F) THEN 25
30 COLOR F,B:PCLS B:FOR L=0 TO 5
35 LINE - (RND(255),RND(191)),PSET
40 CIRCLE (RND(255),RND(191)),RND(100)
50 NEXT: FOR P=0 TO 10
55 PAINT (RND(255), RND(191)), RND(4),F
60 NEXT:FOR H=1 TO 7
65 FOR T=0 TO 600:NEXT T:GOTO 10
```

- PROGRAMA Nº 15

```
1' **COBERTOR NAVAHO***
2'
5  PMODE 3,1
10 PCLS 4
15 SCREEN 1,0
20 COLOR 1,0
25 FOR X=0 TO 255 STEP 10
30 OY=Y
35 Y=30-OY
40 LINE (X,100-Y)-(X+10,100-OY),PSET
45 LINE (X,120+Y)-(X+10,120+OY),PSET
50 NEXT
60 FOR C=2 TO 8
65 PAINT (0,110),C,1
70 NEXT
80 GOTO 5
```

- PROGRAMA Nº 16

```
1' ***LACO PINTADO***
2'
5  PMODE 3,1
10 PCLS
20 SCREEN 1,1
30 DRAW "BM50,180U60BU20U60R60BR20R60D60BD20D60L
   60BL20L60
40 DRAW "BM50,180U60R40BR20R80D20BL20L60BL20L20D20
   R20BR60R20U20
50 DRAW "BM50,180R60U80BU20U40L40BD20D20BD60D20R20U
   60BU20U20L20
```

(Cont.)

```
 60 DRAW "BM50,180U60BU40BR20R60BR20R20U20L20D60BD
    20D20R20
 70 DRAW "BM50,180BR80U40BU20U80
 80 DRAW "BM50,180BU80R80BR20R40
 90 PAINT (85,128),6,8
 95 PAINT (95,78),6,8
 97 PAINT (155,95),6,8
 98 PAINT (135,145),6,8
 99 PAINT (128,185),7,8
100 PAINT (75,150),7,8
101 PAINT (160,150),7,8
102 PAINT (75,75),7,8
103 PAINT (160,75),7,8
104 PAINT (120,110),7,8
110 FOR X=1 TO 600:NEXT X
200 GOTO 5
```

- PROGRAMA Nº 17

```
1' ***DESENHANDO***
2'
3  CLS
5  PRINT @128,STRING$(32,"*"):PRINT  288,STRING$
   (32,"*")
10 PRINT @234, "DESENHANDO"
15 FOR X=1 TO 600:NEXT X
20 CLS
25 PRINT @64, "PRESSIONE (↑) PARA SUBIR,(SETA P/
   BAIXO) PARA DESCER,(<-) PARA ESQUERDA,(->) PARA
   DIREITA,(A) PARA SUDOESTE,(S) PARA SUDESTE,(W)
   PARA NORDESTE,(Q) PARA NOROESTE"
```

(Cont.)

```
 30 PRINT @256, "PRESSIONE <1> PARA RETAS INVISIVEIS,
    <2>,<3> OU <4> PARA RETAS VISIVEIS DE DIFERENTES
    CORES E </> PARA MUDAR A COR DO FUNDO"
 35 PRINT @416, "PRESSIONE <BARRA DE ESPACO> PARA
    PAUSA"
 40 FOR X=1 TO 4800:NEXT X
 45 CC=4: TG=0
 50 PMODE 3,1
 55 PCLS
 60 SCREEN 1,TG
 70 X=128:Y=96:XI=0:YI=0
 80 U$="↑":D$=CHR$(10):W$=CRH$(8):E$=CRH$(9)
 90 NW$="Q":NE$="W":SW$="A":SE$="S"
100 C1$="1":C2$="2":C3$="3":C4$="4"
110 A$=INKEY$
120 IF A$=U$ THEN YI=-1:XI=0: GOTO 240
120 IF A$=D$ THEN YI=1:XI=0:GOTO 240
140 IF A$=W$ THEN XI=-1:YI=0:GOTO 240
150 IF A$=E$ THEN XI=1:Yl=0:GOTO 240
160 IF A$=NE$ THEN XI=1:YI=-1:GOTO 240
170 IF A$=NW$ THEN XI=-1:YI=-1:GOTO 240
180 IF A$=SE$ THEN XI=1:YI=1:GOTO 240
190 IF A$=SW$ THEN XI=-1:YI=1:GOTO 240
200 IF C1$<=A$ AND A$<=C4$ THEN CC=ASC(A$)-48:GOTO
    240
210 IF A$="/" THEN TG=(NOT TG AND 1) OR (TG AND
    NOT 1):GOTO 240
220 SCREEN 1,TG
230 IF A$=" " THEN XI=0:YI=0
240 X=X+XI:Y=Y+YI:IF X<0 THEN X=0
250 IF X>255 THEN X=255
260 IF Y<0 THEN Y=0
270 IF Y>191 THEN Y=191
```

(Cont.)

```
275 IF CC=1 THEN PSET (X,Y,3)
280 PSET (X,Y,CC)
290 GOTO 110
```

- PROGRAMA Nº 18

```
1' ***ROSACEA***
2'
5  PCLEAR 8
10 PMODE 4,1
15 PCLS
20 SCREEN 1,0
25 PI=3.14159
30 A1=0:A2=2*PI
35 N=360:A=50
40 X=(A2-A1)/N
45 FOR I=A1 TO A2 STEP X
50 R=A*COS (4*I)
55 X=R*SIN(I)
60 Y=R*COS(I)
65 PSET (128+X,96+Y,5)
70 NEXT I
75 GOTO 25
```

- PROGRAMA Nº 19

```
1' ***BOMBA RELOGIO***
2'
10 PMODE 4,1
15 PCLS
20 SCREEN 1,1
```

(Cont.)

```
 25 CIRCLE (128,96),80
 30 CIRCLE (128,96),90
 35 PAINT (0,0),5
 40 FOR T=30 TO -30 STEP -1
 45 A=(2*3.1415)*T/60
 50 LINE (128,96)-(75*SIN(A)+128,75*COS(A)+96),PSET
 55 SOUND Q*2+1,20/(Q+1)+1
 60 LINE (128,96)-(75*SIN(A)+128,75*COS(A)+96),PRESET
 65 Q=60-2*T:FOR Y=Q TO OSTEP-1:NEXT
 70 NEXT
 75 CLS
 80 PCLS
 85 PRINT @237, "BOOM!"
 90 SOUND 1,30
 95 PMODE 4,1
100 SCREEN 1,1
105 FOR I=2 TO 200 STEP 2
110 CIRCLE (128,96),I
115 NEXT I
120 SCREEN 1,1
125 FOR X=2 TO 200 STEP 2
130 CIRCLE (128,96),X,,3
135 NEXT X
140 FOR I=2 TO 200 STEP 2
145 CIRCLE (128,96),I,3,.5
150 NEXT I
155 GOTO 155
```

- PROGRAMA Nº 20

```
 1' ***CATAVENTO***
 2'
 5  PCLEAR 8
 50 GOTO 600
```

```
 60 LINE ((255-X),(191-Y))-(X,Y),PSET
 61  J=J+1:IF J>A THEN J=0:A=RND(50)
 63 RETURN
600 REM CATAVENTO GIRANDO
601 FOR I=1 TO 5 STEP 4
602 PMODE 3,I
603 PCLS
604 SCREEN 1,0
605 A=25:X=0:Y=0:J=0
610 FOR X=0 TO 254
612 COLOR X/32+1,5
615 GOSUB 60:NEXT X
620 FOR Y=0 TO 190
623 COLOR Y/24+1,5
625 GOSUB 60:NEXT Y
630 FOR X=255 TO 1 STEP -1
633 COLOR X/32+1,5
635 GOSUB 60:NEXT X
640 FOR Y=191 TO 1 STEP -1
643 COLOR Y/24+1,5
645 GOSUB 60:NEXT Y
650 NEXT I
660 FOR I=1 TO 5 STEP 4
670 PMODE 3,I
680 SCREEN 1,0
690 FOR T=1 TO 30:NEXT T
700 NEXT I
710 GOTO 660
```

- PROGRAMA Nº 21

```
 1' ***CONTANDO***
 2'
 10 CLS
 20 CLEAR 1000
 30 PRINT "ONDE QUER COMECAR A CONTAR?"
 35 INPUT A$
 40 P=LEN(A$)
 50 PRINT :PRINT A$
 60 C=VAL(MID$(A$,P,1))+1
 70 MS$=A$:MR$=RIGHT$(STR$(C),1):PS=P:GOSUB 200:A$=
    MS$
 80 IF C<10 THEN 40
 90 P=P-1
100 IF P=0 THEN IF LEN(A$)=255 THEN PRINT "OVERFLOW":
    END:ELSE A$="1"+A$:GOTO 40
110 GOTO 60
200 LS=LEN(MS$)
210 IF LS<>LEN(MR$)+LS-1 OR PS<1 THEN STOP
220 MS$=LEFT$(MS$,PS-1)+MR$+RIGHT$(MS$,LS-PS)
230 RETURN
```

SUB-ROTINAS CIENTÍFICAS

Estas sub-rotinas permitem executar programas que utilizam funções matemáticas avançadas e que não estão diretamente disponíveis no BASIC normal.

Cada listagem de sub-rotina tem um conjunto de instruções ao lado. Estude-as atentamente, você verá que algumas delas fazem uso de outras para cálculos internos. Você terá que digitar estas "sub-rotinas auxiliares" sempre que as instruções assim o exigirem.

NOTA: A precisão das sub-rotinas é menor que a das funções e operadores matemáticos. Isto se deve a dois fatos:

1 - As sub-rotinas contêm muitos cálculos encadeados, que tendem a aumentar os pequenos erros de operações isoladas.

2 - Estas sub-rotinas são apenas aproximações das funções que elas substituem. Em geral, as sub-rotinas têm precisão até a 5ª ou 6ª casa decimal para a maior parte dos valores permitidos e conforme este valor se aproxima dos limites inferior ou superior da faixa permitida, ocorre uma diminuição da precisão.

RAIZ QUADRADA

Determina : SQR(X), $\sqrt{X}$

Entrada : X, deve ser maior ou igual a zero

Saída : Y

Variáveis usadas : W, Z (internamente)

Sub- rotinas usadas: nenhuma

Como chamar : GOSUB 30030

```
30000 END
30010 REM *RAIZ QUADRADA* ENTRADA X, SAIDA Y
30020 REM VARIAVEIS USADAS: W&Z INTERNAMENTE
30030 IF X=0 THEN Y=0: RETURN
30040 IF X>0 THEN 30060
30050 PRINT "RAIZ DE NUMERO NEGATIVO?": STOP
30060 Y = X * .5 : Z=0
30070 W = (X/Y -Y) * .5
30080 IF (W=0) + (W=Z) THEN RETURN
30090 Y=Y+W : Z=W : GOTO 30070
```

EXPONENCIAÇÃO

Determina : X^Y (X na potência Y)

Entrada : X, Y. Se X for menor que zero, Y deve ser um número inteiro ímpar.

Saída : P

Variáveis usadas: E, L, A, B, C (internamente). O valor de X é alterado.

Sub-rotinas usadas : LOG e EXPONENCIAL

Como chamar : GOSUB 30120

```
30000 END
30100 REM *EXPONENCIACAO* ENTRADA X,Y: SAIDA P
30110 REM VARIAVEIS USADAS E,L,A,B,C,
      INTERNAMENTE
30120 P=1 : E=0 : IF Y=0 THEN RETURN
30130 IF (X<0)AND(INT(Y)=Y) THEN P=1-2*Y+4*
      INT(Y/2): X=-X
30140 IF X<>0 THEN GOSUB 30190: X=Y*L: GOSUB
      30250
30150 P=P*E: RETURN
```

LOGARÍTMO NATURAL E COMUM

Determina : LOG(X) na base e, e LOG(X) na base 10

Entrada : X maior ou igual a zero.

Saída : L é o logarítmo natural, X é o logarítmo decimal

Variáveis usadas: A, B, C (internamente). O valor de X é alterado

Sub-rotinas usadas: nenhuma

Como chamar : GOSUB 30190

```
30000 END
30170 REM *LOG NATURAL E COMUM: ENTRADA X,
      SAIDA L,X
30180 REM L E O LOG NATURAL, X E O LOG DECIMAL
30190 E=0: IF X<0 THEN PRINT "LOG INDEFINIDO
      PARA";X: STOP
30195 A=1: B=2: C=.5
30200 IF X>=A THEN X=C*X: E=E+A: GOTO 30200
30205 IF X<C THEN X=B*X: E=E-A: GOTO 30205
30210 X=(X-.707107)/(X+.707107): L=X*X
30215 L=(((.598979*L+.961471)*L+2.88539)*X+E-.
      5*.693147
30220 IF ABS(L)<1E-6 THEN L=0
30225 X=L*.4342945: RETURN
```

EXPONENCIAL

Determina : EXP(X) (e na potência X)

Entrada : X

Saída : E

Variáveis usadas: L, A (internamente). O valor de X é alterado.

Sub-rotinas usadas: nenhuma

como chamar : GOSUB 30250

```
30000 END
30240 REM *EXPONENCIAL* ENTRADA X, SAIDA E
30245 REM VARIAVEIS USADAS L,A INTERNAMENTE
30250 L=INT(1.4427*X)+1 : IF L<127 THEN 30265
30255 IF X>0 THEN PRINT "LIMITE ULTRAPASSADO":
      STOP
30260 E=0: RETURN
30265 E=.693147*L-X: A=1.32988E-3-1.41316E-4*E
30270 A=((A*E-8.30136E-3)*E+4.16574E-2)*E
30275 E=(((A-.166665)*E+.5)*E-1)*E+1:A=2
30280 IF L<=0 THEN A=.5: L=-L: IF L=0 THEN
      RETURN
30285 FOR X=1 TO L: E=A*E: NEXT X: RETURN
```

TANGENTE

Determina : TAN(X)
Entrada : X em graus
Saída : Y
Variáveis usadas: nenhuma
Sub-rotinas usadas: COSSENO
Como chamar : GOSUB 30310

```
30000 END
30300 REM *TANGENTE* ENTRADA X EM GRAUS,
      SAIDA Y
30310 IF ABS(SIN((90-X)/57.29577951))<1E-7
      THEN PRINT "INDEFINIDO": STOP
30320 Y=SIN(X/57.29577951)/SIN((90-X)/57.
      29577951)
30330 RETURN
```

COSSENO

Determina : COS(X)
Entrada : X em graus
Saída : Y
Variáveis usadas: nenhuma
sub-rotinas usadas: nenhuma
Como chamar : GOSUB 30360

```
30000 END
30350 REM *COSSENO* ENTRADA X EM GRAUS,
      SAIDA Y
30360 Y=SIN((90-X)/57.29577951)
30365 RETURN
```

ARCO COSSENO

Determina : ARCCOS(S), ângulo cujo cosseno é S

Entrada : S, 0 < =S < =1

Saída : Y em graus e W em radianos

Variáveis usadas: X, Z (internamente)

Sub-rotinas usadas: ARCO SENO

Como chamar : GOSUB 30500

```
30000 END
30500 REM *ARCO COSSENO* ENTRADA S, SAIDA Y,W
30510 REM Y EM GRAUS, W EM RADIANOS
30520 GOSUB 30550: Y=90-Y: W=1.570796-W:
      RETURN
```

ARCO SENO

Determina : ARCSEN(S), ângulo cujo seno é S

Entrada : S, 0 < =S < =1

Saída : Y em graus e W em radianos

Variáveis usadas: X, Y (internamente)

Sub-rotinas usadas: nenhuma

Como chamar : GOSUB 30550

```
30000 END
30530 REM *ARCO SENO* ENTRADA S, SAIDA Y,W
30535 REM Y EM GRAUS, W EM RADIANOS
30540 REM VARIAVEIS USADAS X,Z
30550 X=S: IF ABS(S)<=.707107 THEN 30610
30560 X=1-S*S: IF X<0 THEN PRINT S;"FORA DO
      LIMITE": STOP
30565 IF X=0 THEN W=90/57.29577951: GOTO 30630
30570 W=X/2 : Z=0
30580 Y=(X/W-W)/2: IF (ABS(Y)<.1E-8)AND(Y=Z)
      THEN X=W: GOTO 30610
30600 W=W+Y: Z=Y: GOTO 30580
30610 Y=X+X*X*X/6+X*X*X*X*X.075+X*X*X*X*X*X*X*
      4.464286E-2
30620 W=Y+X*X*X*X*X*X*X*X*X*3.038194E-2
30625 IF ABS(S)>.707107 THEN W=1.570796-W
30630 Y=W*57.29577951: RETURN
```

ARCO TANGENTE

Determina : ATN(X), ângulo cuja tangente é X

Entrada : X

Saída : C em graus e A em radianos

Variáveis usadas: B, T (internamente). O valor de X é alterado

Sub-rotinas usadas: nenhuma

Como chamar : GOSUB 30690

```
30000 END
30660 REM *ARCO TANGENTE* ENTRADA X, SAIDA C,A
30670 REM C EM GRAUS, A EM RADIANOS
30680 REM VARIAVEIS USADAS B,T
30690 T=SGN(X): X=ABS(X): C=0
30700 IF X>1 THEN C=1: X=1/X
30710 A=X*X
30720 B=((2.86623E-3*A-1.61657E-2)*A+4.29096E-
      2)*A
30730 B=((((B-7.5289E-2)*A+.106563)*A-.142089)*
      A+.199936)*A
30740 A=((B-.333332)*A+1)*X
30750 IF C=1 THEN A=1.570796-A
30760 A=T*A: C=A*57.29577951: RETURN
```

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

FÓRMULAS TRIGONOMÉTRICAS

Lado a = LA (lado oposto ao ângulo A)
Lado b = LB (lado adjacente ao ângulo A)
Lado c = LC (hipotenusa)
Ângulo A = AA
Ângulo B = AB
Ângulo C = AC

DESCRIÇÃO	FÓRMULA	COMANDO BASIC
Número total de graus de um triângulo	$180^{o} = A + B + C$	TTL = AA + AB + AC
Cálculo da área dados lado A e ângulos B e C	$A = 180 - (B + C)$	AA = 180-(AB+AC) converter AA, AB e AC para radianos
	$\text{Área} = \frac{a^2 \text{ sen B. sen C}}{2 \text{ sen A}}$	AREA = LA↑2*SIN(AB)*SIN(AC)/2*SIN(AA))
Dados os lados a, b e c	$s = 1/2 (a + b + c)$	S = (LA + LB + LC) / 2
	$\text{Área} = \sqrt{s(s-a)(s-b)(s-c)}$	AREA =SQR(S*(S-LA)*(S-LB)*(S-LC))
Lei dos senos	$\frac{a}{b} = \frac{\text{sen A}}{\text{sen B}}$ ou $a = \frac{\text{sen A}}{\text{sen B}} .b$	LA = (SIN(AA)/SIN(AB))*LB

(Cont.)

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

DESCRIÇÃO	FÓRMULA	COMANDO BASIC
Lei dos cossenos	$a^2 = b^2 + c^2 - 2bc.\cos A$ ou $a = \sqrt{b^2 + c^2 - 2bc.\cos A}$	LA = SQR(LB↑2-LC↑2-2*LB*LC*COS (AA))
Lei das tangentes	$\frac{a - c}{a + c} = \frac{\tan 1/2\ (A-C)}{\tan 1/2\ (A+C)}$ ou $\tan 1/2\ (A-C) = \frac{a-c}{a+c} \tan 1/2\ (A+C)$	REM Y = TAN((AA-AC)/2) Y = (LA-LC)/(LA+LC)*TAN((AA+AC)/2)
Dados três lados determinar o ângulo	$s = 1/2\ (a + b + c)$ $r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}}$ $A = 2 \arctan (\frac{r}{s - a})$	S = (LA + LB + LC)/2 R = SQR((S-LA)*(S-LB)*(S-LC)/S) AA = 2*ATN(R/(S-LA))
Equação quadrática	$ax^2 + bx + c = 0$ $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	REM A*X↑2 + B*X + C = 0 Z = B↑2 - 4*A*C X1 = (-B + SQR(Z))/(2*A) 'IF Z =0 X2 = (-B - SQR(Z))/(2*A) 'IF Z =0

(Cont.)

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

DESCRIÇÃO	FÓRMULA	COMANDO BASIC
Equações algébricas	$(a^x)^y = a^{xy}$	Z = (A↑X)↑Y ou Z = A↑(X*Y)
	$a^{-x} = \frac{1}{a^x}$	Z = A↑(-X) ou Z = 1/(A↑X)
	$\log x^y = y \cdot \log x$	Z = LOG(X↑Y) ou Z = Y*LOG(X)
	$\log xy = \log x + \log y$	Z = LOG (X*Y) ou Z = LOG(X)+LOG(Y)
	$\log \frac{x}{y} = \log x - \log y$	Z = LOG(X/Y) ou Z = LOG(X)-LOG(Y)

FUNÇÕES DERIVADAS

FUNÇÃO	COMANDOS DO BASIC EXTENDIDO (X EM RADIANOS)
SECANTE	SEC(X) = 1/COS(X)
COSSENCANTE	CSC(X) = 1/SIN(X)
COTANGENTE	COT(X) = 1/TAN(X)
ARCO SENO	ARCSIN(X) = ATN(X/SQR(-X*X+1))
ARCO COSSENO	ARCCOS(X) = -ATN(X/SQR(-X*X+1)) + 1.5708
ARCO SECANTE	ARCSEC(X) = ATN(SQR(X*X-1)) + (SGN(X) -1)*1.5708
ARCO COSSECANTE	ARCSEC(X) = ATN(1/SQR(X*X-1)) +(SGN(X) -1)*1.5708
ARCO COTANGENTE	ARCCOT(X) = -ATN(X) + 1.5708
SENO HIPERBÓLICO	SINH(X) = (EXP(X) - EXP(-X))/2
COSSENO HIPERBÓLICO	COSH(X) = (EXP(X) + EXP(-X))/2
TANGENTE HIPERBÓLICA	TANH(X)=-EXP(-X)/(EXP(X) + EXP(-X))*2 + 1
SECANTE HIPERBÓLICA	SECH(X) = 2/(EXP(X) + EXP(-X))
COSSECANTE HIPERBÓLICA	COSH(X) = 2/(EXP(X) - EXP(-X))
COTANGENTE HIPERBÓLICA	COTH(X) = EXP(-X)/EXP(X)-EXP(-X)*2+1
ARCO SENO HIPERBÓLICO	ARGSINH(X) = LOG(X+SQR(X*X+1))
ARCO COSSENO HIPERBÓLICO	ARGCOSH(X) = LOG(X+SQR(X*X-1))
ARCO TANGENTE HIPERBÓLICO	ARGTANH(X) = LOG((1-X)/(1-X))/2
ARCO SECANTE HIPERBÓLICO	ARGSECH(X) = LOG((SQR(-X*X+1)+1)/X)
ARCO COSSECANTE HIPERBÓLICO	ARGCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
ARCO COTANGENTE HIPERBÓLICO	ARGCOTH(X) = LOG((X+1)/(X-1))/2

ARCO SENO	-1<X<1
ARCO COSSENO	-1<X<1
ARCO SECANTE	X<-1 ou X>1
ARCO COSSECANTE	X<-1 ou X>1
ARCO COSSENO HIPERBÓLICO	X>1
ARCO TANGENTE HIPERBÓLICO	X*X<1
ARCO SECANTE HIPERBÓLICO	0<X<1
ARCO COSSECANTE HIPERBÓLICO	X<>0
ARCO COTANGENTE HIPERBÓLICO	X*X>1

Certos valores especiais são matematicamente indefinidos mas as funções do BASIC EXTENDIDO podem fornecer respostas inválidas:

TAN e SEC de 90 e 270 graus

COT e CSC de 0 e 180 graus

Por exemplo, TAN(1.5708) retorna um valor mas TAN (90*.01745329) retorna o erro divisão por zero apesar de 90*.01745329 = 1.5708.

Os outros valores que não estão disponíveis para estas funções são:

```
ARCSIN (-1) = - PI/2
ARCSIN ( 1) =   PI/2
ARCCOS (-1) =   PI
ARCCOS ( 1) =   0
ARCSEC (-1) =  -PI
ARCSEC ( 1) =   0
ARCCSC (-1) =  -PI/2
ARCCSC ( 1) =   PI/2
```

RELAÇÃO DE CÓDIGOS ASC II

CARACTER	CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL
ESPAÇO	32	20
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47

CARACTER	CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A
↑	94	5E
↓	10	0A
←	8	08
→	9	09
BREAK	3	03
CLEAR	12	0C
ENTER	13	0D
SHIFT ↑	95	5F
SHIFT ↓	91	5B
SHIFT ←	21	15

(Cont.)

APÊNDICE

CARACTER	CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL
SHIFT ⊖	93	5D
SHIFT CLEAR	92	5C
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C

CARACTER	CÓDIGO DECIMAL	CÓDIGO HEXADECIMAL
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A

CARACTERES DO TECLADO

←	- Apaga o último caracter digitado; move o cursor uma posição para trás.
SHIFT ←	- Volta o cursor para o início da linha atual, apagando tudo que foi digitado.
BREAK	- Interrompe a execução do programa.
CLEAR	- Limpa a tela.
ENTER	- Indica o fim da linha atual.
ESPAÇO	- Gera um caracter em branco e move o cursor uma posição para frente.
SHIFT @	- Provoca uma pausa na execução do programa. Pressione qualquer tecla para continuar.
SHIFT 0	- Permite a passagem do modo letras maísculas para o modo letras minúsculas e vice-versa. As letras minúsculas são representadas por letras maiúsculas em cores reversas.

SÍMBOLOS BASIC

" "	- Indica que a informação entre aspas é uma constante (string).
:	- Separa os comandos escritos numa mesma linha
( )	- Indica ao computador que as operações entre parênteses devem ser executadas primeiro.
;	- Faz com que as constantes e variáveis sejam impressas coladas umas nas outras.

OPERADORES BASIC

+	- combina strings (concatenação).
+	- adição

(Cont.)

-	- subtração.
*	- multiplicação.
/	- divisão.
=	- igualdade.
>	- maior que.
> = ou = >	maior que ou igual a.
< = ou = <	- menor que ou igual a
<	- menor que
<> ou ><	- diferente de
AND	- operador AND(E)
OR	- operador OR(OU)
NOT	- operador NOT(NÃO)

CONVERSÃO DE BASE

DEC.	BINÁRIO	HEX.	OCTAL
0	00000000	00	000
1	00000001	01	001
2	00000010	02	002
3	00000011	03	003
4	00000100	04	004
5	00000101	05	005
6	00000110	06	006
7	00000111	07	007
8	00001000	08	010
9	00001001	09	011
10	00001010	0A	012
11	00001011	0B	013
12	00001100	0C	014
13	00001101	0D	015
14	00001110	0E	016
15	00001111	OF	017
16	00010000	10	020
17	00010001	11	021
18	00010010	12	022
19	00010011	13	023
20	00010100	14	024
21	00010101	15	025
22	00010110	16	026
23	00010111	17	027
24	00011000	18	030
25	00011001	19	031
26	00011010	1A	032
27	00011011	1B	033
28	00011100	1C	034
29	00011101	1D	035
30	00011110	1E	036
31	00011111	1F	037
32	00100000	20	040
33	00100001	21	041
34	00100010	22	042
35	00100011	23	043
36	00100100	24	044
37	00100101	25	045
38	00100110	26	046
39	00100111	27	047
40	00101000	28	050
41	00101001	29	051

DEC.	BINÁRIO	HEX.	OCTAL
42	00101010	2A	052
43	00101011	2B	053
44	00101100	2C	054
45	00101101	2D	055
46	00101110	2E	056
47	00101111	2F	057
48	00110000	30	060
49	00110001	31	061
50	00110010	32	062
51	00110011	33	063
52	00110100	34	064
53	00110101	35	065
54	00110110	36	066
55	00110111	37	067
56	00111000	38	070
57	00111001	39	071
58	00111010	3A	072
59	00111011	3B	073
60	00111100	3C	074
61	00111101	3D	075
62	00111110	3E	076
63	00111111	3F	077
64	01000000	40	100
65	01000001	41	101
66	01000010	42	102
67	01000011	43	103
68	01000100	44	104
69	01000101	45	105
70	01000110	46	1060
71	01000111	47	107
72	01001000	48	110
73	01001001	49	111
74	01001010	4A	112
75	01001011	4B	113
76	01001100	4C	114
77	01001001	4D	115
78	01001110	4E	116
79	01001111	4F	117
80	01010000	50	120
81	01010001	51	121
82	01010010	52	122
83	01010011	53	123

(Cont.)

APÊNDICE

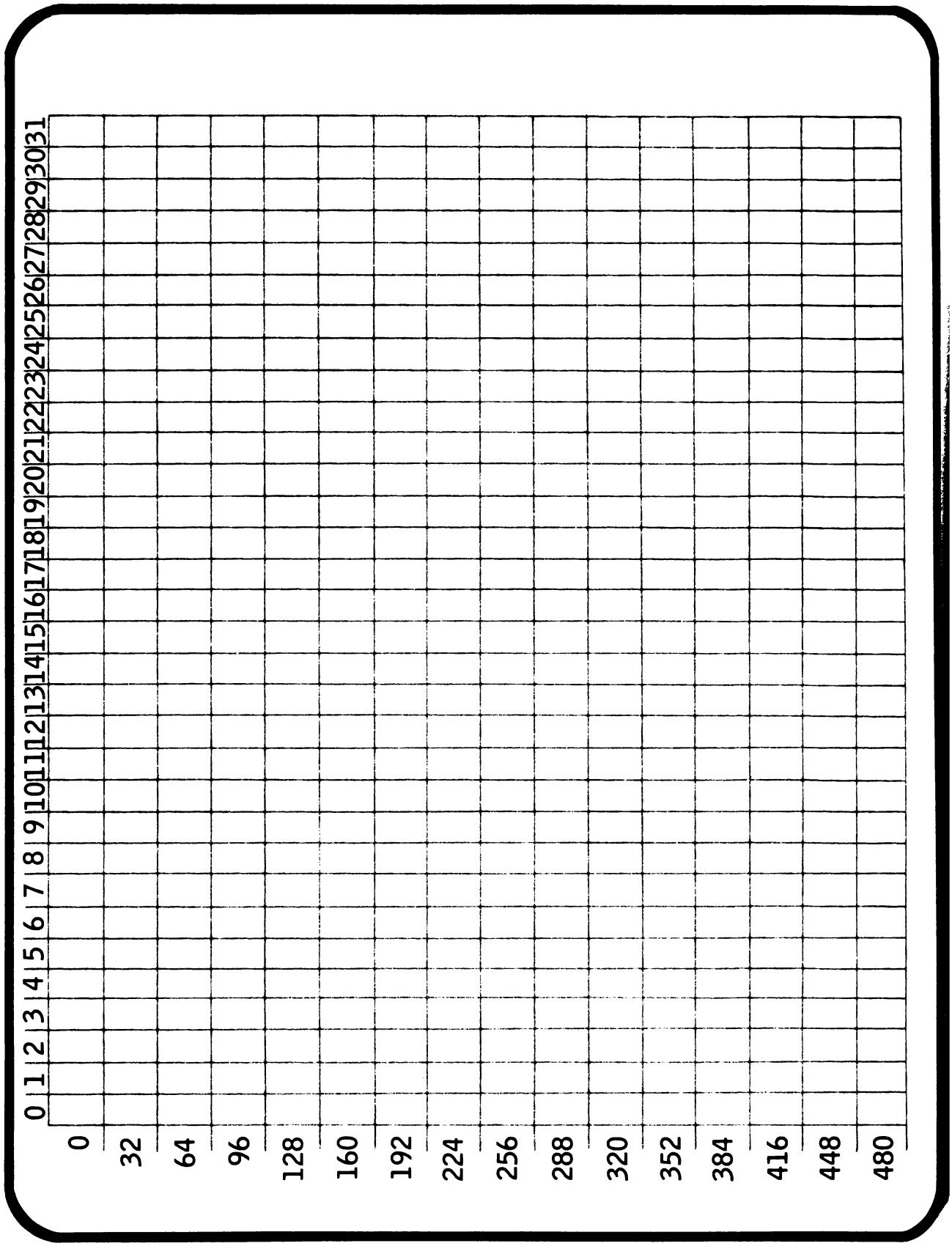
DEC.	BINÁRIO	HEX.	OCTAL
84	01010100	54	124
85	01010101	55	125
86	01010110	56	126
87	01010111	57	127
88	01011000	58	130
89	01011001	59	131
90	01011010	5A	132
91	01011011	5B	133
92	01011100	5C	134
93	01011101	5D	135
94	01011110	5E	136
95	01011111	5F	137
96	01100000	60	140
97	01100001	61	141
98	01100010	62	142
99	01100011	63	143
100	01100100	64	144
101	01100101	65	145
102	01100110	66	146
103	01100111	67	147
104	01101000	68	150
105	01101001	69	151
106	01101010	6A	152
107	01101011	6B	153
108	01101100	6C	154
109	01101101	6D	1550
110	01101110	6E	156
111	01101111	6F	157
112	01110000	70	160
113	01110001	71	161
114	01110010	72	162
115	01110011	73	163
116	01110100	74	164
117	01110101	75	165
118	01110110	76	166
119	01110111	77	167
120	01111000	78	170
121	01111001	79	171
122	01111010	7A	172
123	01111011	7B	173
124	01111100	7C	174
125	01111101	7D	175
126	01111110	7E	176
127	01111111	7F	177
128	10000000	80	200
129	10000001	81	201

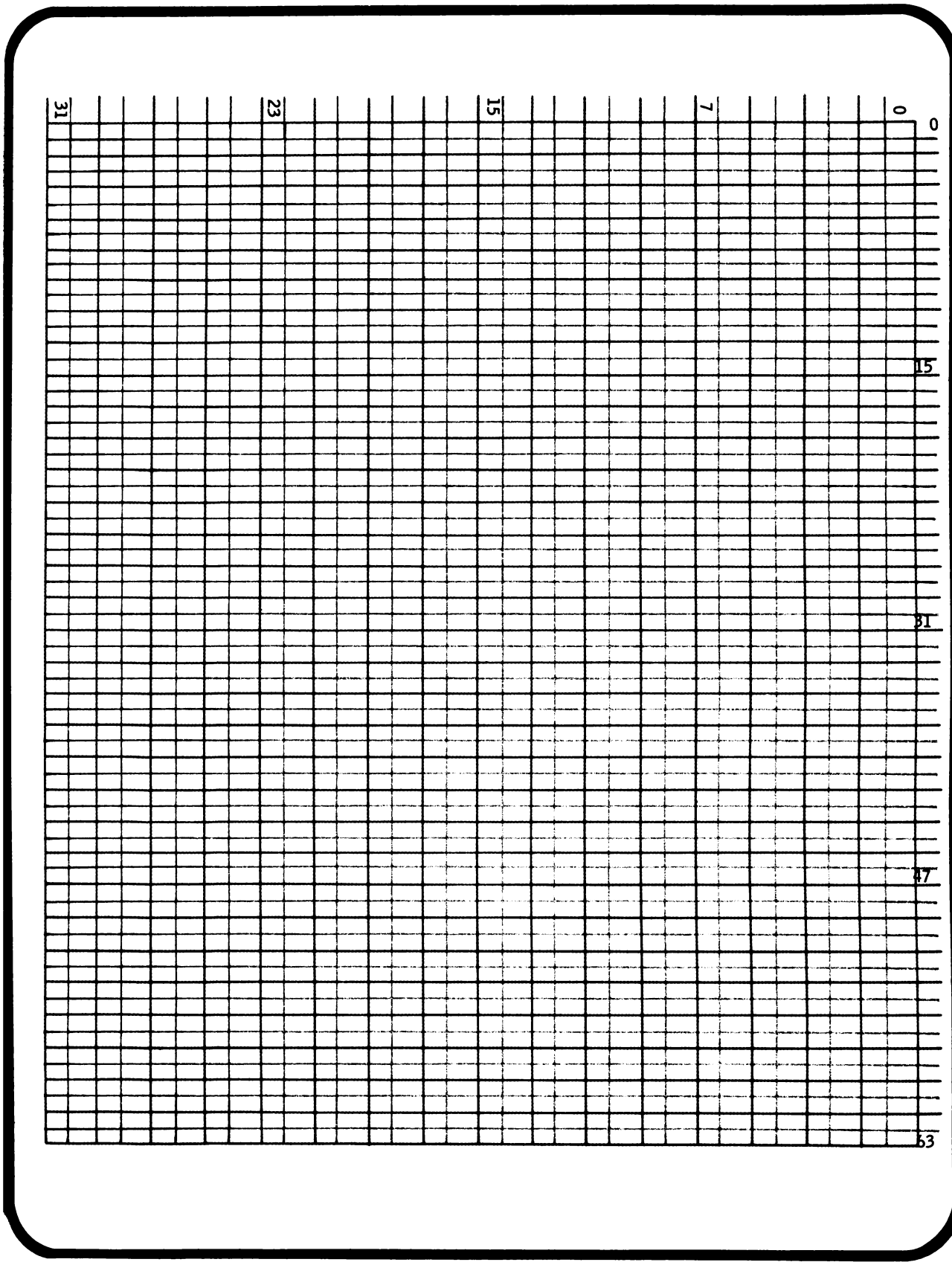
DEC.	BINÁRIO	HEX.	OCTAL
130	10000010	82	202
131	10000011	83	203
132	10000100	84	204
133	10000101	85	205
134	10000110	86	206
135	10000111	87	207
136	10001000	88	210
137	10001001	89	211
138	10001010	8A	212
139	10001011	8B	213
140	10001100	8C	214
141	10001001	8D	215
142	10001110	8E	216
143	10001111	8F	217
144	10010000	90	220
145	10010001	91	221
146	10010010	92	222
147	10010011	93	223
148	10010100	94	224
149	10010101	95	225
150	10010110	96	226
151	10010111	97	227
152	10011000	98	230
153	10011001	99	231
154	10011010	9A	232
155	10011011	9B	233
156	10011100	9C	234
157	10011101	9D	235
158	10011110	9E	236
159	10011111	9F	237
160	10100000	A0	240
161	10100001	A1	241
162	10100010	A2	242
163	10100011	A3	243
164	10100100	A4	244
165	10100101	A5	245
166	10100110	A6	246
167	10100111	A7	247
168	10101000	A8	250
169	10101001	A9	251
170	10101010	AA	252
171	10101011	AB	253
172	10101100	AC	254
173	10101101	AD	255
174	10101110	AE	256
175	10101111	AF	257

(Cont.)

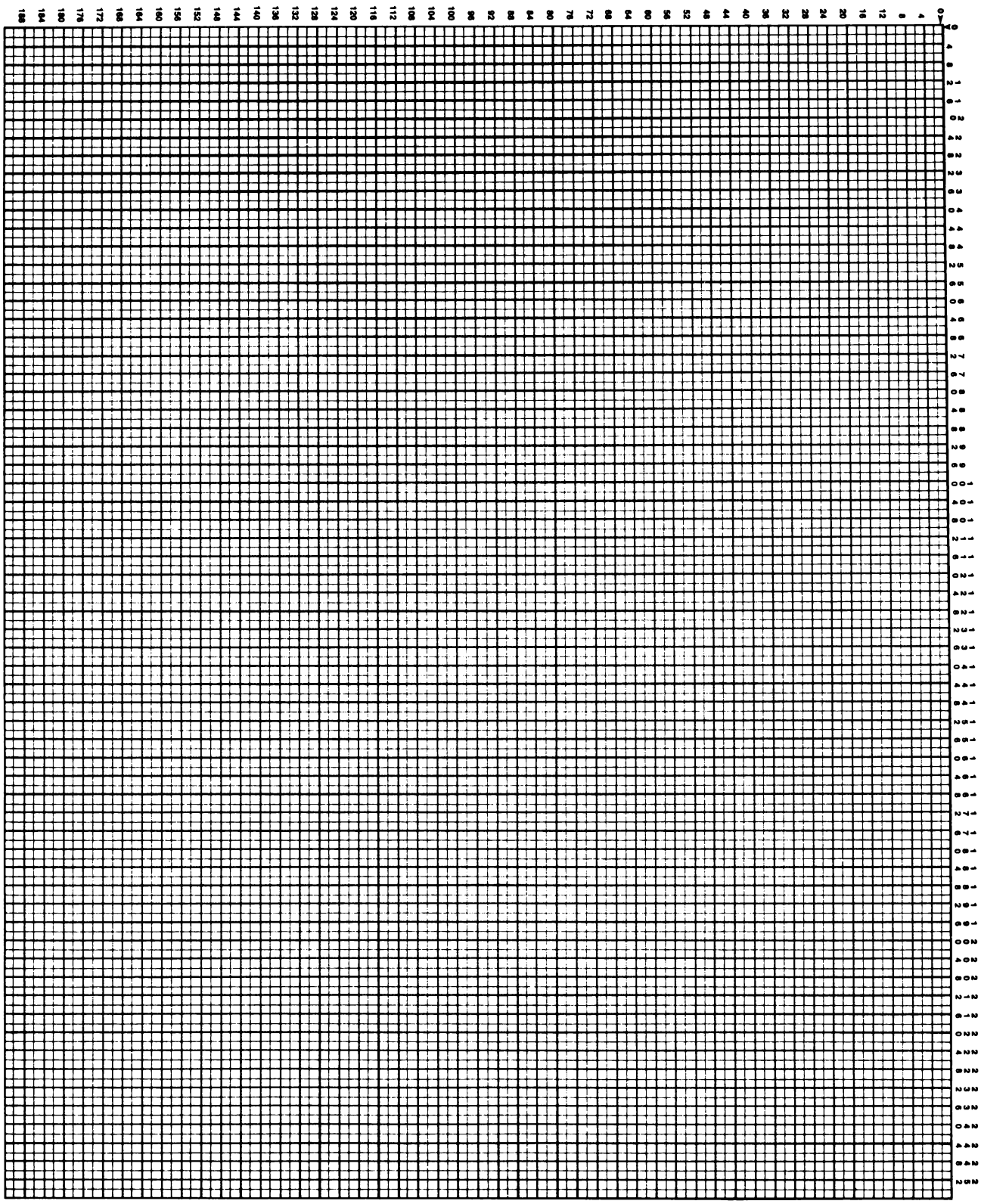
DEC.	BINÁRIO	HEX.	OCTAL
176	10110000	B0	260
177	10110001	B1	261
178	10110010	B2	262
179	10110011	B3	263
180	10110100	B4	264
181	10110101	B5	265
182	10110110	B6	266
183	10110111	B7	267
184	10111000	B8	270
185	10111001	B9	271
186	10111010	BA	272
187	10111011	BB	273
188	10111100	BC	274
189	10111101	BD	275
190	10111110	BE	276
191	10111111	BF	277
192	11000000	C0	300
193	11000001	C1	301
194	11000010	C2	302
195	11000011	C3	303
196	11000100	C4	304
197	11000101	C5	305
198	11000110	C6	306
199	11000111	C7	307
200	11001000	C8	310
201	11001001	C9	311
202	11001010	CA	312
203	11001011	CB	313
204	11001100	CC	314
205	11001101	CD	315
206	11001110	CE	316
207	11001111	CF	317
208	11010000	D0	320
209	11010001	D1	321
210	11010010	D2	322
211	11010011	D3	323
212	11010100	D4	324
213	11010101	D5	325
214	11010110	D6	326
215	11010111	D7	327

DEC.	BINÁRIO	HEX.	OCTAL
216	11011000	D8	330
217	11011001	D9	331
218	11011010	DA	332
219	11011011	DB	333
220	11011100	DC	334
221	11011101	DD	335
222	11011110	DE	336
223	11011111	DF	337
224	11100000	E0	340
225	11100001	E1	341
226	11100010	E2	342
227	11100011	E3	343
228	11100100	E4	344
229	11100101	E5	345
230	11100110	E6	346
231	11100111	E7	347
232	11101000	E8	350
233	11101001	E9	351
234	11101010	EA	352
235	11101011	EB	353
236	11101100	EC	354
237	11101101	ED	355
238	11101110	EE	356
239	11101111	EF	357
240	11110000	F0	360
241	11110001	F1	361
242	11110010	F2	362
243	11110011	F3	363
244	11110100	F4	364
245	11110101	F5	365
246	11110110	F6	366
247	11110111	F7	367
248	11111000	F8	370
249	11111001	F9	371
250	11111010	FA	372
251	11111011	FB	373
252	11111100	FC	374
253	11111101	FD	375
254	11111110	FE	376
255	11111111	FF	377

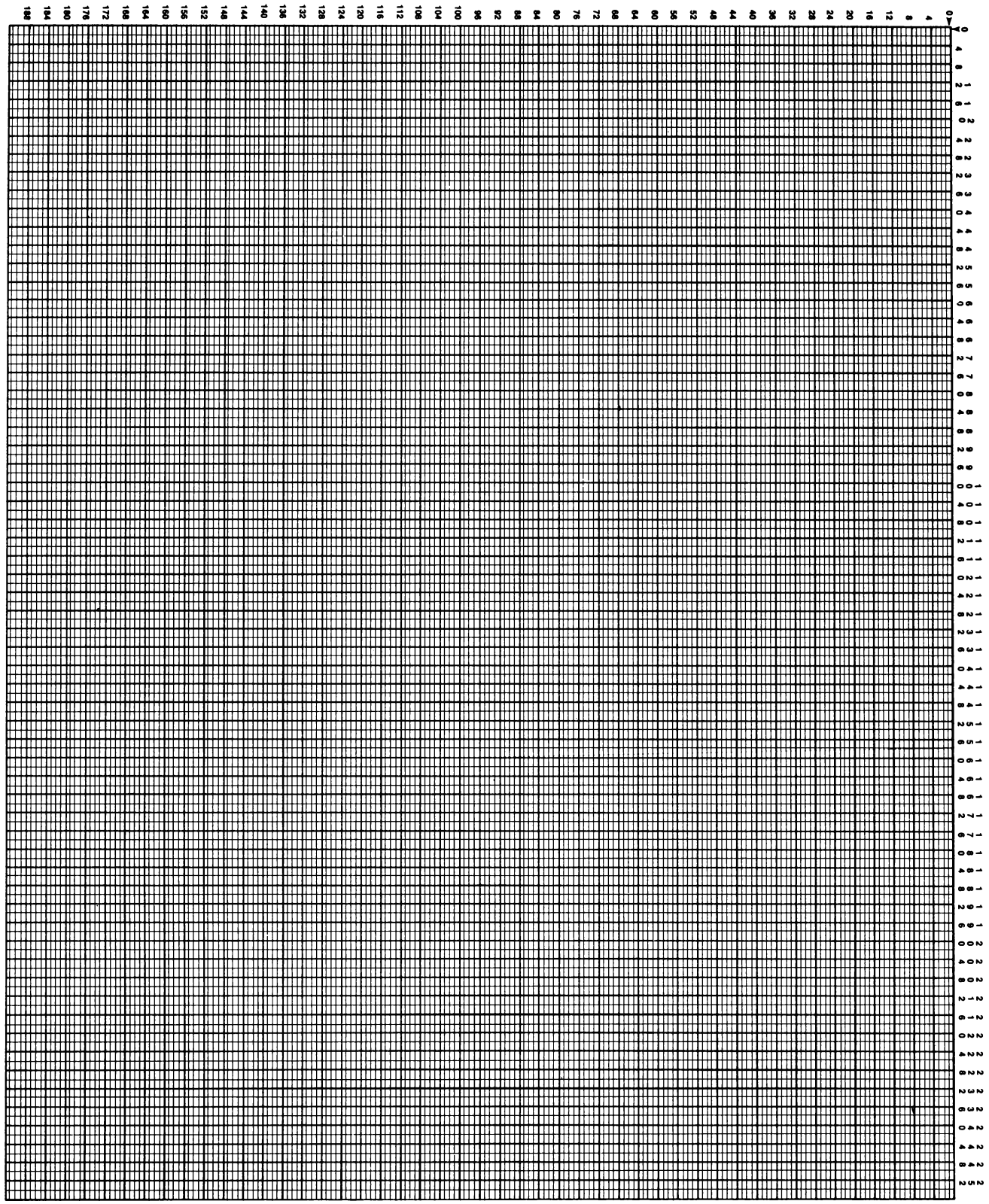

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0
32
64
96
128
160
192
224
256
288
320
352
384
416
448
480


31
23
15
7
0
0
15
31
47
63

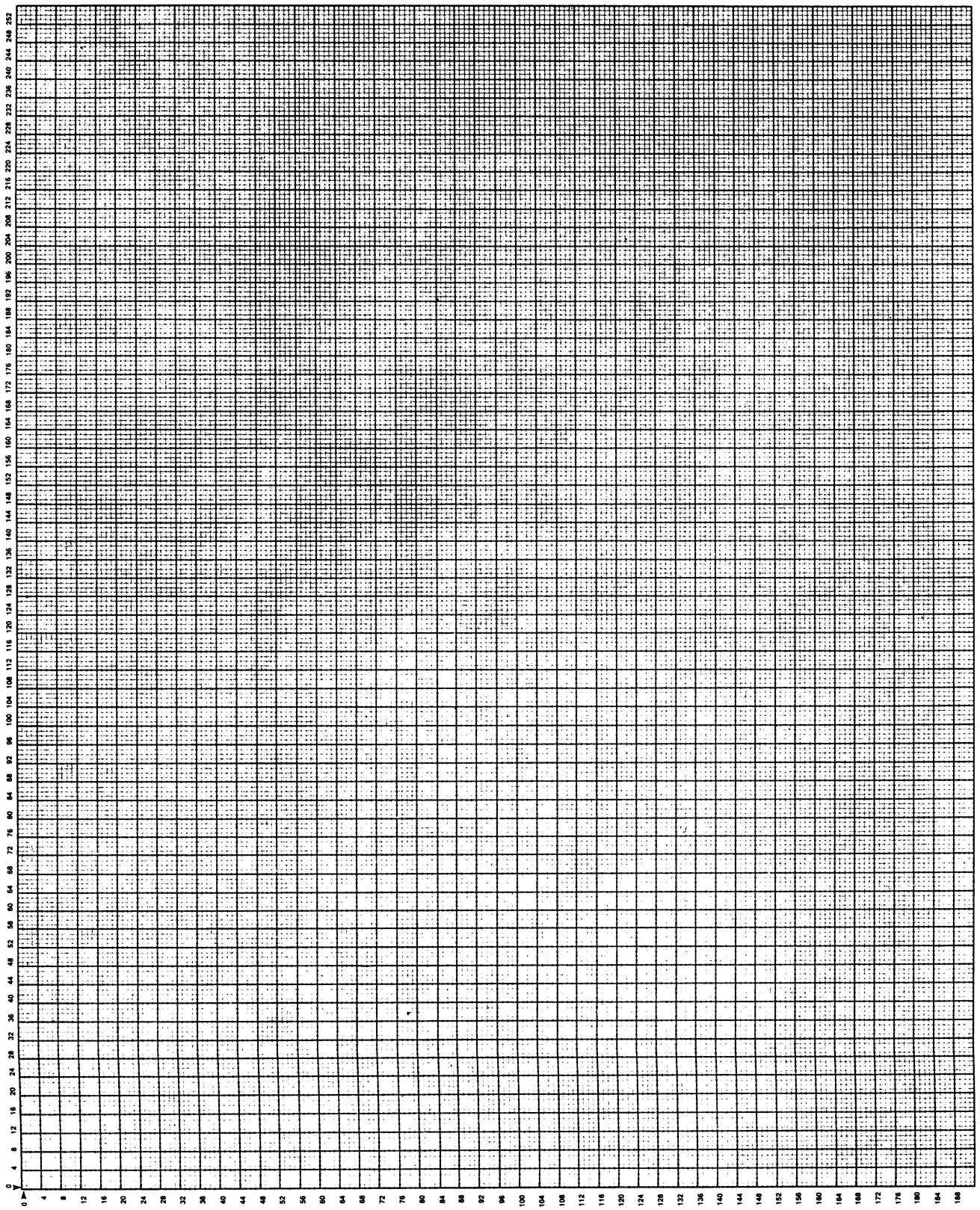
128 x 96



128 x 192



256 x 192



CORES E CARACTERES GRÁFICOS

C O R E S

Esses são os códigos das cores que você pode mostrar na tela.

0 - preto (ausência de cor)	3 - azul	6 - azul piscina
1 - verde	4 - vermelho	7 - roxo
2 - amarelo	5 - bege	8 - laranja

Usando a cor 0 no comando SET, ela não alterará a cor do ponto.

CARACTERES GRÁFICOS

Esses são os códigos dos caracteres gráficos. Para produzí-los, use CHR$ com o código do caracter. Por exemplo, PRINT CHR$(129) mostra na tela o caracter 129.





Para criar estes caracteres em uma das cores abaixo, adicione o número apropriado ao código. Por exemplo, PRINT CHR$(129+16) mostra na tela o caracter 129 em amarelo (exceto a área verde).

+16 - amarelo
+32 - verde
+48 - vermelho
+64 - bege
+80 - azul piscina
+96 - roxo
+112 - laranja

CORES COM BASIC EXTENDIDO

Estes são os códigos para uma das 9 cores que você pode usar com o BASIC EXTENDIDO.

CÓDIGO	COR
0	preto
1	verde
2	amarelo
3	azul
4	vermelho

CÓDIGO	COR
5	bege
6	azul piscina
7	roxo
8	laranja

A tonalidade da cor pode variar um pouco dependendo do ajuste da televisão. A cor 0 (preto) indica ausência de cor.

TABELA DE CONJUNTO DE CORES
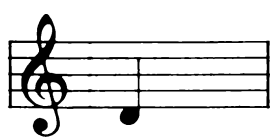
PMODE Nº	CONJUNTO DE COR	COMBINAÇÃO COM DUAS CORES	COMBINAÇÃO COM QUATRO CORES
4	0	preto/verde	-
	1	preto/bege	-
3	0	-	verde/amarelo/azul/vermelho
	1	-	bege/azul pisc./roxo/laranja
2	0	preto/verde	-
	1	preto/bege	-
1	0	-	verde/amarelo/azul/vermelho
	1	-	bege/azul pisc./roxo/laranja
0	0	preto/verde	-
	1	preto/bege	-

TONS MUSICAIS

O computador pode tocar os tons musicais relacionados adiante. Você pode usar um teclado de piano ou outro instrumento musical qualquer para produzir músicas eletrô nicas.

Se estiver usando um piano, o tom do computador para cada tecla, está indicado diretamente sobre a mesma. Por exemplo, o número do tom para o Dó médio (C) é 89.

Se estiver usando outro instrumento, o número do tom de cada nota está indicado embaixo dela. Por exemplo, o número do tom para:



é 108.

Se a nota é um semi-tom abaixo, selecione o número do tom imediatamente anterior.

Por exemplo:

é 99.

Se a nota é um semi-tom acima, selecione o número do tom imediatamente posterior.

Por exemplo.

é 117.

No capítulo IV (BASIC) e XIV (BASIC EXTENDIDO) mostramos como programar o computador para tocar uma canção.

NOTAS MUSICAIS

NÚMERO	NOTA
1	C
2	C≠/D-
3	D/E-
4	E-/D≠
5	E/F-
6	F/E≠

NÚMERO	NOTA
7	F≠/G-
8	G
9	G≠/A-
10	A
11	A≠/B-
12	B

NOTA: O comando PLAY não reconhece a notação "B≠" ou "C-", você tem que usar os números 1 e 12 respectivamente ou substituir C por B≠ e B por C-.
Um erro FC ocorrerá caso tente usar uma dessas notações.

COMANDOS DO VC-50

COMANDO	DESCRIÇÃO	EXEMPLO
ABS	Retorna o valor absoluto da expressão fornecida.	PRINT ABS(-3*4)
ASC	Retorna o código ASCII do primeiro caracter do string fornecido.	PRINT ASC("A") S = ASC (T$)
ATN	Retorna arcotangente em radianos	Y = ATN(X/3)
AUDIO	Liga e desliga a saída do gravador cassete para o auto-falante da televisão	AUDIO ON AUDIO OFF
CHR$	Retorna o caracter ASCII, de controle ou gráfico relativo ao código fornecido.	PRINT CHR$(191) P$ = CHR$(T)
CIRCLE	Desenha um círculo com centro no ponto (x,y), raio y, cor c e a relação altura/comprimento (hw) entre 0 e 4. O círculo pode começar e terminar em um ponto dado (0-1).	CIRCLE (128,96), 50, 4, 1, .5, .75
CLEAR	Reserva espaço de memória para armazenar string (sem o CLEAR, o computador reserva apenas 200 bytes). Se estiver carregando um programa em linguagem de máquina, você pode usar um segundo número para indicar o endere-	CLEAR CLEAR 500 CLEAR 100, 14000

(Continua)

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

COMANDO	DESCRIÇÃO	EXEMPLO
	ço mais alto do BASIC. Este comando também inicializa as variáveis numéricas com zero e as variáveis string com nulos (LEN=0)	
CLOAD	Carrega um programa em BASIC do gravador cassete. Se não for indicado o nome do arquivo, é carregado o primeiro que for encontrado na fita. O nome do arquivo pode ter no máximo 8 caracteres	CLOAD CLOAD "PROGRAM"
CLOADM	Carrega um programa em linguagem de máquina do gravador cassete. Você pode fornecer um número que será somado ao endereço de carregamento do programa.	CLOADM "PROGRAM" CLOADM CLOADM "PROGRAM", 1000
CLOSE	Fecha os arquivos abertos. Cuidado, não se esqueça que este comando esvazia o buffer de comunicação.	CLOSE #-1 CLOSE -2 CLOSE
CLS	Limpa a tela e coloca a cor do código fornecido. Caso não seja indicado nenhum código, é usado a cor verde.	CLS 2 CLS
COLOR	Determina a cor do primeiro plano e a cor do fundo.	COLOR 1,3
CONT	Continua a execução de um programa interrompido por uma instrução STOP ou (BREAK) .	CONT

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
COS	Retorna o cosseno do ângulo medido em radianos.	Y = COS(7)
CSAVE	Salva um programa na fita cassete (o nome do programa pode ter no máximo oito caracteres)... Se for especificado um A após o nome, ele será salvo no formato ASCII.	CSAVE CSAVE "PROGRAM" CSAVE "PROGRAM", A
CSAVEM	Grava um arquivo em linguagem de máquina.	CSAVEM X, 4E, 6F, 5F
DATA	Armazena dados dentro do programa. Use a instrução READ para ler estes dados.	DATA 5, 3, PERAS DATA PAPEL, CANETA
DEF FN	Define uma função numérica	DEF FN(X) = X*3
DEFUSRn	Define o endereço da entrada da função USRn.	DEFUSR5 = 45643
DEL	Permite eliminar linhas de programas. DEL- Elimina todo o programa DEL n Elimina a linha n DEL n- Elimina todas as linhas depois da n DEL -n Elimina todas as linhas até n DEL n-n Elimina todas as linhas entre n e n	 DEL- DEL 25 DEL 25- DEL -200 DEL 10-50

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
DIM	Reserva espaço na memória para vetores matrizes numéricas ou strings.	DIM R(65), W(50) DIM AR$(8,25)
DLOAD	Carrega um programa Basic numa determinada taxa de transmissão (baud). 0 = 300 baud 1 = 1200 baud	DLOAD X,1
DRAW	Desenha uma linha começando num determinado ponto com um determinado comprimento e uma determinada cor. Esta instrução trata também de escalas, traça linhas em branco e executa substrings. Se não for indicado o ponto de início, o computador assume a posição inicial da instrução DRAW anterior ou o ponto (128,96).	DRAW "BM100,100;S10;U25;BR25;ND25;XA$;"
EDIT	Permite reeditar uma linha de programa nC substitui n caracteres nD elimina n caracteres I permite inserir um novo caracter H elimina o resto da linha e permite inserir novos caracteres. L mostra na tela a linha atual e permite correções nSc move o cursor à enésima ocorrência do caracter c. X posiciona o cursor no fim da linha e permite inserir novos caracteres.	EDIT 25

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
	(SHIFT) (↑) sai do controle do comando EDIT n (ESPAÇO) move o cursor n posições para a direta n (←) move o cursor n posições para a esquerda.	
END	Termina o programa. Não é possível usar o comando CONT após uma instrução END.	IF C=5 THEN END END
EOF	Verifica se foi atingido o fim de um arquivo. Se EOF=0 ainda existem dados a serem lidos. Se EOF=1 não há mais dados -1 - gravador cassete 0 - teclado	IF EOF(-1)<> 0 THEN CLOSE IF EOF(-1) = 0 THEN INPUT # -1
EXEC	Transfere o controle para um programa em linguagem de máquina começando no endereço especificado. Se o endereço for omitido, o controle é transferido para o endereço indicado no último comando CLOADM.	EXEC EXEC 32453
EXP	Retorna o exponencial natural (e^n)	Y = EXP(7)
FIX	Retorna a parte inteira	Y = FIX(7.6)
FOR...TO...STEP/NEXT	Cria um loop no programa que o computador repete do primeiro ao último número especificado. Use STEP para indicar de	FOR X=2 TO 5:NEXT X FOR A=1 TO 10 STEP 5: NEXT A

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
	quanto é o incremento a cada passagem pelo NEXT. Se não for indicado o STEP, é usado o número 1.	FOR M=30 TO 10 STEP 5:NEXT M
GET	Lê o conteúdo gráfico de um retângulo para dentro de um vetor para ser usado futuramente pela instrução PUT.	GET (5,20)-(3,8), G
GOSUB	Transfere o controle para uma rotina começando na linha indicada.	GOSUB 1000
GOTO	Desvia a execução do programa para a linha indicada.	GOTO 100
HEX$	Retorna o valor hexadecimal.	PRINT HEX$(30) Y$ = HEX$(X/16)
IF teste THEN ação 1 ELSE ação 2	Faz o teste. Se o resultado for verdadeiro, o computador executa a ação 1. Caso contrário (falso), executa a ação 2.	IF A=5 THEN 30 IF B$="BRANCO" THEN PRINT "CORRETO" ELSE PRINT "ERRADO"
INKEY$	Testa o teclado e fornece o código ASCII da tecla pressionada.	A$ = INKEY$
INPUT	Interrompe a execução do programa e faz o computador esperar a entrada de dados pelo teclado.	INPUT X$ INPUT "NOME";N$

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
INPUT #-1	Lê dados do gravador cassete	INPUT #-1, A
INSTR	Pesquisa a primeira ocorrência do string "alvo" no string "pesquisa" começando na posição "pos". Retorna a posição em que foi achado o "alvo".	PRINT INSTR(5,X$,Y$)
INT	Converte um número fracionário em um inteiro.	X = INT(5.2)
JOYSTK	Retorna a coordenada horizontal ou vertical do joystick esquerdo ou direito. 0 - Coord. horizontal do joystick esquerdo 1 - Coord. vertical do joystick esquerdo 2 - Coord. horizontal do joystick direito 3 - Coord. vertical do joystick direito	X = JOYSTK(0) Y = JOYSTK(1)
LEFT$	Retorna os n primeiros caracteres (contando a partir da esquerda) do string fornecido.	P$ = LEFT$(M$,7)
LEN	Retorna o comprimento em bytes, do string fornecido.	X = LEN(B$)
LET	Atribui um valor a uma variável. Esta instrução é opcional.	LET P$="PRETO"

(Continua)

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

COMANDO	DESCRIÇÃO	EXEMPLO
LIST	Mostra todo ou parte do programa na tela.	LIST LIST 20-30 LIST -100 LIST 1000- LIST 30
LLIST	Imprime todo ou parte do programa na impressora.	LLIST LLIST 50-85 LLIST -30
LINE	Traça uma linha da posição (X1,Y1) até a posição (X2,Y2). Se (X1,Y1) for omitido, é usado o último ponto fornecido ou (128,96). PSET seleciona a cor do primeiro plano e PRESET seleciona a cor do fundo. B traça um retângulo com as coordenadas (X1,Y1) e (X2,Y2) nos cantos opostos. BF preenche este retângulo com a cor do primeiro plano.	LINE (5,3)-(6,6),PSET LINE (10,30)-(20,50),PSET,B LINE (10,30)-(20,50),PSET,BF
LINE INPUT	Lê uma linha inteira do teclado.	LINE INPUT "RESPOSTA";R$
LOG	Retorna o logarítmo natural (base e).	Y = LOG(353)
NEM	Indica qual o espaço que ainda permanece livre na memória do computador.	PRINT NEM
MID$	Retorna um substring do string fornecido começando na posição indicada com um comprimento determinado.	PRINT MID$ (A$,5,8)

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
MID$	Substitui uma parte de um string por outro string.	MID$(A$,14,2)="KS"
MOTOR	Liga ou desliga o gravador cassete.	MOTOR ON MOTOR OFF
NEW	Apaga o conteúdo da memória	NEW
ON...GOSUB...	Transfere o controle para uma das sub-rotinas especificadas dependendo do valor da expressão.	ON Y GOSUB 50,100
ON...GOTO...	Desvia a execução do programa para uma das linhas indicadas dependendo do valor da expressão.	ON X GOTO 100,200
OPEN	Abre um arquivo da seguinte forma: 0 - Teclado -1 - Gravador cassete -2 - Impressora A abertura pode ser: I - Entrada O - Saída	OPEN "I",0,"ARQ" OPEN "O",-1,"DADOS"
PAINT	Desenha um gráfico na tela começando no ponto (X,Y) com uma cor especificada e terminando no bordo da cor especificada	PAINT (10,30),4,2
PCLEARn	Reserva n páginas de memória para gráficos.	PCLEAR 8 PCLEAR 1

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
PCLSc	Limpa e coloca a cor c na tela. Se o código for omitido, é mantida a cor de fundo atualmente em uso (tela-gráfica).	PCLS 3
PCOPY	Copia gráficos da página origem para a página destino.	PCOPY 5 TO 6
PEEK	Retorna o conteúdo do endereço de memória especificado	A = PEEK(32706)
PLAY	Soa o tom da nota (A-G ou 1-12), oitava (O), volume (V), comprimento de nota(L), tempo(T), pausa(P) e permite a execução de substrings.	PLAY "L1;A#;P8;V10;T3;L2;B-;9;XA$;"
PMODE	Indica a resolução gráfica da página de memória reservada para gráficos.	PMODE 4,3
POINT	Testa se o ponto indicado está ativado. O valor retornado pode ser: -1 - o ponto está em modo texto 0 - o ponto está desativado Cod - código da cor que o ponto possui	IF POINT(15,12)=3 THEN PRINT "AZUL"
POKE	Coloca um código (0-255) no endereço de memória especificado	POKE 15872,255
POS	Retorna a posição atual de impressão	PRINT TAB(8) POS(0)

(Continua)

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

COMANDO	DESCRIÇÃO	EXEMPLO
PPOINT	Testa se o ponto gráfico está ativado ou desativado e retorna o código da sua cor.	PPOINT (13,35)
PRESET	Devolve a cor de fundo para um determinado ponto.	PRESET(5,6)
PRINT	Imprime a lista de dados fornecida no periférico indicado. Se você não indicar o periférico, os dados serão mostrados na tela. Consulte o comando OPEN para conhecer os códigos dos periféricos.	PRINT "VC 50" PRINT B$ PRINT #-1,A$,B$ PRINT #-2,A$;"VC 50"
PRINT#-1	Transfere dados para o gravador cassete	PRINT#-1,A
PRINT#-2	Transfere dados para a impressora.	PRINT#-2,CAP$
PRINT TAB	Move o cursor para uma determinada coluna antes de imprimir os dados.	PRINT TAB(5)"NOME"
PRINT USING	Imprime os números num formato pré-determinado. # campo numérico (um dígito por #) . ponto decimal , imprime uma vírgula a cada grupo de três algarismos ** preenche os espaços a esquerda com asteríticos. $ imprime um símbolo $ antes do número.	PRINT USING "# # # #"; 62.2 PRINT USING "##.#";58,6 PRINT USING "####,"; 44.0 PRINT USING "**##.##"; 33.3 PRINT USING "$##.##"; 33.3

(Continua)

COMANDO	DESCRIÇÃO		EXEMPLO
	$$	símbolo $ flutuante	PRINT USING "$$##.#"; 11.544
	**$	símbolos $ e * flutuantes	PRINT USING "**$##. ###";8.444
	+	na primeira posição, imprime o sinal do número no início. Na última posição imprime o sinal logo após o número.	PRINT USING"+##.#";-216
	↑↑↑↑	formato exponencial	PRINT USING "##.#↑↑↑↑"; 546
	-	sinal negativo depois de um número negativo.	PRINT USINT "##.#-"; -534.7
	!	imprimeo primeiro caracter deum string.	PRINT USING "!";"AZUL"
	% espaços %	imprime um string com um comprimento igual a dois, mais o número de espaços entre os símbolos de percentagem.	PRINT USING "% %"; "AMARELO"
PRINT@	Imprime os dados na tela, a partir daposição especificada.		PRINT@ 256,"OI";A$
PSET	Ativa a posição (x,y) da tela com a cor c. Se omitido o código da cor, é usada a cor do primeiro plano.		PSET (5,6,3)
PUT	Transfere o gráfico da partição origem para um retângulo na tela. O tamanhodo retângulo tem que ser igual ao retângulo gerado com a instrução GET.		PUT (3,2)-(5,6),V,PSET

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
READ	Lê o próximo item de uma instrução DATA e atribui o dado a uma variável especificada.	READ A$ READ C,B
REM	Permite a colocação de comentários num programa. Tudo que for digitado após o REM será ignorado pelo computador.	REM ISTO É IGNORADO
RENUM	Permite renumerar as linhas de um programa.	RENUM 1000,5,100
RESET	Desativa o ponto ativado pelo comando SET.	RESET (14,15)
RESTORE	Retorna o indicador da instrução DATA para a sua posição inicial.	RESTORE
RETURN	Transfere o controle do programa para a linha imediatamente abaixo do último GO SUB executado.	RETURN
RIGHT$	Retorna os n primeiros caracteres (contando a partir da direita) do string fornecido.	D$ = RIGHT$(A$,5)
RND	Retorna um número inteiro aleatório entre 1 e o número indicado.	A = RND (20)
RUN	Executa um programa.	RUN

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
SCREEN	Determina o modo da tela (0=texto e 1= gráfico) e o conjunto de cores que será usado (0 ou 1).	SCREEN 1,0
SET	Ativa a posição especificada da tela na cor indicada.	SET(14,13,3)
SGN	Retorna o sinal da expressão numérica fornecida. -1 - se a expressão for negativa 0 - se a expressão for zero 1 - se a expressão for positiva	X = SGN(A*B)
SIN	Retorna o seno do ângulo dado em radianos.	Y = SIN (5)
SKIPF	Pula para o próximo programa do cassete ou para o fim do programa indicado.	SKIPF SKIPF "PGM"
SOUND	Determina um tom com uma determinada duração.	SOUND 128,3
STOP	Interrompe a execução de um programa	STOP
STRING$	Retorna um string de comprimento c contendo um código ASCII ou o primeiro caracter do string fornecido.	A$=STRING$(5,"%")
STR$	Converte uma expressão numérica em string.	S$=STR$(X)

(Continua)

COMANDO	DESCRIÇÃO	EXEMPLO
SQR	Retorna a raiz quadrada.	S = SQR (5+3)
TAB	Inicia a impressão dos dados a partir da coluna indicada.	PRINT TAB(2) "VC 50" PRINT#-2, TAB(5); A$
TAN	Retorna a tangente do ângulo medido em radianos.	Y = TAN (45.2)
TIMER	Retorna o conteúdo do cronômetro ou inicializa-o (0-65535)	PRINT TIMER TIMER = 0
TROFF	Desativa o rastejador de programa (program tracer).	TROFF
TRON	Ativa o rastejador de programa (program tracer).	TRON
USR	Chama uma rotina em linguagem de máquina.	X = USR(Y)
VAL	Converte um string em um número.	A = VAL(B$)
VARPTR	Retorna o endereço de uma variável.	Y = USR(VARPTR(X))

MENSAGENS DE ERRO

/0 Divisão por zero. Foi pedido ao computador para dividir um número por zero, e esta operação é impossível.

AO Tentativa de abrir um arquivo de dados que já se encontra aberto (**A**lready **O**pen).

BS O índice de um vetor está fora dos limites admissíveis. Use a instrução DIM para dimensionar os vetores. Por exemplo, se você escrever no seu programa A(12) sem especificar na instrução DIM que A é um vetor de 12 posições ou mais, você obterá este erro (**B**ad **S**ubscript).

CN Não é possível continuar o programa. Se você utilizar o comando CONT estando no fim (END) do programa, este erro ocorrerá (**C**an **N**ot).

DD Tentativa de redimensionar um vetor. Um vetor só pode ser dimensionado uma vez. Por exemplo, você não pode ter DIM A(12) e DIM A(50) no mesmo programa (**D**imension).

DN Erro no número do periférico. Você só pode usar três tipos de periféricos com os comandos OPEN, CLOSE, PRINT ou INPUT;0,-1 ou -2. Se você tentar usar outro número ocorrerá este erro (**D**evice **N**umber).

DS Instrução direta. Existe uma instrução direta no arquivo. Este erro ocorre se você carregar um programa cujas instruções não tenham número de linha (**D**irect **S**tatement)

FC Chamada incorreta de função. Este erro ocorre quando você usa uma palavra BASIC com um parâmetro (número) fora dos limites admissíveis. Por exemplo SOUND (260, 260) e CLS(10) provocarão este erro. A instrução RIGHT$(S$, 20) também provoca este erro se a variável S$ possuir menos que 20 caracteres. Existem outras situações em que ocorre este erro, como por exemplo, quando usamos vetores com índices negativos (A(-1)) ou quando chamamos funções USR sem termos fornecido (POKE) o seu endereço antes. (**F**unction **C**all)

FD Formato de campo incorreto. Este erro ocorre quando você lê (INPUT) ou grava (PRINT) um dado usando um formato inadequado. Se tentar, por exemplo, ler um string com a instrução INPUT#1,A este erro ocorrerá. (**F**ile **D**ata).

FM Tipo de arquivo incorreto. Este erro ocorre quando você tenta ler (INPUT) dados de um arquivo aberto para saída (OUTPUT), ou tenta gravar (PRINT) dados num arquivo aberto para entrada (INPUT). (**F**ile **M**ode)

ID Instrução direta incorreta. Você só pode usar a instrução INPUT dentro de um programa. Não é possível usá-la como um comando. (**I**llegal **D**irect statement)

IE Tentativa de ler um dado além do fim do arquivo. Use a instrução EOF para descobrir se você atingiu o fim de um arquivo. Quando atingir o fim do arquivo feche-o (CLOSE). (**I**nput past **E**nd)

IO Erro de entrada ou saída. O computador está encontrando dificuldades para ler ou gravar informações num arquivo em fita. (**I**nput/**O**utput)

LS String muito longo. Um string pode ter no máximo 255 caracteres (**S**tring too **L**ong).

NF Instrução NEXT sem FOR correspondente. Você usou uma instrução NEXT sem a instrução FOR correspondente. Este erro também ocorre se você inverter a ordem das instruções NEXT num ninho de FOR/NEXT (**N**ext without **F**or).

NO O arquivo não foi aberto. Você não pode ler ou gravar dados num arquivo sem antes abrí-lo (OPEN) (**N**ot **O**pen).

OD Não há dados suficientes nas instruções DATA para atender a todas as instruções READ. Verifique as instruções DATA e READ do seu programa (**O**ut of **D**ata).

OM Estouro de memória. Você já usou toda a memória disponível do computador (**O**ut of **M**emory).

OS Estouro de string. Não há espaço suficiente na memória do computador para executar as operações envolvendo strings. Use a instrução CLEAR no começo do programa para reservar mais espaço para os strings. (**O**ut of **S**tring).

OV Estouro de variável numérica. Seu programa gerou número muito grande e o computador não tem condições de processar este número (ABS(X)>IX 38) (**OV**erflow).

RG Instrução RETURN sem o GOSUB correspondente. Você usou uma instrução RETURN sem a instrução GOSUB correspondente. (**R**eturn without **G**osub)

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

SN Erro de sintaxe. Este erro ocorre quando você escreve incorretamente uma palavra em BASIC ou usa incorretamente a pontuação, os parênteses ou um caracter ilegal. Digite novamente a linha incorreta . (**Si**ntaxe)

ST Tentativa de usar strings em fórmulas muito complexas. Uma operação com string está muito complexa para ser processada pelo computador. Desdobre esta operação em outras de menor porte. (**ST**ring)

TM Incompatibilidade de tipo de variável. Este erro ocorre quando você tenta atribuir um número a uma variável string (A$=3) ou um string a uma variável numérica (A= "VC50"). (**T**ype **M**ismatch)

UL Linha não definida. Existe no programa uma instrução GOTO ou GOSUB ou qualquer outra instrução de desvio pedindo ao computador que vá para uma linha cujo número não existe (**U**nderfined **L**ine).

MAPA DE MEMÓRIA

HEXADECIMAL	DECIMAL	RESUMO
0000-03FF	0000-1023	RAM usada pelo interpretador Basic
0400-05FF	1024-1535	RAM VIDEO (pode ser movida)
0600-0FFF	1536-4095	RAM disponível
1000-3FFF	4096-16383	RAM disponível
4000-7FFF	16384-32767	RAM disponível
8000-9FFF	32768-40959	Basic Extendido e 2º banco RAM
A000-BFFF	40960-49151	Basic Extendido e 2º banco RAM
C000-FEFF	49152-65279	ROM PACK (CARTUCHO) e 2º banco RAM
FF00-FFFF	65280-65535	Input/Output e controle

		DETALHADO
0003	3	Contador geral
0006	6	Flag de string
0007	7	Flag de coleta de erros
0019-001A	25-26	Pointer início de programa Basic
001B-001C	27-28	Pointer topo do programa e início variáveis
001D-001E	29-30	Pointer topo variáveis e início arrays
001F-0020	31-32	Pointer fim arrays e início memória livre
0021-0022	33-34	Topo do Stack e início do conjunto de strings
0023-0024	35-36	Início da área usada para conjunto de strings
0025-0026	37-38	Pointer para limite da memória (Basic)
0027-0028	39-40	Fim do conjunto strings e início do espaço livre
0033-0034	51-52	Pointer para atual posição de leitura de dados
0037-0038	55-56	Nome da variável atual
0041	65	Maior endereço bloco movido
0043	67	Maior endereço bloco a ser movido
0045	69	Menor endereço bloco movido
0047	71	Menor endereço bloco a ser movido

004B	75	Endereço do drescritor da maior string achada
004F-0054	79-84	Acumulador de ponto flutuante ≠ 1 (FPAC ≠ 1) (6 bytes)
0056	86	Tamanho da string
005C-0061	92-97	Acumulador de ponto flutuante ≠ 2 (FPAC ≠ 2) (6 bytes)
0062	98	Comparação de sinais
0063	99	Byte de precisão extendida
0068-0069	104-105	Linha de programa atual
006C	108	Posição de coluna atual
006F	111	Dispositivo para saída de caracteres (0 = Screem, HFE = Printer, HFF = Tape, 1-16 = Arquivo de disco)
0070	112	Flag de EOF (fim de arquivo) do tape
0071	113	Flag de reset (H55 para reset quente)
0072-0073	114-115	Pointer de reestart (H80C0-Basic)
0074-0075	116-117	Pointer para fim de memória
0078	120	Modo de arquivo (0 = Nada, 1 = Entrada (imput) 2 = Saída (output))
0079	121	Tamanho do Buffer de trabalho do tape
007A-007B	122-123	Pointer para Buffer de trabalho do tape
007C	124	Tipo de bloco de arquivo do tape (0 = Header, 1 = Dados, HFF=EOF)
007D	125	Número de bytes de dados no bloco de I/O do cassete
007E-007F	126-127	Endereço de fim de programa após CLOADM
0080	128	Checksum (conferência de erros de leitura)
0081	129	Número do erro referente a cassete
0082	130	Contador Geral
0083	131	Contador de largura de pulso
0084	132	Flag de subida descida (Rise/Fall)
0085	133	Último valor do seno
0087	135	Última tecla pressionada
0088-0089	136-137	Pointer para posição atual do cursor
008A-008B	138-139	Número de tentativas de leitura serial
008C	140	Frequência do som
008D-008E	141-142	Duração do som
008F	143	Início da área carregada pela ROM
0092	146	Controle do tamanho do carrier não modulado que precede I/O de cassete
0094	148	Cor do cursor

0095-0096	149-150	(Alto e Baixo) Bytes do código de Baud Rate (normalmente H0057)
0097-0098	151-152	Atraso de retorno do carro da impressora (normalmente H0001)
0099	153	Tamanho do campo da vírgula (normalmente H10)
009A	154	Último campo da vírgula (normalmente H70)
009B	155	Tamanho da linha impressora (normalmente H84)
009C	156	Afeta posição do campo da vírgula (normalmente H00)
009D-009E	157-158	Endereço para transferência após CLOADM
009F	159	Início da sub-rotina para obter próximo caracter
00A5	165	Início da sub-rotina para obter o mesmo caracter
00A6	166	Pointer para próximo caracter
00A8-00AA	168-170	Vetor de JUMP para imprimir OK
00AB-00AE	171-174	Área de produto extendido
00AF	175	Flag de TRACE
00B5	181	Cor atual
00B6	182	PMODE atual
00B7-00B8	183-184	Endereço de fim da tela 1
00B9	185	Número de bytes por linha
00BA-00BB	186-187	Endereço da página de gráficos (atual)
00BC	188	HE = com disk driver, H6 = sem disk driver
00BD	189	X_1 (de gráficos)
00BF	191	Y_1 (de gráficos)
00C1	193	Set de cores (1 a 8)
00C3	195	X_2 (de gráficos)
00C5	197	Y_2 (de gráficos)
00D7	215	Temp
00DB	219	Flag de alteração
00E6	230	Baud Rate para DLOAD
00E7	231	Constante de temporização de INPUT
00EA	234	Código da operação
00EB	235	Número do driver
00EC	236	Track do disco
00ED	237	Setor do disco
00EE	238	Endereço do Buffer
00F0	240	Status retornado

0100-0102	256-258	Interrupt 3 de software chamado pelo vetor em HFFF2
0103-0104	259-260	Interrupt 2 de software chamado pelo vetor em HFFF4
0105-0108	261-264	Interrupt 1 de software chamado pelo vetor em HFFFA
0109-010B	265-267	Interrupção não mascarável chamada pelo vetor em HFFFC setada para HD7AE pelo DOS
010C-010E	268-270	Requisição de interrupt chamada pelo vetor em HFFF8 setado para HA9B3 pelo Basic Extendido e para HD7BC pelo DOS
010F-0111	271-273	Vetor de interrupção rápida chamado pelo vetor em HFFF6 setado para HA0F6
0113	275	MSB e LSB bytes do TIMER
0116-0117	278-279	Usado pela função RND
011A	282	Flag para Lock de Shift
011C	284	Constante de retorno do teclado
011D-011F	285-287	Vetor de salto para H8489 - Print OK
0120-013C	288-316	Tabela de códigos tokenizados Byte 1 = Nº de palavras reservadas (comandos) Byte 2,3 = Endereço da tabela Byte 4,5 = Endereço da sub-rotina
0120-0124	288-292	Comandos Basic
0125-0129	293-297	Funções Basic
012A-012E	298-302	Comandos Basic Extendido
012F-0133	303-307	Funções Basic Extendido
0134-0138	308-312	Comandos Basic de disco
0139-013C	313-316	Funções Basic de disco
013E-013F	318-319	Endereço para USR 0
0140-0141	320-321	Endereço para USR 1
0142-0143	322-323	Endereço para USR 2
0144-0145	324-325	Endereço para USR 3
0146-0147	326-327	Endereço para USR 4
0148-0149	328-329	Endereço para USR 5
014A-014B	330-331	Endereço para USR 6
014C-014D	332-333	Endereço para USR 7
014E-014F	334-335	Endereço para USR 8
0150-0151	336-337	Endereço para USR 9
0152-0159	338-345	Tabela de rollover do teclado

015A-015D	346-349	Leitura dos joysticks
015A	346	Joystick esquerdo / acima - abaixo
015B	347	Joystick esquerdo / esquerda - direita
015C	348	Joystick direito / acima-abaixo
015D	349	Joystick direito / esquerda-direita
015E-0160	350-352	Dispositivo aberto (periféricos)/chamado em HA5F6/setado para HC426 pelo DOS
0161-0163	353-355	Checa número do dispositivo (periférico)/chamado em HA5B9/setado para HC838 pelo DOS
0164-0166	356-358	Retorna parâmetros do dispositivo/chamado em HA35F / setado para HC843 pelo DOS
0167-0169	359-361	Saída de caracter / chamado em HA282 / setado para H8273 pelo Extendido / setado para HCB4A pelo DOS
016A-016C	362-364	Entrada de caracter / chamado em HA176 / setado para HBCF1 pelo Extendido / setado para HC58F pelo disco
016D-016F	365-367	Check de arquivo aberto (OPEN) para entrada / chamado em HA3ED / setado para HC818 pelo DOS
0173-0175	371-373	Fecha todos os arquivos abertos / chamado em HA426 / setado para HCA3B pelo DOS
0176-0178	374-376	Fecha um arquivo / chamado em HA42D / setado para H8286 pelo Extendido / setado para HCA4B pelo DOS
0179-017B	377-379	Print using / chamado em HB918 / setado para H8E90 pelo Extendido
017C-017E	380-382	Scanner de item de arquivo / chamado em HB061 / setado para HCC5B pelo disco
017F-0181	383-385	Checa tecla Break / chamado em HA549 / setado para C859 pelo DOS
0182-0184	386-388	Obtém linha do teclado / chamado em HA390 / setado para JMP RTS pelo DOS
0185-0187	389-391	Fim de carregamento (LOAD) de arquivo ASCII / chamado em HA4BF / setado para HCA36 pelo DOS
0188-018A	392-394	Checa fim de arquivo (End Of File) / chamado em HA5CE / setado para HC860 pelo DOS
018B-018D	395-397	Avalia operando / chamado em HB223 / setado para H8846 pelo Extendido / setado para HCDF6 pelo DOS

018E-0190	398-400	Erro de uso /chamado em HAC46 / setado para JMP RTS pelo DOS
0191-0193	401-403	Erro /chamado em HAC49 / setado para H88F0 pelo Extendido / setado para HC24D pelo DOS
0194-0196	404-406	RUN / chamado em HAE75 / setado para H829C pelo Extendido / setado para HC990 pelo DOS
0197-0199	407-409	Hex e Octal / chamado em HBD22 / setado para H87E5 pelo Extendido
019A-019C	410-412	Executa linha / chamado em HAD9E / setado para H82B9 pelo Extendido
019D-019F	413-415	Endereço gráfico / chamado em HA8C4
01A0-01A2	416-418	CLS, GET, PUT, etc / chamado em HA910, H975C, H8AFA, H8162 / setado para HC29A pelo DOS
01A3-01A5	419-421	Tokeniza / chamado em HB821 / setado para H8304 pelo Extendido
01D1	465	Tamanho de arquivo em fita
01D2-01D9	466-473	Nome do arquivo em fita
01DA-02D8	474-728	Buffer do cassete
01DA-01E1	474-481	Nome do arquivo para CLOADM
01E5-01E6	485-486	Endereço de Exec para cassete
01E7-01E8	487-488	Endereço de LOAD para TAPE
02DC	732	TOKEM da primeira palavra do comando Basic
02DD-03DC	733-988	I/O Buffer do console
0400-05FF	1024-1535	Tela de baixa resolução
0600-35FF	1536-13840	Possíveis telas de gráficos
0600	1536	Início de área de programa (sem disco)
0600-06FF	1536-1791	Buffer de disco
0700-07FF	1792-2047	Buffer de disco
0800-0927	2048-2343	Tabela do driver
097E	2430	Tabela do Track atual
0982	2434	Flag do NMI em uso
0983	2435	NMI JMP
0985	2437	Contador de parada do motor
0986	2438	Latch de dados atual
0C00	3072	Início de área de programa (com disco)
0FFF	4095	Topo dos 4 K de RAM

3FFF	16383	Topo dos 16 K de RAM
7FFF	32767	Topo dos 32 K de RAM
FFFF	65535	Topo dos 64 K de RAM (banco 2 selecionado ao invés de ROM)
8000-9FFF	32768-40959	ROM de Basic Extendido
807F	32895	Reset frio (início) para Basic sem procura de tamanho e inicialização de área de trabalho. Reseta pointers para início do programa Basic.
80C0	32960	Reset quente para Basic. Não reseta pointers para início do programa Basic
8183-81EF	33155-33263	Tabela de tokens de comandos Basic Extendido
81F0-821D	33264-33309	Endereço de entrada da sub-rotina
821E-8256	33310-33366	Tabela de TOKEN de funções de Basic Extendido
8257-8272	33367-33394	Endereço de entrada da sub-rotina
82B9	33465	Rotina de Break ou Stop
82BB	33467	Loop do interpretador do Basic Extendido
8378	33656	COS (cosseno)
8381	33665	TAN (tangente)
83B0	33712	ATN (arco tangente)
8446	33862	LOG (logaritmo)
8480	33920	SQR (Raiz quadrada)
84F2	34034	EXP (exponencial)
8524	34084	FIX
8533	34099	EDIT
86A7	34471	TRON (TRACE ON)
86A8	34472	TROFF (TRACE OFF)
86AC	34476	POS (position)
86BE	34494	VARPTR (variable pointer)
874E	34638	STRING$
877E	34686	INSTR
8871	34929	DEF (define)
8968	35176	TIMER
8970	35184	DEL (delete)
8A09	35337	RENUM
8BDD	35805	HEX$
8C18	35864	DLOAD (Down Load)

8DBC	36284	Input caracter modo serial
8E06	36358	Output caracter modo serial
928F	37519	Acha BYTE / Rotina de BIT
92A6	37542	BYTE / BIT; PMODES 0,2,4
92C2	37570	BYTE / BIT; PMODES 1,3
92DD	37597	Tabelas de BIT
9339	37689	PPOINT
9361	37729	PSET
9365	37733	PRESET
93BB	37819	LINE
9444	37956	Desenha Linha Horizontal
946C	37996	Desenha Linha Vertical
94A1	38049	Desenha Linha
94E2	38114	Loop de desenho de linha
9506	38150	Rotinas de mover acima, abaixo, esquerda e direita
9532	38194	PCLS
9546	38214	COLOR
9621	38433	PMODE
9670	38512	SCREEN
968B	38539	PCLEAR
9710	38672	Compara dois pontos
9723	38691	PCOPY
9755	38741	GET
9758	38744	PUT
98EC	39148	PAINT
9A22	39458	PLAY
9CB6	40118	DRAW
9E9D	40605	CIRCLE
A000-BFFF	40960-49151	ROM de BASIC
A000-A001	40960-40961	Endereço de checagem de teclado
A002-A003	40962-40963	Endereço de saída de caracter
A004-A005	40964-40965	Endereço de leitura de cassete
A006-A007	40966-40967	Endereço de entrada de bloco
A008-A009	40968-40969	Endereço de saída de bloco
A00A-A00B	40970-40971	Endereço de entrada de Joystick
A00C-A00D	40972-40973	Endereço de saída de título (Header)
A00E	40974	Reset secundário

A027	40999	Reset primário
A06E	41070	Hard start (início após reset)
A0A6	41126	Checa presença de ROM de disco (DOS)
A0CB	41163	Checa presença de ROM de Basic Extendido
A0D7	41175	Imprime versão
A0E8	41192	Soft start (após reset)
A0F6	41206	FIRQ (first interrupt request) (checa ROM PACK)
A10D	41229	Início da área descarregada na RAM em H8F
A171	41329	Input caracter, Bit 7 limpo, (CLEAR)
A176	41334	Input caracter
A199	41369	Cor do cursor
A1B1	41393	Espera por tecla e leitura de teclado O caracter é retornado no Registro A
A1C1	41409	Checa teclado e obtém tecla se pressionada Z = 1, A = 0 Se não pressionada Z = 0, A = Tecla, B e X preservados
A26E	41582	Tabela de códigos de teclas não alfanuméricas
A282	41602	Saída de caracter para o dispositivo especificado por H6F, todos registros menos CC são preservados
A2BF	41663	Saída de caracter no registro A para impressora
A30A	41738	Saída de caracter no registro A para tela
A390	912	Entra linha do teclado no Buffer H02DD / Retorna registro X com H02DC; ou zero se fim do Buffer
A416	42006	CLOSE
A44C	42060	CSAVE
A46C	42092	Realiza CSAVEM / Requer início do bloco de memória em H-19-HA0 e em H01E7-H01E8 Endereço de transferência em H01E5-H01E6 e nome do arquivo em H01D2-H01D9. Entra com A = 2 e X = 0
A498	42136	CLOAD
A4FE	42238	CLOADM
A53E	42302	EXEC
A564	42340	INKEY$
A59A	42394	Transfere Blocos
A5CE	42446	EOF (End Of File)
A5EC	42476	SKIPF

A5F6	42486	OPEN
A629	42537	Abre arquivo em fita
A681	42625	Acha nome do arquivo
A6FE	42750	Marca piscante no lado da tela
A701	42753	Lê bloco da fita
A70B	42763	Lê bloco do cassete, deve estar em ON e em BIT SYNC / $7C contém tipo de bloco; 0 = Header; 1 = dados; HFF = EOF / H7D contém nº de bytes de dados no arquivo (0-HFF) Z = 1, A = 0 Se não houver erro Z = 0, A = 1 Se houver erro de checksum Z = 0, A = 2 Se houver erro de memória X = tamanho do buffer do bloco de início se não houver erro. U e Y são preservados
A77C	42876	Liga cassete e inicia sincronismo de Bit para leitura. U e Y são preservados, FIRQ e IRQ são mascarados
A7BD	42941	Motor
A7D8	42968	Liga cassete e grava rótulo
A7E5	42981	Grava arquivo na fita
A7F4	42996	Grava bloco na fita (velocidade e rótulo gravados); H7E = endereço do buffer; H7C = Tipo de bloco; H7D = Número de Bytes de dados / X = endereço do Buffer de Bytes de dados / Todos os registros modificados
A85C	43100	Tabela de seno para saída para cassete
A880	43136	SET
A8B1	43185	RESET
A8F5	43253	POINT
A910	43280	CLS
A928	43304	Limpa tela e leva o cursor para posição inicial (HOME)
A937	43319	Imprime copyright (CLS 9)
A94B	43339	SOUND
A956	43350	Gera som
A992	43410	AUDIO
A9B3	43443	Processador de interrupção (60 HZ)

A9C6	43462	JOYSTICK
A9DE	43486	Lê e memoriza valor dos joysticks **esquerdo:** acima/abaixo em H15A esquerda/direita em H15B **direito:** acima/abaixo em H15C esquerda/direita em H15D Y é preservado
AA29	43561	Tabela de endereços de funções
AA51	43601	Tabela de operações para +, -, *, /, , AND, OR, 3 bytes cada (endereço e valor precedente)
AA66	43622	Tabela de nomes dos comandos
AB1A	43802	Tabela de nomes das funções
AB67	43879	Tabela de endereços dos comandos
ABAF	43951	Tabela de códigos de erros
ABE1	44001	Strings de texto
ABF9	44025	Stack de procura para GOSUB ou FOR
AC1E	44062	Espaço na memória
AC20	44064	Move bloco de memória começando no topo H41-H42 é o endereço do topo do destino H43-H44 é o endereço do topo da fonte H45-H46 é o endereço do fim do destino H47-H48 é o endereço do fim da fonte
AC46	44102	Manipulação de erro
AC73	44147	Loop interno
AD17	44311	NEW
AD19	44313	Executa NEW
AD47	44359	FOR
AD9E	44446	Loop do interpretador
ADC6	44486	Executa linha
ADE4	44516	RESTORE
ADEB	44523	Checa tecla BREAK ou pausa SHIFT @
AE02	44546	END
AE09	44553	STOP
AE30	44592	CONT (Continue)
AE41	44609	CLEAR
AE75	44661	RUN

AE86	44678	GO
AE92	44690	GOSUB
AEA4	44708	GOTO
AEC0	44736	RETURN
AEE0	44768	DATA
AEE3	44771	REM ou
AEE8	44776	ELSE
AF14	44820	IF
AF42	44866	ON
AF67	44903	Obtém inteiro não assinalado
AF89	44937	LET
AFF5	45045	INPUT
B046	47686	READ
B0F8	45304	NEXT
B156	45398	Obtém expressão
B1CB	45515	Outra entrada para tabela de operações
B223	45603	Obtém operando
B290	45712	Executa funções
B2D4	45780	Operações AND e OR
B2F4	45828	Operações relacionais (<>=)
B34E	45903	DIM (Dimenciona)
B38F	45968	Criação de variáveis
B3E4	46068	Avalia expressões inteiras
B3ED	46092	Converte nº no FPAC (acumulador de ponto flutuante) em inteiro de complemento de dois com 16 bites (no registro D) Ocorre overflow se >+32767 ou <-32768
B4EE	46318	MEM
B4FD	46333	STR$
B518	46360	Obtém string
B56D	46445	Rotina de alocação de string
B591	46481	Coleta de não reconhecível (raro)
B5D8	46552	Processa um descritor
B5EF	46575	Compacta um string
B681	46721	LEN
B68C	46733	CHR$

B6A0	46752	ASC
B6AB	46763	LEFT$
B6C8	46792	RIGHT$
B6CF	46799	MID$
B716	46869	VAL
B750	46928	PEEK
B757	46935	POKE
B75E	46942	LLIST (comando)
B764	46948	LIST (comando)
B7C2	47042	DESTOKENIZA
B7E6	47078	Destokeniza um TOKEM
B821	47137	Tokeniza
B892	47250	Tokeniza uma palavra
B8F7	47351	PRINT
B97E	47486	TAB
B99C	47516	Imprime texto de string
B9AC	47532	Imprime espaço
B9B4	47540	Início das rotinas de arredondamento com ponto flutuante
B9B9	47545	Subtrai do FPAC1, (acumulador de ponto flutuante 1)
B9C2	47554	Adiciona no FPAC1
BA79	47737	Complemento de dois no FPAC1
BAC5	47813	Constante 1.0
BACA	47818	Multiplica
BB2F	47919	Move reg. X para FPAC2
BB7D	47997	Constante 10.0
BB91	48017	Divide
BC4A	48202	Move FPAC2 para FPAC1
BC5F	48223	Move FPAC1 para FPAC2
BC6D	48237	Testa FPAC1 para zero e sinal
BC7A	48250	SGN (sign)
BC93	48275	ABS (valor absoluto)
BCEE	48366	INT (inteiro)
BD12	48402	Converte String para ponto flutuante
BDB6	48566	Constantes 99999999.9, 999999999 e 1E09
BDCC	48588	Display o valor decimal do registro D

BDD9	48601	Converte FPCAl para ASC II
BEC0	48832	Constante 0,5
BEC5	48837	Série de constantes de 4 Bytes
BF1F	48927	RND (RANDOM)
BF78	49016	SIN (SENO)
BFBD	49085	Constantes 2π; 0,25
BFC8	49096	Série de constantes de 5 bytes
BFF2	49138	Vetores de interrupt e reset
BBF2-BBF3	48114-48115	SWI 3
BBF4-BBF5	48116-48117	SWI 2
BBF6-BBF7	48118-48119	FIRQ
BBF8-BBF9	48120-48121	IRQ
BBFA-BBFB	48122-48123	SWI 1
BBFC-BBFD	48124-48125	NMI
BBFE-BBFF	48126-48127	RESET
C000-D7FF	49152-55295	ROM do DOS
C004	49156	Endereço do DSKCON
C0D4	49364	Start quente para Basic de disco (DOS)
C17F-C1DA	49535-49626	Tabela de TOKEN de comandos DOS
C6C2	50882	KILL
C932	51506	SAVE
C98B	51595	MERGE
C99A	51612	LOAD
CBCF	52175	DIR
CD1A	52506	CVN
CD28	52520	MKN$
CD36	52534	LOC
CD5B	52571	LOF
CDC0	52672	FREE
CDE9	52713	DRIVE
CF3F	53055	RENAME
CF8A	53130	WRITE
CFE0	53216	FIELD
D025	53285	RSET
D026	53286	LSET
D080	53376	FILES
D146	53574	UNLOAD

D175	53624	BACKUP
D2CC	53964	COPY
D3FF	54271	DSKI$
D474	54388	DSKO$
D4AB	54443	DSKINI
D65B	54875	VERIFY
D66C	54892	DSKCON
D6C5	54981	RESTORE
D6FD	55037	Retardo 78 m seg
D705	55045	Setor de leitura / escrita
D7A2	55202	Tabela de endereços de comandos
D7AA	55210	Tabela de Bit para os drivers
D7AE	55214	Manipulação de NMI
D7BC	55228	Manipulação de IRQ
FF00-FFFF	65280-65535	I/O e controle (input/output e controle)
FF00-FF03	65280-65283	PIA (1) (Periferal Interface Adapter)
FF00	65280	Bit 0 - Linha 1 do teclado e botão de joystick direito Bit 1 - Linha 2 do teclado e botão do joystick esquerdo Bit 2 - Linha 3 do teclado Bit 3 - Linha 4 do teclado Bit 4 - Linha 5 do teclado Bit 5 - Linha 6 de teclado Bit 6 - Linha 7 de teclado Bit 7 - Entrada de comparação do joystick
FF01	65281	Bit 0 - Controle do bloco de sincronismo horizontal (≅ 64 µseg) Bit 1 - Entrada de interrupt Bit 2 - Normalmente 1 0 = muda FF00 para registro de direção de dados Bit 3 - SEL 1 (select) LSB das duas linhas multiplexadas de select Bit 4 - Sempre 1 Bit 5 - Sempre 1 Bit 6 - Não usado Bit 7 - Flag de interrupt para sincronismo horizontal.

FF02	65282	Bit 0 - Coluna 1 - Teclado
		Bit 1 - Coluna 2 - Teclado
		Bit 2 - Coluna 3 - Teclado
		Bit 3 - Coluna 4 - Teclado
		Bit 4 - Coluna 5 - Teclado
		Bit 5 - Coluna 6 - Teclado
		Bit 6 - Coluna 7 - Teclado
		Bit 7 - Coluna 8 - Teclado
FF03	65283	Bit 0 - Controle de campo; clock sync ≅ 16,6 m seg.
		Bit 1 - Entrada de interrupt
		Bit 2 - Normalmente 1/ se 0 muda FF02 para registro de direção de dados
		Bit 3 - SEL 2 - MSB das duas linhas multiplexadas de select
		Bit 4 - Sempre 1
		Bit 5 - Sempre 1
		Bit 6 - Não utilizado
		Bit 7 - Flag de interrupção de sincronismo vertical
FF20-FF23	65312-65315	PIA 2 (periferial interface adapter 2)
FF20	65312	Bit 0 - Cassete - Entrada de dados
		Bit 1 - Saída de dados RS232
		Bit 2 - 6 Bit D/A LSB
		Bit 3 - 6 Bit D/A
		Bit 4 - 6 Bit D/A
		Bit 5 - 6 Bit D/A
		Bit 6 - 6 Bit D/A
		Bit 7 - 6 Bit D/A MSB
FF21	65313	Bit 0 - Controle do CD, RS232 entrada de status
		Bit 1
		Bit 2 - Normalmente 1
		Bit 3 - Controle do motor do cassete 0 = OFF / 1=ON
		Bit 4 - Sempre 1
		Bit 5 - Sempre 1
		Bit 6 - Não utilizado
		Bit 7 - Flag de interrupt de CD (RS232)
FF22	65314	Bit 0 - RS232 entrada de dados

FF22	65314	Bit 1 - Saída de som de 1 bit Bit 2 - Entrada de tamanho de RAM Bit 3 - Saída de controle da VDG (Gerador de vídeo) Bit 4 - Saída de controle da VDG (Gerador de vídeo) Bit 5 - Saída de controle da VDG (Gerador de vídeo) Bit 6 - Saída de controle da VDG (Gerador de vídeo) Bit 7 - Saída de controle da VDG (Gerador de vídeo)
FF23	65315	Bit 0 - Controle de interrupção para ROM PACK Bit 1 - Entrada Bit 2 - Normalmente 1 / 0 = muda FF22 para registro de direção de dados Bit 3 - Enable de som de 6 bits Bit 4 - Sempre 1 Bit 5 - Sempre 1 Bit 6 - Não utilizado Bit 7 - Flag de interrupt de ROM PACK
FF40	6544	Latch de saída Bit 0 - Seleção de Drive 0 Bit 1 - Seleção de Drive 1 Bit 2 - Seleção de Drive 2 Bit 3 - Controle do motor Bit 4 - Pré-compensação de escrita Bit 5 - Dupla densidade Bit 6 - Seleção de Drive 3 Bit 7 - Halt Enable
FF48	65352	STATUS do Driver
FF49	65353	Número do track (trilha) do Disco
FF4A	65354	Número do setor do Disco
FF4B	65355	Dados do Disco

(Cont.)

FFE0-FFF1	65504-65521	Não utilizado
FFF2-FFF3	65522-65523	Vetor SWI 3
FFF4-FFF5	65524-65525	Vetor SWI 2
FFF6-FFF7	65526-65527	Vetor FIRQ
FFF8-FFF9	65528-65529	Vetor IRQ
FFFA-FFFB	65530-65531	Vetor SWI 1
FFFC-FFFD	65532-65533	Vetor NMI
FFFE-FFFF	65534-65535	Vetor reset

FFC0 FFC1	65472 65473	CLEAR SET	V0	DISPLAY	1	0	1	0	1	0	1	0
FFC2 FFC3	65474 65475	CLEAR SET	V1	CONTROLE DE MODO VÍDEO	1	1	0	0	1	1	0	0
FFC4 FFC5	65476 65477	CLEAR SET	V2		1	1	1	1	0	0	0	0
FFC6 FFC7	65478 65479	CLEAR SET	F0	OFF-SET do DISPLAY (BINÁRIO) endereço do elemento da esquerda superior igual a 0000 (1/2 K off-set)	N.U.	G6R,G6C	G3R	G3C	G2R	G2C	G1C, G1R	AI, AE, S4, S6
FFC8 FFC9	65480 65481	CLEAR SET	F1									
FFCA FFCB	65482 65483	CLEAR SET	F2									
FFCC FFCD	65484 65485	CLEAR SET	F3									
FFCE FFCF	65486 65487	CLEAR SET	F4									
FFD0 FFD1	65488 65489	CLEAR SET	F5									
FFD2 FFD3	65490 65491	CLEAR SET	F6									
FFD4 FFD5	65492 65493	CLEAR SET	P1	Página≠1 (=0)								
FFD6 FFD7	65494 65495	CLEAR SET	Clock CPU	Refresh transparente	1	0	1	0				
FFD8 FFD9	65496 65497	CLEAR SET	Clock CPU		1	1	0	0				
FFDA FFDD	65498 65501	CLEAR SET	M0		1	0	1	0				
FFDC FFDD	65500 65501	CLEAR SET	M1		1	1	0	0				
FFDE FFDF	65502 65503	CLEAR SET	TY		N.U.		32/64 K	16 K	4 K			

0,9 MHZ apenas

0,9/1,8 MHZ

N.U.(Se utilizado obtém-se +velocidade, porém s/vídeo)

N.U.(Se utilizado obtém-se +velocidade, porém s/vídeo)

Tipo do mapa de memória normalmente=0

COMANDOS POKE, PEEK E EXEC

COMANDO: POKE 178,X (onde X = 0 a 255)
RESULTADO: Permite 256 padrões de cores na tela, com os comandos gráficos.

COMANDO: POKE 174,X (onde X = 0 a 255)
RESULTADO: Permite 256 padrões de cores como cor de fundo, com o comando PCLS.

COMANDO: POKE 383, 158
RESULTADO: Impede listagem.

COMANDO: POKE 383,0
RESULTADO: Retorna comando LIST ao normal, após POKE 383,158

COMANDO: POKE 175,79
RESULTADO: Idem comando TRON

COMANDO: POKE 175,0
RESULTADO: Idem comando TROFF

COMANDO: PEEK (182)
RESULTADO: Retorna número de PMODE em uso; se não estiver em modo gráfico, será retornado zero.

COMANDO: POKE 148,0
RESULTADO: Cursor para de piscar por 3 segundos.

COMANDO: POKE 282,0
RESULTADO: Muda para minúsculas.

COMANDO: POKE 282,255
RESULTADO: Muda para maiúsculas.

COMANDO: EXEC 44539
RESULTADO: Idem INKEY $

COMANDO: EXEC 41175
RESULTADO: Mostra mensagem do Basic normal.

COMANDO: POKE 65495,0
RESULTADO: Aumenta frequência do clock de 0,89 MHZ para 1,7 MHZ. Operações de INPUT/OUTPUT podem não funcionar corretamente com este POKE. Retorne ao normal antes de executá-las.

COMANDO: POKE 65494,0
RESULTADO: Retorna clock ao normal após POKE 65495,0

COMANDO: POKE 113,0
RESULTADO: Produz um cold start (partida a frio) se o reset é acionado ou EXEC 40999.

COMANDO: EXEC 40999
RESULTADO: Produz um cold start. Se utilizado após o POKE 113,0, um cold start imediato é executado. Equivale a ligar e desligar o micro.

COMANDO: POKE 359,60
RESULTADO: Reduz a velocidade de impressão dos caracteres na tela, inclusive para o comando LIST. Não funciona com DRIVER instalado. SCREEN 0,1 após este POKE, muda a tela para laranja.

COMANDO: POKE 359,0 ou POKE 359,126
RESULTADO: Volta ao normal após POKE 359,60

COMANDO: POKE 111,254 :DIR
RESULTADO: Imprime diretório do disco na impressora.

COMANDO: EXEC 49152
RESULTADO: Inicia execução da ROM PACK ou DOS (DISCO)

COMANDO: POKE 65497,0
RESULTADO: Aumenta velocidade da máquina (mais que POKE 65495,0 porém perde-se a tela). Útil para longos processamentos durante o programa.
POKE 65496,0 volta ao normal.

COMANDO: PEEK (275)
RESULTADO: Valor do TIMER (de 0 a 256)

COMANDO: POKE 135,0
RESULTADO: Reseta INKEY $ para nulo.

COMANDO: PEEK(135)
RESULTADO: Código ASCII da última tecla pressionada.

COMANDO: PEEK(55) + PEEK(56)
RESULTADO: Retorna o código ASCII da última variável usada.

COMANDO: POKE 65313,4
RESULTADO: Idem "MOTOR ON".

COMANDO: POKE 65313,52
RESULTADO: Idem "MOTOR OFF".

COMANDO: POKE 1536,52 :POKE 1537,86 :POKE 1538,198 :POKE 1539,1 :POKE 1540,134 :POKE 1541,4 : POKE 1542,61 :POKE 1543,253 :POKE 1544,0 :POKE 1545, 141 :POKE 1546,189 :POKE 1547,169 :POKE 1548, 86 :POKE 1549,53 :POKE 1550,86 :POKE 1551,57 : POKE 360,6 :POKE 361,0
RESULTADO: "Beep" para teclado (emite som a cada tecla pressionada).

COMANDO: PEEK(487)*256+PEEK(488)
RESULTADO: Retorna o endereço de início de um programa em linguagem de máquina.

COMANDO: PEEK(126)*256+PEEK(127) -1
RESULTADO: Retorna o endereço do fim de um programa em linguagem de máquina.

COMANDO: PEEK(157)*256+PEEK(158)
RESULTADO: Retorna o endereço de EXEC de um programa em linguagem de máquina.

COMANDO: POKE 25,6 :NEW
RESULTADO: PCLEAR 0 para o micro sem DISK DRIVER
?MEM - Retorna 31015 (1º banco).

COMANDO: PEEK(25)*256+PEEK(26)
RESULTADO: Retorna endereço de início de um programa em BASIC.

COMANDO: PEEK(31)*256+PEEK(32)
RESULTADO: Retorna endereço do fim de um programa em BASIC.

COMANDO: PEEK(136)*256+PEEK(137)
RESULTADO: Retorna posição do cursor em tela de texto (1024-1535).

COMANDO: PEEK(116)*256+PEEK(117)
RESULTADO: Retorna valor do pointer de memória máxima.

COMANDO: PEEK(39)*256+PEEK(40)
RESULTADO: Retorna valor da memória protegida.

COMANDO: POKE(1024-1535),(33-255)
RESULTADO: Imprime um caracter na tela de texto.

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

COMANDO: POKE(1536-4607),(33-255)
RESULTADO: Imprime um caracter na tela de gráfico.

COMANDO: POKE 25, PEEK(27): POKE 26, PEEK(28)-2: CARREGUE O 2º PROGRAMA: POKE 25,38: POKE 26,1
RESULTADO: "Merge" dois programas (concatena).

COMANDO: PEEK(235)
RESULTADO: Retorna número do DISK DRIVER atual.

COMANDO: PEEK(236)
RESULTADO: Retorna número da trilha atual do diskette.

COMANDO: PEEK(237)
RESULTADO: Retorna o setor atual do diskette.

COMANDO: PEEK(240)
RESULTADO: Retorna os códigos de erro de I/0.

COMANDO: PEEK(274)*256+PEEK(275)
RESULTADO: Retorna valor do TIMER.

COMANDO: CHR$(PEEK(474)+PEEK(475)+PEEK(476)+PEEK(477) + PEEK(478) + PEEK(479) + PEEK(480) + PEEK(481))
RESULTADO: Retorna o nome do último programa carregado do cassete.

COMANDO: POKE 359,255
RESULTADO: Inibe o display, porém os comandos continuam a ser executados.

COMANDO: POKE 359,126
RESULTADO: Retorna ao normal, após POKE 359,255.

COMANDO: POKE 359,57
RESULTADO: Permite que se use qualquer tela de gráficos ou de textos (SCREEN 0,1).

COMANDO: POKE 359,57 :SCREEN 0,1
RESULTADO: Muda tela para laranja e vermelho.

COMANDO: POKE 359,126
RESULTADO: Volta ao normal após POKE 359,57 :SCREEN 0,1 e POKE 359,57.

COMANDO: PEEK(485)*256+PEEK(486)
RESULTADO: Retorna o endereço de EXEC para programas carregados do cassete.

COMANDO: PEEK(104)*256+PEEK(105)
RESULTADO: Retorna número (Opcode) do presente comando.

COMANDO: PEEK(49)*256+PEEK(50)
RESULTADO: Retorna número da linha de dados atual.

COMANDO: PEEK(222)+1
RESULTADO: Retorna oitava atual.

COMANDO: PEEK(255)
RESULTADO: Retorna duração atual da nota.

COMANDO: PEEK(226)
RESULTADO: Retorna tempo atual (som).

COMANDO: PEEK(338)
RESULTADO: Retorna os valores de ROLLOVER das teclas @, H, P, X, 0, 8, ENTER (última tecla pressionada). Se não pressionada igual a 255.

COMANDO: PEEK(339)
RESULTADO: Idem para as teclas A, I, Q, Y, 1, 9, CLEAR.

COMANDO: PEEK(340)
RESULTADO: Idem para as teclas B, J, R, Z, 2, :, BREAK.

COMANDO: PEEK(341)
RESULTADO: Idem para as teclas C, K, S, ↑, 3, ;

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

COMANDO: PEEK(342)
RESULTADO: Idem para as teclas D, L, T,↓ , 4, vírgula.

COMANDO: PEEK(343)
RESULTADO: Idem para as teclas E, M, U, ←, 5, -

COMANDO: PEEK(344)
RESULTADO: Idem para as teclas F, N, V, →, 6, ponto

COMANDO: PEEK(345)
RESULTADO: Idem para as teclas G, O, W, espaço, 7, /, SHIFT

COMANDO: CHR$(PEEK(466)+PEEK(467)+PEEK(468)+PEEK(469) + PEEK(470)+PEEK(471)+PEEK(472)+PEEK(473))
RESULTADO: Retorna o nome do arquivo requisitado.

COMANDO: PEEK(188)
RESULTADO: Retorna "14" se existe DISK DRIVER e "6" se não.

COMANDO: PEEK(193)
RESULTADO: Retorna "8" se usando set de cores "1" e "0" se usando set de cores "0".

COMANDO: PEEK(186)*256+PEEK(187)
RESULTADO: Retorna endereço de início da atual tela de gráficos de alta resolução.

COMANDO: PEEK(183)*256+PEEK(184)
RESULTADO: Retorna endereço de fim da atual tela de gráficos de alta resolução.

COMANDO: PEEK(207)*256+PEEK(208)
RESULTADO: Retorna raio do círculo usado em PMODE 4 (*2 = PMODE 0-3).

COMANDO: PEEK(203)*256+PEEK(204)
RESULTADO: Retorna o centro do círculo (X) em PMODE 4 (*2 = PMODE 0-3).

COMANDO: PEEK(205)*256+PEEK(206)
RESULTADO: Retorna o centro do círculo (Y) em PMODE 4 (*2 = PMODE 0 e 1).

COMANDO: POKE 383,158
RESULTADO: Inibe o comando DIR e LIST com Disco.
POKE 383,126 volta ao normal.

COMANDO: POKE 384, PEEK(114): POKE385, PEEK(115)
RESULTADO: Inibe LIST somente com Driver
POKE 384,200: POKE 385,89 volta ao normal.

APÊNDICE

VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50VARIXVC50

MAXITELA

Maxitela é o nome de um programa utilitário vendido em diskette ou fita cassete. É um programa curto, escrito em linguagem de máquina e que reside juntamente com o programa em Basic no computador, utilizando apenas 2K de memória RAM.

A tela fica formatada para 51x24 caracteres (24 linhas por 51 colunas) com minúsculas reais. Dependendo da versão, pode-se obter até 85x24 caracteres, com letras pretas em fundo branco ou verde.

É implementado também um comando de tratamento de erro (ON ERROR GO TO) além de "repeat" para todas as teclas, automaticamente, ou seja, segurando-se a tecla pressionada, por mais de 2 segundos, inicia-se a repetição.

Pode-se também, mixar gráficos e textos com extrema simplicidade.

O utilitário "MAXITELA" é transparente ao usuário e pode ser carregado e executado também de modo transparente, em um programa em Basic, por exemplo:

Com Diskette:

```
 1   'PROGRAMA
10   LOADM"MAXITELA"
20   EXEC
30   'RESTO DO PROGRAMA
 .
 .
 .
 .
 .
 .
```

Com cassete:

```
 1   'PROGRAMA
10   CLOADM " MAXITELA"
20   EXEC
30  'RESTO DO PROGRAMA
```

No manual do utilitário, o usuário encontrará instruções detalhadas.

VIDEO PACK

VIDEO PACK é uma placa opcional para o microcomputador VARIX VC 50 que formata por HARDWARE o vídeo para 80x24 caracteres, inclusive minúsculas.

O cartão já possue saída de vídeo composto para ser ligado ao monitor de vídeo profissional.

Com esta placa e com um monitor de vídeo, obtem-se um Display de altíssima qualidade, além de permitir edição de tela integral, com inúmeros comandos. Maiores detalhes serão encontrados no respectivo manual (compatível com placa CPM).

Varix

VC 50