Indo além com o CP 400 Color

PAULO ADDAIR

EDITELE

Indo além com o CP 400 COLOR

Paulo Addair

CIP - Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

Addair, Paulo, 1957-
A179i Indo além com o CP 400 color / Paulo Addair.
— São Paulo : EDITELE, 1985.

1. BASIC (Linguagem de programação para computadores) 2. CP 400 (Computador) - Programação 3. Microcomputadores - Programação I. Título.

17. CDD-651.8
18. -001.642
18. -001.6424

85-1537

Índices para catálogo sistemático:

1. BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8 (17.) 001.6424 (18.)
2. CP 400 : Computadores : Programação : Processamento de dados 651.8 (17.) 001.642 (18.)
3. Microcomputadores : Programação : Processamento de dados 651.8 (17.) 001.642 (18.)

EDITELE

Editora Técnica Eletrônica Ltda.
Rua Casa do Ator, 1060 — Vila Olímpia
CEP 04546 — São Paulo — SP
Caixa Postal 30141

Impresso no Brasil

APRESENTAÇÃO

A idéia deste livro surgiu da necessidade que experimentamos de suplementar as informações do próprio manual do CP 400. Em suas 288 páginas, apesar de repletas de informações úteis sobre o equipamento, não foi possível abranger sequer um terço das reais possibilidades de programação desse versátil micro.

É claro que não se pretende abranger os outros dois terços agora, de uma vez. Mesmo porque, para isso, seria necessário passar laconicamente por vários detalhes do equipamento. E isso não ajudaria em nada ao programador menos experiente, ao usuário cujo primeiro contato com um micro tenha sido o próprio CP 400.

O que se pretende é discutir alguns recursos cuidadosamente selecionados, mesclando inclusive, quando possível, linguagem de máquina e BASIC. Assim, o leitor vai percebendo, gradualmente, as reais dimensões da ferramenta que possui, numa tentativa de dominar essa nova tecnologia.

Esse domínio de uma nova tecnologia é de vital importância, os japoneses sabem disso! Mesmo que não se tenha a autoria da técnica, deve-se tentar absorvê-la e dominá-la. E para isso é necessário critério, bom senso, um pouco de criatividade e, naturalmente, acesso às informações "estratégicas". Não é imprescindível o acesso a todas as informações, pois, se o tivermos às básicas, as outras a gente descobre por tentativa e erro.

Outra coisa à qual demos especial importância nesse livro é o método. Já no primeiro capítulo colocamos uma proposta de organização para seus programas, agilizando a correção das listagens. Mais à frente, voltamos a falar em método, sugerindo um programa para organizar suas fitas de programas. Isso será muito importante, pois é vital que se gaste o tempo que lidamos com a máquina aprendendo e criando e não procurando um determinado programa ou revisando outro.

Em resumo, o que se espera com esse livro é colaborar modestamente na exploração dos recursos de um micro em cujo futuro confiamos e que, para tomar seu lugar ao sol, precisa, antes de mais nada, de bons programadores. Só assim evitaremos "macaquear" o criador da tecnologia, e estaremos absorvendo e dominando sua técnica para, a partir daí, ir além...

a meus filhos
Paulo Marcelo e
Carlos Alexandre

SUMÁRIO

I

Recursos de programação

Cap. 1 — Prova dos noves para conferir listagens

Cap. 2 — Localização e edição automáticas de erro

Cap. 3 — Depuração de programas byte por byte

Cap. 4 — Listagem super-rápida

Cap. 5 — Uma ajudazinha na introdução dos dados

1 PROVA DOS NOVES PARA CONFERIR LISTAGENS

Antes de começar a explorar os recursos do seu micro, quero fazer uma pergunta: como é que você vai saber se copiou tudo certo sem ter que revisar letra por letra todo o texto digitado? Você tem alguma idéia?

Quando me ensinaram a fazer contas com grandes números (mais de dois algarismos cada!!), mostraram-me também um meio de saber se o resultado estava correto sem ter de refazer toda a conta. Isso era um alívio... já pensou no meu esforço, aos oito anos de idade, para fazer uma conta do tipo 985 mais 1024? Mesmo hoje, sem um teclado à mão, isso seria muito difícil!

Mas existia a salvadora "prova dos noves". Nunca entendi direito como é que ela funcionava, mas uma coisa eu sabia: ela nunca falhava. Com o tempo, fui ganhando prática e confiando mais na minha aritmética (sem contar que comprei uma calculadora), e nunca mais precisei tirar a prova.

Agora estou precisando de algo como a prova dos noves para saber, rápida e seguramente, se o programa que introduzi está correto. Certos erros de digitação são muito difíceis de detectar apenas dando uma olhada superficial na tela. É necessário um método mais completo, algo que basta olhar para se decretar: "A linha 12434 está errada!"

Se você é um programador que acompanha as várias publicações técnicas que estão por aí já deve ter notado que o pessoal lá das redações não está muito interessado em facilitar as coisas para você. Nas revistas nacionais não existe um mínimo de cuidado com as listagens. Às vezes não se consegue sequer ler o que está escrito, dada a pobreza da listagem impressa.

Algumas revistas simplesmente padronizam tudo, vaticinando que todas as listagens serão em 34 colunas. É uma boa tentativa mas... para que serve, se cada micro possui uma largura diferente de linha na tela, que varia de 32 a 80 colunas, mas nunca 34!? O programador assim nunca terá uma identificação visual entre o que está lendo e o que está digitando. Revistas mais amadurecidas, infelizmente estrangeiras, mostram cada programa como ele realmente aparece na tela. Assim, se você está começando uma nova linha e o primeiro caractere não é o mesmo que o da revista, você já sabe que errou alguma coisa na linha anterior.

Infelizmente, não podemos esperar que nossas revistas amadureçam tanto. Mesmo porque já estão aparecendo novas alternativas para se colocar um programa na memória, como o telesoftware e o MSsave, por exemplo. Este últi-

mo, aliás, uma boa iniciativa de uma revista nacional, que parte para uma solução mais comercial, vendendo fitas com os programas, sem no entanto mudar os critérios de listagens da revista.

Então, qual a solução? Na minha opinião, o meio mais seguro de saber se cometemos erros é ainda a prova do noves. No caso de listagens, não exatamente uma prova dos noves, mas dos 255, que é o mesmo processo, só que aplicando a base numérica do computador, o byte. Como em aritmética "humana", a base é dez e a prova é dos "noves"; se usarmos o valor máximo do byte, 256, a prova deve ser dos 255.

Como você deve saber, toda a informação armazenada no micro está representada por valores numéricos na memória, representados pelos tais "bytes". Isso vale também para o programa, que fica codificado numa determinada parte da memória destinada só para ele. Assim, se queremos ter a prova dos noves de um programa, basta somarmos os conteúdos na memória e, toda vez que se ultrapassar 255, subtrairmos esse valor do resultado. No final teremos um valor representativo de cada linha e de todo o programa.

Na realidade essa é apenas uma das muitas maneiras de se garantir as listagens. Se você consultar revistas como a *RAINBOW*, por exemplo, verá que isso é feito de forma diferente, com resultados diferentes. Eu particularmente prefiro o meu método, que é capaz de identificar exatamente qual linha está errada.

O PROGRAMA

Para implementar a prova dos noves, criei o programa na Listagem 1-1, como resultado do confronto de vários programas similares, aproveitando as boas soluções de uns e contornando as deficiências de outros.

O programa, além de calcular e mostrar a prova de todas as linhas, fornece a prova dos noves para cada grupo de dez linhas. É este valor que está indicado ao lado de cada listagem em BASIC neste livro. Não há restrição quanto à complexidade das linhas, podendo estas terem múltiplas instruções separadas por dois pontos. A única restrição é quanto à numeração das linhas, que não pode exceder a 62999, pois a rotina da prova dos noves começa em 63000.

Ao lado da listagem você encontrará um exemplo de utilização do programa, com a prova dos noves dele mesmo. Eu tirei a prova de uma forma meio "excêntrica". Renumerei a rotina com uma instrução RENUM, carreguei a rotina novamente com a numeração original e a executei. É óbvio que isso não vale para garantir a própria rotina. É como inventar um novo padrão de medida, e depois querer medir ele com ele mesmo para ver se está certo. Por isso, peço desculpas, mas esse primeiro programa **tem que ser corrigido letra por letra.** Nos próximos programas sim, os valores podem ser então comparados com os fornecidos pelo autor (como no caso deste livro) ou acrescentados no final das linhas correspondentes, usando para isso o comando de edição (EDIT), nos programas que você criar.

Para assegurar a exatidão de suas listagens, siga os seguintes passos:

— Certifique-se de que o seu programa está rodando sem problemas.

— Acrescente a rotina da prova dos noves (você pode digitá-la no final de cada

programa ou usar o programa do Capítulo 10).
— Chame a rotina com o comando RUN 63000.
— Anote os valores para cada linha e/ou os valores para os blocos de dez linhas, como aparecem na tela.
— Insira no final de cada linha o valor correspondente à mesma. Use para isso a instrução EDIT (subcomando X), colocando um apóstrofo seguido do valor da prova até aquela linha.

Esse procedimento não deve ser usado em linhas com instruções DATA. Nesses casos, é melhor fazer o próprio programa realizar uma soma dos valores na instrução DATA e verificar ao final da leitura dos mesmos.

Ao fornecer a listagem do seu programa para alguém, essa pessoa deverá copiar o programa ignorando o apóstrofo (sim, porque "apóstrofe" é figura de linguagem) e os valores, inserindo a rotina da prova dos noves. Depois disso ela roda a rotina (RUN 63000) e compara os valores obtidos com o fornecido na cópia que você deu.

Com os valores em mãos, identifica-se as linhas onde houve erro de digitação, corrige-se através de EDIT e torna-se a rodar a rotina da prova (RUN 63000 de novo). Repete-se este procedimento até não encontrar mais disparidades entre a listagem fornecida e a digitada.

Este procedimento permite um método de depuração (*debbuging*, para aqueles que acham que o Português não dá *status*) bastante prático, principalmente quando o programa possui uma listagem muito longa ou complexa. Você não precisa necessariamente usá-lo, podendo até, a partir dele, chegar a um outro, ainda melhor.

Listagem 1-1 — Rotina da prova dos noves

```
63000 CL=PEEK(25)*256+PEEK(26)
63010 CLS:BN=BN+1:BT=0:PRINT" LI
NHA","PROVA":PRINT
63020 FOR I=1 TO 10:LN=PEEK(CL+2
)*256+PEEK(CL+3)
63030 IFLN<63000 THEN PRINTLN,:N
L=PEEK(CL)*256+PEEK(CL+1):ELSEI=
11:GOTO 63060
63040 FOR J=CL+2 TO NL-1:IF PEEK
(J)=58 AND PEEK(J+1)=131 THEN J=
NL:ELSE CS=CS+PEEK(J):IFCS>254TH
ENCS=CS-255
63050 NEXTJ:PRINTCS:CL=NL:BT=BT+
CS
63060 NEXTI:IFLN<63000THENPRINT#
-2,"":PRINT#-2,"LINHA:";LN,CS:PR
INT#-2,"" ELSE PRINT#-2,"FIM:  "
;CS:END
63070 INPUT"PRESSIONE QUALQUER T
ECLA";BT:GOTO 63010
```

```
FIM:    215
```

2 LOCALIZAÇÃO E EDIÇÃO AUTOMÁTICAS DE ERRO

O que acontece quando existe um erro no programa que você está rodando? O computador mostra um lacônico "MN Erro na XXX", onde MN é um mnemônico que indica o tipo de erro detectado e XXX, o número da linha onde ele ocorreu. A partir dessa informação, você digita um invariável LIST XXX ENTER, analisa a linha problemática e, quase sempre, parte para um EDIT XXX ENTER.

Em todo esse procedimento, o maior problema é localizar exatamente qual das instruções que compõem a linha está incorreta. A dificuldade é, na minha opinião, diretamente proporcional ao quadrado do comprimento da linha. Mas, para que todo esse ritual se o próprio computador foi capaz de detectar o erro? É óbvio que, se um erro foi detectado dentro de uma linha, o computador, de uma forma ou de outra, sabe *exatamente* onde está o problema. O que acontece depois é que a máquina "generaliza", dizendo apenas o número da linha.

É possível então melhorar significativamente o funcionamento da rotina de tratamento de erros. Podemos acrescentar uma rotina em linguagem de máquina no CP 400 capaz de, ao ocorrer um erro, acionar automaticamente a função de edição, posicionar o cursor sobre a parte da linha onde o erro foi detectado e permitir que você tenha apenas o trabalho de corrigir o problema.

Para rodar o programa em BASIC listado neste capítulo em um outro micro que não o CP 400, deve-se dispor de uma versão com Extended Color BASIC. No CP 400 com ou sem disco ou em compatíveis com DISK BASIC, não há restrições. A rotina carregada independe de sua posição na memória, podendo ser colocada em qualquer parte da RAM.

A ROTINA

O interpretador BASIC do CP 400 utiliza uma sub-rotina armazenada nas posições &H9F a &HAA que pega os caracteres um por um dentro da linha de programa. Quando o interpretador encontra um erro, o conteúdo de &HA6 e &HA7 aponta para o último caractere lido pelo interpretador.

As linhas em BASIC são armazenadas num formato "remissivo", ou seja, as palavras-chave não são armazenadas por extenso, mas sim representadas por

um código de até dois bytes. Isso torna o programa mais compacto (imagine armazenar LINEINPUT por extenso!) e, conseqüentemente, mais rápido. Por outro lado, fica difícil para o operador ler a linha diretamente da memória.

Para que o operador possa identificar corretamente onde ocorreu o erro, a indicação deste deve ser colocada dentro da representação "legível" da linha, ou seja, a linha deve ser decifrada. Felizmente, existe uma rotina em &HB7CB que realiza essa função e coloca a linha decifrada na memória auxiliar (buffer) indicada pelo registrador Y.

Agora precisamos definir os vários passos necessários para que as coisas saiam como esperamos. Em primeiro lugar, precisamos fazer com que a linha seja decifrada e que o cursor pare sobre o erro. Para isso colocamos um zero na posição de erro. Isso faz com que a rotina de decodificação ("decifração" fica meio esquisito) acredite que aquele é o final da linha. As linhas 00350 a 00450 salvam a posição do erro, recolocam o caractere original e continuam a decodificação.

As linhas 00490 a 00580 imprimem o número da linha e o seu conteúdo, assim como EDIT o faria, e então acionam a edição, via rotina &H855C.

Nosso único problema agora é como fazer com que o interpretador BASIC realize tudo isso automaticamente, logo após ter sido detectado um erro. O interpretador passa pelas posições &H15E a &H1A8 várias vezes em diversas situações. A rotina do interpretador que coloca um caractere na tela (CHROUT — $A282) é importante, pois a nova instrução EDIT deve ser acionada após ter sido impresso "Erro na".

Este desvio pode ser usado para verificar qual instrução está sendo colocada na tela. A mensagem "Erro na" fica armazenada nas posições $ABE1 a $ABEB da ROM. A rotina CHROUT espera por esse endereço no registrador X.

Existe um problema nesse método. "Erro" e "na" são consideradas mensagens separadas na memória do computador. Por isso, verificamos apenas se a mensagem "na" foi impressa. No entanto, a mensagem "Break na" também utiliza o endereço que checamos no nosso método. Isso vai fazer com que a edição seja acionada também quando houver um STOP ou o pressionamento da tecla BREAK, e a linha editada será a última que foi executada. Resumindo, ninguém é perfeito!

Mas, voltemos à elucubração técnica. A rotina CHROUT é checada sempre antes de completar a saída na tela. Mudando-se o endereço de retorno usado no final desta rotina, a nova rotina de erro não será acionada até que CHROUT termine. Este endereço é armazenado no indicador de pilha pelo registrador S. Já que na tabela de desvio os endereços estão no terceiro nível e cada nível ocupa dois bytes, devemos inserir o endereço da nova rotina de erro a partir do sexto byte; assim o registrador S vai operar apropriadamente.

A sub-rotina nas linhas 060 a 160 insere o novo desvio na tabela. Esta rotina permite que um desvio que já esteja na posição $167 da tabela seja executado imediatamente após a rotina de manipulação de erro. Este método é mais complicado, porém, mantém a compatibilidade com outras rotinas que também utilizam esse desvio.

A sub-rotina em linguagem de máquina até pode trabalhar com micros que não disponham de Extended BASIC (que é o COLOR BASIC do CP 400), mas aí não daria para usar o programa em BASIC que listamos aqui, pois ele depende de algumas instruções que só o Extended possui.

O programa na Listagem 2-2 é que vai carregar a sub-rotina em linguagem

de máquina sem a necessidade de um EDITOR/ASSEMBLER. As linhas 20 e 25 verificam quanta memória dispõe seu sistema e automaticamente limpam pelo menos 1k para a rotina em L.M. (Linguagem de Máquina).

Espero que esse pequeno utilitário lhe facilite a vida nessas intermináveis horas, nas quais a gente fica "caçando as bruxas" daqueles programas temperamentais que relutam em funcionar... Boa sorte!

Listagem 2-1 — A rotina em linguagem de máquina

```
                    00010 ****************************************
                    00020 *              AUTO EDIT               *
                    00030 *     PAULO ADDAIR -- JAN/85           *
                    00040 ****************************************
7F00                00050         ORG  $7F00
                    00060 ***** REAJUSTA DESVIO ANTES DA SAIDA DE CARACTERE ($A282)
7F00 8E0167         00070         LDX  #$167          POSICAO PARA VINDA DE CHROUT
7F03 A680           00080         LDA  ,X+            PEGA RETORNO DE DESVIO ATUAL
7F05 A78D0020       00090         STA  DESVIO,PCR     E O COLOCA NO FIM
7F09 10AE84         00100         LDY  ,X              DA NOVA ROTINA
7F0C 10AF8D0019     00110         STY  1+DESVIO,PCR   PEGA ENDERECO DO NOVO DESVIO
7F11 318D0008       00120         LEAY VEERRO,PCR      E O COLOCA NA TABELA DESVIO
7F15 10AF84         00130         STY  ,X
7F18 867E           00140         LDA  #$7E           DESVIO NO 6809E.
7F1A A71F           00150         STA  -1,X           ARMAZENA-O NA TABELA.
7F1C 39             00160         RTS
                    00170 ***** VERIFICA SE "ERRO NA" FOI IMPRESSO ***
7F1D 8CABEA         00180 VEERRO CMPX #$ABEA         IMPRIMINDO "ERRO NA"?
7F20 2607           00190         BNE  DESVIO         NAO. FACA O DESVIO NORMAL.
7F22 318D0006       00200         LEAY ERRO,PCR       PEGA ENDERECO DO NOVO DESVIO
7F26 10AF66         00210         STY  6,S            ARMAZENA COMO ULTIMO RETORNO.
7F29 39             00220 DESVIO RTS                  SALVA 3 BYTES
7F2A 39             00230         RTS
7F2B 39             00240         RTS
                    00250 ****************************************
                    00260 * NOVA ROTINA DE MANIPULACAO DE ERRO *
                    00270 ****************************************
7F2C BDB958         00280 ERRO    JSR  $B958          APAGA O FINAL LINHA NA TELA
7F2F DC68           00290         LDD  $68            PEGA NUMERO ATUAL DA LINHA
7F31 DD2B           00300         STD  $2B            E O ARMAZENA
7F33 BDBDCC         00310         JSR  $BDCC          IMPRIME NUMERO DA LINHA ATUAL
7F36 BDB9AC         00320         JSR  $B9AC          IMPRIME ESPACO
7F39 BDAD01         00330         JSR  $AD01          POE POSICAO DO NUMERO DE LINHA
7F3C 2547           00340         BCS  FIMERR         ERRO SE NAO FOR NUMERO CORRETO
7F3E 108E02DD       00350         LDY  #$2DD          POSICAO DO BUFFER DO TECLADO
7F42 A69F00A6       00360         LDA  [$A6]          POSICAO DO ERRO NA LINHA BASIC
7F46 3402           00370         PSHS A              SALVA AQUELE CARACTERE
7F48 6F9F00A6       00380         CLR  [$A6]          IGUALA A ZERO P/ FIM DE LINHA
7F4C 3004           00390         LEAX 4,X            PEGA COMECO DA LINHA BASIC
7F4E BDB7CB         00400         JSR  $B7CB          REPOE O CARACTERE DE ERRO
7F51 3502           00410         PULS A              PEGA O CARACTERE DE ERRO
7F53 A79F00A6       00420         STA  [$A6]          O O COLOCA DE NOVO NA LINHA
7F57 301F           00430         LEAX -1,X           REAJUSTA P/ DECIFRAR A LINHA
7F59 3420           00440         PSHS Y              SALVA POSICAO ERRO NO BUFFER
7F5B BDB7CB         00450         JSR  $B7CB          DECIFRA O RESTO DA LINHA
7F5E 1F20           00460         TFR  Y,D            POE FINAL DO BUFFER EM D
7F60 8302DE         00470         SUBD #$2DE          PEGA COMPRIMENTO DO BUFFE
7F63 DDD7           00480         STD  $D7            SALVA PARA A ROTINA DE EDICAO
7F65 8E02DD         00490         LDX  #$2DD          INICIO DO BUFFER
7F68 BD85B4         00500         JSR  $85B4          IMPRIME BUFFER
7F6B BDB958         00510         JSR  $B958          APAGA O FINAL DA LINHA
7F6E BDBDCA         00520         JSR  $BDCA          IMPRIME O NUMERO DE LINHA
7F71 BDB9AC         00530         JSR  $B9AC          IMPRIME ESPACO
7F74 8E02DD         00540         LDX  #$2DD          POSICAO DO BUFFER
7F77 3506           00550         PULS D              PEGA POSICAO DO ERRO NO BUFFER
7F79 8302DD         00560         SUBD #$2DD          DISTANCIA DO INICIO DO BUFFER
7F7C BD85B6         00570         JSR  $85B6          IMPRIME BUFFER ATE' O ERRO
7F7F BD855C         00580         JSR  $855C          CHAMA ROTINA DE EDICAO
7F82 7EAC73         00590         JMP  $AC73          VAI PARA LOOP DE COMANDO BASIC
7F85 7EAED2         00600 FIMERR  JMP  $AED2          CHAMA ERRO UL
     0000           00610         END
```

Listagem 2-2 — O programa em BASIC

```
5 ' LOCALIZAÇAO E EDICAO
7 ' PAULO ADDAIR  --  JUL/85
10 CLS:CLEAR200,PEEK(&H74)*256
20 X=PEEK(&H74)*256
30 FORY=X TO X+133
40 READ Z$:Z=VAL("&H"+Z$)
50 POKEY,Z
60 C=C+Z
70 NEXTY
80 IF C<>15312THENPRINT"PROBLEMA
S NA LINHA DATA":END
90 EXECX
100 PRINT"EDICAO AUTOMATICA ATIV
ADA"
110 P=PEEK(25)*256+PEEK(26):POKE
P,0:POKEP+1,0:POKE&H2B,0:POKE&H2
C,0
120 DATA8E,01,67,A6,80,A7,8D,00,
20,10,AE,84,10,AF,8D,00,19
130 DATA 31,8D,00,08,10,AF,84,86
,7E,A7,1F,39,8C,AB,EA,26,07
140 DATA 31,8D,00,06,10,AF,66,39
,39,39,BD,B9,58,DC,68,DD,2B,BD,B
D,CC
150 DATA BD,B9,AC,BD,AD,01,25,47
,10,8E,02,DD,A6,9F,00,A6,34,02
160 DATA 6F,9F,00,A6,30,04,BD,B7
,CB,35,02,A7,9F,00,A6,30,1F,34,2
0
170 DATA BD,B7,CB,1F,20,83,02,DE
,DD,D7,8E,02,DD,BD,85,B4,BD,B9,5
8
180 DATA BD,BD,CA,BD,B9,AC,8E,02
,DD,35,06,83,02,DD,BD,85,B6,BD,8
5,5C
190 DATA 7E,AC,73,7E,AE,D2
```

```
LINHA: 180        165

FIM:   204
```

3 DEPURAÇÃO DE PROGRAMAS BYTE POR BYTE

Que vantagem Maria leva? Pra que saber os endereços e conteúdos de onde está armazenado o programa em BASIC? Não é melhor simplesmente alterar a linha de programa usando EDIT?

Vamos por partes. Em primeiro lugar, a Maria só vai levar vantagem se for uma programadora experiente. O utilitário que estamos propondo "abre" a parte da memória onde está o programa, permitindo que a gente altere o programa sem a interferência do interpretador BASIC. Isso é especialmente útil quando vamos usar linguagem de máquina, ou, então, quando precisamos fazer certas mudanças críticas na listagem armazenada.

Por isso esse utilitário, contrariando a sua aparência, pode se tornar essencial para o programador que precisa compactar suas listagens alterando diretamente a memória, evitando assim as regras impostas pelo interpretador.

O utilitário que batizamos "Depurador" mostra as linhas de programa byte por byte, identificando a posição na memória de cada byte, seu conteúdo decimal e seu significado.

Entre as aplicações desse programa está a alteração de nome de variáveis, através de POKEs diretamente na memória, sem precisar editar as linhas. Você pode ainda colocar dispositivos de segurança no programa. Por exemplo, pode colocar certas linhas que impeçam o programa de rodar e, quando quiser executá-lo, você modifica o primeiro byte dessas linhas, inserindo um apóstrofo no primeiro byte delas, o que fará com que essas linhas sejam consideradas comentários.

Para fazer proteções mais elaboradas, pode-se verificar através de PEEKs o conteúdo de determinados bytes onde foram colocados, por exemplo, os direitos autorais de seu programa. Qualquer mudança nesses bytes pode ser detectada e o programa pode simplesmente ser impedido de rodar. Isso permite ir mais além. Você pode pegar esses valores na mensagem de direitos autorais, colocá-los em outra parte da memória mais à frente e verificar mais tarde se os valores ainda estão lá. Assim, ninguém poderá mudar a mensagem sob pena de deslocar o restante da memória, tirando aqueles valores de suas posições originais e, conseqüentemente, inutilizando o programa.

O "Depurador" está dividido em dois programas para operação com sistemas que possuam disco. O primeiro, na Listagem 3-1, é o "Preparador", encarregado de preparar um arquivo de dados com todas as palavras-chave necessárias para a interpretação das listagens. Ele só será usado uma vez. O segundo pro-

grama, na Listagem 3-2, é o "Depurador" propriamente dito. Ele deve ser armazenado em formato ASCII no disquete (SAVE "DEPURADOR", A) para que possa ser fundido (usando-se o comando MERGE) com o programa que estiver na memória. Como ele começa pela linha 62000, deve-se tomar o cuidado de não deixar o programa a ser depurado com linhas além desse número.

Quando tiver os dois programas na memória (o "Depurador" e o que será analisado) introduza o comando RUN 62000. O programa mostrará as opções disponíveis. A primeira é listar o programa a partir da primeira linha na memória. Essa opção é indicada quando se quer analisar o programa por inteiro, buscando os pontos de destaque de sua listagem. Por outro lado, quando sabemos exatamente onde queremos "mexer", a opção mais indicada é começar a listagem por aquele determinado endereço. A última opção permite listar o programa a partir de um determinado número de linha, para analisar uma sub-rotina, por exemplo. Tenha sempre em mente que essa é a opção mais demorada, pois o "Depurador" é um programa em BASIC e, conseqüentemente, lento.

Para utilizar esse programa com sistemas sem disquete, apenas mude as instruções de disquete para suas equivalentes em fita. A "fusão" dos programas pode ser feita com o utilitário mostrado no Capítulo 10.

Listagem 3-1 — Preparador de arquivo de códigos

```
200 CLS3:PRINT@8,"PREPARADOR";:D
IMCH$(200):FORX=1TO137:READCH$(X
):NEXT:OPEN"O",#-1,"CODIGO":FORX
=1 TO137:PRINT#-1,CH$(X):NEXTX:P
RINT@64,"CODIGOS CARREGADOS";:EN
D
210 DATAFOR,GO,REM,":REM",ELSE,I
F,DATA,PRINT,ON,INPUT,END,NEXT,D
IM,READ,RUN,RESTORE,RETURN,STOP,
POKE,CONT,LIST,CLEAR,NEW,CLOAD,C
SAVE,OPEN,CLOSE,LLIST,SET,RESET,
CLS,MOTOR,SOUND,AUDIO,EXEC,SKIPF
,TAB(,TO,SUB,THEN,NOT,STEP,OFF,+
,-,*,/,^,AND,OR,>,=,<,DEL
220 DATAEDIT,TRON,TROFF,DEF,LET,
LINE,PCLS,PSET,PRESET,SCREEN,PCL
EAR,COLOR,CIRCLE,PAINT,GET,PUT,D
RAW,PCOPY,PMODE,PLAY,DLOAD,RENUM
,FN,USING,DIR,DRIVE,FIELD,FILES,
KILL,LOAD,LSET,MERGE,RENAME,RSET
,SAVE,WRITE,VERIFY,UNLOAD,DSKINI
,BACKUP,COPY,DSKI$,DSKO$
230 DATASGN,INT,ABS,USR,RND,SIN,
PEEK,LEN,STR$,VAL,ASC,CHR$,EOF,J
OYSTK,LEFT$,RIGHT$,MID$,POINT,IN
```

```
KEY$,MEM,ATN,COS,TAN,EXP,FIX,LOG
,POS,SQR,HEX$,VARPTR,INSTR,TIMER
,PPOINT,STRING$,CVN,FREE,LOC.LOF
,MKN$,AS
```

```
FIM:    176
```

Listagem 3-2 — Depurador de programas byte por byte

```
62000 CLS3:CLEAR2000
62010 LV=2:LL=0:L=1:ST=PEEK(25)*
256+PEEK(26)-1
62020 PRINT@256,"COMECAR PELO IN
ICIO?";:GOSUB62120:IFW$()"N"THEN
62040ELSEPRINT@320,"(N)UMERO DE
LINHA OU (E)NDERECO";:GOSUB62120
:IFW$="E"THENPRINT@384,"";:INPUT
"ENDERECO";W$:ST=VAL(W$):GOTO620
40
62030 PRINT@384,"";:INPUT"NUMERO
 DA LINHA";LL
62040 CLS3:PRINT@12,"FORMATO";:P
RINT@66,"LINHA  END.  VALOR SIGN
IF. ";:DIMCH$(600):OPEN"I",#1,"C
ODIGO":FORX=1 TO 127:CH$(X)=CHR$
(X):NEXTX:FORX=128TO224:INPUT#1,
CH$(X):NEXTX:FORX=383 TO 422
62050 INPUT#1,CH$(X):NEXTX:CLOSE
#1
62060 FORX=ST TO65535:P=PEEK(X):
IFZZ()OTHENZZ=0:NEXTX
62070 IFP(32 AND PEEK(X+3)(65THE
NL=PEEK(X+3)*256+PEEK(X+4):X=X+4
:NEXTX
62080 IFP=255THENP=255+PEEK(X+1)

62090 IFL)=LL THENLV=LV+1:PRINT@
LV*32+2,USING"#####  %  %  ###
 %      %";L;HEX$(X);P;CH$(P);:I
FLV=15THENLV=2:GOSUB62120
62100 IFP)254THENZZ=1
62110 NEXT
```

```
62120 W$=INKEY$:IFW$=""THEN62120
ELSEIFW$=CHR$(13)THENENDELSERETU
RN
```

```
LINHA: 62090     111
FIM:   227
```

4 LISTAGEM SUPER-RÁPIDA

Sempre que estou revisando um programa, uma das coisas que mais me aborrece é ter que ficar digitando os intermináveis comandos "LIST daqui pra lá". É um procedimento bastante repetitivo e improdutivo que, com o tempo, acaba se tornando um empecilho para uma boa revisão.

Seria muito bom se o CP 400 e seus similares dispusessem de um recurso como o NEWDOS ou o DOS500, que rodam no CP 500. Esse recurso consiste simplesmente em usar as teclas de seta (para cima e para baixo) para listar as linhas de programa. Um exemplo: digamos que a última linha listada tenha sido a de número 100 (você acionou um LIST-100, por exemplo) e você queira ver a próxima, a 110. A única maneira de fazer isso com o micro é digitar LIST110 ENTER, ou seja, pressionar nove teclas. Se você estivesse operando um CP 500 com disco, bastaria pressionar a tecla de seta para baixo e imediatamente a próxima linha na memória surgiria na tela.

O programa que proponho neste capítulo permite ao CP 400 dispor de um recurso desse tipo. Isso vai facilitar muito as revisões das suas listagens e até contribuir, não agravando um possível mau humor, quando você estiver tentando achar um "gato" no seu software.

O PROGRAMA

A rotina de listagem automática está na realidade em linguagem de máquina, porém o programa que a coloca na memória está em BASIC. Esse programa utiliza uma série de recursos avançados que passaremos a discutir. Mais uma coisa, quando você terminar a digitação, não deixe de copiá-lo em fita ou disco antes de rodar, pois ele apaga a porção em BASIC da memória, deixando apenas a rotina em linguagem de máquina operando.

Inicialmente o programa localiza o topo da memória na linha 310. Com essa informação, ele separa espaço suficiente na memória para acomodar a rotina em linguagem de máquina (linha 320) e localizar o novo topo da memória, na linha seguinte.

As quatro linhas a seguir (340 a 370) lêem os valores decimais correspon-

dentes à rotina e colocam-nos na área de memória previamente reservada. Esta área está protegida contra futuros NEWs, CLOADs ou LOADs, o que significa que todos os programas que forem colocados na memória não afetarão esta rotina. Cuidado para que os seus programas que usam rotinas em linguagem de máquina não borrem esta área. A linha 380 ativa a rotina e a 390 localiza o início do programa em BASIC que a carregou. A linha seguinte coloca dois zeros no começo daquele programa para que o computador o considere "apagado". Na linha 410, outros dois zeros são colocados na memória, desta vez nos endereços usados pelo computador para marcar a linha onde ele está trabalhando.

A ROTINA

A primeira parte do código carregado altera duas posições de memória para que o computador salte para a rotina SUPERLIST toda vez que uma tecla seja pressionada.

A segunda parte é a sub-rotina, para onde o computador salta quando qualquer tecla é pressionada. Essa sub-rotina verifica se a tecla digitada foi a seta para cima e, nesse caso, muda o valor decimal de 94 para 1, a fim de que não seja impresso o colchete.

A terceira parte é a principal da rotina. Ela verifica qual seta foi pressionada, localiza o número e o endereço da linha imediatamente posterior ou anterior, conforme o caso, e ativa a sub-rotina na ROM, que procederá à decodificação e listagem dessa linha.

OBSERVAÇÕES FINAIS

Se você usar o SUPERLIST com vários programas, não se esqueça de executar um LIST da parte que pretende analisar logo após carregar cada programa. Isso deve ser feito sempre, pois o computador mantém registro de onde estava a última linha de programa, mesmo que um novo programa seja colocado na memória. Essa informação só é adaptada ao novo programa após um LIST Dependendo de onde estiver esse endereço, o SUPERLIST pode encontrar a próxima linha do novo programa; caso contrário, o computador poderá "travar" Se isso acontecer, não se esquente: acione RESET, digite LIST e ENTER e tudo voltará ao normal.

Listagem 4-1 — Programa de carregamento do SUPERLIST

```
300 ' LISTAGEM RAPIDA
301 ' PAULO ADDAIR -- JUL/85
302 CLS
310 TM=PEEK(39)*256+PEEK(40)
```

```
320 CLEAR200,TM-240
330 TM=PEEK(39)*256+PEEK(40)
340 FORX=TM TO TM+223
350 READA
360 POKEX,A
370 NEXTX
380 EXECTM
390 P=PEEK(25)*256+PEEK(26)
400 POKEP,0:POKEP+1,0
410 POKE&H2B,0:POKE&H2C,0
420 DATA49,141,0,220,190,1,107,1
75,164,48,141,0,36,191,1,107,49,
141,0
430 DATA 214,190,1,104,175,164,4
8,141,0,4,191,1,104,57,129,94,38
,8,111
440 DATA 130,90,134,1,50,98,57,1
10,157,0,185,52,119,51,141,0,169
,129
450 DATA 1,39,20,129,10,38,10,14
1,26,236,70,221,43,174,72,141,12
2,53,119
460 DATA 110,157,0,145,141,10,23
6,66,221,43,174,68,141,106,32,23
8,158
470 DATA 25,236,132,38,4,50,98,3
2,228,220,43,39,60,16,131,255,25
5,39,54
480 DATA 237,66,175,68,16,174,13
2,16,174,164,38,5,237,70,175,72,
57,16
490 DATA 163,2,39,14,52,6,236,2,
237,66,53,6,175,68,174,132,32,23
7,16
500 DATA 174,132,16,174,164,39,2
,174,132,236,2,237,70,175,72,32,
218,236
510 DATA 2,237,70,175,72,237,66,
175,68,16,174,132,16,174,164,39,
200,175
520 DATA 68,236,2,237,66,174,132
,38,238,32,188,52,16,189,189,204
,189
530 DATA 185,172,53,16,189,183,1
94,206,2,221,166,192,39,5,189,18
5,177,32,247,189,185,92,57
```

```
LINHA: 390          252
LINHA: 490          7
FIM:   34
```

5 UMA AJUDAZINHA NA INTRODUÇÃO DOS DADOS

O sucesso de um programa depende muito mais de sua simplicidade de operação do que propriamente dos recursos que oferece. É claro que os recursos são importantes também, mas o grau de dificuldade em utilizá-los pode levar o usuário a optar por um similar, não tão poderoso, mas que seja mais rápido de assimilar e prático de operar.

Por isso, o ponto crítico de todo programa é o modo como o operador deve introduzir os dados. No CP 400 existem vários recursos exclusivos que podem ser explorados para se conseguir uma tela de introdução extremamente funcional.

Baseado em alguns dos inúmeros programas que surgem nas revistas especializadas, cheguei a um programa que permite a introdução e edição de vários itens de uma só vez na mesma tela. O programa pode ser usado como sub-rotina do seu programa de cadastramento, processamento de texto, planilha ou qualquer outro que requeira a introdução e visualização conjunta de uma série de informações na tela.

A filosofia do programa é relativamente simples. Cada item possui um determinado número de bytes que limita seu comprimento e uma posição inicial na tela. O programa demarca com uma cor diferente o campo permitido para introdução de cada item. O operador pode introduzir os dados em qualquer ordem, e até voltar atrás e modificar um item já introduzido. Um cursor controlado pelas quatro teclas de direção indica onde a próxima introdução será colocada. Se o operador apenas usar as setas e a tecla ENTER, não produzirá nenhuma alteração no que foi introduzido.

O programa na Listagem 5-1 foi desenvolvido para controlar os 6 itens que compõem uma ficha de um cadastro hipotético. Na Tabela 5-1 está uma relação das alterações que devem ser realizadas para se adaptar o programa para outras aplicações. No caso do nosso exemplo, cada item deve ser identificado pelo seu comprimento em bytes e pela posição do primeiro caractere do item na tela. Para essa posição você deve utilizar o *lay-out* da tela para PRINT@, que está na página 266 do seu manual de operação. Lembre-se sempre de que essa posição mais o comprimento do item não podem ultrapassar 511, que é a última posição disponível na tela do CP 400. Se isso acontecer, o programa simplesmente ignorará o comprimento que você definiu para o item e adotará valores de forma a preencher o que restou no vídeo.

Como recurso especial, esse programa dispõe de uma pequena rotina em linguagem de máquina para demarcar os campos de introdução dos itens. Essa

rotina utiliza os endereços reais na memória dos pontos iniciais e finais de cada campo. O endereço inicial de um dado campo é calculado pela fórmula: EI = 1024 + PI; onde EI é o endereço e PI, a posição inicial para PRINT@. O endereço final é dado pela fórmula: EF = EI + CI − 1; onde EF é o endereço final e CI, o comprimento do item, sempre maior que zero. A Tabela 5-2 mostra os valores de PI e CI para os nossos 6 itens de exemplo.

Ainda sobre a rotina em linguagem de máquina, que está armazenada na linha 61030, o primeiro valor é exatamente o código do caractere gráfico que preencherá o campo com a cor desejada. Esse código é calculado pela fórmula: CC = 127 + 16*COR; onde CC é o código do caractere e COR, o código da cor utilizada para cada item. No exemplo foram usadas as cores cinza (COR = 5) e amarela (COR = 2), mas existem outras opções: preto, 0; azul, 3; vermelho, 4; ciano (tá no "Aurélio", é um "azul-esverdeado"), 6; magenta (vermelho ligeiramente arroxeado), 7; e laranja, 8. O verde (COR = 1) também pode ser utilizado, mas para isso a cor de fundo deve ser mudada, acrescentando-se um código de cor na instrução CLS da linha 61040.

Listagem 5-1 — Processador de tela

```
1 CLEAR1000,&H7DFF
2 DIMB$(511),C$(5),CC(5),PI(5),C
I(5):LM=5:TC$=CHR$(8)+CHR$(9)+CH
R$(10)+CHR$(94)+CHR$(13)+CHR$(12
)
61000 'ROTINA DE EDICAO DE TELA
61010 FORT=32256TO32270
61020 READD:POKET,D:NEXTT:DEFUSR
0=32257
61030 DATA 207,142,0,0,182,126,0
,167,128,140,0,0,45,246,57
61040 CLS:PRINT@0,"NOME:";:PRINT
@32,"ENDERECO:";:PRINT@152,"N.:"
;:PRINT@160,"CIDADE:";:PRINT@224
,"CEP:";:PRINT@245,"ESTADO:";
61050 DATA 207,5,25
61060 DATA 159,64,35
61070 DATA 159,155,5
61080 DATA 207,167,30
61090 DATA 159,228,5
61100 DATA 175,254,2
61110 FORR=0TOLM:READCC(R),PI(R)
,CI(R)
61120 E1=1024+PI(R):E2=E1+CI(R)
61130 E3=INT(E1/256):E4=E1-E3*25
6
61140 E5=INT(E2/256):E6=E2-E5*25
```

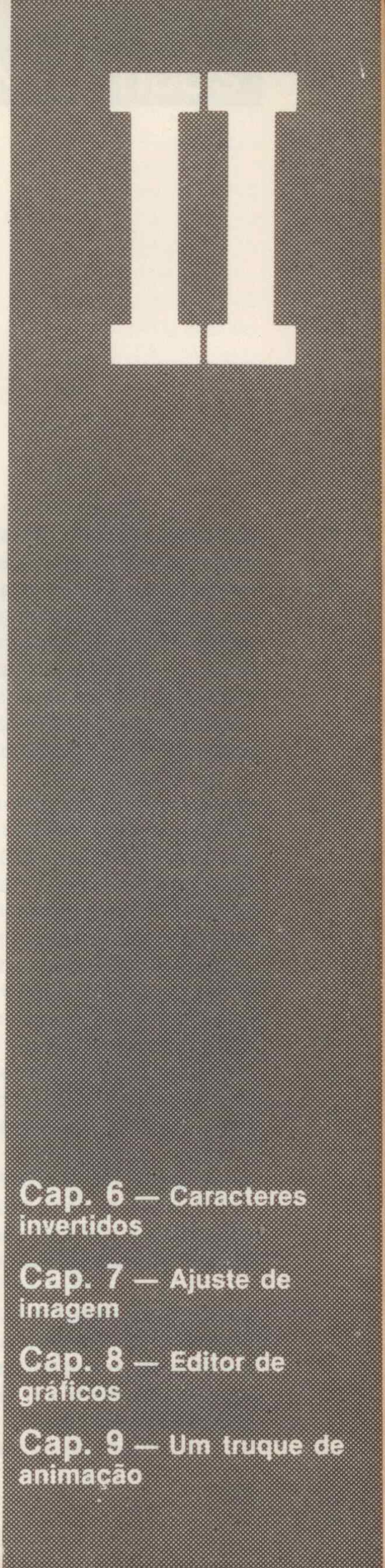
```
6
61150 POKE32256,CC(R):POKE32258,
E3:POKE32259,E4:POKE32266,E5:POK
E32267,E6
61160 K=USR0(I):NEXT
61170 CP=PI(0)
61180 IFCP)=PI(L)+CI(L)THENIF L=
LM THENL=0:CP=PI(L)ELSEL=L+1:CP=
PI(L)
61190 E=1024+CP:CE=PEEK(E)
61200 A$=INKEY$:POKEE,CC(L)-1:PO
KEE,CC(L):IFA$=""THEN61200ELSEPR
INT@CP,B$(CP);
61210 AA=INSTR(1,TC$,A$):ON AA G
OTO 61240,61260,61270,61290,6127
0,61310
61220 B$(CP)=A$:PRINT@CP,B$(CP);

61230 CP=CP+1:GOTO61180
61240 CP=CP-1:IFCP(PI(L)THENIF L
(1 THEN L=LM ELSEL=L-1ELSE61180
61250 CP=PI(L)+CI(L)-1:GOTO61190
61260 CP=CP+1:GOTO61180
61270 L=L+1:IF L)LM THENL=0
61280 CP=PI(L):GOTO61190
61290 L=L-1:IFL(0THENL=LM
61300 CP=PI(L):GOTO61180
61310 FORI=0 TO LM
61320 FORC=PI(I) TO PI(I)+CI(I)
61330 C$(I)=C$(I)+B$(C)
61340 NEXTC,I
61350 ' AQUI ENTRA UM RETURN SE
FOR USADA COMO SUB-ROTINA
```

Tabela 5-1 — Relação das alterações necessárias no programa

linha 2: mudar os índices das matrizes C$, CC, PI, e CI para N-1 (onde N é o número de itens que se deseja usar). Mudar também o valor de LM para N-1.

linhas 61050 a 61100: determinação dos parâmetros dos itens. Devem obedecer a seqüência DATA CC, PI, CI (onde CC é o código do caracteré que deve preencher o campo do item; PI, a posição inicial do item para um PRINT@; CI, o número de caracteres para cada item).

Tabela 5-2 — Valores de CC, PI e CI do nosso exemplo				
item	CC	PI	CI	Cor
NOME:	207	5	25	cinza
ENDEREÇO:	159	64	35	amarelo
N.:	159	155	5	amarelo
CIDADE:	207	167	30	cinza
CEP:	159	228	5	amarelo
ESTADO:	175	254	2	azul



Recursos de vídeo

6 CARACTERES INVERTIDOS

Você sabe como é que se faz para escrever letras verdes em fundo preto na tela, ao contrário do normal? Certo, é só pressionar as teclas SHIFT e 0 juntas e digitar o texto normalmente. Mas, e os números e outros símbolos? No manual e no cartão de referência não tem nenhuma dica sobre isso! Então o jeito é descobrir sem a ajuda do manual. Aliás, o manual aborda quase tudo superficialmente, apenas para que a gente comece a conhecer seus recursos. Na verdade, se o manual fosse mais fundo, provavelmente teria o dobro de páginas e aí muita gente iria desistir lá pelo meio do primeiro capítulo, quando começasse a se falar em "descrição funcional da pinagem do conector de expansão".

Mas, voltando ao assunto, é possível imprimir números e símbolos em negativo na tela, sim. Existem, inclusive, duas maneiras de se fazer isso. A primeira é imprimir o texto usando os caracteres de CHR$(96) a CHR$(127), em instruções PRINT normais. Esse método permite imprimir todas as letras mais seis símbolos, todos em negativo. A outra forma é colocando, com POKE, os códigos correspondentes aos negativos de letras, números, pontuação e símbolos diretamente na memória de texto do CP 400. Essa parte da memória fica entre os bytes 1024 (canto superior esquerdo da tela) e 1535 (canto inferior direito).

O programa na Listagem 6-1 ilustra essas duas técnicas, colocando inclusive o texto normal para que se possa confrontá-los. As duas primeiras linhas mostram o texto normal. A terceira linha mostra os caracteres negativos usando PRINT e CHR$. Essa técnica, usando códigos de 96 a 127, não inclui um espaço em negativo, mas pode ser utilizado o caractere gráfico 128 que nada mais é que um bloco negro.

Outra opção para um espaço em negativo é usar POKE n, 32. Nesse caso o espaço vai ficar ligeiramente esverdeado, pois você estará usando outra técnica de impressão. Essa técnica é mostrada na terceira divisão da tela, onde é usada a instrução POKE para gerar 64 caracteres em negativo. Os códigos usados vão de 0 a 63.

Para caracteres em negativo usa-se, de preferência, um fundo negro. Isso é conseguido rapidamente com CLS0. Mas, cuidado... se o ajuste de brilho do seu TV estiver muito alto, a tela poderá ficar cheia de manchas esverdeadas em torno dos caracteres. A causa dessas manchas é a própria cor de fundo dos caracteres em negativo, que não é exatamente preta, mas de um tom levemente esverdeado. Pode-se eliminar essas manchas simplesmente colocando (com PO-

KE) o código 32 em toda a memória da tela. Assim toda a tela fica levemente esverdeada... se não consegue vencê-lo, una-se a ele!

Outra mudança possível na cor dos caracteres é a que se consegue com SCREEN0,1. Como está explicado no manual, a tela normal de texto é fornecida com SCREEN0,0. Essa instrução define caracteres negros sobre fundo verde. Com SCREEN0,1 obtemos uma tela laranja com caracteres negros levemente avermelhados. No entanto, essa condição só permanece enquanto não se usar nenhuma instrução que trate a tela, como PRINT, INPUT ou SET. Uma boa sugestão para se manter a tela laranja é fazer o computador entrar em um ciclo com FOR/NEXT, INKEY$, EXEC44539, PLAY ou SOUND.

Listagem 6-1 — Caracteres em negativo

```
490 'CARACTERES EM NEGATIVO
495 'PAULO ADDAIR    JUN/85
500 CLS:PRINT"32 <--          print
      --> 63":P=32:C=32:GOSUB5
90
510 PRINT@96,"64 <--       print p
oke     --> 95":GOSUB590
520 PRINT@192,"96 <--          prin
t       --> 127":GOSUB590
530 PRINT@288,"0 <--             pok
e         --> 31":P=1344:C=0:GOSU
B600
540 PRINT@384,"32 <--            pok
e         --> 63":P=1440:GOSUB600

550 PRINT@448,"<-- VEJA AS LINHA
S READ/DATA -->";
560 FORP=1504TO1535:READD$:D=VAL
("&H"+D$):POKEP,D:NEXT
570 DATA11,15,01,0C,11,15,05,12,
20,20,14,05,03,0C,01,20,01,0C,14
,05,12,01,20,20,01,13,20,03,0F,1
2,05,13
580 EXEC44539:SCREEN0,1:EXEC4453
9:SCREEN0,0:GOTO580
590 FORC=C TOC+31:PRINT@P,CHR$(C
):P=P+1:NEXT:P=P+64:RETURN
600 FORC=C TOC+31:POKEP,C:P=P+1:
NEXT:RETURN
```

```
LINHA: 570        156
FIM:   157
```

7 AJUSTE DE IMAGEM

A característica mais marcante do CP 400 Color está exatamente em seus recursos gráficos e de cores. Mas para se conseguir a melhor qualidade possível é preciso que o aparelho de TV esteja perfeitamente ajustado. Aqueles que conhecem um pouco do assunto sabem que esses ajustes, principalmente o de convergência, requerem aparelhos especiais, capazes de gerar os padrões de imagem necessários. Esses aparelhos podem ser conseguidos em forma de kit ou então montados, mas são sempre caros.

Mas se você tem um CP 400, ele mesmo pode se encarregar de gerar esses padrões, sem nenhum custo adicional. O programa deste capítulo gera uma imagem em preto e branco (cinza, para ser exato), para ajuste de luminância (escala de cinza); um conjunto de barras coloridas para ajuste de saturação e matiz; e imagens com linhas verticais, horizontais e diagonais para ajuste dinâmico de convergência. Outro padrão gerado pelo programa é uma tela pontilhada, para ajuste estático de convergência. Para facilitar, o ponto central é destacado por um pequeno círculo a sua volta. Além disso, um grande círculo ajuda a checar a linearidade.

Para ajustar o seu TV, você deve sempre começar pela escala de cinza (pressione A) seguida pela centralização e linearidade (use C). Verifique também a nitidez da imagem com essa opção. A qualquer momento você pode consultar o menu para rever as opções disponíveis, pressionando M. Passando para a opção P, que é o teste de convergência estática, certifique-se de ajustar o receptor para obter o máximo de definição entre os pontos. Em seguida, verifique novamente a nitidez (C, de novo). O próximo ajuste é o teste dinâmico de convergência, usando os padrões de linhas verticais, horizontais e diagonais (H, V e D). Para maior eficiência do ajuste de convergência, deixe o nível de cor no mínimo.

Apesar de ser um procedimento demorado, o uso desse programa é muito gratificante, pois resultará numa imagem nítida, sem aquelas inconvenientes sombras na separação das cores. É claro que, se os ajustes forem realizados por um técnico, o resultado será melhor ainda, pois ele poderá abrir o aparelho de TV e realizar ajustes individuais nos próprios circuitos geradores da imagem. Em hipótese alguma tente abrir o seu aparelho para fazê-los, pois existem tensões elevadas lá dentro, da ordem de milhares de volts e só um técnico saberá como lidar com elas. Você deve se limitar a regular a imagem através dos controles de brilho, matiz, saturação, contraste, cor e sintonia fina de seu receptor. Caso o aparelho possua sintonia automática (AFT), desligue-a para proceder aos ajustes, podendo acioná-la novamente quando terminar.

Listagem 7-1 — Gerador de padrões para TV

```
690 'GERADOR DE PADROES PARA TV
695 'PAULO ADDAIR              MAI/85
700 CLS:OP$="MDCPHVIFA"
710 GOSUB770
720 GOSUB810:GOTO780
730 PMODE4,1
740 SCREEN1,1
750 COLOR5,0
760 PCLS
770 RETURN
780 A$=INKEY$:IFA$=""THEN780
790 S=INSTR(1,OP$,A$):ON S GOSUB
 860,880,1010,1040,910,960,1110,
1150,1140
800 GOTO780
810 PCLS:CLS
820 CLS
830 FORY=0 TO 21:FORX=0 TO63
840 SET(X,Y,INT(X/8)+1):NEXTX,Y
850 PRINT@384," mENU   CiRCULO p
ONTOS  VaRRED.":PRINT@416," cENT
RAL.dIAGON.hORIZ.   vERT.":PRINT
@448," fIM"
860 SCREENO
870 RETURN
880 GOSUB730
890 DRAW"BM128,96U95D190U95L127R
254BL127E127G254BM128,96F127H254
"
900 RETURN
910 GOSUB730
920 FORY=12TO180STEP14
930 LINE(0,Y)-(256,Y),PSET
940 NEXTY
950 RETURN
960 GOSUB730
970 FORX=2TO180STEP14
980 LINE(X,0)-(X,192),PSET
990 NEXTX
1000 RETURN
1010 GOSUB910
1020 GOSUB970
1030 RETURN
```

```
1040 GOSUB730
1050 FORX=2TO254STEP14
1060 FORY=12TO180STEP14
1070 PSET(X,Y,5)
1080 NEXTY,X
1090 CIRCLE(128,96),7,5
1100 RETURN
1110 GOSUB730
1120 CIRCLE(128,96),92,5
1130 RETURN
1140 GOSUB730:PCLS5:RETURN
1150 CLS:END
```

```
LINHA: 770          121

LINHA: 870          246

LINHA: 970          42

LINHA: 1070         23

FIM:   208
```

8 EDITOR DE GRÁFICOS

Se você está pensando que este é mais um daqueles programas que só servem para "pintar o sete" na tela e depois gravar o resultado em uma fita ou disco, está muito enganado! Quando decidi criar este programa, o meu maior problema não era criar os desenhos, mas fazer com que o programa que iria utilizá-los os criasse. A técnica de trabalhar com imagens gravadas é indicada para sistemas com disquete, que são muito mais rápidos que as fitas e não necessitam de uma seqüência rígida de gravação das imagens.

Quando o próprio programa gera as imagens fica muito complicado escrevê-lo, pois temos que introduzir cada instrução na ordem certa e com os parâmetros corretos. Aí então testamos o programa, verificamos o resultado, corrigimos a rotina e testamos de novo. E assim até que a droga da rotina faça o que queremos! Para mim essa sempre foi a pior parte da programação de rotinas gráficas. O longo e tedioso processo de depuração da listagem.

Por isso existem tantos programas para "edição de gráficos". Você simplesmente pressiona algumas teclas para gerar as figuras no vídeo e, quando estiver satisfeito, aperta uma tecla, dá um nome à imagem e pronto! Uma cópia de sua obra (ou devemos dizer um original) é automaticamente incluída no diretório de seu disco, ao lado do Procalc e do Zaxxon.

O nosso "Editor de gráficos" oferece as mesmas facilidades para gerar a imagem. A grande diferença é que, como resultado, ele gera uma "sub-rotina gráfica" no lugar de simplesmente gravar a imagem. Essa sub-rotina é que será gravada, se você quiser. Ou então anexada a um ou mais programas que você tenha desenvolvido. Na realidade a sub-rotina nada mais é do que uma reprodução dos comandos e parâmetros que você utilizou para gerar a imagem, em BASIC. Você pode até escolher a numeração das linhas da sub-rotina para acomodá-la perfeitamente no seu programa principal.

A operação do programa é tanto por menu, para selecionar o comando e parâmetros, como por edição de tela, para posicionar um cursor sobre o desenho indicando os limites da figura que se quer desenhar. No menu você indica, movendo um cursor com as setas e pressionando ENTER, qual comando deverá ser realizado. Se for necessário algum parâmetro complementar, como cor por exemplo, o cursor passa para a posição correspondente e você faz a introdução. Suponhamos que a escolha foi traçar um círculo. Depois de entrar com a cor desejada, a tela gráfica surge, mostrando um pequeno cursor. Posicione,

sempre com as setas, o cursor na posição que será o centro e pressione ENTER. O programa espera então que você leve o cursor até um ponto por onde a circunferência deve passar. Uma vez introduzido o segundo ponto o programa traça a circunferência sem que seja necessário introduzir valores como pares ordenados ou raio.

As linhas e os retângulos também são definidos por dois pontos, da mesma forma que a circunferência. No caso dos retângulos, você deve definir os pontos extremos de uma das diagonais, e o programa faz o resto. Se você pressionar SHIFT juntamente com as setas, o cursor se deslocará dez pontos de uma só vez. Caso pressione qualquer tecla errada, o programa retorna ao menu para um novo comando.

Quando você tiver terminado sua criação, selecione a linha correspondente à gravação, pressione ENTER e introduza o nome do arquivo BASIC que será criado. Você deve também escolher o número inicial e o incremento das linhas da rotina que será gravada nesse arquivo.

O PROGRAMA

No início do programa são definidas algumas variáveis cujos valores permanecerão inalterados durante toda a execução. Estas variáveis armazenam valores freqüentemente utilizados e representam uma maneira mais prática de se manipular esses valores do que ficar incluindo instruções IF/THEN ao longo de todo o programa.

A seguir é acionada a rotina que cria a tela de comandos, nas linhas 2065 a 2140, e retorna à linha 1270 para varrer o teclado aguardando a introdução de uma tecla de comando. As linhas 1285 e 1290 testam se a tecla pressionada foi uma seta para cima ou uma para baixo (códigos ASCII 94 ou 10). Se foi uma dessas, o valor da variável CM é ajustado de acordo com o caso. Se a tecla foi um ENTER, a linha 1315 passa o controle para uma das rotinas de comando, com a instrução ON CM GOTO.

As linhas 1320 a 1415 manipulam a introdução dos valores iniciais das coordenadas x e y. É possível mudar essa rotina para usar joysticks, mas não é aconselhável. Lembre-se de que os joysticks têm uma resolução baixa (0 a 63) se comparada com a resolução da tela gráfica (0 a 191 em y e 0 a 255 em x) e isso dificulta o posicionamento preciso do cursor.

Os dados sobre a rotina capaz de gerar a imagem que está sendo editada na tela são armazenados na matriz L$. LP aponta o lugar onde será colocado o próximo elemento nessa matriz. Cada elemento dessa variável equivale a um comando dado para criar a imagem. Como a matriz é definida no começo do programa com 50 elementos, é possível elaborar figuras com até 50 comandos, o que permite criar ilustrações relativamente complexas no vídeo.

A rotina de gravação está nas linhas 1865 a 1940. Ela opera originalmente com fita, mas pode ser facilmente adaptada para disquete. Para começar, ela grava instruções de inicialização do modo gráfico com PMODE3,1 PCLS e SCREEN1,1. Depois ela pega os elementos de L$ e acrescenta os números de linhas, segundo especificado no início da própria rotina. Para terminar, é colocada uma linha de ciclo infinito, tipo N GOTO N, para fixar a página gráfica na tela. Pode-se substituir GOTO N por um RETURN, se necessário.

Com a sua rotina gráfica gravada, você pode passar a se preocupar com o restante do programa que a utilizará. Quando este estiver pronto, utilize o programa de fusão do Capítulo 10 para inserir a rotina gráfica. Se dispor de disquete, basta usar o comando MERGE. Para disquetes, mude todas as instruções que usam #-1, para #1. Assim você passa a usar o disquete em vez da fita. Se preferir rodar a rotina isoladamente, use CLOAD "nome de arquivo" e RUN (ou RUN "nome" para disquete).

Listagem 8-1 — Editor de gráficos

```
1195 '  EDITOR GRAFICO
1200 CLEAR 1000:PMODE3,1:PCLS:DI
ML$(50)
1205 D$=CHR$(94)+CHR$(9)+CHR$(10
)+CHR$(8)+CHR$(95)+CHR$(93)+CHR$
(91)+CHR$(21)
1210 CS$(0)="VRD AMR AZL VRM":CS
$(1)="CNZ CIA MAG ORA"
1215 CLS:BL$=STRING$(31,32):FC=4
:CM=1:CS=0:BC=1:X=127:Y=91
1220 C$(1)=" VERDE    ":C$(2)=" A
MARELO ":C$(3)=" AZUL     ":C$(4)
=" VERMELHO":C$(5)=" CINZA    ":C
$(6)=" CIANO    ":C$(7)=" MAGENTA
 ":C$(8)=" LARANJA "
1225 GOTO 2065
1230 PRINT@CP,CHR$(143);:C$=INKE
Y$:IF C$="" THEN PRINT@CP,CHR$(1
75);:GOTO1230
1235 RETURN
1240 C=INSTR(1,"12345678",C$)
1245 IF C=0 THEN RETURN
1250 IF CS=1 AND C<4 THEN C=C+4
1255 IF CS=0 AND C>4 THEN C=C-4
1260 RETURN
1265 'INTRODUCAO COMANDO
1270 C$=INKEY$:IF C$="" THEN 127
0
1275 PRINT@449,"O PROGRAMA ESTA'
 COM"LP"LINHAS";
1280 CP=CM*32:GOSUB 1230
1285 IF ASC(C$)=94 THEN CM=CM-1
1290 IF ASC(C$)=10 THEN CM=CM+1
1295 IF CM<1 THEN CM=13
1300 IF CM>13 THEN CM=1
1305 IF ASC(C$)=13 THEN 1315
```

```
1310 GOTO 1280
1315 ON CM GOTO 1425,1465,1465,1
670,1745,1740,1515,1590,1590,180
0,1950,1965,1865
1320 PSET(X,Y,BC)
1325 PC=PPOINT(X,Y)
1330 C$=INKEY$:IF C$="" THEN 133
0
1335 C=INSTR(1,D$,C$)
1340 PSET(X,Y,PC)
1345 ON C GOTO 1355,1360,1365,13
70,1375,1380,1385,1390
1350 C=ASC(C$):RETURN
1355 Y=Y-1:GOTO1395
1360 X=X+1:GOTO1395
1365 Y=Y+1:GOTO1395
1370 X=X-1:GOTO1395
1375 Y=Y-10:GOTO1395
1380 X=X+10:GOTO1395
1385 Y=Y+10:GOTO1395
1390 X=X-10:GOTO1395
1395 IF X>255 THEN X=255
1400 IF X<0 THEN X=0
1405 IF Y>191 THEN Y=191
1410 IF Y<0 THEN Y=0
1415 PC=PPOINT(X,Y):PSET(X,Y,(PC
 AND 7)+1):GOTO 1330
1420 'MUDANCA DE CORES
1425 CP=46:GOSUB 1230
1430 IF C$="1" THEN CS=1:GOTO 14
45
1435 IF C$="0" THEN CS=0:GOTO 14
45
1440 SOUND 10,5: GOTO 1425
1445 PRINT@45,CS;CS$(CS);
1450 L$(LP)="SCREEN 1,"+C$:LP=LP
+1
1455 GOTO 1275
1460 'COR DE TRACO & FUNDO
1465 CP=81:GOSUB 1230:GOSUB 1240

1470 IF C=0 THEN SOUND 10,5:GOTO
 1465
1475 PRINT@80,C;C$(C);:BC=C
1480 CP=113:GOSUB 1230:GOSUB 124
0
```

```
1485 IF C=0 THEN SOUND 10,5:GOTO
 1480
1490 PRINT@112,C;C$(C);:FC=C
1495 COLOR FC,BC
1500 L$(LP)="COLOR"+STR$(FC)+","
+STR$(BC):LP=LP+1
1505 GOTO 1275
1510 'CIRCULOS
1515 CP=242:GOSUB 1230:GOSUB 124
0
1520 IF C=0 THEN 1270
1525 CC=C:PRINT@241,CC;C$(CC);
1530 SCREEN 1,CS
1535 GOSUB 1320: IF C<>13 THEN 1
275
1540 XO=X:YO=Y
1545 GOSUB 1320:IF C<>13 THEN 12
75
1550 R=INT(SQR((XO-X)^2+(YO-Y)^2
))
1555 IF AR=1 THEN 1610
1560 CIRCLE(XO,YO),R,CC
1565 L$(LP)="CIRCLE("+STR$(XO)+"
,"+STR$(YO)+"),"+STR$(R)+","+STR
$(CC)
1570 LP=LP+1
1575 IF AR=1 THEN 1610
1580 GOTO 1270
1585 'ELIPSES & ARCOS
1590 CP=271:GOSUB 1625:ST$=N$:ST
=N
1595 CP=283:GOSUB 1625:SP$=N$:SP
=N
1600 CP=304:GOSUB 1625:HW$=N$:HW
=N
1605 AR=1:GOTO 1530
1610 L$(LP)="CIRCLE("+STR$(XO)+"
,"+STR$(YO)+"),"+STR$(R)+","+STR
$(CC)+","+HW$+","+ST$+","+SP$
1615 CIRCLE(XO,YO),R,CC,HW,ST,SP

1620 AR=0:LP=LP+1:GOTO1270
1625 'INTRODUCAO NUMERICA
1630 PRINT@CP,"    ";:N$=""
1635 GOSUB1230:IF ASC(C$)=8 AND
LEN(N$)>0 THEN N$=LEFT$(N$,LEN(N
$)-1):CP=CP-1:GOTO1635
```

```
1640 IF ASC(C$)=13 THEN 1660
1645 IF ASC(C$)<46 OR ASC(C$)>57
THEN SOUND 10,5:GOTO 1635
1650 N$=N$+C$:IF LEN(N$)=4THEN16
60
1655 PRINT@CP,C$;:CP=CP+1:GOTO 1
635
1660 N=VAL(N$):RETURN
1665 'LINHAS
1670 SCREEN 1,CS
1675 GOSUB 1320:IF C<>13 THEN 12
75
1680 XO=X:YO=Y:PSET(X,Y,FC)
1685 GOSUB 1320:IF C<>13 THEN 12
75
1690 LINE(XO,YO)-(X,Y),PSET
1695 GOSUB 1725
1700 L$(LP)=L$(LP)+"PSET":LP=LP+
1
1705 GOSUB 1320:IF C<>13 THEN 12
75
1710 LINE -(X,Y),PSET
1715 GOSUB 1730
1720 L$(LP)=L$(LP)+"PSET":LP=LP+
1:GOTO 1705
1725 L$(LP)="LINE("+STR$(XO)+","
+STR$(YO)+")-("+STR$(X)+","+STR$
(Y)+"),":RETURN
1730 L$(LP)="LINE-("+STR$(X)+","
+STR$(Y)+"),":RETURN
1735 'QUADRADOS
1740 F=1
1745 SCREEN 1,CS
1750 GOSUB 1320:IF C<>13 THEN 12
80
1755 XO=X:YO=Y:PSET(X,Y,FC)
1760 GOSUB 1320:IF C<>13 THEN 12
80
1765 IF F THEN LINE(XO,YO)-(X,Y)
,PSET,BF
1770 IF F=0 THEN LINE(XO,YO)-(X,
Y),PSET,B
1775 L$(LP)="LINE("+STR$(XO)+","
+STR$(YO)+")-("+STR$(X)+","+STR$
(Y)+"),PSET,B"
1780 IF F THEN L$(LP)=L$(LP)+"F"
```

```
1785 LP=LP+1
1790 F=0:GOTO 1270
1795 'COLORIR
1800 CP=329:GOSUB 1230:GOSUB 124
0
1805 IF C=0 THEN 1275
1810 PP=C:PRINT@CP-1,PP;C$(PP);
1815 CP=350:GOSUB1230:GOSUB 1240

1820 IF C=0 THEN 1275
1825 BB=C:PRINT@350,BB;
1830 SCREEN 1,CS
1835 GOSUB1320:IF C<>13 THEN 127
5
1840 PAINT(X,Y),PP,BB
1845 L$(LP)="PAINT("+STR$(XO)+",
"+STR$(YO)+"),"+STR$(PP)+","+STR
$(BB)
1850 LP=LP+1
1855 GOTO1270
1860 'GRAVACAO
1865 GOSUB1995:GOSUB2030
1870 PRINT@449,"PREPARE O GRAVAD
OR"

1875 C$=INKEY$:IF C$="" THEN 187
5
1880 PRINT@480,"GRAVANDO.";
1885 OPEN "O",-1,N$
1890 X$=STR$(NL)+" PMODE 3,1:PCL
S:SCREEN 1,1":PRINT#-2,X$
1895 NL=NL+IN
1900 FOR I=0 TO LP
1905 X$=STR$(NL+I*IN)+" "+L$(I)
1910 PRINT#-2,X$
1915 IF I/5=INT(I/5) THEN PRINT"
. ";
1920 NEXTI:PRINT@480,BL$;
1925 NL=NL+(I+1)*10:X$=STR$(NL)+
" GOTO"+STR$(NL)
1930 PRINT#-1,X$
1935 CLOSE
1940 GOTO2070
1945 'VER A FIGURA
1950 SCREEN 1,CS
1955 GOTO 1270
```

```
1960 'SALTAR UM ARQUIVO
1965 CP=400:GOSUB 1995
1970 PRINT@449," PREPARE O GRAVA
DOR"
1975 C$=INKEY$:IF C$="" THEN 197
5
1980 PRINT@480,"SALTANDO. ";
1985 SKIPF N$
1990 GOTO2070
1995 'NOME DE PROGRAMA
2000 N$="":CP=417:PRINT@CP,"NOME
 DO PROGRAMA:";:CP=434
2005 GOSUB 1230
2010 IF ASC(C$)=13 THEN RETURN
2015 PRINT@CP,C$;
2020 N$=N$+C$:IF LEN(N$)=8 THEN
RETURN
2025 CP=CP+1:GOTO 2005
2030 'NUMERO LINHA E INCREMENTO
2035 PRINT@417,"LINHA INICIAL
   INCREM."
2040 CP=432:GOSUB 1625:NL=INT(N)

2045 IFNL<1 THEN NL=1
2050 CP=443:GOSUB 1625:IN=INT(N)

2055 IFIN<1 THEN IN=1
2060 RETURN
2065 'MENU
2070 CLS:PRINT@8,"EDITOR DE GRAF
ICOS";
2075 PRINT@33,"CORES ATIVAS";CS;
CS$(CS);
2080 PRINT@65,"COR DE FUNDO    1
 VERDE";
2085 PRINT@97,"COR DO TRACO    4
 VERMELHO"
2090 PRINT@129,"LINHAS";
2095 PRINT@161,"QUADRADO";
2100 PRINT@193,"QUAD.CHEIO";
2105 PRINT@225,"CIRCULO   COR";
2110 PRINT@257,"ARCO   INICIO
   PARADA";
2115 PRINT@289,"ELIPSE   H/W";
2120 PRINT@321,"COLORIR
    BORDA";
```

```
2125 PRINT@353,"VER A FIGURA";
2130 PRINT@385,"SALTAR UM ARQUIV
O";
2135 PRINT@417,"GRAVAR O PROGRAM
A";
2140 GOTO 1275
```

```
LINHA: 1240     16      LINHA: 1740     100
LINHA: 1290     32      LINHA: 1790     203
LINHA: 1340     103     LINHA: 1840     107
LINHA: 1390     128     LINHA: 1890     247
LINHA: 1440     63      LINHA: 1940     240
LINHA: 1490     37      LINHA: 1990     178
LINHA: 1540     172     LINHA: 2040     40
LINHA: 1590     123     LINHA: 2090     15
LINHA: 1640     108     LINHA: 2140     172
LINHA: 1690     148     FIM:    172
```

9 UM TRUQUE DE ANIMAÇÃO

Quando a gente quer fazer animação com o computador, a solução mais óbvia parece ser criar os vários quadros em diferentes páginas de vídeo, armazenar os elementos do cenário em variáveis indexadas (com GET) e movimentá-los na tela com PUT. Simples, não?

Não! Não é simples nem tão pouco rápido. Quando precisamos trabalhar com cenários relativamente complexos, o programa torna-se muito lento e complicado. Mas existe uma forma muito mais prática, rápida e simples de fazer animação no seu CP 400. Basta mover *a tela toda* no lugar de mover os elementos do cenário.

Calma! Não rasgue esse livro ainda! Mover a tela toda ao invés de partes dela não é como puxar o piano em vez da banqueta. Se você souber como mudar o endereço na memória que diz ao computador onde começa a página de vídeo, isso é fácil. Só que o manual não diz que endereço é esse! Não faz mal, eu conto.

O seu CP 400 Color, e todos os seus compatíveis, permite que você defina a posição da tela de alta resolução em qualquer ponto da memória. A única restrição é que o endereço inicial deve ser sempre múltiplo de 512. Como está explicado no manual, cada modo gráfico utiliza um determinado número de página de memória para formar um quadro (uma tela gráfica). O modo 0, por exemplo, usa 1 página por quadro, o que representa 1536 bytes ou 3 vezes 512. Então, se mudarmos o início da tela 512 bytes para mais ou para menos, o efeito, para quem observar a TV, será um deslocamento de um terço da imagem para cima ou para baixo. Isso no modo 0, o de mais baixa resolução. Nos modos 3 ou 4, o deslocamento corresponde a 1/12 da imagem. Portanto, mudando esse endereço, fazemos a imagem se deslocar continuamente pela tela, criando um efeito de animação sem ter que mexer em um byte sequer dentro da memória de vídeo. Nada mais simples e rápido!

O programa da Listagem 9-1 utiliza o modo 3, que é o que na minha opinião apresenta uma animação melhor. Por isso as explicações neste capítulo vão se limitar a esse modo, apesar de o efeito ser possível para qualquer outro.

O endereço inicial é definido pelo "registrador de seleção de página" dentro do CP 400. Esse registrador consiste em um conjunto de sete bits que indicam em qual múltiplo de 512 está o início da tela gráfica. Se usarmos o valor 3, o endereço inicial será 1536 que é 3 vezes 512. Portanto o RSP (registrador de seleção de página) conterá o valor 3 em binário, ou seja, 0000011.

Para você ter acesso a esse registrador em BASIC, ele está representado na memória por vários endereços. Para ativar ou desativar cada bit, você deve dar um POKE em uma determinada posição, segundo a Tabela 1.

Tabela 1 — POKEs do RSP

Bit	Desativar	Ativar
0	65478	65479
1	65480	65481
2	65482	65483
3	65484	65485
4	65486	65487
5	65488	65489
6	65490	65491

Para representar esse registrador em um programa basta usar uma variável inteira que passa a ser um pseudoregistrador. A transferência do valor desse pseudoregistrador para o registrador real é realizada por uma sub-rotina que analisa bit a bit esse valor e executa os POKEs necessários. Assim fica fácil operar o RSP. Você armazena o valor, chama a sub-rotina e pronto! Está mudado o endereço da tela.

O funcionamento da sub-rotina é muito simples. O valor armazenado em RSP é comparado, via instrução AND, com as oito potências de 2 que formam um byte, cada uma representando um dos bits ativos. Para cada comparação existem duas opções a seguir: ativar ou desativar o bit correspondente àquela potência.

Mas mudar o conteúdo do RSP não é a única providência a se tomar quando se quer realizar animação. Existem dois outros registradores que devem ser ajustados antes de se começar e ainda um endereço real de memória que possui informações importantes. Os dois registradores são o do "gerador de vídeo gráfico" e o de "controle". Para o nosso caso, basta colocar as seguintes instruções na rotina de animação:

```
POKE 65472,0:POKE 65475,0:POKE 65477,0
```

e

```
POKE 65314,224 OR (PEEK(65314)AND7)
```

O endereço que fica faltando arrumar quando se usa animação é o que indica o início da tela gráfica para as funções que precisam de coordenadas para operarem. Nada mais lógico! Se você está deslocando a varredura da imagem que se vê no monitor, a posição das coordenadas que você enxerga na TV também muda. O canto esquerdo superior já não é mais o endereço de memória que era, e todo o resto da tela também não. Assim, se você pretende mexer na memória gráfica durante a animação, então deve manter esse endereço atualizado. Esse endereço fica nas posições 186 e 187 da memória, representando os bytes mais e menos significativos, respectivamente (se você está acostumado com o formato de endereços do Z80, lembre-se que o 6809 usa o byte mais significativo no primeiro endereço).

Listagem 9-1 — Demonstração de animação

```
5 'EFEITOS DE ANIMACAO
10 PCLEAR8:PMODE3,1:PCLS:SCREEN1
,0
20 FORI=1TO200
30 X=RND(255):Y=RND(191):C=RND(4
)
40 PSET(X,Y,C):NEXTI
50 CIRCLE(127,180),10:SOUND128,1
0
60 PMODE3,5:SCREEN1,0
70 FORJ=1 TO4:PCOPY J TO J+4:NEX
TJ
80 FORRSP=18TO7STEP-1
90 GOSUB 60000
100 SOUNDRSP*10,6
110 NEXTRSP:GOTO80
60000 '  ROTINA DE ANIMACAO
60010 IF(RSP AND 1)=1THENPOKE654
79,0ELSEPOKE65478,0
60020 IF(RSP AND 2)=2THENPOKE654
81,0ELSEPOKE65480,0
60030 IF(RSP AND 4)=4THENPOKE654
83,0ELSEPOKE65482,0
60040 IF(RSP AND 8)=8THENPOKE654
85,0ELSEPOKE65484,0
60050 IF(RSP AND 16)=16THENPOKE6
5487,0ELSEPOKE65486,0
60060 IF(RSPAND32)=32THENPOKE654
89,0ELSEPOKE65488,0
60070 IF(RSPAND64)=64THENPOKE654
91,0ELSEPOKE65490,0
60080 POKE 65472,0:POKE65475,0:P
OKE65477,0
60090 POKE 65314,224 OR (PEEK(65
314) AND 7)
60100 RETURN
```

```
LINHA: 90          240

LINHA: 60070       6

FIM:   206
```

Recursos de fita cassete

10 CMERGE: UM PROGRAMA DE FUSÃO PARA FITA

Se você já trabalhou com programas em disco, deve ter notado que o DOS possui um comando muito útil para criação de programas extensos: o MERGE. Esse comando permite que você acrescente uma rotina gravada em disco ao programa que está na memória. Assim você pode fazer uma "biblioteca de sub-rotinas" para criar os seus programas com mais racionalidade.

Infelizmente, o CP 400 sem disquete não dispõe desse recurso. Qualquer ampliação no programa tem que ser feita "na unha". Uma tremenda injustiça para com os usuários dessa configuração, pois são eles que mais precisam de recursos para mexer com programas. Quando se dispõe de disquetes, normalmente se utiliza muito mais os programas já prontos, chamados "aplicativos".

Mas é possível usar alguns recursos não explorados no manual do Color (nem do CP 400 nem do TRS-80) para se criar um utilitário capaz de fazer exatamente o que o MERGE faz: você especifica quais programas que estão na fita serão fundidos e o MERGE faz o resto. Carrega o primeiro programa na memória e passa a carregar o segundo. Se houver linhas no primeiro programa com numeração que se repete no segundo, a linha do primeiro é substituída. Por isso o processo é chamado de fusão: os programas se fundem na memória formando um só programa final, sem repetição de linhas e na ordem correta.

Os recursos "esquecidos" nos manuais dos compatíveis com o CP 400 são detalhes aparentemente tão sem importância, que não fazem muita falta. O primeiro é que pode-se gravar um programa na fita em formato ASCII. O que quer dizer isso? Os arquivos gravados nesse formato podem ser lidos linha por linha por um outro programa, utilizando a instrução INPUT #-1.

Ou seja, podemos fazer um programa capaz de ler outro programa, ou vários outros... e é isso mesmo que nós queremos. O segundo recurso é que os arquivos em ASCII podem ser carregados na memória e executados como programas, desde que o seu conteúdo esteja sintaticamente correto. Isso significa que podemos processar os programas lidos com INPUT #-1, criando um terceiro que será gravado (com PRINT #-1) linha por linha e posteriormente recarregado como programa (CLOAD) e executado. Pode parecer meio complicado agora, mas rodando o programa você compreenderá melhor.

Grave os programas que você quer fundir usando o comando CSAVE "nome",A ENTER na ordem em que eles devem ser carregados pelo CMERGE. O detalhe do ",A" é importante, pois faz o computador gravar o programa no

formato ASCII, tratando cada linha como se fosse uma string gravada por um comando PRINT#-1. Apesar de não ser mencionada nos manuais, essa opção é comum nos sistemas com disco. Suas desvantagens são a demora e o espaço maior necessários para o programa.

Com os programas devidamente gravados, carregue o utilitário (CLOAD "CMERGE") e rode-o. Ele vai perguntar pelos nomes dos programas a serem fundidos. O segundo é o que tem precedência em caso de linhas com numeração equivalente. Além disso, você deve introduzir também o nome do arquivo que conterá o programa resultante. Depois disso o CMERGE pedirá para preparar a fita e o gravador para carregar os programas. Acione o botão PLAY para tocar a fita e pressione ENTER no Color. O motor do gravador vai começar a rodar e parar alternadamente até carregar completamente os dois programas. Os nomes aparecerão na tela à medida que forem sendo carregados.

Ao terminar a fusão dos programas, o CMERGE pedirá para trocar a fita. Você pode simplesmente acionar RECORD e PLAY no gravador , pressionar ENTER e o novo programa será gravado em seguida aos outros dois. Porém, é bom sempre manter uma certa distância entre as gravações na fita. Alguns segundos são suficientes.

Depois de gravar o programa final, o nosso utilitário encerra suas operações. Agora você pode carregar e rodar o seu novo programa. Pode até gravá-lo novamente, sem a opção ASCII (,A) para acelerar o processo.

CONCLUSÕES

A fusão não introduz erros ou qualquer outro tipo de modificação nos programas, a menos que estes se baseiem de alguma forma no endereçamento real das listagens originais. Se os dois programas utilizarem READ e DATA também há a possibilidade de mau funcionamento. Nesse caso é melhor reavaliar a operação dessas funções na listagem final.

O maior problema causado pela fusão é sem dúvida a limitação de memória quando combinando extensos programas. O CMERGE na Listagem 10-1 está dimensionado para máquinas de 32k. Para equipamentos com apenas 16k, mude a linha 10 para: 5 CLS:PCLEAR1:CLEAR10000:DIML$(650). Apesar de possuir uma pequena proteção contra estouro de memória, é sempre frustante, após algum tempo de espera, receber um "Memoria insuficiente" como resposta. Por isso, muito cuidado com o tamanho dos programas que pretende fundir.

CSORT: ORDENAÇÃO DE ARQUIVOS

Com muito poucas alterações, é possível transformar o CMERGE em um programa capaz de organizar arquivos de dados, como cadastros ou malas diretas. Em sistemas maiores esses programas são chamados de SORT (do inglês, ordenar). A Listagem 10-2 mostra como fica o nosso CSORT.

Listagem 10-1 — CMERGE: fusão de programas em fita

```
2200 CLS:PCLEAR1:CLEAR20000:DIML
$(1300):E$=CHR$(32):X=0
2210 PRINT"CMERGE":PRINT"POR PAU
LO ADDAIR"
2220 LINEINPUT"PRIMEIRO ARQUIVO?
 ";N1$
2230 LINEINPUT"SEGUNDO  ARQUIVO?
 ";N2$
2240 LINEINPUT"RESULTANTE? ";N3$

2250 OPEN"I",#-1,N1$
2260 GOSUB2420
2270 IFNOTEOF(1)THENIFMEM>255THE
N2260ELSE2450
2280 CLOSE
2290 OPEN"I",#-1,N2$
2300 GOSUB2420
2310 FORI=0TOX-2:N1=VAL(L$(I))
2320 IF NL>N1 THEN2350
2330 IF NL<N1 THENFORJ=X TOI+1ST
EP-1:L$(J)=L$(J-1):NEXTJ:L$(I)=L
$(X):L$(X)="":GOTO2360
2340 L$(I)=L$(X-1):L$(X-1)="":X=
X-1:GOTO2360
2350 NEXTI
2360 IFNOTEOF(1)THENIFMEM>255THE
N2300ELSE2450
2370 INPUT"PREPARE PARA GRAVAR E
 enter";ZZ$
2380 CLOSE:OPEN"O",#-1,N3$
2390 FORI=0 TO X-1
2400 PRINT#1,L$(I)
2410 NEXTI:CLOSE:CLS:PRINT"ARQUI
VO "N3$" GRAVADO":END
2420 LINEINPUT#-1,L$(X):PRINTL$(
X)
2430 NL=VAL(L$(X)):IF NL=0THEN24
20
2440 X=X+1:RETURN
2450 PRINT"MEMORIA INSUFICIENTE"
```

```
LINHA:  2290        141
LINHA:  2390        144
FIM:    97
```

Listagem 10-2 — CSORT: ordenação de arquivo em fita

```
2200 CLS:PCLEAR1:CLEAR20000:DIML
$(1300):E$=CHR$(32):X=0
2210 PRINT"CSORT":PRINT"POR PAUL
O ADDAIR"
2220 LINEINPUT"QUAL  ARQUIVO? ";
N1$
2230 L$(0)=CHR$(255)
2240 LINEINPUT"RESULTANTE? ";N3$

2250 OPEN"I",#-1,N1$
2260 GOSUB2420
2310 FORI=0TOX-2
2320 IF L$(X-1)>=L$(I) THEN2350
2330 FORJ=X TOI+1STEP-1:L$(J)=L$
(J-1):NEXTJ:L$(I)=L$(X):L$(X)=""
:GOTO2360
2350 NEXTI
2360 IFNOTEOF(1)THENIFMEM>255THE
N2260ELSE2450
2370 INPUT"PREPARE PARA GRAVAR E
 enter";ZZ$
2380 CLOSE:OPEN"O",#-1,N3$
2390 FORI=0 TO X-1
2400 PRINT#1,L$(I)
2410 NEXTI:CLOSE:CLS:PRINT"ARQUI
VO "N3$" GRAVADO":END
2420 LINEINPUT#-1,L$(X):PRINTL$(
X)
2440 X=X+1:RETURN
2450 PRINT"MEMORIA INSUFICIENTE"
```

```
LINHA:  2330        139
LINHA:  2450        231
FIM:    231
```

11 CLOAD MELHORADO

Aqui está uma pergunta que todo feliz proprietário de um CP 400 com 64k de RAM sempre faz ao quase sempre despreparado vendedor que o atendeu: "Quer dizer que esse modelo dispõe de quatro vezes mais memória que a versão de 16k?" E a invariável resposta: "Não... na realidade usando o BASIC o senhor só vai ter acesso a 32k, o restante só com unidade de disco..." Peraí! Pra que comprar mais memória se eu não posso usar? Pode tratar de tirar metade da RAM e me fazer um desconto...

Não é bem assim. Você não precisa comprar uma unidade de disco para usar todos os quilobytes do seu micro. Você só precisa saber como está distribuída a memória do CP 400 e como fazer para utilizá-la.

O CP 400 Color é controlado por um microprocessador de 8 bits, o 6809E, que é capaz de acessar diretamente 64k de memória. Nesses 64k deve-se incluir o interpretador BASIC, que está gravado em ROM (nos primeiros modelos era EPROM), ocupando 16k, e o espaço reservado para o cartucho, que são outros 16k. Assim, tirando-se 16k de ROM mais 16k de cartucho, ficamos com apenas 32k para nossos programas e dados em BASIC.

Isso significa que uma parte dos 64k de RAM deve ficar "bloqueada" durante a utilização desses recursos. Quando usamos uma unidade de disco, toda a RAM é liberada, copiando-se o interpretador em uma parte dela e acrescentando-se novos recursos da própria unidade de disco. Por isso não podemos usar cartuchos quando usamos o disco.

Mas essa liberação da RAM e a cópia do interpretador podem ser realizadas sem a unidade de disco. O programa da Listagem 11-6 realiza essas duas funções. Tente executá-lo. Para indicar que a operação foi bem sucedida, a linha 2610 muda a mensagem "OK" do interpretador para caracteres em negativo.

Você já deve ter sacado a importância desse macete. Claro! Se tudo passar para a RAM, então você vai poder alterar o próprio interpretador BASIC, melhorando suas rotinas ou adaptando-o às suas necessidades, ficando assim com um micro "personalizado".

Só tem um problema: é preciso conhecer muito bem a arquitetura do micro e, de quebra, dominar a linguagem de máquina. Por isso é bom começar com calma, pegando um comando simples, só pra tomar contato com a fera.

E, sem dúvida, um comando simples mas que aborrece bastante o operador é o CLOAD. Quando ele não consegue ler o primeiro arquivo que encontra, imprime "I/O Erro" e desiste. O caso mais comum que provoca isso é quando

você põe uma fita para tocar a partir do meio de um programa, geralmente o anterior ao que você quer. Isso não deveria ser considerado um erro, mas é e, como resultado, você tem que digitar CLOAD novamente para carregar o arquivo desejado.

Se você não mudar o modo de operação do CLOAD, a solução é tocar a fita para localizar o começo da gravação, retroceder um pouco e preparar para tocar novamente. Isso envolve um processo braçal, e por que não dizer oral, que mais incomoda do que resolve. Portanto, mãos à obra! Vamos tentar descobrir como essa rotina opera!

No manual do CP 400 existe um capítulo inteiro sobre linguagem de máquina que fala alguma coisa sobre as rotinas de manipulação de arquivos em fita. Essas rotinas compreendem a WRTLDR, que ativa o gravador e envia um "cabeçalho" que precederá o arquivo, e várias outras. Se você entendeu tudo o que está lá, então provavelmente nem precisava perder tempo com este livro! Aquele capítulo não diz nada realmente sobre como o arquivo fica gravado na fita ou sobre os processos de gravação. Ele serve apenas como orientação para programadores mais experientes.

Para decifrar esse mistério, vamos analisar a Tabela 11-1, onde está representado um arquivo genérico da forma como é gravado em fita.

Tabela 11-1 — Formato dos arquivos em fita

&H55	Cabeçalho composto por 128 bytes &H55 (85 decimal ou 01010101).
•	Estes bytes formam aquele tom característico no início dos arquivos.
•	
•	
&H55	
&H55	Byte "líder". Mais um byte &H55, só para assegurar que o motor do gravador atingiu a velocidade "de cruzeiro".
&H3C	Byte de sincronismo. Indica o início de um bloco de dados.
&H00	Tipo de bloco. O zero indica título.
&H20	Oito bytes compõem o título do arquivo. Se não for definido nenhum
•	nome, os bytes serão todos &H20 (espaços em branco).
•	
&H20	
&H00	Tipo de arquivo. O 00 indica um arquivo em BASIC; 01 indica um arquivo de dados; e 02, um arquivo em linguagem de máquina.
&HFF	Byte de formato. FF indica informação gravada em binário. 00 indica formato ASCII.
&H00	Byte de continuidade. 00 indica gravação contínua; FF denota lacunas (geralmente, arquivo de dados).

B+S Endereço de partida, no caso de programas em linguagem de máquina
B–S (B+S é o Byte mais Significativo e B–S, o menos Significativo).

B+S Endereço de carga. Define o ponto por onde deve ser iniciado o car-
B–S regamento da memória.

B+S Número total de bytes gravados, para assegurar correção da leitura.
B–S

&H55 Novo cabeçalho (128 bytes).
•
&H55

bloco de dados

&H55 Byte "líder".
&H3C Byte de sincronismo.
&H01 Tipo de bloco. 01 indica dados e FF, final de arquivo.
&HFF Comprimento do bloco. De 00 a FF.
&H00 Seqüência de até 255 bytes que compõem o dado propriamente dito.
•
&H00
B+S Total de verificação, somando os dados, tipo de bloco e comprimento
B–S de bloco.

Blocos adicionais não possuem um cabeçalho (128 bytes &H55).

Bloco final

&H55 Byte de "lacuna".
&H3C Byte de sincronismo.
&HFF Byte de fim de bloco.
&H00 Comprimento do bloco.

Os blocos são escritos pelas rotinas mencionadas no "famoso" Capítulo 19. WRTLDR aciona o motor e grava o cabeçalho. Os demais são gravados por BLKOUT e lidos por BLKIN. Quando o interpretador BASIC reconhece um comando CLOAD, ele passa o controle para uma rotina localizada no endereço &HA498 da ROM. Essa rotina está parcialmente comentada na Listagem 11-1.

A rotina verifica se existe um M após CLOAD, para saber se é um comando CLOADM. Em caso afirmativo, passa o controle para a sub-rotina em &HA4FE, que vai carregar arquivos em linguagem de máquina. Outra sub-rotina importante é a que batizamos de "busca de arquivo" e está comentada na Listagem 11-2. Essa sub-rotina é utilizada tanto por CLOAD como por CLOADM. Há ainda a rotina em &HA681, chamada aqui de "busca de nome", que está comentada na Listagem 11-3.

O segredo para corrigir o comando CLOAD está na verificação do tipo de bloco lido. Durante sua operação, é ativada a rotina em &HA701, que lê um blo-

co e coloca o tipo na posição &H7C. A instrução ORB (&HA698 na Listagem 11-3) verifica se trata-se de um bloco de título e/ou se existem erros. Se não for um título, a rotina devolve o controle do processamento, imprimindo a mensagem "I/O Erro".

Se a fita começar a ser tocada do meio do arquivo, é claro que não será lido o título e, portanto, ocorrerá erro. Mas, se a gente começa a tocar a fita ali, é porque aquele programa não nos interessa, mas sim os próximos. Então a rotina deve simplesmente ignorar qualquer bloco que não seja título e não evocar um erro no primeiro que aparecer.

Bem, já está na hora de deixarmos tanta "teoria" de lado e partirmos para a prática! O programa fonte para aperfeiçoar o CLOAD está na Listagem 11-4. A rotina coloca um desvio na sub-rotina "busca de arquivo" em &HA698, fazendo o controle passar para a nova rotina de manipulação do carregamento. Essa rotina, que começa efetivamente na linha 180 da Listagem 11-4, antes de mais nada "empilha" os valores dos registradores para evitar "gatos no programa" (não é a tradução correta de *bug*, mas é tupiniquim). Em seguida vem o trecho do programa que opera o carregamento da forma que queremos e, finalmente, uma rotina de encerramento, restituindo os valores dos registradores e retornando à rotina do CLOAD.

Se você não dispõe de um Editor/Assembler, como o do Capítulo 18, ou não quer usá-lo, pode introduzir e rodar o programa "CORRIGE", na Listagem 11-5. Esse programa em BASIC faz toda a correção sem precisar de nenhum programa adicional.

Espero que essa dica sirva para ampliar mais um pouco a sua visão de como opera esse incrível micro, que por enquanto é o único capaz de alterar o que vem gravado de fábrica na ROM. Se bem que é claro que a alteração se faz na cópia da ROM na RAM, sem prejuízo do sistema original, que retomará sua programação original assim que se desligar e religar o equipamento.

Listagem 11-1 — Rotina CLOAD

```
A498    CLR  78   LIMPA PARAMETROS DE ARQUIVO
A49A    CMPA #4A  EXISTE UM 'M' DEPOIS DE CLOAD
A49C    REQ  A4FE SE HOUVER, VAI PARA CLOADM
A49E    LEAS S+2  RESTAURA PILHA E RETORNA
A4A0    JSR  A5C5 ARMAZENA ARQUIVO PEDIDO
A4A3    JSR  A648 VAI PARA 'BUSCA DE ARQUIVO'
```

Listagem 11-2 — Rotina de busca de arquivo

```
A648    TST  78   VERIFICA STATUS DE ARQUIVO
A64A    BNE  A61C SE ABERTO, ERRO
A64C    BSR  A681 VAI PARA 'BUSCA DE NOME'
A64E    BNE  A619 VERIFICA SE HA' ERROS
A650    CLR  79   LIMPA CONTADOR
A652    LDX  #1DA POSICAO DO NOME ENCONTRADO
```

Listagem 11-3 — Rotina de busca de nome

```
A681  LDX  #1DA LOCAL ONDE VAI O NOME
A684  STX  7E   EMPILHA O ENDERECO DO NOME
A686  LDA  68   INDICADOR DA INSTRUCAO ATUAL
A688  INCA
A689  BNE  A696 SE NAO SALTAR
A68B  JSR  A928 VAI LIMPAR A TELA
A68E  LDX  88   PEGA POSICAO DO CURSOR
A690  LDB  #53  "S"
A692  STB  ,X++ POE E ABRE NA TELA
A694  STX  88   EMPILHA POSICAO DO CURSOR
A696  BSR  A701 LE CABECALHO E BLOCO
A698  ORB  7C   VERIFICA SE E' BLOCO DE TITULO
A69A  BNE  A6D0 RETORNO E ERRO DE IMPRESSAO
A69C  LDX  #1DA LOCAL DO ARQUIVO ACHADO
A69F  LDU  #1D2 LOCAL DO ARQUIVO PEDIDO
A6A2  LDB  #8   CARREGA CONTADOR COM 8 ESPACOS
A6A4  CLR  ,-S
A6A6  LDA  ,X+  CARREGA COM CARACTERE
A6A8  LDY  68   LOCAL DO INDICADOR ATUAL
A6AB  LEAY 1,Y  ROTINA PARA COMPARAR CARACTERE
A6AD  BNE  A6B4  LIDO COM O DO NOME PEDIDO.
  .
  .
  .
A6C3  BEQ  A6CB SE FOR A LETRA, CONTINUA
A6C5  BSR  A6D1 SENAO, SALTA O RESTO DO ARQUIVO
A6C7  BNE  A6D0 EM CASO DE ERRO, RETORNA (RTS)
A6C9  BRA  A686 SENAO, COMECA O ARQUIVO DE NOVO
A6CB  LDA  #46  "E" PARA ENCONTRADO
A6CD  BSR  A6F8 POE "E" NA TELA
A6CF  CLRA
A6D0  RTS
```

Listagem 11-4 — Rotina de correção do CLOAD

```
00100 PRINC  NOP
00110        LDX  #$A698     POE A ROTINA NO BASIC
00120        LDA  #$7E       DESVIO
00130        STA  ,X+
00140        LEAY INICIO,    INDICA A ROTINA 'INICIO'
00150        STY  ,X
00160        RTS             VOLTA AO BASIC
00170 INICIO PSHS A,B,X,Y,CC    GUARDA TUDO
00180        LDA  $7C        PEGA O TIPO DO BLOCO
```

```
00190        BNE  RELE  SE NAO E TITULO, LE DE NOVO
00200        PULS A,B,X,Y,CC PEGA TUDO DE VOLTA
00210        ORB  $7C   DEVOLVE OS BYTES TROCADOS
00220        LBNE $A6D0
00230        JMP  $A69C
00240 RELE   PULS A,B,X,Y,CC      REPOE A PILHA
00250        JMP  $A696      LE OUTRO BLOCO
00260        END
```

Listagem 11-5 — Programa "CORRIGE" em BASIC

```
2680 '  CLOAD+
2690 ' PAULO ADDAIR - MAI/85
2700 GOSUB2750:EI=B:GOSUB2750:EF
=B
2710 FORI=EI TO EF:GOSUB 2750:PO
KE I,B:NEXT I
2720 END
2730 DATA FD00,FD24,8E,A6,98,86,
7E,A7,80,31,8D,00,04,10,AF,84,39
,34,37
2740 DATA 96,7C,26,0B,35,37,DA,7
C,10,26,6A,C6,7E,A6,9C,35,37,7E,
A6,96
2750 READ A$:B=VAL("&H"+A$)
2760 RETURN
```

```
FIM:    50
```

Listagem 11-6 — Liberação da RAM e cópia da ROM

```
2490 '  ATIVA 64K E COPIA A ROM
2500 FOR I=32382 TO 32407:READ B
:POKE I,B:NEXT I
2510 EXEC 32382
2520 DATA 26,80,142,128,0,166,13
2,183,255,223,167,128,183,255,22
2,140
2530 DATA 255,0,38,241,183,255,2
23,28,175,57
```

```
2540 'DESATIVA A PROTECAO
2550 POKE&H055,&H0D:FOR I=&HA068
 TO &HA071:POKE I,&H12:NEXT I
2560 'VETOR DE RESET
2570 E1=PEEK(114):E2=PEEK(115)
2580 POKE 114,0:POKE115,&HF8
2590 POKE&HF8,&H12:POKE&HF9,&H7F
:POKE&HFA,&HFF:POKE&HFB,&HDF
2600 POKE&HFC,&H7E:POKE&HFD,E1:P
OKE&HFE,E2
2610 PRINT"TODA A RAM ESTA' ATIV
ADA":POKE44014,ASC("o"):POKE4401
5,ASC("k")
```

```
LINHA: 2580      7

FIM:    40
```

12 ORGANIZAÇÃO DOS PROGRAMAS EM FITA

Qual a melhor maneira de organizar os inúmeros programas que desenvolvemos, adaptamos ou simplesmente copiamos? O ideal é gravar em disquetes, mas você já viu o preço? E, além do mais, o disquete só é realmente útil quando temos uma aplicação digna de suas vantagens, que são a grande capacidade de armazenamento de dados (e não especificamente de programas) e a velocidade. É um verdadeiro desperdício comprar um acionador de disco, que chega a custar mais que o micro, apenas para catalogar programas em BASIC! É óbvio que, se você já dispõe de um sistema com disco, o gravador torna-se dispensável, mesmo porque você *tem que usar o dinheiro que gastou*!

Mas apesar de barato, não é muito simples lidar com programas em fita Existe toda uma técnica para lidar com os programas armazenados. Os problemas provocados por esse sistema são tantos que tornam extremamente difícil manter-se uma programateca (alguns prefeririam *softeca*) organizada. Sem contar que o programador, com o tempo, vai perdendo o "pique da coisa", deixando de lado as enfadonhas anotações sobre o que está gravando, em que fita e em que pontos da fita o programa começa e termina.

A primeira coisa que se deve ter em mente é que a fita a ser usada não precisa ser de longa duração. Mesmo os sistemas com 32k de RAM (em BASIC) não precisam mais do que 5 minutos de fita para armazenar programas. Você pode até mesmo "preparar" suas fitas, cortando trechos de cinco a seis minutos de uma fita nova (de preferência, metálica ou óxido de cromo) e rebobinando esses trechos cortados em cassetes usados (aquela coleção de BIG HITS de 1970, por exemplo!). Com uma fita metálica C60, por exemplo, você pode preparar até 10 "fitas especiais" para programas do seu micro que comportam, cada uma, perto de 10 programas médios. Se você decidir por essa solução, use cassetes fixados por parafusos, pois estes são de montagem mais confiável.

A grande vantagem da fita de curta duração é o tempo reduzido de localização do programa e de rebobinamento. Mas ainda falta um recurso muito importante para se manter as fitas organizadas: o índice dos programas gravados. O ideal é ter essa informação gravada logo no começo da fita. E ela deve ser carregada na memória e mostrada rapidamente; caso contrário, não tem vantagem nenhuma sobre o papel e o lápis. O grande problema é que usar a instrução PRINT #-1 envolve uma perda considerável de tempo. Os blocos gravados comportam apenas 254 bytes, o que é pouco para um índice que se preze, e cada

um possui toda uma informação complementar de sincronismo e identificação que, para o nosso propósito, é perfeitamente dispensável.

Então, a solução para gravarmos um índice completo com todas as informações a respeito da fita envolve fazer uma cópia direta do índice devidamente preparado na memória em fita. Isso quer dizer que um trecho da RAM deve ser gravado em fita! E qual a instrução que faz isso? O manual não faz referência direta a esse recurso, mas ele existe.

A instrução CSAVEM se encarrega disso (você se lembra de ter lido alguma coisa sobre essa instrução em algum lugar?). No manual é mencionada apenas a CLOADM (pág. 240), que realiza a operação inversa, o carregamento de uma informação previamente gravada em uma certa área de memória.

Mas, sejamos justos com o manual do CP 400. Devemos dizer que essa instrução também não é mencionada em detalhes no manual do TRS80 Color Computer, aparecendo laconicamente apenas num resumo das funções e instruções do equipamento (pág. 192 do *Going Ahead With Extended Color BASIC*). E, para piorar, estava errado!

Impressionante... Eles erraram... Erraram quando disseram que o primeiro parâmetro da instrução CSAVEM é uma variável numérica. Na realidade, este parâmetro é o nome do arquivo que será criado em fita. Os outros três estavam certos. São valores numéricos (constantes ou variáveis) que definem, respectivamente, os endereços inicial, final e de execução da rotina a ser gravada em fita. Nenhum dos parâmetros é opcional, apesar de o nome do arquivo poder ser uma string nula,

Para o nosso caso, onde usaremos a instrução para gravar dados em fita, como endereço de execução foi escolhido o 350. Isso, porque nesse endereço existe uma instrução de retorno, no caso do arquivo ser executado acidentalmente. Assim podemos usar uma instrução de manipulação de rotinas em linguagem de máquina para manipular blocos de dados, com vantagens sobre o método convencional.

COMO USAR O PROGRAMA

Grave o programa DIRFITA (Listagem 12-1) no ínicio de todas as fitas que você estiver usando. Ele deve ser sempre o primeiro programa na fita, por uma questão de funcionalidade. Rode o programa e crie um índice para cada fita, gravando-o logo após o DIRFITA. Para cada novo arquivo que você criar, rode o programa e acrescente os novos dados no índice, não esquecendo principalmente o nome e os pontos onde começa e termina a gravação. Assim você terá sempre uma maneira prática para manter controle sobre os seus arquivos sem ter que ficar guardando aquelas anotações nem sempre legíveis.

Listagem 12-1 — Diretório em fita

```
10 MX=13
20 POKE 500,MX
30 CLEARMX*28+100
```

```
40 MX=PEEK(500)
50 P=0:ST=0:EN=0
60 DIM NX$(MX)
2800 FOR X=0 TO MX
2810 NX$(X)=STRING$(27,46):NEXT
2820 P=VARPTR(NX$(MX))
2830 ST=PEEK(P+2)*256+PEEK(P+3)
2840 EN=PEEK(39)*256+PEEK(40)
2850 CLS:INPUT"CRIAR UM DIRETORI
O";A$:IF A$="S" THEN2880
2860 CLS:PRINT@232,"CARREGANDO I
NDICE";
2870 LOADM"INDICE"
2880 CLS:FORX=0TOMX
2890 PRINTSTR$(X+1);TAB(4);NX$(X
)
2900 IF (X+1)/14-INT((X+1)/14)=0
 THEN A$=INKEY$:IFA$="" THEN2900
:ELSE IF ASC(A$)=9 THEN2920ELSEI
F X(MX THENCLS
2910 NEXT
2920 INPUT" DIGITE NUMERO DE LIN
HA";X
2930 IFX=0 THEN 2980
2940 F=1
2950 PRINTTAB(3);")";:LINEINPUTN
X$
2960 MID$(NX$(X-1),1)=NX$
2970 CLS:GOTO2880
2980 IFF=0 THEN 3050
2990 CLS:PRINT" REBOBINE A FITA"

3000 PRINTTAB(10);"PREPARE PARA
GRAVAR"
3010 PRINT"PRESSIONE QUALQUER TE
CLA";
3020 A$=INKEY$:IFA$=""THEN3020
3030 PRINT@232,"GRAVANDO INDICE.
..";
3040 SAVEM"INDICE",ST,EN,350
3050 CLS
```

```
LINHA:  2830        161
LINHA:  2930        244
LINHA:  3030        48
FIM:    33
```

IV

Recursos de disco

13 UM DIRETÓRIO MAIS PRÁTICO

Este programa mostrará os arquivos em seu disquete de uma maneira mais racional do que o comando DIR. Ele coloca quatro nomes de arquivo por linha, o que permite mais nome na tela e evita a inconveniência do deslocamento em disquetes muito cheios.

O programa permite ainda ter uma listagem de todos os arquivos com uma determinada extensão (por exemplo, /BAS), para facilitar ao usuário a busca da informação desejada. Além disso, é possível solicitar uma listagem completa dos arquivos presentes no disquete. Essa listagem irá incluir até mesmo os arquivos já cancelados pelo comando KILL e que ainda não tenham sido sobrepostos por novas informações.

Para acionar o programa digite simplesmente RUN "DIR" ENTER. Não há problema em usar o próprio nome do comando original, pois o nosso DIR será um programa gravado no disco com o nome DIR/BAS. Assim que o programa começa a rodar, ele lhe perguntará sobre qual drive utilizar. Você deve digitar o número do drive; se omiti-lo será utilizado o drive em uso no momento do acionamento do programa. Depois disso, introduza a extensão que você quer ver pesquisada. Se preferir uma listagem completa de todos os arquivos, use simplesmente o asterisco (*). Depois de pressionar ENTER, você verá todos os nomes de arquivos no disquete, juntamente com o espaço ainda livre nessa unidade.

Apesar desse programa ter sido desenvolvido para um sistema com dois drives, não há grandes dificuldades para adaptá-lo a sistemas com mais de duas unidades.

Arquivos cancelados podem ser identificados pela ausência do primeiro caractere no nome. Por exemplo, se o arquivo "MIRA" for cancelado por um comando KILL no DOS, quando você pedir uma listagem completa com o asterisco, esse arquivo vai aparecer como "IRA", desde que nada tenha sido gravado sobre ele. Os programas que você não vê com o comando DIR, do sistema operacional, mas aparecem com o programa DIR, são aqueles cancelados e cujo espaço ainda não foi utilizado para nenhuma outra informação.

Agora que você já sabe como operar mais essa ferramenta para incrementar o seu sistema com disco, introduza o programa e grave-o no drive 0 com o nome "DIR/BAS". Assim você terá à mão um indispensável utilitário.

Listagem 13-1 — Um diretório mais prático

```
10 CLEAR1000
3100 CLS:PRINT"** DIRETORIO ** D
RIVE?";
3110 R$=INKEY$:IFR$=""THEN3110
3120 IFVAL(R$)=1THENDRIVE1:D=1:P
RINT" 1"ELSEPRINT" 0":D=0
3130 PRINTSTRING$(32,159);
3140 INPUT"EXTENSAO";Z$:IFZ$=""T
HENZ$="BAS"ELSEIFLEN(Z$)<3THENZ$
=Z$+STRING$(3-LEN(Z$),32)ELSEZ$=
LEFT$(Z$,3)
3150 PRINT"EXTENSAO    /";Z$
3160 FORX=3TO11
3170 DSKI$ D,17,X,A$,B$
3180 C$=A$+LEFT$(B$,127)
3190 FORN=0TO7
3200 N$(N)=MID$(C$,N*32+1,8)
3210 E$(N)=MID$(C$,N*32+9,3)
3220 NEXTN
3230 FORN=0TO7
3240 IFZ$<>"*   "THEN3270
3250 IFLEFT$(N$(N),1)=CHR$(0)THE
N N$(N)=CHR$(207)+RIGHT$(N$(N),7
)
3260 GOSUB3370:GOTO3280
3270 IFE$(N)=Z$ANDLEFT$(N$(N),1)
<>CHR$(0)THENGOSUB3370
3280 Q=PEEK(1517)
3290 IFQ<>96GOSUB3360
3300 NEXTN
3310 NEXTX
3320 POKE65344,0
3330 PRINT:PRINTFREE(D)"BLOCOS L
IVRES NO DRIVE ";D
3340 DRIVE0
3350 END
3360 N=N+1:GOSUB3370:RETURN
3370 PC=PEEK(136)*256+PEEK(137)
3380 IFFL=0THENPRINTN$(N);:FL=1:
GOTO3440
3390 FORK=1TOLEN(N$(N)):CC=ASC(M
ID$(N$(N),K,1))
3400 IFCC=0THENCC=32
```

```
3410 IF CC>31 AND CC<64THEN POKE
 PC,CC ELSE POKE PC,CC-64
3420 FL=0:PC=PC+1:IFPC>1503THENP
RINT@511,"":PRINT@448,;ELSEPRINT
@PC-1024,;
3430 NEXTK
3440 RETURN
```

```
LINHA: 3180        54
LINHA: 3280        143
LINHA: 3380        87
FIM:   4
```

14 RACIONALIZAÇÃO DE DISCOS

Depois de algum tempo utilizando meu CP 400 com unidade de disco, descobri um ponto em comum entre todos os disquetes que eu já tinha utilizado: todos, sem exceção, estavam uma "bagunça"! Isso porque eu nunca sabia, ao começar um determinado trabalho, exatamente quanto espaço de cada disco seria utilizado. Então, o que no começo era disco para processamento de texto acabava virando "coração de mãe"; alguns utilitários, vários programas inacabados, um ou dois arquivos de texto e muitas rotininhas e programas intermediários que depois de melhorados foram esquecidos.

Aí me ocorreu a idéia: por que não fazer um programa para "passar a limpo" os meus discos? O que eu precisava era de um utilitário que pegasse cada arquivo de um dado disco, me perguntasse se devia ou não transferi-lo e, em caso afirmativo, copiasse o dito programa em um novo disquete. Não querendo a cópia, o programa ia em frente e repetia o processo com o próximo arquivo.

O resultado dessa idéia é uma espécie de comando "BACKUP condicional", que não vai duplicar todo o disco original, mas apenas os arquivos que você quer que sejam transferidos. Ao utilizá-lo, a operação ocorrerá automaticamente no caso de o seu sistema contar com dois drives.

Dispondo de apenas uma unidade de disco, o programa pedirá que sejam trocados os discos na medida em que faz as cópias. O problema, nesse caso, é que a troca será feita para cada arquivo copiado e aí você pode até achar que se é pra ter tanto trabalho, trocando os discos, nem precisa do programa. Mas não é bem assim. Lembre-se de que com um drive, além de trocar os disquetes, você vai ter que digitar o comando COPY "nome/ext" para cada arquivo que quiser gravar.

CONSIDERAÇÕES

Existem alguns problemas com relação à duplicação de arquivos via comando COPY que merecem atenção. O primeiro deles envolve o espaço disponível no disquete de destino, onde deve ser gerada a duplicata do arquivo. Se o arquivo for maior que o espaço restante, o programa será interrompido por um erro "DF".

O segundo caso que pode gerar erro é a existência de um arquivo no disquete de destino com o mesmo nome do que será duplicado. Nesse caso, o código de erro será "AE" e o programa "aborta", ou seja, pára de rodar.

Finalmente, o último tipo de problema que pode ocorrer envolve a própria rotina da instrução COPY. Se houver algum problema na leitura do arquivo (setor defeituoso, diretório ilegível etc.), o micro simplesmente "trava". É fácil perceber quando isso ocorre, pois o LED piloto do drive apaga mas não aparece o "OK" na tela. Se isso acontecer, pressione as teclas RESET para retomar o controle do micro.

Para evitar esses problemas, tenha cuidado ao selecionar quais arquivos devem ser duplicados e em que discos deve ser feita a duplicação. De qualquer forma, esses problemas existem mesmo para quem quiser fazer a duplicação "na unha". Boa sorte!

Listagem 14-1 — Backup condicional

```
3500 PMODE0:PCLEAR1:CLEAR5000
3510 DIM A$(18),B$(72)
3520 '    MENU
3530 CLS:PRINT@3,"COMPACTADOR DE
 DISCOS"
3540 PRINT@100,"DISCO FONTE:   ";

3550 A$=INKEY$:IFA$=""THEN3550
3560 S=VAL(A$):IFS>3THEN3550
3570 PRINTS
3580 PRINT@132,"DISCO DESTINO:";

3590 A$=INKEY$:IFA$=""THEN3590
3600 D=VAL(A$):IFD>3THEN3580
3610 PRINTD
3620 PRINT@225,"PRESSIONE enter
PARA COMECAR"
3630 A$=INKEY$:IFA$=""THEN3630
3640 '       CARREGA DIRETORIO
3650 Y=3:FORX=1 TO 17 STEP 2
3660 DSKI$ S,17,Y,A$(X),A$(X+1)
3670 Y=Y+1:NEXT X
3680 '       SEPARA POR ARQUIVO
3690 FOR X=0 TO 17
3700 B$(X*4+1)=MID$(A$(X+1),1,11
)
3710 B$(X*4+2)=MID$(A$(X+1),33,1
1)
3720 B$(X*4+3)=MID$(A$(X+1),65,1
1)
```

```
3730 B$(X*4+4)=MID$(A$(X+1),97,1
1)
3740 NEXTX
3750 '      ORDENA OS ARQUIVOS
3760 FORX=1 TO72
3770 IF LEFT$(B$(X),1)=CHR$(0) T
HEN 3970
3780 IF LEFT$(B$(X),1)=CHR$(255)
 THEN 4000
3790 NA$=LEFT$(B$(X),8)+"/"+MID$
(B$(X),9,3)
3800 PRINT@290,NA$;" COPIAR (S/N
)";
3810 PRINT@316,CHR$(206);:PRINT@
316,CHR$(205);:PRINT@316,;
3820 A$=INKEY$:IF A$="" THEN3810
 ELSE IF A$="S" THEN PRINT"S" EL
SE PRINT"N":GOTO 3970
3830 SOUND80,2
3840 IF S=D THEN 3900
3850 '      COPIA COM 2 DRIVES
3860 A1$=":"+RIGHT$(STR$(S),1)
3870 A2$=":"+RIGHT$(STR$(D),1)
3880 COPY NA$+A1$ TO NA$+A2$
3890 GOTO3970
3900 '      COPIA NO MESMO DRIVE
3910 A1$=":"+RIGHT$(STR$(S),1)
3920 COPY NA$+A1$
3930 SOUND 100,5:CLS
3940 PRINT"INSERIR DISCO FONTE E
"
3950 PRINT"PRESSIONE enter";
3960 LINEINPUTA$
3970 '      PASSA AO PROXIMO ARQ.
3980 SOUND 180,1
3990 NEXTX
4000 '      FIM DO PROGRAMA
4010 PRINT:PRINT"OUTRO DISCO?"
4020 A$=INKEY$:IF A$=""THEN4020
4030 IF A$="S" THEN RUN
```

```
LINHA: 3590        112
LINHA: 3690        67
LINHA: 3790        16
LINHA: 3890        136
LINHA: 3990        103
FIM:   129
```

15 BACKUPS MAIS PRÁTICOS PARA SEUS DISCOS

O que fazer quando a temida frase "I/O Erro" surge na tela, indicando mais um disco pifado, bem na hora de rodar aquela cópia em disco do Zaxxon que você demorou tanto para conseguir? Primeiro, xingue bastante... Quando terminar, tente novamente a operação onde ocorreu o erro e, caso persista, o último recurso é pegar o backup e copiá-lo novamente no disco pifado. Evite usar o backup em substituição do original, pois se ocorrer um novo erro, você não terá mais a que recorrer. O disco de backup deve ser usado apenas para "recondicionar" o disquete com defeito, fazendo-se uma cópia daquele. Não se esqueça de reformatar o disco defeituoso.

Mas... e se eu não tiver um backup?! Aí você terá um problema sério nas mãos. Não há como recuperar a informação perdida, pois o setor fica simplesmente ilegível para a unidade de disco. Certos utilitários, como o do Capítulo 17, oferecem meios de se recondicionar arquivos que contenham setores nestas condições, mas o trabalho é bastante complexo.

Geralmente o que é feito consiste em uma "dissecação" do disco, mostrando o conteúdo que se conseguiu ler no setor defeituoso. Esse setor terá que ser reconstituído manualmente, ou seja, os bytes errados devem ser editados. Isso é relativamente simples se o arquivo for de dados (principalmente no formato ASCII), pois pode-se localizar o defeito em uma listagem anterior. Mas se o arquivo for binário ou de comando, isso é quase impossível. Portanto, tenha sempre backups de seus programas e arquivos de dados.

Os programas e arquivos de parâmetros não alteráveis devem ser mantidos em discos "matrizes" que só serão usados no caso da cópia em uso pifar. Já os arquivos de dados ou programas editados devem ser copiados "sempre" depois de usados. O esquecimento dessa regra pode pôr a perder todo o trabalho realizado entre a última cópia e o momento da falha.

UMA TÉCNICA MAIS PRÁTICA

O grande problema ao recuperar um disquete defeituoso é que o computador vai copiar todos os setores novamente, os bons e os maus. Isso quer dizer

que o computador não copia apenas o setor onde ocorreu o erro, mas todos os setores gravados no disco. Como o disquete é dividido em 35 trilhas concêntricas, numeradas de 0 a 34, cada uma contendo 18 setores, serão copiados 630 setores e só um está com problemas.

E o pior é que, em 95% dos casos, é sempre na mesma trilha: a número 17. O motivo é simples; essa é a trilha do diretório e quase todas as operações com disco acessam, de uma forma ou de outra, as informações aí contidas. Você pode verificar se foi a trilha 17 que deu problema simplesmente introduzindo o comando DIR. Se foi, a mensagem de erro aparecerá.

Partindo daí, é fácil pensar num sistema que agilize os backups que você necessita. Você só precisa copiar a trilha do diretório; assim, além de agilizar a cópia, você economiza em disquetes, pois com um disquete apenas você poderá ter até 34 diretórios gravados, e, conseqüentemente, 34 discos protegidos contra a principal causa de problemas em seu sistema. A trilha restante é o diretório do próprio disco, que chamaremos de "backup mestre".

O programa deste capítulo copia apenas a trilha do diretório e realiza toda a manipulação do backup mestre. Assim que você tiver o programa rodando, surgirá na tela as seis opções, como segue.

Gravar o diretório no B.M. (Backup Mestre) — Com esta opção você pode fazer o backup de um determinado disco. Você deve seguir as instruções dadas na tela, introduzindo o disquete do qual você quer fazer o backup e pressionando ENTER. Feito isto, o programa vai copiar o diretório (o conteúdo da trilha 17) nas matrizes D$ e E$. Troque o disco pelo Backup Mestre e pressione ENTER novamente. O programa verifica então se existe alguma trilha disponível. Se houver, ele pedirá o nome do disquete, que pode ter até doze letras, e grava o diretório no Backup Mestre. Cuidado para não chamar dois diretórios pelo mesmo nome. Isso poderia interferir na operação das demais opções.

Mostrar os nomes dos disquetes com backup no B.M. — Você pode querer saber quais os disquetes que já tem em backup.

Recuperação de diretório — Utilize esta função quando você tiver um "I/O Erro" na trilha 17 de um dado disquete. O programa vai mostrar os nomes dos diretórios gravados. Introduza o número correspondente ao disquete que você quer recuperar. O programa vai carregar na memória os dados referentes à trilha defeituosa e pedir a troca de disquetes. Coloque o disquete danificado e pressione ENTER. Agora você já pode usar novamente o seu disquete.

Cancela backup de determinado disquete — Esta função é parecida com o comando KILL do BASIC Disco. Ela apaga diretórios já não mais necessários no seu B.M., permitindo mais espaço para novos disquetes. Não deve ser usada para cancelar backups de disquetes ainda em uso.

Mostrar um diretório — Você pode com essa opção verificar o conteúdo de um determinado backup no seu B.M. para saber, por exemplo, se é realmente a última versão do referido disquete.

Fim — Devolve o controle ao BASIC.

Em todas as opções você pode retornar ao menu inicial pressionando a barra de espaço.

O programa foi criado para um micro com 64 kB, mas pode rodar em equipamentos de 16 kB também. Para isso, antes de digitar o programa, introduza: PMODE0:POKE25,14:POKE3584,0:NEW ENTER. Isto fará com que o computador libere 6 kB antes destinados para os gráficos de alta resolução. (No caso específico do CP 400, não é necessário esse recurso, pois o equipamento sempre opera com 64 kB ao se instalar uma unidade de disco.)

Listagem 15-1 — Backup Mestre

```
4100 '     BACKUP MESTRE
4105 '   PAULO ADDAIR  --  MAR/85

4110 CLEAR3500:DIM  NI$(64),TI$(
34),EL(34):CLS2:PRINT@64,"":PRIN
T:PRINT
4115 PRINT@65,"DISCO DE BACKUP M
ESTRE (B.M)"
4120 PRINT@195,"<1> GRAVA DIRETO
RIO NO B.M.";
4125 PRINT@227,"<2> MOSTRA BACKU
PS DO B.M. ";
4130 PRINT@259,"<3> RECUPERA UM
DISQUETE   ";
4135 PRINT@291,"<4> CANCELA UM D
IRETORIO   ";
4140 PRINT@323,"<5> MOSTRA UM DI
RETORIO    ";
4145 PRINT@355,"<6> TERMINA O PR
OGRAMA     ";
4150 PRINT@487,"QUAL A SUA ESCOL
HA ";
4155 A$=INKEY$:IFA$=""THEN4155
4160 A=VAL(A$):IFA<1 OR A>6 THEN
 4155
4165 IFA=6THENCLS:END
4170 CLS:IF A>1ANDA<6 THEN PRINT
@224,"INSIRA O BACKUP MESTRE E e
nter"ELSE IF A=1THENPRINT@225,"I
NSIRA O DISCO P/ COPIAR O DIR.":
PRINT@263,"E PRESSIONE enter"
4175 A$=INKEY$:IF A$=" "THEN4110
ELSEIFA$<>CHR$(13)THEN4175ELSEON
 A GOTO 4180,4270,4300,4375,4450
```

```
4180 'GRAVA DIRETORIA
4185 CLEAR4000:T=0:X=0:Y=0:DIMD$
(10),E$(10),F$(35)
4190 FORX=2TO11
4195 DSKI$ 0,17,X,D$(Y),E$(Y):Y=
Y+1
4200 NEXT X
4205 CLS:PRINT@224,"INSIRA O B.M
. E enter"
4210 A$=INKEY$:IF A$=" " THEN 41
10 ELSE IFA$<>CHR$(13) THEN 4210

4215 GOSUB 4570
4220 FORX=1 TO 2:FORY=1 TO17
4225 F$(Y*X)=MID$(B$(X),Z*15+2,1
):F=ASC(F$(Y*X))
4230 T=T+1
4235 IF F=255 THEN 4245
4240 Z=Z+1:NEXTY:Z=0:NEXTX:CLS:P
RINT@235,"DISCO CHEIO":FOR X=1 T
O 650:NEXTX:GOTO4110
4245 CLS:PRINT@234,"NOME DO DISC
O":PRINT@266,"";:INPUTNM$:IFLEN(
NM$)>12THEN4245
4250 IFLEN(NM$)<12THENNM$=NM$+ST
RING$(12-LEN(NM$),32)
4255 MID$(B$(X),Z*15+2,1)="0":MI
D$(B$(X),Z*15+3,12)=NM$
4260 GOSUB 4600:X=0:Y=0:FORX=2 T
O11
4265 DSKO$ 0,T,X,D$(Y),E$(Y):Y=Y
+1:NEXTX:GOTO 4110
4270 '    LISTA BACKUPS
4275 CLS3:PRINT@2,"LISTA DE DIRE
TORIOS COPIADOS";
4280 GOSUB 4625:PRINT@422,"enter
 PARA REVER";:PRINT@454,"espaco
PARA DESISTIR";
4285 A$=INKEY$
4290 A$=INKEY$:IF A$=" "THEN 411
0ELSE IF A$<>CHR$(13) THEN 4290
4295 GOTO4275
4300 '    RECUPERAR UM DIRETORIO
4305 CLS3:PRINT@4,"RECUPERACAO D
E DIRETORIO";
4310 GOSUB4625:PRINT@420,"enter _
```

```
PARA REVER";:PRINT@452,"s PARA S
ELECIONAR ";:PRINT@484,"espaco P
ARA DESISTIR";
4315 A$=INKEY$
4320 A$=INKEY$:IFA$=" "THEN4110E
LSEIFA$=CHR$(13) THEN 4300 ELSE
IF A$<>"S"THEN 4320
4325 PRINT@416,"DIGITE NUMERO DO
 DIRETORIO":PRINT@448,"":PRINT@4
90,"                ";:PRINT@442,"";
:INPUT T:IF T<1 OR T>34THEN 4325

4330 X=0:Y=1:FOR X=2 TO 11
4335 DSKI$ 0,T,X,D$(Y),E$(Y)
4340 Y=Y+1:NEXTX:X=0:Y=0
4345 CLS:PRINT@225,"INSIRA DISCO
 A RECUPERAR E enter";
4350 A$=INKEY$:IFA$=""THEN4350
4355 FORX=2 TO 11:Y=Y+1
4360 DSKO$ 0,17,X,D$(Y),E$(Y)
4365 NEXT X
4370 GOTO 4110
4375 '    CANCELAR DIRETORIO
4380 CLS3:PRINT@8,"CANCELAR DIRE
TORIO";
4385 GOSUB 4625:PRINT@420,"enter
 PARA CONTINUAR";:PRINT@452,"c P
ARA CANCELAR";:PRINT@484,"espaco
 PARA DESISTIR";
4390 A$=INKEY$
4395 A$=INKEY$:IF A$="C" THEN 44
05ELSE IF A$=" "THEN 4110 ELSE I
F A$<>CHR$(13) THEN 4395
4400 GOTO4380
4405 PRINT@416,"":PRINT@448,"":P
RINT@416,"NUMERO DO DIRETORIO";:
INPUTK:IFK<10RK>34THEN4110
4410 PRINT@448,"CONFIRME (sim OU
 nao)";:INPUT A$:IF A$<>"SIM"THE
N4110
4415 GOSUB4570:Z=0:T2$=TI$(K):FO
RY=1 TO 34:IF Y>17 THEN X=2 ELSE
 X=1
4420 IFMID$(B$(X),Z*15+3,12)=T2$
 THEN 4440
4425 Z=Z+1:IF Y=17 GOSUB 4435
```

```
4430 NEXTY:CLS:PRINT@264,"DIRETO
RIO NAO ENCONTRADO":FORX=1TO1000
:NEXT:GOTO4110
4435 Z=0:RETURN
4440 MID$(B$(X),Z*15+2,1)=CHR$(2
55):MID$(B$(X),Z*15+3,12)=STRING
$(12,32)
4445 GOSUB4600:GOTO4110
4450 '     MOSTRAR UM DIRETORIO
4455 CLS3:PRINT@4,"MOSTRAR UM DI
RETORIO";
4460 GOSUB4625:PRINT@420,"enter
PARA REVER";:PRINT@452,"s PARA S
ELECIONAR";:PRINT@484,"espaco PA
RA DESISTIR";
4465 A$=INKEY$
4470 A$=INKEY$:IFA$=""THEN4470 E
LSE IF A$=" " THEN 4110 ELSE IF
A$=CHR$(13)THEN4450
4475 PRINT@448,"":PRINT@416,"DIG
ITE O NUMERO DO DIRETORIO";:INPU
TV:PRINT@448,""
4480 IFV<1ORV>34THEN4110
4485 FORX=3TO11
4490 DSKI$ 0,V,X,D$,E$
4495 F$=D$+LEFT$(E$,127)
4500 FORZ=0 TO7:NM$=MID$(F$,Z*32
+1,8):EXT$=MID$(F$,Z*32+9,3)
4505 IFASC(NM$)=0THEN4515ELSEIFA
SC(NM$)=255THEN4520
4510 Q=Q+1:NI$(Q)=NM$+"/"+EXT$
4515 NEXTZ
4520 NEXTX
4525 R=1
4530 CLS3:PRINT@5,"DIRETORIO NUM
ERO";V;:Y=64
4535 FORX=R TOQ
4540 PRINT@Y,NI$(X);
4545 IF X/24=INT(X/24) THEN 4555

4550 Y=Y+16:NEXTX
4555 PRINT@484,"enter PARA CONTI
NUAR";
4560 A$=INKEY$:IFA$<>CHR$(13)THE
N4560
4565 IFX>Q THEN 4110 ELSE R=X+1:
```

```
GOTO4530
4570 '      PEGA TRILHA
4575 DSKI$ 0,0,1,A$(1),A$(2):DSK
I$ 0,0,2,A$(3),A$(4)
4580 A$(2)=LEFT$(A$(2),127):A$(4
)=LEFT$(A$(4),127)
4585 B$(1)=A$(1)+A$(2):B$(2)=A$(
3)+A$(4)
4590 A$(1)="":A$(2)="":A$(3)="":
A$(4)=""
4595 RETURN
4600 '      GRAVA TRILHAS
4605 A$(1)=LEFT$(B$(1);128):A$(2
)=RIGHT$(B$(1),127)+"0":A$(3)=LE
FT$(B$(2),128):A$(4)=RIGHT$(B$(2
),127)+"0"
4610 B$(1)="":B$(2)=""
4615 DSKO$ 0,0,1,A$(1),A$(2):DSK
O$ 0,0,2,A$(3),A$(4)
4620 RETURN
4625 '  MOSTRA O DIRETORIO
4630 GOSUB 4570
4635 Z=0:FORY=1 TO 34
4640 IF Y<18 THEN X=1 ELSE X=2
4645 TI$(Y)=MID$(B$(X),Z*15+3,12
):EL(Y)=ASC(MID$(B$(X),Z*15+2,1)
)
4650 Z=Z+1:IFY=17THENZ=0
4655 NEXT Y
4660 FORX1=0 TO1:FORX=1 TO 12:XO
=0:GOSUB4680:XO=12:GOSUB4680
4665 NEXTX:IFX1=0THENEXEC44539:P
RINT@64,STRING$(255,175);:PRINT@
319,STRING$(190,175);
4670 NEXTX1
4675 RETURN
4680 'IMPRESSAO NO VIDEO
4685 IF XO+24*X1+X>34THEN4705
4690 PO=32+32*X:IF XO THENPO=PO+
16
4695 IF EL(24*X1+X+XO)=255 THEN
TI$(24*X1+X+XO)=STRING$(12,46)
4700 PRINT@PO,"";:PRINTUSING"##
%              %";24*X1+X+XO;TI$(24*
X1+X+XO);
4705 RETURN
```

```
LINHA:  4145        101
LINHA:  4195        143
LINHA:  4245        81
LINHA:  4295        59
LINHA:  4345        192
LINHA:  4395        173
LINHA:  4445        151
LINHA:  4495        67
LINHA:  4545        231
LINHA:  4595        59
LINHA:  4645        36
LINHA:  4695        68
FIM:    224
```

16 MUDANDO ENDEREÇOS

Uma das características mais notáveis do microprocessador usado no CP 400, o 6809E, é a conveniência de se escrever programas em linguagem de máquina independentes de sua posição física na memória. Isso é especialmente útil quando utilizamos uma mesma rotina em L.M. em vários programas diferentes. Não precisamos nos preocupar em recalcular os endereços referidos pela rotina.

No COLOR BASIC existe ainda um meio simples de deslocar (o termo correto é "relocar") uma rotina dentro da memória. Pode ser colocado um valor após o nome de arquivo na instrução LOADM que corresponde ao deslocamento que se pretende impor à rotina dentro da memória. Assim, podemos, já no próprio carregamento da rotina, estabelecer um endereço mais alto para seu armazenamento, colocando-a, se quisermos, até mesmo no topo da memória.

Infelizmente, não podemos colocar um valor de deslocamento (chamado *offset* pelos americanos e pelos técnicos tupiniquins que não sabem traduzir do inglês) negativo nessa instrução. Isso quer dizer que não se pode, através da instrução LOADM, trazer uma rotina que estava no topo da RAM para o início.

Mas essa aparente deficiência estimulou muitos programadores a idealizarem programas em BASIC e utilitários em L.M. capazes de realizar essa tarefa. A maioria deles optou por utilizar malhas FOR/NEXT com farta distribuição de PEEKs e POKEs e alguns cálculos "intrínsecos" para mover a rotina de um ponto para outro da memória. Simples, não?

Não! Uma malha FOR/NEXT simplifica tanto as coisas para o programador que a maioria deles acaba se "viciando" nessas facilidades. Toda vez que se fala de alguma coisa repetitiva, ou que "aparenta" ser repetitiva, enfia-se logo um FOR, coloca-se as instruções e termina-se com o invariável NEXT. Não tem erro. Se, no nosso caso, tiverem que ser transferidos 4 mil bytes por exemplo, a malha será repetida para cada um deles. Não deixa de ser uma solução, mas um pouco de conhecimento sobre como uma rotina é armazenada pode evitar toda essa "parafernália algorítmica".

Eu tenho sempre em mente que o bom programa não é aquele que "faz o que deve ser feito", mas sim aquele que "utiliza os recursos do micro para fazer o que deve ser feito". Vamos tentar explicar isso mostrando a maneira mais racional de relocar uma rotina em L.M. para qualquer ponto da memória.

A TÉCNICA

Toda rotina em L.M. armazenada em disco é precedida por três números que são a chave para o nosso problema. Esses números definem o comprimento da rotina, o endereço de carregamento e o seu endereço de execução. O que nós precisamos é pegar essa informação, calcular os novos endereços e gravá-los de volta no arquivo. Dessa forma, quando for solicitado o carregamento da rotina por um LOADM, o próprio CP 400 vai se encarregar de colocá-la na posição correta. Assim, mexemos em apenas seis bytes (cada valor é representado por dois bytes) e o micro faz o resto.

O programa da Listagem 16-1 realiza essa operação de uma forma bem simples. Ele abre o arquivo que contém a rotina em L.M. para acesso direto, com comprimento de registro igual a 1. Isso é conveniente pois com a função LOF (último registro no arquivo) consegue-se exatamente o comprimento do arquivo. Note que mesmo o arquivo gravado em disco por um comando SAVEM, exclusivo para linguagem de máquina, pode ser manipulado como arquivo de acesso direto (vulgo randômico) por um programa em BASIC.

Qualquer rotina que tenha sido gravada (SAVEM) em disco pode ser relocada por esse programa. Porém, alguns programas comerciais podem gerar problemas por não usar as técnicas convencionais de gravação.

Listagem 16-1 — Mudando de endereço

```
4800 CLS:CLEAR 1000
4810 PRINT"NOME ARQUIVO:";:LINEI
NPUT FI$
4820 IFFI$=""THENEND
4830 OPEN"D",#1,FI$,1:IFLOF(1)=0
THENCLOSE:KILLFI$:RUN
4840 FIELD#1,1ASF$:LF=LOF(1)
4850 FORQ=1 TO5:GET#1,Q:BY(Q)=AS
C(F$):NEXTQ
4860 B=0:FORQ=LF-4 TO LF:B=B+1
4870 GET#1,Q:BE(B)=ASC(F$):NEXTQ

4880 LD$=HEX$(BY(4)*256+BY(5))
4890 E$=HEX$(BE(4)*256+BE(5))
4900 LN$=HEX$(BY(2)*256+BY(3))
4910 PRINT
4920 PRINT"ENDERECO DE CARGA = "
;LD$
4930 PRINT"ENDERECO FINAL    = "
;HEX$(VAL("&H"+LD$)+VAL("&H"+LN$
))
4940 PRINT"ENDERECO DE PART. = "
```

```
;E$
4950 PRINT"COMPRIMENTO          = "
;LN$
4960 PRINT
4970 PRINT"NOVO END. CARGA    = "
;:LINEINPUTNL$
4980 IF NL$="" THEN CLOSE:RUN
4990 OF=VAL("&H"+E$)-VAL("&H"+LD
$)
5000 NL=VAL("&H"+NL$)
5010 NE=NL+OF
5020 BY(4)=INT(NL/256)
5030 BY(5)=NL-256*BY(4)
5040 BE(4)=INT(NE/256)
5050 BE(5)=NE-256*BE(4)
5060 PRINT"NOVO ENDER. FINAL  =
";HEX$(NL+VAL("&H"+LN$))
5070 PRINT"NOVO ENDER. PART.
= ";HEX$(BE(4)*256+BE(5))
5080 FORQ=1 TO5:LSET F$=CHR$(BY(
Q)):PUT#1,Q:NEXTQ
5090 B=0:FORQ=LF-4 TO LF:B=B+1
5100 LSET F$=CHR$(BE(B)):PUT#1,Q
:NEXTQ
5110 CLOSE
5120 PRINT:PRINT"PRESSIONE enter
 PARA RECOMECAR";:LINEINPUTX$
5130 RUN
```

```
LINHA: 4890        78
LINHA: 4990        148
LINHA: 5090        169
FIM:   187
```

17 UM PROGRAMA PARA CORRIGIR ERROS NO DISCO

Ninguém pode explorar os recursos do disco sem ter a principal ferramenta para isso: um programa para consertar o disco caso algo saia errado. É claro! Já que quase todo conhecimento que adquirimos está baseado no velho método da tentativa e erro, devemos estar preparados para poder amenizar pelo menos parte das conseqüências.

Todos os modelos de computadores possuem no mínimo um utilitário capaz de "dissecar" o disquete, permitindo que se leia qualquer coisa gravada nele e se altere o que for necessário. Sem dúvida, o utilitário desse tipo que ficou mais conhecido é o SUPERZAP, que roda no CP 500 com NEWDOS e permite a edição de qualquer setor do disquete, incluindo o próprio sistema operacional.

O PROGRAMA

O ZAP400, mostrado na Listagem 17-1, opera de maneira muito parecida com o SUPERZAP. Ele pede inicialmente os números da trilha e setor por onde se quer começar a edição. Uma vez introduzidos esses parâmetros, ele lê a informação diretamente no disquete, usando a instrução DSKI$, e a coloca na tela em formato ASCII.

Você pode então analisar o conteúdo do disquete paginando com as teclas de mais e de menos (+ /; e = / −). A paginação permite que se passe para o próximo setor ou para o anterior sem alterar o conteúdo do que está na tela. Ao localizar o setor onde quer fazer as alterações, pressione M para modificar.

No modo de modificação, um cursor aparece sobre o conteúdo do setor na tela, mostrando qual byte será modificado. O valor hexadecimal do byte onde está o cursor é mostrado no alto da tela, à direita do número do setor. As quatro setas servem para movimentar o cursor dentro do setor. Usando-se SHIFT e as setas para esquerda ou direita, o cursor se move dois bytes, agilizando a movimentação. A barra de espaço aciona a modificação do byte indicado e o programa solicita a introdução do valor do novo conteúdo.

Depois de alterado um valor pode-se continuar editando o setor sem prejuízo para o processo. Quando a edição estiver completa, basta pressionar G pa-

ra gravar o novo conteúdo no disco. Se houver algum problema durante a edição, a tecla F finaliza o processo sem que as alterações na tela sejam passadas para o disco.

, A Tabela 17-1 mostra um resumo dos comandos possíveis com o ZAP400. Espero que esse programa não seja muito solicitado para correção de erros no disquete provocados por programas problemáticos, mas sim para outras finalidades mais interessantes. Mas isso você deve descobrir com o tempo, sem dúvida através de algumas tentativas e vários erros. Boa sorte!

Tabela 17-1 — Resumo dos comandos do ZAP400

Modo de paginação

Tecla	Função
+	Passa para o próximo setor
–	Passa para o setor anterior
CLEAR	Seleciona um novo setor e nova trilha
M	Ativa modificação

Modo de modificação

Tecla	Função
←	Move um byte à esquerda
→	Move um byte à direita
SHIFT ←	Move dois bytes à esquerda
SHIFT →	Move dois bytes à direita
↑	Move um byte para cima
↓	Move um byte para baixo
barra de espaço	Ativa alteração do byte sob o cursor
F	Finaliza o modo de modificação
G	Grava o novo conteúdo do setor no disco

Listagem 17-1 — ZAP400: um programa para corrigir erros no disco

```
5200 CLS2:PRINTTAB(3)"INSERIR O
DISCO NO DRIVE O"
5205 PRINTTAB(4);"PRESSIONE QUAL
QUER TECLA
5210 A$=INKEY$:IFA$=""THEN5210
5215 CLS2:CLEAR1000
5220 PRINT@8,"EDITOR DE TRILHAS"
;
5225 PRINT@33,"TRILHA ";:PC=40:G
OSUB5555:T=VAL(R$)
5230 IFT<OORT>35THEN5225
5235 PRINT@49,"SETOR ";:PC=55:GO
SUB5555:S=VAL(R$)
5240 IFS<1 OR S>18THEN5235
5245 DSKI$ O,T,S,A$,B$
```

```
5250 CLS2:PRINT@8,"  AGUARDE  ";

5255 FORA=1152 TO 1279
5260 POKE A,ASC(MID$(A$,A-1151,1
))
5265 NEXTA
5270 FORA=1280 TO 1407
5275 POKE A,ASC(MID$(B$,A-1279,1
))
5280 NEXTA
5285 PRINT@8,"EDITOR DE TRILHAS"
;
5290 P=0:PRINT@64,"TRILHA";T:PRI
NT@77,"SETOR";S:PRINT@88,"VALOR
";HEX$(P);
5295 PRINT@96,STRING$(32,"-");
5300 PRINT@416,"PRESSIONE m PARA
 MUDAR";
5305 PRINT@448,"+ OU - PARA PAGI
NAR   ";
5310 PRINT@384,STRING$(32,"-");
5315 T$=INKEY$:IFT$=""THEN5315
5320 IFT$="+"ORT$="-"THENGOSUB53
40:GOTO5245 ELSEIFT$="-"ORT$="="
THENGOSUB5350:GOTO5245
5325 IFT$="M" THEN GOSUB 5360
5330 IFT$=CHR$(12)THEN5215
5335 GOTO5285
5340 S=S+1:IFS>18THENT=T+1:S=1:I
FT>34THENT=0
5345 RETURN
5350 S=S-1:IFS<1THENT=T-1:S=18:I
FT<0THENT=34
5355 RETURN
5360 X=1152

5365 P=PEEK(X):PRINT@94,HEX$(P)
5370 FORA=1 TO25
5375 POKEX,207
5380 T$=INKEY$:IFT$<>""THEN5400E
LSE NEXTA
5385 FORA=1 TO25
5390 POKEX,P
5395 T$=INKEY$:IFT$<>""THEN5405E
LSENEXTA
5400 POKEX,P
5405 IFT$=CHR$(8)THENX=X-1 ELSE
```

```
IFT$=CHR$(9)THENX=X+1
5410 IFT$=CHR$(93)THENX=X+2 ELSE
 IFT$=CHR$(21) THENX=X-2
5415 IFT$=CHR$(94)THENX=X-32
5420 IFT$=CHR$(10)THENX=X+32
5425 IFX<1152 THEN X=X+32 ELSE I
FX>11407 THEN X=X-32
5430 IFT$="F" THEN RETURN
5435 IFT$=" " THEN GOSUB5465
5440 IFT$="G" THEN GOSUB5490
5445 IFX<1152 THENX=1152 ELSEIFX
>1407 THEN X=X-32
5450 PRINT@416,"espaco PARA MODI
FICAR              ";
5455 PRINT@448,"f PARA FINALIZAR
  g PARA GRAVAR";
5460 GOTO 5365
5465 PRINT@448,STRING$(32,32);:P
RINT@448,"MODIFICAR ("HEX$(P)")
PARA ";
5470 LINEINPUTP$:P=VAL("&H"+P$)
5475 IF P<0 OR P>255 THEN 5465
5480 POKE X,P
5485 RETURN
5490 PRINT@448,"GRAVANDO EM DISC
O..                ";
5495 A$="":B$=""
5500 FORA=1152 TO 1279
5505 A$=A$+CHR$(PEEK(A))
5510 NEXTA
5515 FORA=1280 TO 1407
5520 B$=B$+CHR$(PEEK(A))
5525 NEXTA
5530 DSKO$ 0,T,S,A$,B$
5535 PRINT@448,"PRESSIONE QUALQU
ER TECLA";
5540 A$="":B$=""
5545 R$=INKEY$:IFR$=""THEN5545
5550 RETURN
5555 '     INTRODUCAO
5560 R$=""
5565 R1$=INKEY$:PRINT@PC,R$;:IFL
EN(R$)<3ANDR1$>="0"ANDR1$<="9"TH
ENR$=R$+R1$:GOTO5565:ELSEIFR1$<>
CHR$(13)THEN5565
5570 RETURN
```

```
LINHA: 5245        26
LINHA: 5295        55
LINHA: 5345        233
LINHA: 5395        0
LINHA: 5445        127
LINHA: 5495        169
LINHA: 5545        161
FIM:   31
```

V

Recursos avançados

18 COLOR ASSEMBLER: MONTANDO ROTINAS EM LINGUAGEM DE MÁQUINA

Você deve ter notado que boa parte dos recursos explorados neste livro envolvem, de uma forma ou de outra, pequenas rotinas em linguagem de máquina (L.M.). A grande vantagem de se utilizar essas rotinas é essencialmente a velocidade. Mas também podemos ressaltar a economia de memória e a flexibilidade que se consegue com essas rotinas. E isso tudo nos leva a uma pergunta importante: se a L.M. é tão vantajosa, por que usar o BASIC?

Talvez a única vantagem que o BASIC leva sobre a L.M. esteja na facilidade de programação. Pela sua estrutura sintática (o modo como as coisas são escritas) e pelas palavras-chave que o compõem, o BASIC é considerado uma linguagem de alto nível, muito próxima da linguagem humana, mais especificamente do inglês. É claro que nenhum americano fala '10 PRINT "HELLO" :INPUT "HOW DO YOU DO";A$ quando quer cumprimentar alguém, mas *você* sim, quando dá ordens ao micro, o faz em inglês. Naturalmente, o número de linha, os dois pontos, o ponto e vírgula e o nome de variável A$ tiram muito da beleza do idioma de Shakespeare, mas ainda assim a linguagem continua de alto nível...

Essa semelhança com um idioma comum é que torna tão atraente programar em BASIC. No entanto, o programador mais experiente tende sempre a usar ao máximo os recursos do microprocessador. E quanto mais se explora o "chip", mais a linguagem de máquina se faz necessária.

Para atender aqueles que acham que já estão em condições de baixar o nível de seus programas (no bom sentido, é claro!) ou que podem falar diretamente com o micro de homem para máquina (sem "intérpretes"), resolvi colocar neste livro um programa em BASIC (alto nível) que permite a programação em L.M. (baixo nível?!). Assim surgiu o Editor/Assembler em BASIC, da Listagem 18-1.

O PROGRAMA

O 6809E é um dos mais poderosos microprocessadores de 8 bits do mercado e ainda o mais simples de programar. Sua estrutura interna, com duas pilhas, registradores de índice com 16 bits e um conjunto de instruções muito bem

escolhido, supre boa parte das deficiências dos processadores tradicionais, como o 6502 e o Z80.

O programa na Listagem 18-1 permite acesso a todos os recursos de programação do 6809E, incluindo todos os modos de endereçamento, o conjunto completo de instruções e até algumas pseudo-instruções. Essas pseudo-instruções permitem algumas funções adicionais durante a programação, agilizando a linguagem original.

O programa roda sem problemas em micros de 64k. Para adaptá-lo para 16k, basta mudar o argumento da instrução PCLEAR na linha 5800 para 1. As três páginas reservadas originalmente servem como um espaço para se colocar as rotinas que você for executar. Mas uma página já representa 1,5 quilobyte de memória e dificilmente uma rotina em L.M. ocupa tanto espaço.

CONJUNTO DE INSTRUÇÕES

O conjunto de instruções padrão do 6809E é composto por 139 mnemônicos diferentes. Mnemônico é um tipo especial de abreviatura, que faz lembrar a palavra ou expressão original pelas suas partes mais fáceis de reconhecer. Um exemplo prático é chamar São Paulo de "Sampa". Mas, voltando ao nosso micro, quase todos esses mnemônicos utilizam várias formas de endereçamento, o que multiplica o número de instruções do micro. Para ter uma relação desses mnemônicos, rode o programa (RUN ENTER) e, quando aparecer o cursor na tela à direita do número da linha, pressione BREAK. Com o programa interrompido, digite:

```
FOR I=1 TO 139:PRINT LEFT$(MN$(I),4);STRING$(4,32);:NEXT ENTER
```

Todos os mnemônicos seguem a nomenclatura da própria Motorola, criadora do 6809, com exceção do ANDC, que equivale ao ANDCC na documentação original. Por falar em documentação original, o Capítulo 22 relaciona e descreve todos os mnemônicos do microprocessador. Explica também os vários modos de endereçamento e sua arquitetura.

ARGUMENTOS

Os argumentos modificam significativamente o código objeto correspondente ao mnemônico utilizado. A Tabela 18-1 relaciona todos os formatos possíveis de argumentos utilizados pelo 6809E. O programa se baseia principalmente no comprimento do argumento utilizado, por isso não se deve acrescentar espaços ao escrever um programa.

Nos casos em que é permitido o endereçamento indireto, o argumento pode ser escrito entre colchetes. Para acionar os colchetes no teclado, use SHIFT ↓ para abrir e SHIFT → para fechar.

Os argumentos válidos para as instruções de troca e transferência são todos de três caracteres de comprimento, no formato M,N, onde M e N são ambos registradores de 8 bits (A, B, C ou Z) ou de 16 bits (D, X, Y, U, S, P). O registrador C é o de código de condição (CC) e o Z é o de página direta (DP).

Para instruções de manipulação de pilha (*push* ou *pull*), pode-se usar como argumento um único registrador ou uma lista de registradores separados por vírgulas. Nesse caso, o registrador D não pode ser usado, pois trata-se de um pseudo-registrador. Se precisar, use "A,B" no lugar de "D".

No lugar dos registradores de pilha U ou S, você pode colocar uma seta para cima. Assim, no lugar de PULS D,U pode-se ter um PULS A,B,↑. A ordem na qual os registradores aparecem na instrução não importa, mas note que nos casos de listas de registradores, o comprimento do argumento sempre é ímpar.

Tabela 18-1 — Formatos de argumentos válidos

formato	compr.	descrição	uso indireto
#nn	3	byte imediato	não
#nnnn	5	palavra imediata	não
nn	2	página direta	não
nnnn	4	por extenso*	sim
+nn,I	5	deslocamento em 5 bits	não
−nn,I	5	deslocamento em 5 bits	não
,I+	3	pós-incremento	sim
,I++	4	pós-incremento de 2	sim
,−I	3	pós-decremento	sim
,−−I	4	pós-decremento de 2	sim
,I	2	índice simples	sim
A,I	3	deslocamento do acumulador	sim
B,I	3	deslocamento do acumulador	sim
D,I	3	deslocamento do acumulador	sim
nn,I	4	deslocamento de 8 bits	sim
nnnn,I	6	deslocamento de 16 bits	sim
nn,PCR	8	deslocamento de 8 bits	sim
nnnn,PCR	8	deslocamento de 16 bits	sim

Legendas

nn	valor hexadecimal de dois dígitos (um byte)
nnnn	valor hexadecimal de quatro dígitos (dois bytes)
I	representa um dos registradores de 16 bits (X, Y, U ou S)
PCR	registrador contador de programa
A,B	acumulador de 8 bits
D	acumulador de 16 bits (A e B juntos)

Obs.: Eu, pessoalmente, prefiro usar o termo em português "por extenso" quando me refiro ao modo de endereçamento "extended". Muitos programadores e técnicos da área preferem o termo "extendido".

RÓTULOS

Qualquer linha do programa pode ser identificada por um rótulo (conhecido por *label*) opcional. Isso permite que se associe nomes a endereços, valores ou rotinas muito usados. Esses rótulos tornam mais legível o programa fonte, facilitando a estruturação das rotinas e agilizando a programação. Eles devem sempre ter quatro caracteres de comprimento, sendo que o primeiro caractere pode ser qualquer letra de G a Z e os demais, letras ou números. São usados em substituição a constantes hexadecimais de quatro dígitos nos argumentos das instruções em geral, com exceção da pseudo-instrução DATA.

A MEMÓRIA

Toda a memória necessária é definida na linha 5800. Se o seu programa fonte precisar mais do que 50 linhas, você deve modificar o valor de S naquela linha e talvez reservar mais espaço para strings (instrução CLEAR). O argumento de PCLEAR, como já foi comentado, pode ser diminuído para 1, aumentando em 3 quilobytes a memória disponível, permitindo inclusive rodar o programa em micros com 16k. Outras formas de aumentar a memória disponível é renumerar as linhas com um RENUM1,1 ENTER, ou tirar todos os comentários da Listagem 18-1. Em último caso, você pode tirar algumas rotinas, como a tela de auxílio com os comandos possíveis ou a listagem dos *opcodes*.

PSEUDO-INSTRUÇÕES

O nosso Editor/Assembler possui quatro pseudo-instruções: END, EQU, ORG e DATA. Elas devem ser colocadas no campo de mnemônicos e todas requerem argumentos. END indica o final do programa e seu argumento é o endereço de execução deste, após compilado e carregado na memória. Esse endereço pode ser tanto uma constante hexadecimal quanto um rótulo, e não precisa corresponder ao início físico do programa. Quando você introduz a linha que contém essa pseudo-instrução, o Editor automaticamente sai do modo de introdução, passando para o de comando.

EQU é usada para dar nomes a constantes hexadecimais de dois bytes e endereços (geralmente, externos à rotina), que serão empregados com certa freqüência durante o programa. Assim, por exemplo, você não precisa tentar lembrar toda hora qual o endereço da rotina "Busca de nome" (mencionada no Capítulo 11). Basta definir NOME EQU A681 e toda vez que precisar da rotina é só usar NOME no lugar do endereço. Como é fácil de perceber, todas as pseudo-instruções EQU devem aparecer logo no começo do programa.

ORG acerta o contador de endereço, que é responsável pelo posicionamento da rotina na memória. Normalmente aparece apenas uma vez no programa, lo-

go após as EQUs. Seu argumento é um endereço hexadecimal de dois ou quatro dígitos ou um rótulo previamente definido.

A quarta pseudo-instrução, DATA, representa um modo prático de incluir strings (seqüências de caracteres) ou valores hexadecimais no programa. O seu argumento pode ser uma mensagem entre apóstrofos (DATA 'CP 400') ou uma seqüência de valores hexadecimais (DATA 41424344). Deve-se ter cuidado ao utilizar esse recurso para que os dados não fiquem em um endereço executado pelo restante da rotina.

O EDITOR

Ao rodar o programa da Listagem 18-1, a primeira coisa que aparece é uma "planilha" para introdução das linhas do programa fonte. Essa planilha é composta por uma coluna com o número da linha que se está introduzindo, outra para os rótulos (chamados "Labels" pelos programadores em Assembly), uma para os mnemônicos e a última para os argumentos. Um cursor indica em qual campo será introduzido sua próxima digitação. Se o campo onde está o cursor deve ficar em branco, simplesmente digite ENTER. A introdução do mnemônico END finaliza a introdução do programa fonte, passando o controle para o modo de comando. Nessa condição, as seguintes teclas podem ser acionadas:

"/" — Fornece uma lista dos comandos possíveis.
"L" — Lista o programa fonte.
"O" — Lista os *opcodes* (códigos hexadecimais gerados pela compilação).
"M" — Fornece o número de bytes ainda disponíveis.
"F" — Permite gravar ou carregar um programa usando uma fita cassete.
"E" — Eliminar uma linha.
"I" — Inserir uma linha.
"S" — Substituir uma linha.
"A" — Compilar ("Assemblar") um programa fonte previamente carregado (de uma fita) ou recém-digitado.
"P" — Preparar uma rotina já compilada para execução, colocando todo o código hexa na memória.
"E" — Executar uma rotina já preparada. Se o final do seu programa for um RTS, o controle retorna ao modo de comando do Editor/Assembler.
"T" — Terminar o programa.

Se o seu programa for longo, é sempre uma boa medida gravá-lo em fita antes de tentar a execução, só por precaução. Existe uma infinidade de erros que não são detectados pelo Editor/Assembler.

Listagem 18-1 — Editor/Assembler em BASIC

```
5800 PCLEAR3:CLEAR500:CLS:PRINT@
6,"EDITOR / ASSEMBLER":PRINT"LIN
HA LABEL  MNEM.  ARG.":S=50:DIML
```

```
$(S),M$(S),O$(S),AD(S),OC$(S):GO
SUB6270:I=0:R$="DXYUSPABCZ":N$="
0123456789AB":S$="CABZXY^P"
5805 I=I+1:GOSUB6835:PRINT@BY,US
ING" ####";I;:BY=BY+6:GOSUB6810:
L$(I)=LH$:BY=BY+7:PRINT@BY,"";:G
OSUB6810:M$(I)=LH$:BY=BY+7:PRINT
@BY,"";:GOSUB6810:O$(I)=LH$:PRIN
T
5810 IFM$(I)="END"THENN=I:ELSE58
05
5815 X$=INKEY$:IFX$="L"THEN5880
5820 IFX$=CHR$(12)THENCLS
5825 IFX$="M"THENPRINTMEM
5830 IFX$="E"THENE=USR1(0)
5835 IFX$="I"THEN5895
5840 IFX$="/" ORX$="?" THEN CLS:
PRINT"lISTAR","oPCODES","mEMORIA
","fITA",,,"dELETAR","iNSERIR","
sUBSTITUIR",,,,"pREPARAR","eXECU
TAR","tERMINAR":GOTO5815
5845 IFX$="O"GOSUB6440
5850 IFX$="P"GOSUB6590
5855 IFX$="F"GOTO6725
5860 IFX$="D"THEN5915
5865 IFX$="A"THEN5975
5870 IFX$="T"THENCLS:PRINT"FIM D
E OPERACAO":END
5875 IFX$="S"THEN5930ELSE5815
5880 FORI=1 TON
5885 PRINTI;TAB(3)L$(I)TAB(9)M$(
I)TAB(14)O$(I)TAB(21)OC$(I)
5890 NEXTI:GOTO5815
5895 INPUT"APOS";K:N=N+1
5900 FORQ=N+1 TOK+1STEP-1
5905 L$(Q)=L$(Q-1):M$(Q)=M$(Q-1)
:O$(Q)=O$(Q-1)
5910 NEXTQ:L$(K+1)="":M$(K+1)=""
:O$(K+1)="":GOSUB5965:GOTO5815
5915 INPUT"#";K:FORI=K TON
5920 L$(I)=L$(I+1):M$(I)=M$(I+1)
:O$(I)=O$(I+1)
5925 NEXTI:N=N-1:GOSUB5965:GOTO5
815
5930 INPUT"#";K
5935 PRINT"lABEL, mNEMONICO OU a
RG."
```

```
5940 X$=INKEY$:IFX$=""THEN5940
5945 IFX$="L"THENPRINTL$(K):INPU
TL$(K)
5950 IFX$="M"THENPRINTM$(K):INPU
TM$(K)
5955 IFX$="A"THENPRINTO$(K):LINE
INPUT"?";O$(K)
5960 GOSUB5965:GOTO5815
5965 FORI=1 TON:OC$(I)="":NEXTI:
RETURN
5970 '       MONTADOR
5975 GOSUB5965:FORI=1 TON-1:PRIN
TI;
5980 M$=M$(I):O$=O$(I):OC$="":LO
=LEN(O$):PB=0
5985 IFM$="ORG"ORM$="EQU"THENGOS
UB6545:AD(I)=VAL("&H"+Q$):GOTO62
60
5990 IFM$()"DATA"THEN6005
5995 IFASC(O$)()39THENOC$(I)=O$:
GOTO6005
6000 FORQ=2 TOLO-1:OC$(I)=OC$(I)
+HEX$(ASC(MID$(O$,Q,1))):NEXTQ
6005 AD(I)=AD(I-1)+.5*LEN(OC$(I-
1))
6010 IFM$="DATA"THEN6260
6015 IFM$="TFR"ORM$="EXG"THEN664
5
6020 IF LEN(M$)=3THENM$=M$+" "
6025 FOR J=1 TO M
6030 T$=LEFT$(MN$(J),4)
6035 IFT$()M$THEN6045
6040 T$=MN$(J):LT=LEN(T$):GOTO60
50
6045 NEXTJ:GOTO6265
6050 IF ASC(T$)=80 THEN6685
6055 GOSUB6425
6060 IF LO=0 THEN 6070
6065 GOSUB6390:IFII=1 OR CP>0 TH
EN6105
6070 IFLT>6 THEN6080
6075 OC$(I)=RIGHT$(T$,2)+MID$(O$
,2):GOTO6250
6080 IFLO=3 OR LO=5 THENBP=5:LO=
LO-1:GOTO6095
```

```
6085 IFLO=2THENBP=7
6090 IFLO=4THENBP=9
6095 O$=RIGHT$(O$,LO):IF LO=4 GO
SUB 6545:O$=Q$
6100 OC$(I)=MID$(T$,BP,2)+O$:GOT
O6250
6105 OC$(I)=MID$(T$,11,2)
6110 PB=128+16*II:FORQ=LO TO LO-
2STEP-1
6115 Q$=MID$(O$,Q,1):RV=0
6120 IFQ$="X"THEN6145
6125 IFQ$="Y"THEN RV=32:GOTO6145

6130 IFQ$="U"THEN RV=64:GOTO6145

6135 IFQ$="S"THEN RV=96:GOTO6145

6140 NEXTQ
6145 PB=PB+RV:DV$=""
6150 ON LO GOTO 6265,6155,6180,6
210,6160,6225,6265,6235
6155 PB=PB+4:GOTO6240
6160 PB=VAL("&H"+MID$(O$,2,2))
6165 IF ASC(O$)=45 THEN PB=-PB
6170 PB=PB AND 31:PB=PB+RV
6175 DV$="":GOTO6240
6180 Q1$=LEFT$(O$,1):Q3$=RIGHT$(
O$,1)
6185 IFQ1$="A"THEN PB=PB+6:GOTO6
240
6190 IFQ1$="B"THEN PB=PB+5:GOTO6
240
6195 IFQ1$="D"THEN PB=PB+11:GOTO
6240
6200 IFQ3$="+"THEN 6240
6205 PB=PB+2:GOTO 6240
6210 IF ASC(O$)=ASC(",") THEN PB
=PB+1:GOTO6240
6215 IFCP=0 GOSUB6545:PB=&H9F:DV
$=Q$:GOTO6240
6220 PB=PB+8:DV$=LEFT$(O$,2):GOT
O6240
6225 IF MID$(O$,5,1)=","THEN PB=
PB+9:O$=LEFT$(O$,4):GOSUB6545:DV
$=Q$:GOTO6240
6230 PB=PB+12:DV$=LEFT$(O$,2):GO
```

```
TO 6240
6235 O$=LEFT$(O$,4):GOSUB6545:DV
$=Q$:PB=PB+13
6240 PB$=HEX$(PB):IF LEN(PB$)=1
THEN PB$="0"+PB$
6245 OC$(I)=OC$(I)+PB$
6250 OC$(I)=PR$+OC$(I)+DV$
6255 GOSUB6450
6260 DV$="":NEXTI:PRINT"OK":GOTO
5815
6265 GOSUB6720:GOTO5815
6270 M=139:DIM MN$(M):FORI=1TOM
6275 READMN$(I):NEXTI:RETURN
6280 DATAABX 3A,ADCA8999B9A9,AD
CBC9D9F9E9,ADDA8B9BBBAB,ADDBCBDB
FBEB,ADDDC3D3F3E3,ANDA8494B4A4
6285 DATAANDBC4D4F4E4,ANDC1C????
??,ASL ??087868,ASLA48,ASLB58,AS
R??077767,ASRA47,ASRB57
6290 DATABCC ????24??,BCS ??????5
??,BEQ ????27??,BGE ????2C??,BGT
 ????2E??,BHI ????22??
6295 DATABHS ????24??,BITA8595B5
A5,BITBC5D5F5E5,BLE ????2F??,BLO
 ????25??,BLS ????23??
6300 DATABLT ????2D??,BMI ????2B
??,BNE ????26??,BPL ????2A??,BRA
 ????20??,BRN ????21??,BSR ????8
D??,BVC ????28??,BVS ????29??
6305 DATACLR ??0F7F6F,CLRA4F,CLR
B5F,CMPA8191B1A1,CMPBC1D1F1E1,CM
PD8393B3A3A,CMPS8C9CBCACB
6310 DATACMPU8393B3A3B,CMPX8C9CB
CAC,CMPY8C9CBCCACA,COM ??037363,
COMA43,COMB53;CWAI3C??????,DAA 1
9
6315 DATADEC ??0A7A6A,DECA4A,DEC
B5A,EORA8898B8A8,EORBC8D8F8E8,EX
G 1E??????,INC ??0C7C6C,INCA4C
6320 DATAINCB5C,JMP ??0E7E6E,JSR
 ??9DBDAD
6325 DATALBCC????24A,LBCS????25?
?A,LBEQ????27??A,LBGE????2C??A
6330 DATALBGT????2E??A,LBHI????2
2??A,LBHS????24??A,LBLE????2F??A
,LBLO????25??A,LBLS????23??A
```

```
6335 DATALBLT????2D??A,LBMI????2
B??A,LBNE????26??A,LBPL????2A??A
,LBRA????16??,LBRN????21??A
6340 DATALBSR????17??,LBVC????28
??A,LBVS????29??A
6345 DATALDA 8696B6A6,LDB C6D6F6
E6,LDD CCDCFCEC,LDS CEDEFEEEA
6350 DATALDU CEDEFEEE,LDX 8E9EBE
AE,LDY 8E9EBEAEA,LEAS??????32
6355 DATALEAU??????33,LEAX??????
30,LEAY??????31,LSL ??087868,LSL
A48,LSLB58,LSR ??047464,LSRA44,L
SRB54,MUL 3D,NEG ??007060
6360 DATANEGA40,NEGB50,NOP 12,OR
A 8A9ABAAA,ORB CADAFAEA,ORCC1A,P
SHS34,PSHU36,PULS35,PULU37
6365 DATAROL ??097969,ROLA49,ROL
B59,ROR ??067666,RORA46,RORB56,R
TI 3B,RTS 39,SBCA8292B2A2
6370 DATASBCB C2D2F2E2,SEX 1D,STA
 ??97B7A7,STB ??D7F7E7,STD ??DDF
DED,STS ??DFFFEFA,STU ??DFFFEF
6375 DATASTX ??9FBFAF,STY ??9FBF
AFA,SUBA8090B0A0,SUBBC0D0F0E0,SU
BD8393B3A3,SWI 3F,SWI23FA,SWI33F
B
6380 DATASYNC13,TRF 1F??????,TST
 ??0D7D6D,TSTA4D,TSTB5D
6385 '    IND/IND?
6390 II=0:CP=0
6395 IF ASC(O$)=91THEN II=1:LO=L
O-2:O$=MID$(O$,2,LO)
6400 FORQ=1 TOLO
6405 Q$=MID$(O$,Q,1)
6410 IFQ$=","THEN CP=Q:RETURN
6415 NEXTQ:RETURN
6420 '    PREBYTE
6425 PR$="":IFLT/2=INT(LT/2)THEN
RETURN
6430 PR$="10":IF RIGHT$(T$,1)="B
" THEN PR$="11"
6435 LT=LT-1:T$=LEFT$(T$,LT):RET
URN
6440 FORK=1 TON-1
6445 PRINTK;TAB(5);L$(K);TAB(11)
;HEX$(AD(K));TAB(17)OC$(K):NEXTK
```

```
:RETURN
6450 F$=LEFT$(M$,1):F2$=LEFT$(M$
,2)
6455 IFF2$="LB"THEN6500
6460 IFF2$="BI" OR F$()"B" THEN
RETURN
6465 GOSUB6545
6470 BV=AD(I)+2:BV=VAL("&H"+Q$)-
BV
6475 IFBV<0THEN BV=256+BV
6480 IFBV<0 OR BV>255THEN BV=0:G
OSUB6720
6485 H$=HEX$(BV):IFLEN(H$)=1THEN
 H$="0"+H$
6490 OC$(I)=LEFT$(OC$(I),2)+H$
6495 RETURN
6500 LC=LEN(OC$(I)):LC=LC-4
6505 OC$(I)=LEFT$(OC$(I),LC)
6510 GOSUB6545
6515 WV=AD(I)+4:WV=VAL("&H"+Q$)-
WV
6520 IFWV<0THEN WV=65536+WV
6525 IFWV<0 OR WV>65535THEN 6265

6530 H$=HEX$(WV)
6535 IFLEN(H$)<4THEN H$="0"+H$:G
OTO6535
6540 OC$(I)=OC$(I)+H$:RETURN
6545 O=ASC(O$):IFO<71 OR O>90THE
N Q$=O$:RETURN
6550 Q$="0000"
6555 FORQ=1 TO N-1
6560 IFO$=L$(Q)THEN 6570
6565 NEXTQ:GOSUB6720:RETURN
6570 IF Q>I THEN PRINT"PARA FREN
TE"
6575 Q$=HEX$(AD(Q))
6580 IFLEN(Q$)<4THENQ$="0"+Q$:GO
TO6580
6585 RETURN
6590 FORI=1 TO N-1
6595 O$=OC$(I):L=LEN(O$)
6600 IFL=0THEN 6630
6605 A=AD(I)
6610 FORQ=1 TOL-1 STEP2
6615 V=VAL("&H"+MID$(O$,Q,2))
```

```
6620 POKEA,V:PRINTHEX$(A);" ";HE
X$(V);" ";:A=A+1
6625 NEXTQ
6630 NEXTI
6635 O$=O$(N):GOSUB6545:PRINT
6640 DEFUSR1=VAL("&H"+Q$):RETURN

6645 R1$=LEFT$(O$,1):R2$=RIGHT$(
O$,1)
6650 FORQ=1 TO10
6655 Q$=MID$(R$,Q,1):NC$=MID$(N$
,Q,1)
6660 IFR1$=Q$THEN N1$=NC$
6665 IFR2$=Q$THEN N2$=NC$
6670 NEXTQ
6675 O$="1E":IFM$="TFR"THEN O$="
1F"
6680 OC$(I)=O$+N1$+N2$:GOTO6260
6685 BV=0:FORQ=1 TO LO STEP2
6690 Q$=MID$(O$,Q,1)
6695 EP=INSTR(S$,Q$):IFEP=0THEN6
705
6700 BV=BV+2^(EP-1)
6705 NEXTQ
6710 H$=HEX$(BV):IFLEN(H$)=1THEN
 H$="0"+H$
6715 OC$(I)=RIGHT$(T$,2)+H$:GOTO
6260
6720 PRINT"ERRO LINHA";I:RETURN
6725 INPUT"L S";X$
6730 IFX$="S"THEN 6770
6735 IFX$<>"L"THEN 6800
6740 GOSUB 6805
6745 OPEN"I",#-1,"CODES"
6750 INPUT#-1,N
6755 FORI=1 TON
6760 INPUT#-1,L$(I),M$(I),O$(I)
6765 NEXTI:GOSUB5965:GOTO6800
6770 GOSUB6805
6775 OPEN"O",#-1,"CODES"
6780 PRINT#-1,N
6785 FORI=1 TON
6790 PRINT#-1,L$(I),M$(I),O$(I)
6795 NEXTI
6800 PRINT"ok":CLOSE#-1:GOTO5815

6805 INPUT"PREPARE O GRAVADOR";Q
```

```
$RETURN
6810 'INTRODUCAO
6815 LH$=""
6820 TE$=INKEY$:PRINT@BY+LEN(LH$
),CHR$(206);:PRINT@BY,LH$;" ";:I
F TE$="" THEN6820 ELSE IF TE$=CH
R$(8) AND LEN(LH$)>0 THEN LH$=LE
FT$(LH$,LEN(LH$)-1):GOTO 6820
6825 IF TE$=CHR$(13) THEN RETURN

6830 LH$=LH$+TE$:GOTO 6820
6835 'POSICAO DO CURSOR
6840 BY=PEEK(136)*256+PEEK(137)-
1024
6845 IF BY>479 THEN PRINT@511,""
:PRINT@448,"";:GOTO 6840
6850 RETURN
```

```
LINHA: 5845     128     LINHA: 6395     120
LINHA: 5895     31      LINHA: 6445     111
LINHA: 5945     42      LINHA: 6495     30
LINHA: 5995     162     LINHA: 6545     72
LINHA: 6045     98      LINHA: 6595     203
LINHA: 6095     68      LINHA: 6645     93
LINHA: 6145     232     LINHA: 6695     127
LINHA: 6195     145     LINHA: 6745     48
LINHA: 6245     113     LINHA: 6795     7
LINHA: 6295     231     LINHA: 6845     245
                        FIM:   99
LINHA: 6345     178
```

19 COLOR DISASSEMBLER: INTERPRETADOR DE ROTINAS

Para complementar o seu "Pacote de Programação em L.M.", que já conta com um Editor/Assembler, aqui está o programa que faz a função inversa: a "decodificação" das rotinas em L.M. na memória. É óbvio que esse programa não tem como finalidade fornecer o programa fonte da rotina gerada pelo Editor do capítulo anterior. Isso é desnecessário! Mas muitas vezes, você vai precisar analisar certas rotinas que serão necessárias ao seu programa.

Essas rotinas, quase sempre, não têm programas fonte, pois são as que estão na ROM do micro, como a do CLOAD, mencionada no Capítulo 11. Assim, para entender essas rotinas e, a partir daí, poder explorá-las da melhor forma, é necessário ter uma listagem mais legível do que uma série de endereços e valores hexadecimais.

O PROGRAMA

O interpretador de rotinas (conhecido como *DISASSEMBLER* no meio técnico) apresentado na Listagem 19-1 roda em apenas 7k de memória e fornece uma listagem fonte com todos os mnemônicos e argumentos padrões do 6809E. Como no capítulo anterior, o registrador CC é uma exceção, pois nesse programa ele é representado por C, simplesmente. Assim, a instrução ANDCC aparece como ANDC, bem como todas as demais que utilizarem esse registrador.

A presença da seta para cima como argumento de uma instrução de troca, transferência ou manipulação de pilha representa o indicador de pilha complementar ao referido no próprio mnemônico. Por exemplo, PULS ↑ significa tirar do indicador de pilha do sistema o valor a ser colocado no indicador de pilha do usuário. Em outras palavras, tirar de S e colocar em U.

A listagem resultante trabalha com valores em hexadecimal e está dividida em quatro campos: endereço, *opcodes* (podem ser dados), mnemônico e argumentos. Como todo programa desse tipo, existe um problema inerente à própria natureza do processo. Como o programa foi escrito para interpretar *todo o conteúdo da memória* como rotina em linguagem de máquina, as informações referentes a tabelas ou dados incorporados à rotina também serão

considerados *opcodes* executáveis. Isso provoca um erro de interpretação que faz o programa perder o controle sobre o que está decodificando. Para usar um chavão antigo, o programa "perde o fio da meada". Para contornar esse problema, nunca use o programa para analisar grandes rotinas de uma só vez. Vá por partes, identificando cada desvio na seqüência natural da listagem e tentando delimitar as tabelas que existirem.

CONSIDERAÇÕES FINAIS

A operação do programa em si não tem grandes segredos. Basta responder às perguntas iniciais com o endereço por onde a análise deve começar, em decimal. Você pode usar endereços hexadecimais, se preferir. Basta que o endereço seja precedido por um "&H". Para interromper a listagem, voltando à pergunta inicial, pressione qualquer tecla.

Espero que esse programa, juntamente com o Editor, dê-lhe condições de explorar um pouco mais os recursos de seu micro. Naturalmente, os programas profissionais de edição/compilação e de interpretação são infinitamente mais eficientes que os aqui apresentados. Estes são meramente introdutórios e têm por propósito unicamente permitir-lhe um primeiro contato com o 6809E. Mesmo assim, os resultados dependem muito mais da sua dedicação e habilidade do que dos recursos ora apresentados.

Listagem 19-1 — COLOR DISASSEMBLER: interpretador de rotinas

```
6890 'DISASSEMBLER
6895 'PAULO ADDAIR  --  JUN/85
6900 PCLEAR1:RK$="CABDXY^P"
6905 DIMX$(15):FORI=0 TO15:READX
$(I):NEXT
6910 DATAD,X,Y,U,S,PC,-,-,A,B,C,
DP,-,-,-,-
6915 DIM R$(3)
6920 FORI=0 TO3:READR$(I):NEXT
6925 DATA X,Y,U,S
6930 DIM M$(15,15)
6935 FORI=0 TO15:FORJ=0 TO15
6940 READM$(I,J):NEXTJ,I
6945 DATANEG 2,-,-,COM 2,USR 2,-
,ROR 2,ASR 2,ASL 2,ROL 2,DEC2,-,
INC 2,TST 2,JMP 2,CLR 2
6950 DATA-,-,NOP 1,SYNC1,-,-,LBR
A8,LBSR8,-,DAA 1,ORCC4,-,ANDC4,S
EX 1,EXG 5,TFR 5
6955 DATABRA 3,BRN 3,BHI 3,BLS 3
```

```
,BCC 3,BCS 3,BNE 3,BEQ 3,BVC 3,B
VS 3,BPL 3,BMI 3,BGE 3,BLT 3,BGT
 3,BLE 3
6960 DATALEAX9,LEAY9,LEAS9,LEAU9
,PSHS4,PULS4,PSHU4,PULU4,-,RTS 1
,ABX 1,RTI 1,CWAI4,MUL 1,-,SWI 1
SWI 2SWI 3
6965 DATANEGA1,-,-,COMA1,LSRA1,-
,RORA1,ASRA1,LSLA1,ROLA1,DECA1,-
,INCA1,TSTA1,-,CLRA1
6970 DATANEGB1,-,-,COMB1,LSRB1,-
,RORB1,ASRB1,LSLB1,ROLB1,DECB1,-
,INCB1,TSTB1,-,CLRB1
6975 DATANEG 9,-,-,COM 9,LSR 9,-
,ROR 9,ASR 9,LSL 9,ROL 9,DEC 9,-
,INC 9,TST 9,JMP 9,CLR 9
6980 DATANEG 7,-,-,COM 7,LSR 7,-
,ROR 7,ASR 7,LSL 7,ROL 7,DEC 7,-
,INC 7,TST 7,JMP 7,CLR 7
6985 DATASUBA4,CMPA4,SBCA4,SUBD6
CMPD6CMPU6,ANDA4,BITA4,LDA 4,-,E
ORA4,ADCA4,ORA 4,ADDA4,CMPX6CMPY
6CMPS6,BSR 3,LDX 6LDY 6,-
6990 DATASUBA2,CMPA2,SBCA2,SUBD2
CMPU2,ANDA2,BITA2,LDA 2,STA 2,EO
RA2,ADCA2,ORA 2,ADDA2,CMPX2CMPY2
CMPS2,JSR 2,LDX 2LDY 2,STX 2STY
2
6995 DATASUBA9,CMPA9,SBCA9,SUBD9
CMPD9CMPU9,ANDA9,BITA9,LDA 9,STA
 9,EORA9,ADCA9,ORA 9,ADDA9,CMPX9
CMPY9CMPS9,JSR 9,LDX 9LDY 9,STX
9STY 9
7000 DATASUBA7,CMPA7,SBCA7,SUBD7
CMPD7CMPU7,ANDA7,BITA7,LDA 7,STA
 7,EORA7,ADCA7,ORA 7,ADDA7,CMPX7
CMPY7CMPX7,JSR 7,LDX 7LDY 7,STX
7STY 7
7005 DATASUBB4,CMPA4,SBCB4,ADDD6
,ANDB4,BITB4,LDB 4,-,EORB4,ADCB4
,ORB 4,ADDB4,LDD 6,-,LDU 6LDS 6,
-
7010 DATASUBB2,CMPB2,SBCB2,ADDD2
,ANDB2,BITB2,LDB 2,STB 2,EORB2,A
DCB2,ORB 2,ADDB2,LDD 2,STD 2,LDU
 2LDS 2,STU 2STS 2
```

```
7015 DATASUBB9,CMPB9,SBCB9,ADDD9
,ANDB9,BITB9,LDB 9,STB 9,EORB9,A
DCB9,ORB 9,ADDB9,LDD 9,STD 9,LDU
 9LDS 9,STU 9STS 9
7020 DATASUBB7.,CMPB7,SBCB7,ADDD7
,ANDB7,BITB7,LDB 7,STB 7,EORB7,A
DCB7,ORB 7,ADDB7,LDD 7,STD 7,LDU
 7LDS 7,STU 7STS 7
7025 DIM P$(3,15)
7030 FOR I=0 TO3:FORJ=0 TO15:REA
DP$(I,J):NEXTJ,I
7035 DATA"+0,R","+1,R","+2,R","+
3,R","+4,R","+5,R","+6,R","+7,R"
,"+8,R","+9,R","+A,R","+B,R","+C
,R","+D,R","+E,R","+F,R"
7040 DATA"-10,R","-F,R","-E,R","
-D,R","-C,R","-B,R","-A,R","-9,R
","-8,R","-7,R","-6,R","-5,R","-
4,R","-3,R","-2,R","-1,R"
7045 DATA",R+",",R++",",-R",",--
R",",R","B,R","A,R",-,"**,R","##
##,R",-,"D,R","**,PC","####,PC",
-,-
7050 DATA-,"[,R++]",-,"[,--R]","
[,R]","[B,R]","[A,R]",-,"[**,R]"
,"[####,R]",-,"[D,R]","[**,PC]",
"[####,PC]",-,"[####]"
7055 INPUT"iMPRESSORA OU vIDEO";
IV$
7060 IF IV$="I" THEN IV=1 ELSE I
V=0
7065 INPUT"INTRODUZA ENDERECO";M

7070 H$=HEX$(PEEK(M)):IFLEN(H$)=
1THENH$="0"+H$
7075 SM=M
7080 I=VAL("&H"+LEFT$(H$,1))
7085 J=VAL("&H"+RIGHT$(H$,1))
7090 IFI=1 AND J<2 THEN 7435
7095 '     OPCODE DE 2 BYTES
7100 S$=M$(I,J):IFS$="-"THENS$="
---   ":O$=CHR$(39)+CHR$(16*I+J)+
CHR$(39):GOTO7125
7105 S$=LEFT$(S$,5)
7110 AT=VAL(RIGHT$(S$,1))
7115 ON AT GOTO 7205,7210,7215,7
235,7270,7275,7280,7285,7315
```

```
7120 ' MODOS DE ENDERECAMENTO
7125 S$=LEFT$(S$,4)+"    "+O$
7130 O$=""
7135 T$=HEX$(SM)
7140 IF LEN(T$)<4THEN T$="0"+T$:
GOTO7140
7145 T$=T$+"  "
7150 FORI=SM TO M:V$=HEX$(PEEK(I
)):IF LEN(V$)=1THEN V$="0"+V$
7155 T$=T$+V$:NEXTI
7160 PRINT:PRINTT$;TAB(15)S$;
7165 IF IV=1 THENPRINT#-2,"":PRI
NT#-2,T$;TAB(15)S$;
7170 M=M+1
7175 RR$=INKEY$
7180 IFRR$=""THEN7070
7185 PRINT:GOTO7055
7190 ' VERIFICA PEEK(M)
7195 M=M+1:V$=HEX$(PEEK(M)):IFLE
N(V$)=1THEN V$="0"+V$
7200 RETURN
7205 O$="":GOTO7125
7210 GOSUB7195:O$=V$:GOTO7125
7215 M=M+1:V=PEEK(M):IFV>127THEN
 V=V-256
7220 V=M+1+V:O$=HEX$(V)
7225 IFLEN(O$)<4THEN O$="0"+O$:G
OTO7225
7230 GOTO7125
7235 GOSUB7195:O$="#"+V$
7240 IFASC(S$)<>80THEN 7125
7245 N=VAL("&H"+V$):O$=""
7250 FORI=0 TO7
7255 IF (N AND (2^I))>OTHEN O$=O
$+","+MID$(RK$,I+1,1):IFRIGHT$(O
$,1)="D"THEN O$=O$+"P"
7260 NEXT I
7265 O$=MID$(O4,2):GOTO 7125
7270 GOSUB7195:R1=VAL("&H"+LEFT$
(V$,1)):R2=VAL("&H"+RIGHT$(V$,1)
):O$=X$(R1)+","+X$(R2):GOTO7125
7275 GOSUB7195:O$="#"+V$:GOSUB71
95:O$=O$+V$:GOTO7125
7280 GOSUB7195:O$=V$:GOSUB7195:O
$=O$+V$:GOTO7125
7285 GOSUB7195:O$=V$:GOSUB7195:O
```

```
$=O$+V$
7290 V=VAL("&H"+O$)
7295 IFV>32767THEN V=V-65536
7300 V=M+1+V
7305 IF V<O THEN O$="---"ELSE O$
=HEX$(V)
7310 GOTO7125
7315 GOSUB7195
7320 N1=VAL("&H"+LEFT$(V$,1))
7325 N2=VAL("&H"+RIGHT$(V$,1))
7330 R=N1 AND 6:R=R/2:RG$=R$(R)
7335 I=2*INT(N1/8)+(N1 AND 1)
7340 J=N2:O$=P$(I,J)
7345 L=LEN(O$)
7350 FOR I=1 TO L
7355 IF MID$(O$,I,1)="R"THEN7370

7360 NEXTI
7365 GOTO7375
7370 MID$(O$,I,1)=RG$
7375 FORI=1 TOL
7380 IFMID$(O$,I,1)="*"THEN 7400

7385 IFMID$(O$,I,1)="#"THEN 7415

7390 NEXTI
7395 GOTO 7430
7400 GOSUB7195
7405 MID$(O$,I,4)=V$
7410 GOTO 7430
7415 GOSUB7195:T$=V$:GOSUB7195
7420 T$=T$+V$
7425 MID$(O$,I,4)=T$
7430 GOTO 7125
7435 GOSUB7195
7440 IFLEFT$(V$,1)<>"2"THEN7470
7445 L1=2
7450 L2=VAL("&H"+RIGHT$(V$,1))
7455 T$=M$(L1,L2)
7460 S$="L"+LEFT$(T$,3)+RIGHT$(T
$,1)
7465 GOTO 7110
7470 P=6+5*J
7475 L1=VAL("&H"+LEFT$(V$,1))
7480 L2=VAL("&H"+RIGHT$(V$,1))
7485 T$=M$(L1,L2)
```

```
7490 S$=MID$(T$,P,5)
7495 IFLEN(S$)=OTHEN S$="---   ":
O$="---":GOTO7125
7500 GOTO7110
```

```
LINHA: 6935      98

LINHA: 6985      49

LINHA: 7035      225

LINHA: 7085      76

LINHA: 7135      248

LINHA: 7185      161

LINHA: 7235      124

LINHA: 7285      139

LINHA: 7335      140

LINHA: 7385      133

LINHA: 7435      223

LINHA: 7485      188
FIM:   85
```

20 AS ENTRADAS E SAÍDAS DO CP 400

Para explorar com eficiência o 6809, não adianta apenas ter habilidade com a linguagem desse processador. É necessário também um certo conhecimento da arquitetura do computador como um todo. O microprocessador controla toda uma série de circuitos auxiliares e, sem eles, nenhum processamento pode ser realizado. Pelo menos nada que o micro fizer poderá ser visto, ouvido, gravado e o que você tiver que passar, ele não receberá. Para citar apenas os mais óbvios: o gerador de vídeo e o próprio teclado. Mas existem vários outros, que formam o que é conhecido como entradas/saídas de um microcomputador.

Todos esses circuitos possuem informação que o microprocessador utiliza. Em contrapartida, eles também precisam de certas informações que o micro lhes envia. As tabelas a seguir mostram as entradas/saídas do CP 400, seus endereços e informações relacionadas. Com esses dados esperamos ampliar a gama de aplicações de suas rotinas tanto em L.M. como em BASIC.

O principal circuito nessas tabelas é o Adaptador de Interface Periférica ou PIA (*Peripheral Interface Adapter*). É ele que faz a conexão do microprocessador com todos os outros circuitos auxiliares (conhecidos como "periféricos"). Ele se divide em várias portas de entrada e/ou saída de informação. Essas portas são representadas por endereços de memória que, por sua vez, se dividem em oito sinais chamados *bits*. Cada bit tem uma função específica relacionada nas tabelas. Para ativar uma dada função ou saber determinada condição, o 6809E deve enviar ou ler aquele bit dentro do endereço. Em BASIC, você pode acessar facilmente esses bits através da instrução POKE (como foi usado no Capítulo 9, "Um truque de animação"). No entanto, seja cuidadoso ao utilizar qualquer porta ou endereço mencionado aqui, pois o menor descuido pode representar problemas, inclusive com perda do controle do micro. Nesse caso a solução é desligar o equipamento e esperar uns dez segundos para religá-lo.

Tabela 20-1 — Porta de controles diversos dos drives

(armazenar o valor em &HFF40 — 65344)

Bit	Função
0	Seleção de drive (quando em 1 o drive 0 está ativo)
1	Seleção de drive (quando em 1 o drive 1 está ativo)
2	Seleção de drive (quando em 1 o drive 2 está ativo)
3	Acionamento dos motores (em 1 aciona todos os motores)
4	Pré-compensação de gravação (ativada em 1)
5	Dupla densidade (ativada em 1)
6	Seleção de drive (quando em 1, o drive 3 está ativo)
7	Sincronização de espera (ativada em 1)

Tabela 20-2 — Porta de controle da PIA (&HFF48 — 65352)

Valor Hexa	Dec.	Comando
&H03	3	Retorna a cabeça de gravação do drive selecionado à trilha 0
&H13	19	Posiciona a cabeça de gravação do drive selecionado na trilha especificada no registrador de trilha
&H80	128	Lê o setor determinado pelo registrador de setor
&HA0	160	Grava o setor determinado pelo registrador de setor
&HF0	240	Grava (formata) uma trilha. Usado por DSKINI
&HD0	208	Termina a função atual imediatamente

Tabela 20-3 — Modos de operação do vídeo (definidos pela porta PIA 2B e pela SAM)

Resolução		Número	PIA 2B				SAM		
X	Y	de cores	Bit 7 A/G	Bit 6 MG2	Bit 5 MG1	Bit 4 MG0	V2	V1	V0
256	192	2	1	1	1	1	1	1	0
128	192	4	1	1	1	0	1	1	0
128	192	2	1	1	0	1	1	0	1
128	96	4	1	1	0	0	1	0	0
128	96	2	1	0	1	1	0	1	1
128	64	4	1	0	1	0	0	1	0
128	64	2	1	0	0	1	0	0	1
64	64	4	1	0	0	0	0	0	1
64	32	8*(alfa)	0	0	0	0	0	0	0
64	48	8*	0	0	0	1	0	0	0
64	64	8*	0	0	0	0	0	1	0
64	96	8*	0	0	0	0	1	0	0
64	192	8*	0	0	0	0	1	1	0

Obs.: O bit 3 da PIA 2B seleciona qual conjunto de cores será utilizado. Assim, existem sempre dois conjuntos de cores possíveis, que podem conter 2, 4 ou 8 (nos modos semigráficos indicados por um asterisco) cores.

Figura 20-1 — Matriz de leitura do teclado

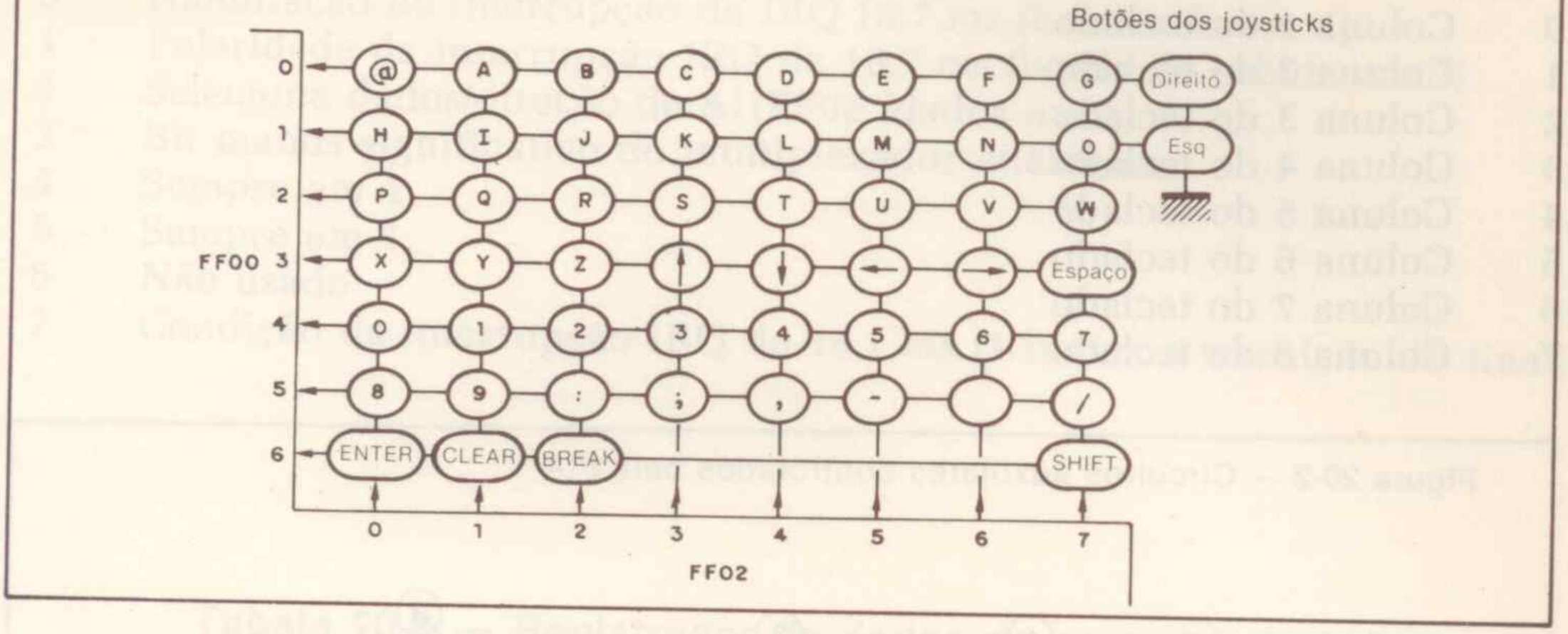


Tabela 20-4 — Registrador de dados da porta 1A da PIA (&HFF00 — 65280)

Bit	Função
0	Linha 1 do teclado e botão do joystick direito
1	Linha 2 do teclado e botão do joystick esquerdo
2	Linha 3 do teclado
3	Linha 4 do teclado
4	Linha 5 do teclado
5	Linha 6 do teclado
6	Linha 7 do teclado
7	Entrada para o comparador do joystick

Tabela 20-5 — Registrador de controle da porta 1A da PIA. (&HFF01 — 65281)

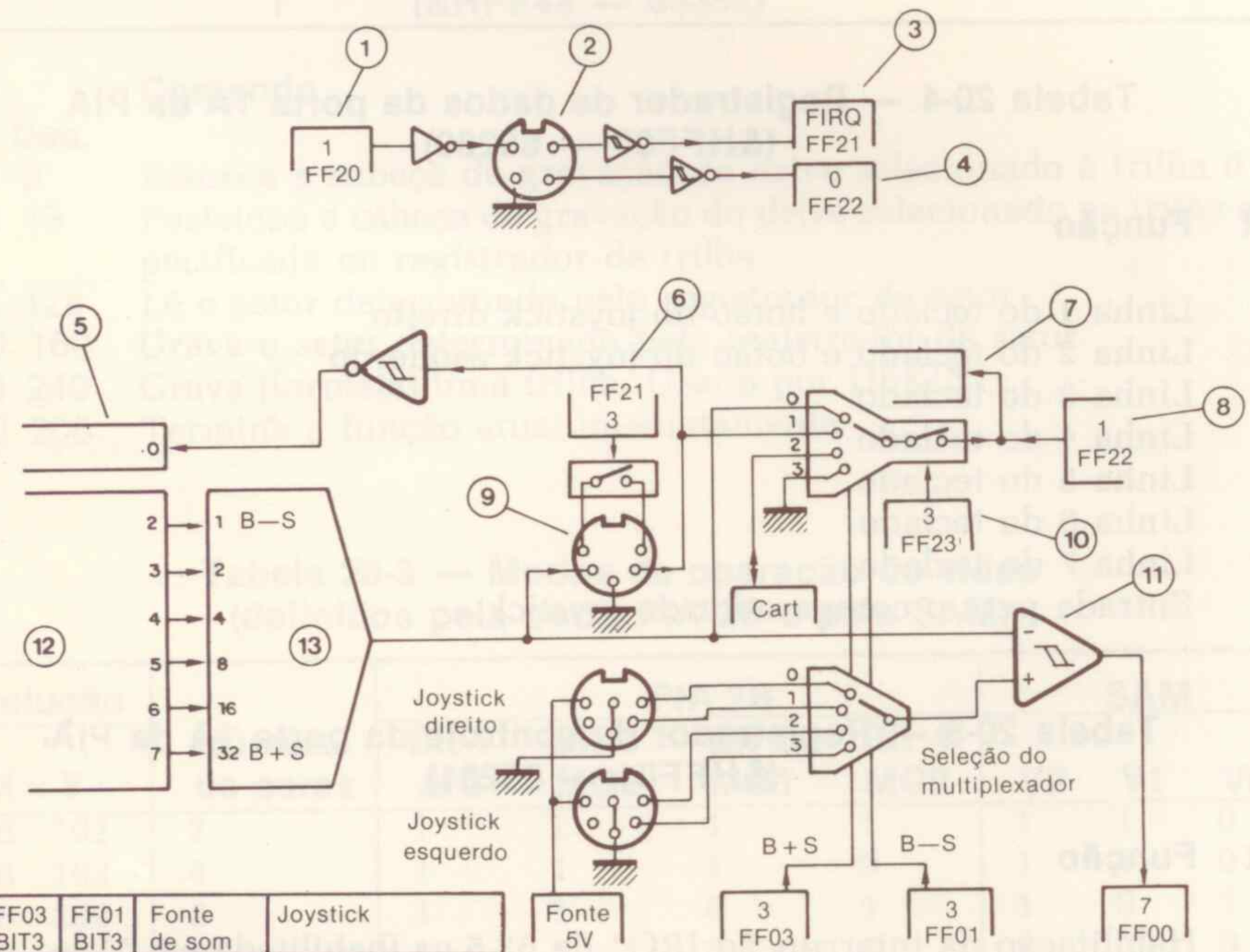
Bit	Função
0	Habilitação da interrupção IRQ* de 63.5 µs (habilitada em 1)
1	Polaridade da interrupção IRQ de 63.5 µs (borda de subida em 1)
2	Seleciona dados/direção de &HFF00 (dados em 1)
3	Bit menos significativo do multiplexador analógico
4	Sempre em 1
5	Sempre em 1
6	Não usado
7	Condição da interrupção de 63.5 µs (1 indica que houve transição no sinal)

***Obs.:** IRQ é uma das entradas de interrupção do 6809. Consulte o Capítulo 22 para maiores informações.

Tabela 20-6 — Registrador de dados da porta 1B da PIA (&HFF02 — 65282)

Bit	Função
0	Coluna 1 do teclado
1	Coluna 2 do teclado
2	Coluna 3 do teclado
3	Coluna 4 do teclado
4	Coluna 5 do teclado
5	Coluna 6 do teclado
6	Coluna 7 do teclado
7	Coluna 8 do teclado

Figura 20-2 — Circuitos auxiliares controlados pela PIA



	FF03 BIT3	FF01 BIT3	Fonte de som	Joystick
0	0	0	Conv. D/A	eixo X do direito
1	0	1	Cassete	eixo Y do direito
2	1	0	Cartucho	eixo X do esquerdo
3	1	1	Sem som	eixo Y do esquerdo

1. Saída RS 232C
2. Tomada serial (RS 232C)
3. Entrada de detecção de portadora
4. Entrada RS 232C
5. Entrada do gravador cassete
6. Controle do motor (REMOTE)
7. Saída de som
8. Som de 1 bit
9. Tomada K-7
10. Habilitação do multiplexador de som
11. Entrada do comparador do joystick
12. Saída do Conv. D/A FF20
13. Entrada de 0 a 63
    Saída de 0 a 5V

Tabela 20-7 — Registrador de controle da porta 1B da PIA (&HFF03 — 65283)

Bit	Função
0	Habilitação de interrupção de IRQ 16.7 ms (habilitada em 1)
1	Polaridade da interrupção IRQ de 16.7 ms (borda de subida em 1)
2	Seleciona dados/direção de &HFF02 (dados em 1)
3	Bit menos significativo do multiplexador analógico
4	Sempre em 1
5	Sempre em 1
6	Não usado
7	Condição da interrupção IRQ de 16.7 ms (1 indica a transição do sinal)

Tabela 20-8 — Registrador de dados da porta 2A da PIA (&HFF20 — 65312)

Bit	Função
0	Entrada de dados do gravador cassete (se 1, a entrada é menor que −1.5V)
1	Saída serial de dados da RS 232C (se 1, a saída fica em −10V)
2 a 7	Valor da saída para o conversor digital/analógico. Esses seis bits compõem um valor qualquer entre 0 e 31. O bit 7 é o mais significativo

Tabela 20-9 — Registrador de controle da porta 2A da PIA (&HFF21 — 65313)

Bit	Função
0	Habilitação da detecção (via FIRQ**) de portadora da RS 232C (sinal CD do padrão serial). Habilitada em 1
1	Polaridade da detecção de portadora da RS 232C (borda de subida em 1)
2	Seleciona dados/direção de &HFF20 (dados em 1)
3	Controle do motor do gravador (pino REMOTE). Ativa em 1
4	Sempre em 1
5	Sempre em 1
6	Não usado
7	Condição da detecção de portadora (1 indica transição do sinal)

****Obs.:** FIRQ é uma das interrupções por hardware de que o 6809 dispõe. É chamada *fast interrupt* e tem prioridade sobre IRQ (veja o Capítulo 22).

Tabela 20-10 — Registrador de dados da porta 2B da PIA (&HFF22 — 65314)

Bit	Função
0	Entrada serial de dados da RS 232C (entrada menor que +1V em 1)
1	Saída de 1 bit para o gerador de som (normalmente 0)
2	Entrada do tamanho da RAM (0 = 4k, 1 = 16k e alta impedância = 64k)
3	Seleção de conjunto de cor para o Gerador de Vídeo (GV)
4	GM0 do GV e INT/EXT (EXT em 1). Vide Tabela 20-3
5	GM1 Idem
6	GM2 Idem
7	Modo gráfico/alfa (gráfico em 1)

Tabela 20-11 — Registrador de controle da porta 2B da PIA (&HFF23 — 65315)

Bit	Função
0	Habilitação da detecção de interrupção (via FIRQ) do cartucho (habilitada em 1)
1	Polaridade da detecção de interrupção do cartucho (borda de subida em 1)
2	Seleciona dados/direção de &HFF22 (dados em 1)
3	Habilitação de geração de som por 6 bits (habilitado em 1)
4	Sempre em 1
5	Sempre em 1
6	Não usado
7	Condição da detecção de interrupção do cartucho (1 indica transição do sinal)

Tabela 20-12 — Registradores gerais

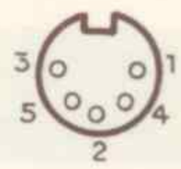
Registrador	Endereço
De comando da PIA	&HFF48 - 65352
De trilha	&HFF49 - 65353
De setor	&HFF4A - 65354
De dados	&HFF4B - 65355

Tabela 20-13 — Registrador de controle da SAM

Ativa		Função	Desativa	
Hexa	**Decimal**		**Hexa**	**Decimal**
FFDE	65502	(TY) — Tipo do mapa de memória (0 = 32k, 1 = 64k)	FFDF	65503
FFDC	65500	(M0) — Tipo de RAM:	FFDD	65501
FFDA	65498	(M1) 00 = 4k 01 = 16k 10 = 64k	FFDB	65499
FFD8	65496	(R1) — Freqüência de operação	FFD9	65497
FFD6	65494	(R0) 00 = .9 MHz,01 = dependente de endereço	FFD7	65495
FFD4	65492	(P1) — Páginação de memória:	FFD5	65493
FFD2	65490	(F6) — Endereço inicial da tela	FFD3	65491
FFD0	65488	(F5) mostrada (vide Cap. 9)	FFD1	65489
FFCE	65486	(F4)	FFCF	65487
FFCC	65484	(F3)	FFCD	65485
FFCA	65482	(F2)	FFCB	65483
FFC8	65480	(F1)	FFC9	65481
FFC6	65478	(F0)	FFC7	65479
FFC4	65476	(V2) — Modos de vídeo	FFC5	65477
FFC2	65474	(V1) (veja Tabela 20-3)	FFC3	65475

Figura 20-3 — Configuração dos pinos do CP 400

1. SERIAL. Conector DIN de 4 pinos



2. K-7. Conector DIN de 5 pinos

3. JOY. Conector DIN de 6 pinos

21 O CABO DE COMUNICAÇÃO SERIAL

Você sabia que a saída SERIAL, atrás de seu micro, não é *apenas* uma saída de dados para ligar na impressora? Na realidade, é uma interface de comunicação serial, capaz de tanto receber quanto transmitir dados, segundo um padrão internacional definido pela EIA — *Electronics Industries Association* — que é uma Associação das Indústrias de Eletrônica norte-americana (só pra variar!).

Esse padrão foi batizado RS 232C e define os sinais que são necessários para que a transmissão (e, conseqüentemente, a recepção do outro lado) ocorra com o máximo de confiabilidade e funcionalidade. A informação propriamente dita (os bytes a serem enviados) é transmitida por um único canal, bit por bit. Esse modo de operação é chamado "serial", pois a informação é desmembrada em vários sinais que são transmitidos seqüencialmente. Daí o nome técnico "Interface Serial Padrão RS 232C".

A COMPATIBILIDADE

Apesar de obedecer a essa norma, existe um pequeno detalhe na saída SERIAL do CP 400 que impede que se conecte o micro a qualquer equipamento do mesmo padrão: o conector não segue um padrão muito comum entre as interfaces seriais! O conector do micro é uma tomada DIN de 4 pinos e a grande maioria dos equipamentos que segue o padrão RS 232C usa uma tomada DB-25, que tem uma forma achatada e é composta por 25 pinos, distribuídos em duas fileiras (veja a figura 21-1). Conseqüentemente, não dá para ligar diretamente no CP 400.

Existe ainda um outro detalhe importante que diz respeito ao tipo de equipamento que você quer conectar ao seu micro. Em comunicação de dados existem dois tipos básicos de equipamento: o DCE (*Data Communication Equipment*), que é um equipamento de comunicação de dados; e o DTE (*Data Terminal Equipment*), ou equipamento terminal de dados. Para exemplificar, um modem usado para interligar dois micros via telefone pode ser considerado um DCE, enquanto uma impressora é um DTE.

Mas a diferença não é apenas conceitual. Existe uma diferença de pina-

gem (função dos pinos no conector) entre um DCE e um DTE. Existem vários pares de pinos que devem ser invertidos quando se quer mudar de um DTE para um DCE. No caso do CP 400, como o micro só utiliza quatro pinos, vamos discutir apenas a inversão que ele requer. Em um DTE, o pino 2 é utilizado como saída de dados do terminal, enquanto o pino 3 é a entrada. Já em um DCE as funções desses pinos estão invertidas, sendo o pino 2 a entrada e o pino 3, a saída.

O CABO DE COMUNICAÇÃO

Com essas informações, juntamente com as funções de cada pino (relacionadas no Apêndice B do manual do CP 400), é possível montar um cabo para interligar o micro a qualquer equipamento RS 232C, ou então inverter as ligações dos pinos 2 e 3 se você já tiver o cabo e mudar de equipamento. Obviamente, a melhor opção é comprar um cabo de adaptação entre o micro e o equipamento. Mas, devido aos inúmeros modelos de DTEs e DCEs disponíveis, muitas vezes fica difícil encontrar o cabo mais indicado.

A figura 21-1 mostra como devem ser as ligações do cabo RS 232C para um DCE, enquanto a figura 21-2 mostra o mesmo cabo preparado para um DTE.

OS PARÂMETROS DA RS 232C

Para uma perfeita utilização da sua saída serial com qualquer equipamento, é necessário que você saiba como fazer para adequar suas características às do equipamento utilizado.

Essas características dependem de valores armazenados na memória que podem ser alterados por comandos POKEs em BASIC. As tabelas 21-1 e 21-2 relacionam as características, seus endereços, valores possíveis e iniciais. Com essas informações é possível interligar o CP 400 com, virtualmente, qualquer equipamento RS 232C.

Tabela 21-1 — Características da RS 232C

Característica	Endereço B-S	Endereço B+S
Baud Rate (taxa de transmissão)	149	150
Atraso de linha	151	152
Largura do espaçamento de tabulação*	153	
Último ponto de tabulação*	154	
Largura da linha na impressora*	155	
Posição atual de impressão*	156	

Obs.: Características usadas exclusivamente com impressoras.

Tabela 21-2 — Valores disponíveis para a RS 232C

Baud Rate	POKE em	Valor B-S 150	Valor B+S 149
50 bauds		88	4
75 bauds		227	2
110 bauds		246	1
134.5 bauds		153	1
120 bauds		202	1
150 bauds		110	1
300 bauds		180	0
600* bauds		87	0
1200 bauds		40	0
1800 bauds		25	0
2000 bauds		23	0
2400 bauds		18	0
3600 bauds		10	0
4800 bauds		7	0
7200 bauds		3	0
9600 bauds		1	0
Tabulação da impressora	POKE em	**153**	
A cada 16 colunas*		16	
Último ponto de tabulação	POKE em	**154**	
Inicialmente em 112 (7 pontos)*		112	
Largura da linha na impressora	POKE em	**155**	
Inicialmente em 132 colunas*		132	
Posição atual de impressão POKE em		**156**	
Inicialmente na primeira coluna*		0	

*Obs.: Valor inicial.

Espero que com essas dicas você consiga implementar ainda mais o seu sistema, rompendo novas barreiras para ingressar, definitivamente, na era da Informação.

Figura 21-1 — Cabo de comunicação serial com DCE

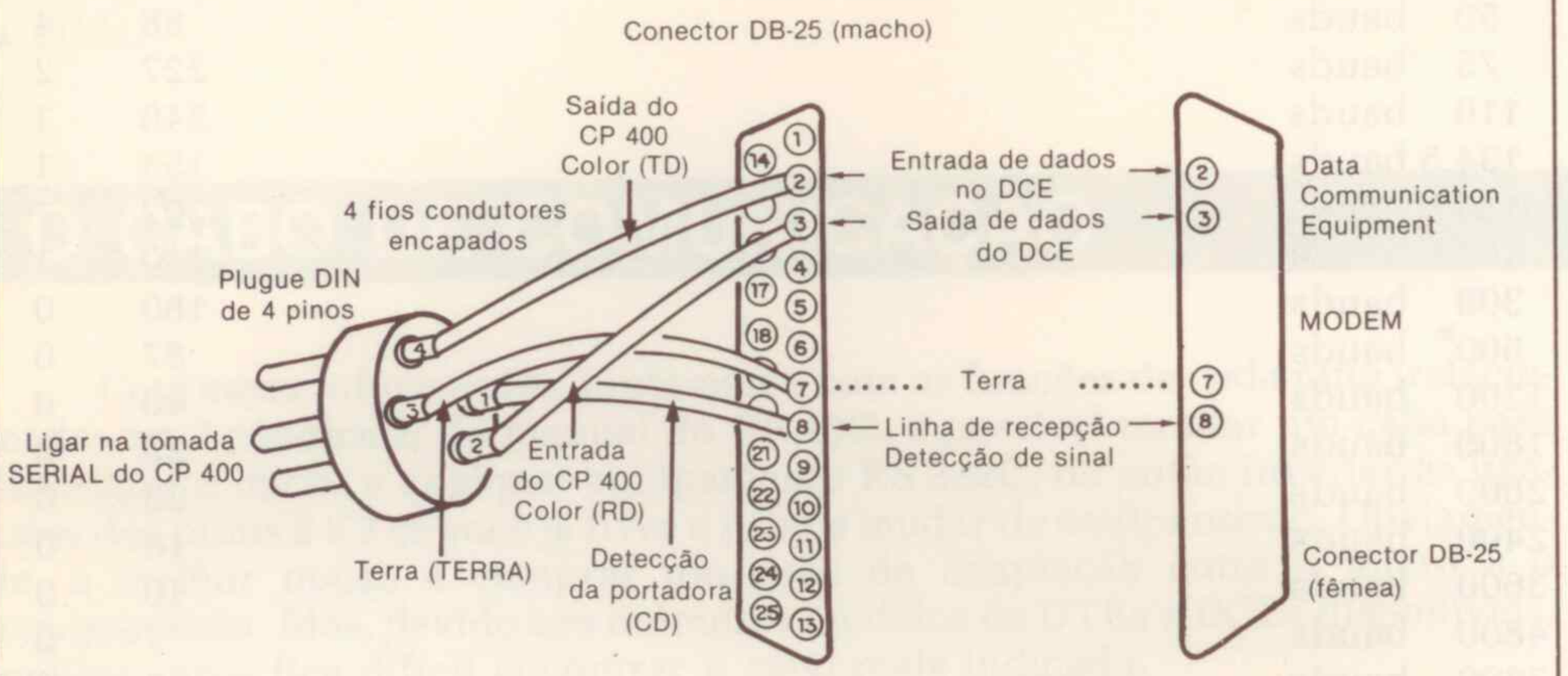
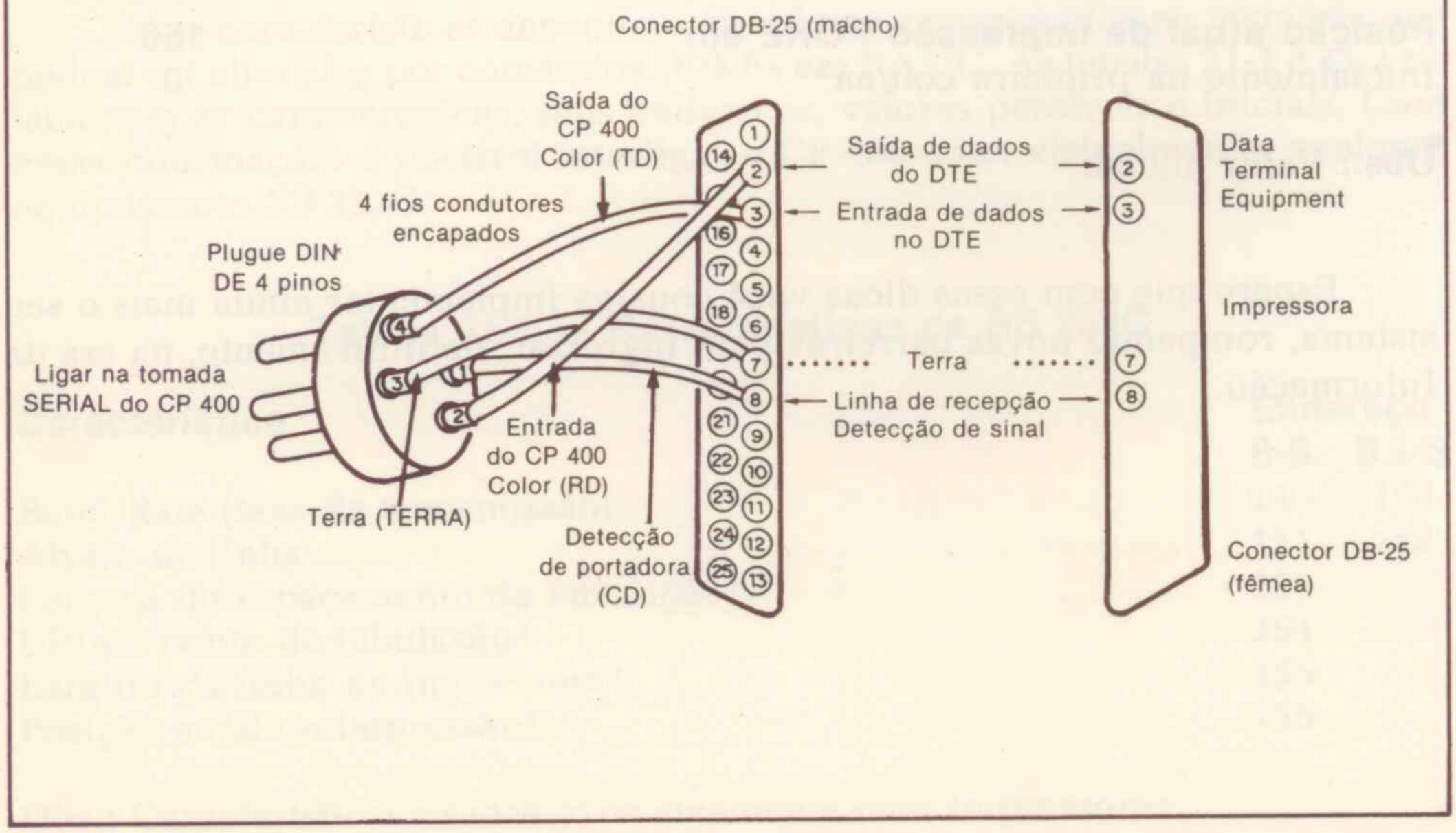


Figura 21-2 — Cabo de comunicação serial com DTE



22 AS INSTRUÇÕES DO MICROPROCESSADOR 6809E

Para facilitar a vida daqueles que pretendem ir um pouco mais além do que rodar os programas deste livro, este capítulo apresenta de forma sintética a arquitetura interna do 6809E, seus modos de endereçamento e conjunto de instruções. Dessa forma, é possível estruturar rotinas simples em linguagem de máquina.

Naturalmente, é necessário algum conhecimento de programação em L.M., mas para quem já tem uma boa "fluência" em BASIC não é tão dolorosa a mudança. A definição do algoritmo (descrição das etapas a serem cumpridas) serve tanto para o BASIC quanto para L.M.; a grande diferença é que todo o trabalho tem que ser realizado pelo programa em L.M.

Em BASIC, por exemplo, você não precisa definir um lugar da memória para os seus dados, pois o interpretador faz isso por você. Em L.M. você divide a memória antes mesmo de digitar o primeiro mnemônico. Em compensação, você obtém um programa compacto, rápido e eficiente (se você não tem experiência, não conte com isso antes do décimo programa).

REGISTRADORES INTERNOS DO 6809

O 6809 possui dois acumuladores de oito bits, que podem ser usados como um de 16, um registrador de código de condição (ou *status*), dois registradores de índice (ou indexadores), dois registradores de 16 bits, dois indicadores de pilha, um contador de programa e um registrador de página direta. A figura 22-1 ilustra esta estrutura.

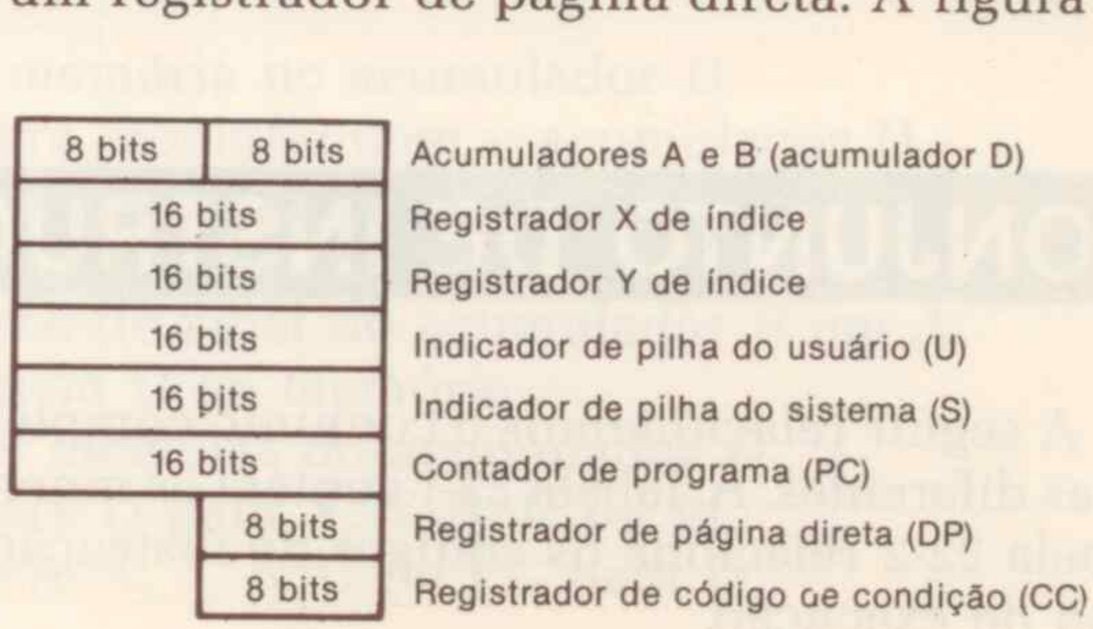


Figura 22-1 — Estrutura de registradores do 6809

O registrador de código de condição é composto por 5 "sinalizadores" (*flags*). Um sinalizador é uma espécie de bit que não faz parte da informação, mas indica uma determinada condição desta. As condições que o registrador CC controla são as seguintes:

Carry/Borrow (*flag* C): "vai um"/"vem um" em operações aritméticas;
Overflow (O ou V): "estouro" em operações aritméticas;
Zero (Z): indica que uma operação resultou em zero;
Sign (S ou N): sinalizador de sinal ou negativo;
Half-carry (H): "vai um"/"vem um" em operações com meio byte (BCD).

Os sinalizadores ocupam as seguintes posições dentro do registrador de código de condição:

7	6	5	4	3	2	1	0	← número do bit
E	F	H	I	N	Z	V	C	← registrador CC

Note que além dos sinalizadores existem mais 3 bits nesse registrador. Esses bits envolvem o modo como o 6809E trata as interrupções.

O 6809 possui diferentes formas de interromper o processamento normal de um programa. A interrupção pode ser requerida por software, através das instruções SWI, SWI2 ou SWI3, ou por hardware, por intermédio de sinais aplicados a determinados pinos do circuito integrado.

Existem quatro interrupções por hardware: RESET (pino 37); NMI (pino 2); IRQ (pino 3); e FIRQ (pino 4). O sinal de RESET, na realidade, não é bem uma interrupção, mas uma reinicialização. É como se o 6809 recebesse uma ordem para começar tudo de novo. O sinal NMI sempre interrompe o processamento do micro, sendo a interrupção de maior prioridade e não mascarável. Mascaramento é a possibilidade que se tem de bloquear uma interrupção, evitando que o processamento pare.

O sinal FIRQ, por exemplo, é mascarável através do bit F do registrador CC. Quando este bit está em 1, o processamento não pode ser interrompido por aquele sinal.

Já o sinal IRQ é mascarável via bit I, da mesma forma que FIRQ o é por F.

O bit E do registrador CC tem a função de sinalizar quando há uma interrupção. As interrupções NMI, IRQ, SWI, SWI2 ou SWI3 colocam este bit em 1 e empilham todo o conteúdo dos registradores internos do 6809, enquanto que a FIRQ o faz igual a 0 e empilha apenas o contador de programa e o registrador CC. Assim, ao terminar a rotina de interrupção (via instrução RTI), o processador verifica o bit E. Se estiver em 1, o 6809 busca na pilha o conteúdo de todos os seus registradores; caso contrário, apenas os do contador de programa e o do registrador CC.

CONJUNTO DE INSTRUÇÕES

A seguir relacionamos o conjunto completo de instruções do 6809 de duas formas diferentes. A Tabela 22-1 contém os mnemônicos e suas funções, enquanto a Tabela 22-2 relaciona os códigos de instrução, espaço de memória ocupado e tempo de execução.

Tabela 22-1 — Conjunto de instruções do 6809

Instruções de 8 bits com memória e acumulador

Mnemônico	Operação
ADCA, ADCB	Soma memória no acumulador com "vai um"
ADDA, ADDB	Soma memória no acumulador
ANDA, ANDB	Soma lógica (AND) de memória com acumulador
ASL, ASLA, ASLB	Deslocamento aritmético à esquerda
ASR, ASRA, ASRB	Deslocamento aritmético à direita
BITA, BITB	Teste de bit de memória com acumulador
CLR, CLRA, CLRB	Apaga acumulador ou memória
CMPA, CMPB	Compara memória com acumulador
COM, COMA, COMB	Complementa acumulador ou memória
DAA	Ajuste decimal do acumulador A
DEC, DECA, DECB	Decrementa acumulador ou memória
EORA, EORB	OU exclusivo entre memória e acumulador
EXG r1, r2	Troca r1 com r2
INC, INCA, INCB	Incrementa acumulador ou memória
LDA, LDB	Carrega acumulador
LSL, LSLA, LSLB	Deslocamento lógico à esquerda
LSR, LSRA, LSRB	Deslocamento lógico à direita
MUL	Multiplicação sem sinal (A×B→D)
NEG, NEGA, NEGB	Negação de acumulador ou memória
ORA, ORB	OU entre memória e acumulador
ROL, ROLA, ROLB	Rotação à esquerda
ROR, RORA, RORB	Rotação à direita
SBCA, SBCB	Subtrai memória do acumulador com "vem um"
STA, STB	Armazena acumulador na memória
SUBA, SUBB	Subtrai memória do acumulador
TST, TSTA, TSTB	Testa acumulador ou memória
TFR r1, r2	Transfere r1 para r2

Obs.: r1 e r2 representam os registradores A, B, CC, DP, X, Y, S, U.

Instruções de 16 bits com memória e acumulador

Mnemônico	Operação
ADDD	Soma memória no acumulador D
CMPD	Compara memória com o acumulador D
EXG D, r	Troca D com X, Y, S, U ou PC
LDD	Carrega D com a memória
SEX	Extensão do sinal do acumulador B em A
STD	Armazena D na memória
SUBD	Subtrai memória do acumulador D
TFR D, r	Transfere D para X, Y, S, U ou PC
TFR r, D	Transfere X, Y, S, U ou PC para D

Instruções com indicadores/indexadores

Mnemônico	Operação
CMPS, CMPU	Compara memória com indicador de pilha
CMPX, CMPY	Compara memória com registrador de índice (indexador)
EXG r1, r2	Troca D, X, Y, S, U ou PC com D, X, Y, S, U ou PC
LEAS, LEAU	Carrega endereço efetivo no indicador de pilha
LEAX, LEAY	Carrega endereço efetivo no indicador de índice
LDS, LDU	Carrega indicador de pilha com a memória
LDX, LDY	Carrega indicador de índice com a memória
PSHS	Põe A, B, CC, DP, D, X, Y, U ou PC na pilha do sistema
PSHU	Põe A, B, CC, DP, D, X, Y, U ou PC na pilha do usuário
PULS	Retira A, B, CC, DP, D, X, Y, U ou PC da pilha do sistema
PULU	Retira A, B, CC, DP, D, X, Y, U ou PC da pilha do usuário
STS, STU	Armazena indicador de pilha na memória
STX, STY	Armazena indicador de índice na memória
TRF r1, r2	Transfere D, X, Y, S, U ou PC para D, X, Y, S, U ou PC
ABX	Soma acumulador B em X (sem sinal)

Instruções de desvio

Mnemônico	Operação
	Desvio simples
BEQ, LBEQ	Desvio se igual (o prefixo L indica argumento de 16 bits)
BNE, LBNE	Desvio se diferente
BMI, LBMI	Desvio se menos
BPL, LBPL	Desvio se mais
BCS, LBCS	Desvio se "carry" igual a 1
BCC, LBCC	Desvio se "carry" igual a 0
BVS, LBVS	Desvio se "estouro" igual a 1
BVC, LBVC	Desvio se "estouro" igual a 0
	Desvio com sinal
BGT, LBGT	Desvio se maior (com sinal)
BGE, LBGE	Desvio se maior ou igual (com sinal)
BEQ, LBEQ	Desvio se igual
BLE, LBLE	Desvio se menor ou igual (com sinal)
BLT, LBLT	Desvio se menor (com sinal)

Desvio sem sinal

BHI, LBHI	Desvio se maior (sem sinal)
BHS, LBHS	Desvio se maior ou o mesmo (sem sinal)
BEQ, LBEQ	Desvio se igual
BLS, LBLS	Desvio se menor ou o mesmo (sem sinal)
BLO, LBLO	Desvio se menor (sem sinal)

Outros desvios

BSR, LBSR	Desvio para sub-rotina
BRA, LBRA	Desvio incondicional
BRN, LBRN	Desvio nunca (instrução sem efeito)

Instruções diversas

Mnemônico	Operação
ANDCC	Operação lógica E no registrador CC
CWAI	O mesmo que ANDCC, só que fica esperando interrupção
NOP	Sem operação (similar a BRN)
ORCC	Operação lógica OU no registrador CC
JMP	Salto
JSR	Salto para sub-rotina
RTI	Retorna de uma interrupção
RTS	Retorna de uma sub-rotina
SWI, SWI2, SWI3	Interrupção por software
SYNC	Sincroniza com a linha de interrupção

Tabela 22-2 — Códigos de operação, espaço de memória e tempo de execução

Modo de endereçamento	Inerente			Imediato			Direto			Por extenso			Indexado/ Indireto			Relativo			
				dado 8 ou dado 16			end8			end16			veja Tab 22-3			rótulo ou desl.			
Mnemônico	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	Obs.
ABX	3A	3	1																
ADCA				89	2	2	99	4	2	B9	5	3	A9	4+	2+				
ADCB				C9	2	2	D9	4	2	F9	5	3	E9	4+	2+				
ADDA				8B	2	2	9B	4	2	BB	5	3	AB	4+	2+				
ADDB				CB	2	2	DB	4	2	FB	5	3	EB	4+	2+				
ADDD				C3	4	3	D3	6	2	F3	7	3	E3	6+	2+				
ANDA				84	2	2	94	4	2	B4	5	3	A4	4+	2+				
ANDB				C4	2	2	D4	4	2	F4	5	3	E4	4+	2+				
ANDCC				1C	3	2													
ASL							08	6	2	78	7	3	68	6+	2+				
ASLA	48	2	1																
ASLB	58	2	1																
ASR							07	6	2	77	7	3	67	6+	2+				
ASRA	47	2	1																
ASRB	57	2	1																
BCC																24	3	2	
BCS																25	3	2	
BEQ																27	3	2	
BGE																2C	3	2	
BGT																2E	3	2	
BHI																22	3	2	
BHS																24	3	2	
BITA				85	2	2	95	4	2	B5	5	3	A5	4+	2+				
BITB				C5	2	2	D5	4	2	F5	5	3	E5	4+	2+				

Modo de endereçamento	Inerente			Imediato			Direto			Por extenso			Indexado/ Indireto			Relativo			
				dado 8 ou dado 16			end8			end16			veja Tab 22-3			rótulo ou desl.			
Mnemônico	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	Obs.
BLE																2F	3	2	
BLO																25	3	2	
BLS																23	3	2	
BLT																2D	3	2	
BMI																2B	3	2	
BNE																26	3	2	
BPL																2A	3	2	
BRA																20	3	2	
BRN																21	3	2	
BSR																8D	7	2	
BVC																28	3	2	
BVS																29	3	2	
CLR							0F	6	2	7F	7	3	6F	6+	2+				
CLRA	4F	2	1																
CLRB	5F	2	1																
CMPA				81	2	2	91	4	2	B1	5	3	A1	4+	2+				
CMPB				C1	2	2	D1	4	2	F1	5	3	E1	4+	2+				
CMPD				10 83	5	4	10 93	7	3	10 B3	8	4	10 A3	7+	3+				
CMPS				11 8C	5	4	11 9C	7	3	11 BC	8	4	11 AC	7+	3+				
CMPU				11 83	5	4	11 93	7	3	11 B3	8	4	11 A3	7+	3+				
CMPX				8C	4	3	9C	6	2	BC	7	3	AC	6+	2+				
CMPY				10 8C	5	4	10 9C	7	3	10 BC	8	4	10 AC	7+	3+				
COM							03	6	2	73	7.	3	63	6+	2+				
COMA	43	2	1																
COMB	53	2	1																
CWAI				3C	20	2													
DAA	19	2	1																
DEC							0A	6	2	7A	7	3	6A	6+	2+				
DECA	4A	2	1																
DECB	5A	2	1																
EORA				88	2	2	98	4	2	B8	5	3	A8	4+	2+				
EORB				C8	2	2	D8	4	2	F8	5	3	E8	4+	2+				
EXG				1E	8	2													
INC							0C	6	2	7C	7	3	6C	6+	2+				2
INCA	4C	2	1																
INCB	5C	2	1																
JMP							0E	3	2	7E	4	3	6E	3+	2+				
JSR							9D	7	2	BD	8	3	AD	7+	2+				
LBCC																10 24	5(6)	4	1
LBCS																10 25	5(6)	4	1
LBEQ																10 27	5(6)	4	1
LBGE																10 2C	5(6)	4	1
LBGT																10 2E	5(6)	4	1
LBHI																10 22	5(6)	4	1
LBHS																10 24	5(6)	4	1
LBLE																10 2F	5(6)	4	1
LBLO																10 25	5(6)	4	1
LBLS																10 23	5(6)	4	1
LBLT																10 2D	5(6)	4	1
LBM																10 2B	5(6)	4	1
LBNE																10 26	5(6)	4	1
LBPL																10 2A	5(6)	4	1
LBRA																16	5	3	
LBRN																10 21	5	4	
LBSR																17	9	3	
LBVC																10 28	5(6)	4	1
LBVS																10 29	5(6)	4	1
LDA				86	2	2	96	4	2	B6	5	3	A6	4+	2+				
LDB				C6	2	2	D6	4	2	F6	5	3	E6	4+	2+				
LDD				CC	3	3	DC	5	2	FC	6	3	EC	5+	2+				
LDS				10 CE	4	4	10 DE	6	3	10 FE	7	4	10 EE	6+	3+				
LDU				CE	3	3	DE	5	2	FE	6	3	EE	5+	2+				
LDX				8E	3	3	9E	5	2	BE	6	3	AE	5+	2+				
LDY				10 8E	4	4	10 9E	6	3	10 BE	7	4	10 AE	6+	3+				
LEAS													32	4+	2+				
LEAU													33	4+	2+				
LEAX													30	4+	2+				
LEAY													31	4+	2+				
LSL							08	6	2	78	7	3	68	6+	2+				
LSLA	48	2	1																

Modo de endereçamento	Inerente			Imediato			Direto			Por extenso			Indexado/ Indireto			Relativo			
				dado 8 ou dado 16			end8			end16			veja Tab 22-3			rótulo ou desl.			
Mnemônico	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	H	C	B	Obs.
LSLB	58	2	1																
LSR							04	6	2	74	7	3	64	6+	2+				
LSRA	44	2	1																
LSRB	54	2	1																
MUL	3D	11	1																
NEG							00	6	2	70	7	3	60	6+	2+				
NEGA	40	2	1																
NEGB	50	2	1																
NOP	12	2	1																
ORA				8A	2	2	9A	4	2	BA	5	3	AA	4+	2+				
ORB				CA	2	2	DA	4	2	FA	5	3	EA	4+	2+				
ORCC				1A	3	2													
PSHS				34	5+	2													
PSHU				36	5+	2													2, 3
PULS				35	5+	2													2, 3
PULU				37	5+	2													2, 3
ROL							09	6	2	79	7	3	69	6+	2+				2, 3
ROLA	49	2	1																
ROLB	59	2	1																
ROR							06	6	2	76	7	3	66	6+	2+				
RORA	46	2	1																
RORB	56	2	1																
RTI	3B	6/15	1																
RTS	39	5	1																
SBCA				82	2	2	92	4	2	B2	5	3	A2	4+	2+				
SBCB				C2	2	2	D2	4	2	F2	5	3	E2	4+	2+				
SEX	1D	2	1																
STA							97	4	2	B7	5	3	A7	4+	2+				
STB							D7	4	2	F7	5	3	E7	4+	2+				
STD							DD	5	2	FD	6	3	ED	5+	2+				
STS							10 DF	6	3	10 FF	7	4	10 EF	6+	3+				
STU							DF	5	2	FF	6	3	EF	5+	2+				
STX							9F	5	2	BF	6	3	AF	5+	2+				
STY							10 9F	6	3	10 BF	7	4	10 AF	6+	3+				
SUBA				80	2	2	90	4	2	B0	5	3	A0	4+	2+				
SUBB				C0	2	2	D0	4	2	F0	5	3	E0	4+	2+				
SUBD				83	4	3	93	6	2	B3	7	3	A3	6+	2+				
SWI	3F	19	1																
SWI2	10 3F	20	2																
SWI3	11 3F	20	2																
SYNC	13	2	1																
TFR				1F	7	2													
TST							0D	6	2	7D	7	3	6D	6+	2+				2
TSTA	4D	2	1																
TSTB	5D	2	1																

Obs.: Na tabela,
H = código objeto da instrução (em hexadecimal)
C = número de ciclos de máquina para executar a instrução
(um ciclo dura ≅ 1,117 μs)
B = número de bytes para compor a instrução na memória
dado 8 = valor de 8 bits
dado 16 = valor de 16 bits
*1 — O número de ciclos entre parênteses se aplica quando houver desvio
*2 — O dado imediato no código objeto desta instrução é sempre uma especificação de registro embutida
*3 — Uma instrução PSH ou PUL requer 5 ciclos mais um para cada byte manipulado

Tabela 22-3 — Modos de endereçamento indireto e indexado

Tipo	Forma	Não indireto Formato Opcode		+C	+B	Indireto Formato Opcode		+C	+B
			pós-byte				pós-byte		
Deslocamento constante de r	Sem desl.	,r	1rr00100	0	0	[,r]	1rr10100	3	0
	desl. 5 bits	n,r	0rrnnnnn	1	0		Fica c/ 8 bits		
	desl. 8 bits	nn,r	1rr01000	1	1	[nn,r]	1rr11000	4	1
	desl. 16 bits	mmnn,r	1rr01001	4	2	[mmnn],r	1rr11001	7	1
Deslocamento de acumulador	desl. de A	A,r	1rr00110	1	0	[A,r]	1rr10110	4	0
	desl. de B	B,r	1rr00101	1	0	[B,r]	1rr10101	4	0
	desl. de D	D,r	1rr01011	4	0	[D,r]	1rr11011	7	0
Auto decrem./ increm. de r	pós-incr. 1	,r+	1rr00000	2	0		inviável		
	pós-incr. 2	,r++	1rr00001	3	0	[,r++]	1rr10001	6	0
	pós-decr. 1	,—r	1rr00010	2	0		inviável		
	pós-decr. 2	,--r	1rr00011	3	0	[,--r]	1rr10011	6	0
Deslocamento constante de PC	desl. 8 bits	rót,PCR	1xx01100	1	1	[rót,PCR]	1xx11100	4	1
	desl., 16 bits	rót,PCR	1xx01101	5	2	[rót,PCR]	1xx11101	8	2
Indireto por extenso	end. 16 bits					[mmnn]	10011111	5	2

Obs.: r = X, Y, U ou S
xx = não importa
rr: 00 = X, 01 = Y, 10 = U e 11 = S
rót = rótulo
PCR = registrador contador de programa (PC)

Índice remissivo

Impresso em offset

GRÁFICA EDITORA HAMBURG
Avenida Bogaert, 64
Vila das Mercês São Paulo
Fone: 914-0233
CEP 04298

com filmes fornecidos pelo editor

Um livro dedicado à exploração de alguns dos inúmeros recursos desse versátil micro. Numa linguagem clara e direta, são apresentados e discutidos vários aspectos únicos do CP 400, envolvendo recursos de programação, de vídeo, de utilização de fita cassete e disco. Complementando essas informações, a última parte da obra é inteiramente dedicada ao hardware. Aí são explicadas a arquitetura interna do microprocessador 6809E, sua linguagem e os vários meios (vídeo, teclado, interface serial etc.) de que ele dispõe para se comunicar com o mundo exterior.

Uma característica deste livro é que para cada novo recurso discutido é apresentado um programa completo. Ao todo, são 21 programas independentes e modulares, que podem tanto ser executados isoladamente como agrupados para formar novas aplicações.

RECURSOS EXCLUSIVOS

- Detecção e edição automáticas de erro
- Listagem super-rápida
- Caracteres em negativo
- Editor de gráficos
- Deslocamento diferente para animação
- CMERGE: um programa para fusão em fita
- Racionalização de discos
- Backups mais práticos para os discos
- ZAP400: um programa para corrigir erros em disco
- COLOR ASSEMBLER: um EDITOR/ASSEMBLER em BASIC
- COLOR DISASSEMBLER: um interpretador de rotinas em BASIC

SOBRE O AUTOR

Paulo Addair é editor de livros e manuais de informática, já tendo coordenado a publicação de inúmeras obras. Seus trabalhos mais conhecidos são as adaptações para o português dos dois volumes do livro "BASIC Para Crianças" e o texto dos manuais do CP 200 e do CP 500. Mais recentemente, trabalhou na edição do manual do CP 400, juntamente com o pessoal do departamento técnico da Prológica.

EDITELE

EDITORA TÉCNICA ELETRÔNICA LTDA.