TIMEX COMPUTER

2048
MICROCOMPUTADOR



MANUAL DO UTILIZADOR

TIMEX COMPUTER

2048
MICROCOMPUTADOR
MANUAL DO UTILIZADOR

Charles F. Durang
Autor

udith Richland
)esenhador Gráfico

Horácio Mariano
:oordenador Gráfico

1985 de TMX Portugal Lda
1982 de Sinclair Research Limited

Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, seja electrónico, mecânico, de fotocópia, de gravação ou outros, sem prévia autorização por escrito da TMX Portugal, Lda.

Fotocomposição, Montagem e Impressão:
ROLIMPRE — Artes Gráficas, Lda.
ALMADA

Índice dos Assuntos

I — Primeiros passos

Introdução: ***Você e o Timex Computer 2048*** 1

Capítulo 1: ***Como Ligar o Computador*** 5

Capítulo 2: ***Descubra o Teclado*** 11
K L C E G DELETE
CAPS SHIFT CAPS LOCK
SYMBOL SHIFT
TRUE VIDEO
INVERSE VIDEO

Capítulo 3: ***Como Dar Ordens ao Computador*** 27
PRINT ENTER
" " Strings SQR

Capítulo 4: ***Como Usar Programas Pré-gravados*** 37
LOAD RUN SAVE VERIFY
LINE MERGE
REM Mensagens

Capítulo 5: ***Como Usar as Cores*** 55
INK PAPER BORDER

Capítulo 6: ***Como Desenhar Linhas e Círculos*** 59
PLOT DRAW CIRCLE

Capítulo 7: ***O Som*** 65
BEEP

Índice dos Assuntos

II — Programação para Principiantes

Capítulo 8: **Como Escrever Um Programa** 71
NEW Line#s GOTO
scroll? BREAK CONT

Capítulo 9: **Como Organizar o Ecrã** 79
; , AT TAB : ↑↓
EDIT

Capítulo 10: **Poupe Tempo e Espaço Usando Variáveis** 89
LET = VARIÁVEIS
$ Variáveis String
CLS CLEAR

Capítulo 11: **Matemáticas Com o TC 2048** 95
+ – / ** ( )
RND RAND INT

Capítulo 12: **Programas Que Pedem Informação** 103
INPUT STOP
READ DATA RESTORE

Capítulo 13: **Programas Que se Repetem: Ciclos** 113
FOR...TO NEXT STEP
LIST

Capítulo 14: **Programas Que Decidem Ramificação** 121
IF...THEN <<= >
>= <>

Capítulo 15: **Programas Dentro de Programas: Subrotinas** 133
GOSUB RETURN

Capítulo 16: **Matrizes** 141
DIM «Subscrito»
«Corte» String
SAVE DATA

III — Características Especiais do TC 2048

Capítulo 17: **Gráficos** 149
SCREEN$

Capítulo 18: **Gráficos a Definir Pelo Utilizador** 161
BIN POKE USR

Capítulo 19: **Tempo e Movimento** 167
INKEY$ PAUSE

Capítulo 20: **A Cor** 175
BRIGHT FLASH OVER
INVERSE

Capítulo 21: **Como Obter Informações Especiais do TC 2048** 183
POINT ATTR
CODE CHR$

Índice dos Assuntos

Capítulo 22: **Como Usar a Impressora** 189

LPRINT LLIST COPY

Capítulo 23: **«Entradas» e «Saídas» (INPUT/OUTPUT)** 197

PEEK IN OUT OPEN CLOSE
FORMAT ERASE MOVE CAT
RESET

APÊNDICES

Apêndice A: *Revisão do BASIC do TC 2048* 203

Apêndice B: *O Conjunto de Caracteres* 227

Apêndice C: *Modos de Apresentação do Ecrã e Memória* 235

Apêndice D: *As Variáveis do Sistema* 248

Apêndice E: *Como usar o Código Máquina* 254

Apêndice F: *Tabela de «comandos»* 258

Apêndice G: *Índice* 266

Apêndice H: *Lista de Códigos* 274

Apêndice I: *Como utilizar o Joystick* 278

"Comandos" adicionais cobertos nos Apêndices:

PEEK DEF FN FN
STR$ VAL VAL$ LEN
EXP SGN SIN COS
TAN LN PI ASN ATN

Você e o Timex Computer 2048

Síntese do Capítulo

Este Capítulo trata da utilidade de um computador e refere a extraordinária capacidade do seu TC 2048.



Para que serve um computador?
O seu novo computador Timex Computer 2048 é um instrumento muito especial. É uma ferramenta que pode aumentar a capacidade da sua inteligência tal como o desporto o ajuda a desenvolver os seus músculos.
Para o principiante, é fácil de utilizar e de compreender. Para o «expert», é uma máquina extremamente sofisticada e poderosa.

Vamos explorar a comparação que estabelecemos.

Foi desde o princípio dos tempos que os homens inventaram ferramentas para os ajudarem a conseguir os seus fins, seja para reforçar o seu vigor físico, seja para desenvolver as suas capacidades mentais.

Para o homem era difícil recordar-se de quantidades ou números grandes, tendo utilizado métodos como simples cortes em varas, ou seixos dentro de um saco, para contabilizar, por exemplo,

o número de ovelhas do seu rebanho.

Era duro, para nós, manipular os números, pequenos ou grandes, pelo que criámos sistemas escritos matemáticos e máquinas cada vez mais complexas que nos conduziram até às máquinas somadoras ou de calcular.

Também se tornava difícil executar maçadoras operações repetitivas sem produzir erros, pelo que, de novo, inventámos máquinas e sistemas (como a «contabilidade») para nos ajudarem a não nos despistarmos.

O computador é a máquina derradeira ou definitiva para auxiliar o nosso cérebro. Ele pode recordar — e encontrar — vastas quantidades de informação. Pode manipular essa informação muito para além daquilo que se torna possível «nas nossas cabeças». Pode realizar fastidiosas operações repetitivas, vezes sem conta, sem nunca produzir um erro. Pode realizar, para nós, aquilo que nos seria difícil ou maçador executar e com muito maior rapidez.

Tal como se nos torna impossível pregar pregos apenas com as mãos e podemos fazê-lo com o auxílio de um martelo, pela mesma razão recorremos ao computador para conseguirmos realizar aquilo que não podemos fazer sozinhos.

Tal como os homens necessitam de orientar o seu trabalho, assim também se torna imprescindível decidir que tipo de problemas desejamos resolver com o auxílio do computador e que caminho queremos seguir para obter os resultados pretendidos. O computador é um bom trabalhador mas você é o seu chefe.

Algumas das nossas outras ferramentas possuem capacidades para mais do que um único uso — com um martelo de orelhas tanto se pregam como se arrancam pregos — mas o computador tem milhares de aplicações possíveis. É tão

«generalista» como o próprio ser humano e pode realizar toda a classe de espantosas tarefas diferentes... desde que lhe diga como.

Vejamos algumas das coisas que o seu Timex Computer 2048 pode fazer e como poderá orientá-lo para que os seus esforços sejam bem sucedidos.

Aplicações do Seu Timex Computer 2048

O Timex Computer 2048 é uma máquina que pode utilizar para diversas finalidades, de múltiplas formas. Poderá começar a utilizá-lo dentro de poucos minutos se recorrer a programa pré-registados para:

- manter os registos caseiros;
- jogar;
- complementar a educação dos seus filhos;
- assisti-lo no seu trabalho;
- e aprender mais sobre os próprios computadores.

Dissemos que o computador poderia desempenhar muitas tarefas, mas apenas se lhe ensinassem como. Na segunda parte deste Manual mostrar-lhe-emos como poderá escrever os seus próprios programas — listas de instruções para o computador. E daí partiremos para mais longe, para programação mais avançada; com este e com outros livros poderá vir a tornar-se tão perito na matéria quanto possa desejar. O Timex Computer 2048 pode acompanhá-lo, tão longe quanto decida, nesta viagem: Nada vulgar num computador doméstico, ele pode armazenar até 48 ***kilobytes*** aproximadamente — 48 mil caracteres — de informação e possui capacidades avançadas no domínio dos gráficos e da cor, incluindo o «modo» de duplo ecrã para animação, o «modo» de largura total e «modo» de extensão em cor.

Uma das características do TC 2048 é poder gravar os seus próprios programas de forma a reproduzi-los quantas vezes o desejar, sempre que necessite de recorrer a eles para a realização duma determinada tarefa. De onde se pode inferir que também os programas escritos por outros programadores poderão, identicamente, ser utilizados no seu próprio computador. Poderá introduzir um programa específico no seu computador para que ele realize a tarefa que lhe interessa em determinado momento: desde os exercícios educacionais para as crianças, os jogos de «aventuras», a memorização ou arquivo de receitas e respectivas quantidades ou pesos. Pode até ser um precioso auxílio quanto à maneira como deverá realizar os seus registos para o pagamento de impostos... sem que, para tudo isso, tenha de aprender a programar!

Temos a esperança de que eventualmente venha a escrever os seus próprios programas — é um bom exercício mental, mesmo na fase inicial — mas, antes de mais, vamos dizer-lhe como poderá usar o seu TC 2048, imediatamene, com programas pré-gravados que existem disponíveis em largas quantidades.

A PARTE 1 deste Manual informá-lo-á sobre a ligação do computador e sobre o seu teclado. Também o ensinará a utilizar programas pré--gravados. Na PARTE 2 estudaremos as bases da programação e na PARTE 3 algumas das características especiais do TC 2048.

Os APÊNDICES incluem informações úteis a que poderá recorrer em qualquer momento e também contêm algum material destinado a técnicos conhecedores de computadores que pretendam obter detalhes relacionados com o funcionamento interno do Timex Computer 2048.

Os nossos parabéns por ter aderido à Era do Computador! Certamente que achará nisso bastante satisfação, utilidade e um meio educacional perfeito.

Como Ligar o Computador 1

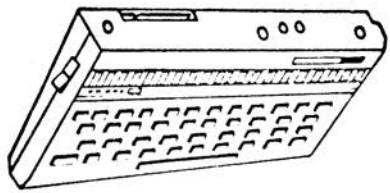
Síntese do Capítulo

Este capítulo mostra-lhe como ligar o TC 2048 ao seu Televisor e Gravador de «Cassettes» e como começar a utilizá-lo imediatamente.



Na caixa encontra-se tudo o que necessita para começar a utilizar imediatamente o seu Timex Computer 2048. Basta-lhe o seu próprio televisor (a cores, de preferência) e um gravador de «cassettes».

Eis aquilo que deve possuir:



O Computador TC 2048
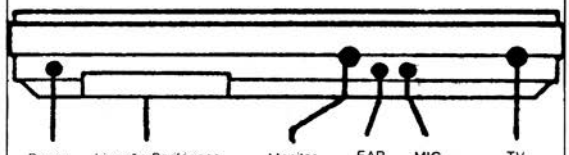
Capítulo 1: Como Ligar o Computador

1. O próprio Timex Computer 2048. Se bem que extremamente leve é tão poderoso como aqueles computadores que, ainda não haverá dez anos, ocupavam completamente uma grande sala.

Na parte de cima do computador está o teclado. Na parte de trás encontram-se diversas tomadas referenciadas com as indicações TV, POWER, MIC, EAR e MONITOR, assim como uma fenda maior rectangular na qual poderão ligar-se diversos periféricos, entre os quais a Impressora Timex Printer 2040, Microdrives, Sistema Floppy Disc Drive Timex, etc.



ON/OFF
Lado direito do computador



Parte de trás do computador



Ligação Joystick
Lado esquerdo do computador

No lado direito encontrará um interruptor para ligar ou desligar o computador. Do lado esquerdo existe ainda uma tomada para ligar um «joystick».



Cabo TV

2. Um cabo para ligar ao televisor.

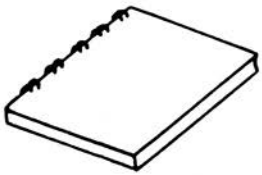


Cabo audio de fichas duplas

3. Um cabo com fichas duplas, mais curto e destinado a ligar o gravador de «cassettes» ao computador.



4. Um transformador de alimentação, com uma ficha para ligar à tomada de corrente eléctrica e outra para introduzir na tomada POWER do computador.



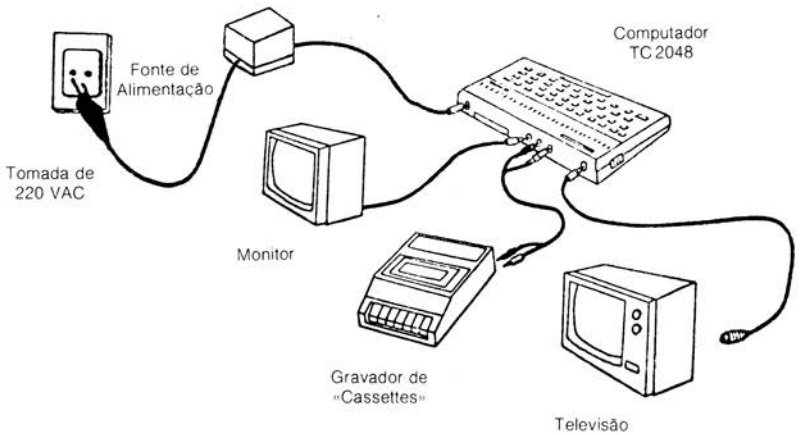
Manual

5. Este Manual.

6. Algum «Software» — (programas gravados em «cassette») — para o ajudar na sua iniciação.

Eis como poderá ligar rapidamente o seu TC 2048 (volte a página para ver os detalhes):

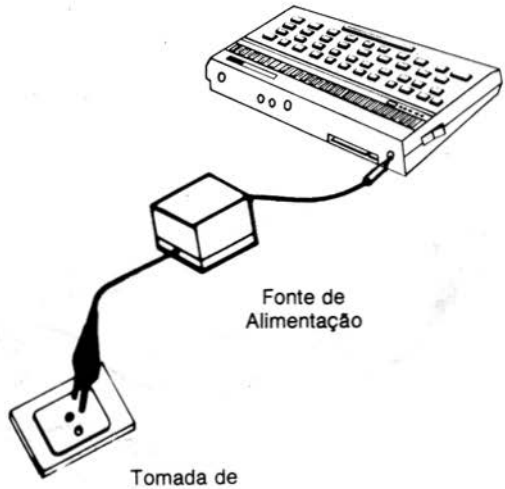
A imagem mostra-lhe como é possível utilizar, com o TC 2048, quer o seu televisor quer um Monitor Especial se bem que baste apenas um televisor (UHF - canal 36) para que o computador funcione.



Primeiro: Desligue a antena de Televisão do seu aparelho.

Introduza, nessa mesma tomada de antena do seu televisor, o cabo que foi referido em 2.
A ficha que fica livre, na outra ponta do cabo, deve ser ligada à tomada TV, na parte de trás do computador.

Segundo: Ligue o transformador de alimentação à corrente. No transformador existe um segundo cabo que termina com uma ficha metálica estreita. Deve ligá-la à tomada POWER que se encontra na parte de trás do computador.

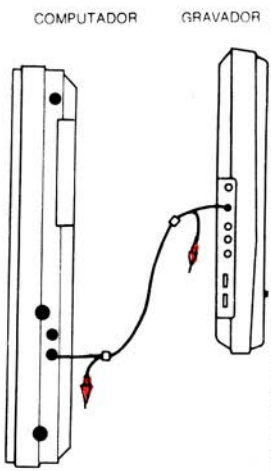


Tomada 220 VAC + Fonte de alimentação

Capítulo 1: Como Ligar o Computador



Terceiro: Ligue o seu televisor. Sintonize-o para o segundo canal (UHF) e ajuste-o para uma frequência um pouco mais elevada do que aquela em que costuma receber as imagens da Televisão. Baixe completamente o volume de som do televisor.
Agora ligue o computador através do interruptor situado no lado esquerdo.
Deve aparecer uma mensagem no fundo do ecrã, indicando o «copyright» do sistema, o que significa que o computador está pronto para trabalhar.
Se a imagem não estiver suficientemente nítida sintonize melhor o seu televisor.



Ligar EAR com EAR
Ligar MIC com MIC

Quarto: Ligue o seu gravador de «cassettes» ao computador, com o cabo de audio de fichas duplas. Ligue a tomada de auscultador do seu gravador à tomada EAR do computador para que este fique apto a receber programas gravados nas «cassettes» ou a tomada de microfone do gravador à tomada MIC do computador para que este possa gravar programas modificados ou produzidos por si.
Verifique cuidadosamente se as fichas que ligam EAR com EAR ou MIC com MIC têm as cores correspondentes.
Àcerca deste assunto voltamos a falar no Capítulo 4.

NOTA: A imagem do seu televisor deverá ser clara; se notar interferências experimente, na devida ordem, o seguinte:

1. Ajuste melhor o botão de sintonia do televisor. (Se ele dispuser de sintonia automática desligue-a). Seguidamente ajuste o controlo de brilho e de contraste e, eventualmente, o oscilador horizontal (por vezes este controlo situa-se na parte de trás do televisor).
2. Verifique bem se, de facto, o canal de sintonia no seu televisor está correcto. Como dissemos deve estar em UHF (um pouco acima do local onde normalmente «apanha» o segundo canal de TV).
3. Desloque o computador, afastando-o do televisor ou, se lhe for possível, coloque-o a um nível mais baixo.

4. Tente ligar o computador a uma tomada de corrente diferente daquela a que está ligado o televisor. Por vezes sucede que as tomadas instaladas em paredes opostas estão ligados a circuitos independentes, o que será favorável.
5. Pode ser vantajoso utilizar um cabo de ligação (blindado) maior, entre o televisor e o computador, de maneira a poder afastá-los ainda mais.
6. Consulte um técnico de reparações de televisores. É possível que o seu aparelho necessite de algum ajuste.

Agora está em condições de utilizar o seu Timex Computer 2048.

Descubra o teclado 2

Sintese do Capítulo

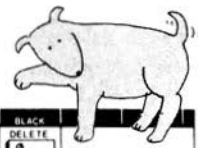
De como os cursores K, L, C, E, G, tal como as teclas CAPS SHIFT e SYMBOL SHIFT lhe permitem ter acesso a todas as funções de todas as teclas.

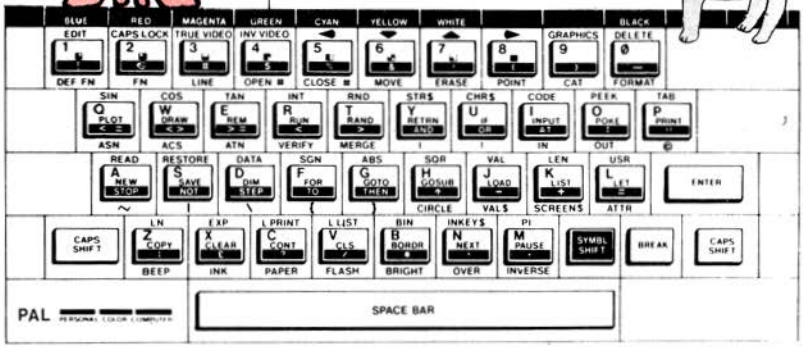
Investigue DELETE, CAPS LOCK, TRUE VIDEO e INVERSE VIDEO e aprenda também como funcionam as teclas das setas para a direita e para a esquerda.

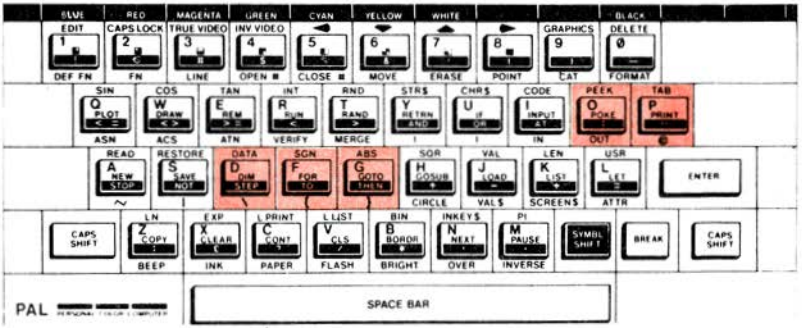


O teclado

Colocará o Timex Computer 2048 a trabalhar através de ligeiras pressões exercidas em cada uma das teclas do teclado. Note bem que não dissemos «carregar nas teclas» como se estivesse a escrever à máquina. É muito mais fácil do que isso, como veremos.







À primeira vista o teclado poderá parecer extremamente complexo. Cada uma das teclas contém cinco ou seis indicações. Mas verificará, muito rapidamente, como é fácil utilizá-lo.

Se sabe escrever à máquina notará que as indicações mais destacadas, em cada tecla — letras e números na maior parte dos casos — estão colocadas como numa máquina de escrever.

A novidade, tanto para os dactilógrafos como para quem não sabe escrever à máquina, é que não terá de escrever (dactilografando-os na totalidade) os comandos ou palavras de ordem para o computador executar. Em vez disso verificará que, nessas teclas, existem já tais «comandos».

Em muitos casos, os «comandos» estão impressos na própria tecla (ou muito perto) da letra pela qual a palavra começa, podendo estar também impressos por cima.

Notará, por hipótese, que a palavra **PRINT** está na tecla da letra P e que **POKE**, tal como **PEEK** estão situadas sobre e por cima, respectivamente, da tecla que fica do lado esquerdo.

Dê, também, uma olhadela às teclas D, F e G e repare nas palavras que elas têm escritas ou que estão por cima delas.

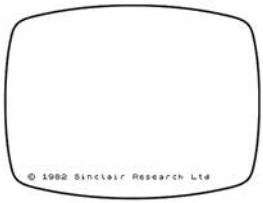
Todas as palavras e símbolos impressos em cada tecla, ou à sua volta, significam obviamente,

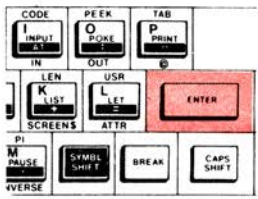
que cada tecla pode perfazer mais do que uma função, quando pretender dar instruções ao computador.

O que a tecla «significa» para o computador, quando a premir, depende de duas coisas:

1. Do «cursor» que está no ecrã do Televisor (já vamos ver o que é um cursor, dentro de momentos);
2. De estar ou não a premir uma das teclas **CAPS SHIFT** ou **SYMBOL SHIFT**, enquanto toca noutra qualquer tecla.

Pratiquemos um pouco com o teclado para ver como é fácil conseguir, de cada tecla, todas as suas funções. Mas temos de pedir-lhe que faça exactamente aquilo que vamos dizer. Depois disso poderá então experimentar sózinho.

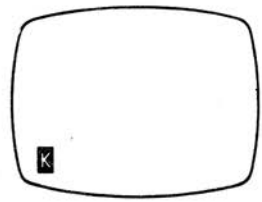




O Cursor K... Palavras nas Teclas

Ligue o seu computador e o televisor tal como vimos no Capítulo antecedente. Deve obter a mensagem de «copyright» na parte inferior do ecrã.

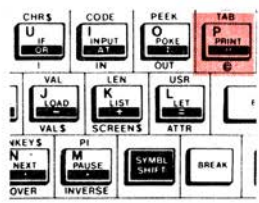
Toque agora na tecla **ENTER**.



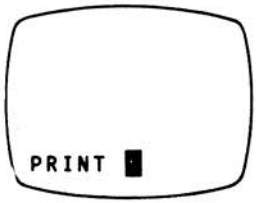
Em vez da mensagem de «copyright» — que desapareceu — surgiu um **K** a piscar, no canto inferior esquerdo do ecrã. (Imprime-se, na realidade, alternadamente, um K preto sobre um fundo branco ou um K branco sobre fundo preto).

Ajuste, se necessário, a sintonia do seu televisor para que obtenha a maior nitidez e recorte possível.

Vamos utilizar, neste exemplo, as teclas de que mais terá de servir-se no seu Timex Computer 2048. Comece pela tecla P, situada no fim da segunda fila de teclas, quase no canto superior direito do teclado.



Se o cursor que está no ecrã é o **K** toque na letra P, com uma ligeira pressão, deixando-a logo a seguir.



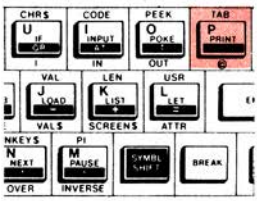
Aconteceram duas coisas: Apareceu a palavra **PRINT** escrita no canto inferior esquerdo do ecrã e o cursor mudou para **L** intermitente.

K significa palavra da tecla (KEYWORD). Porque em português a tradução do vocábulo inglês «Key» tanto pode ser tecla como chave. Com a introdução de microcomputadores em Portugal e a sua vulgarização, alguns tradutores adoptaram «palavra-chave» em vez de «palavra da tecla» e assim ficou. Contudo nós iremos utilizar a palavra «comando».



Portanto quando o cursor **K** está no ecrã, a tecla premida fará surgir o seu «comando» (tal como **PRINT**, **POKE**, **INPUT**, etc.).

O cursor também se deslocou para um ponto à frente da palavra **PRINT**. O cursor marca a posição, no ecrã, onde se imprimirá seja o que for que venha a seguir; mas veremos, também, como é possível movimentar o cursor de forma a poder colocar o texto onde o desejarmos ou, até, como chegar ao local onde quisermos fazer eventuais alterações

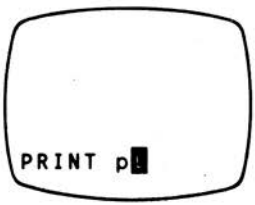


Premir e largar a tecla P

O Cursor L ... os Caracteres principais em cada Tecla.

Agora «tecle» novamente o P premindo e abandonando logo a tecla.

Obterá a impressão de um p minúsculo e o ecrã apresentar-se-á como pode ver na figura.



Deve ter notado que estamos a recomendar que, depois de premir, largue logo a tecla. É que o TC 2048 dispõe de um sistema de repetição automática. Se premir uma tecla durante um segundo, ou um pouco mais, o computador repetirá a impressão do caracter dessa mesma tecla e não deixará de o fazer enquanto não a abondonar.

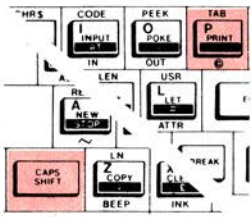
Lembre-se sempre disto: deve premir a tecla ligeiramente e abandoná-la logo a seguir, se, efectivamente, apenas pretende um caracter ou um «comando».



«Tecle» mais algumas letras e alguns números.

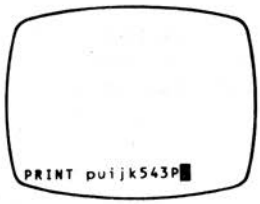


L significa LETRA. Quando o cursor L está no ecrã a pressão de uma tecla origina a impressão do carácter principal que está destacado na tecla... (letra logo acima do «comando» ou número, caso se trate de tecla numérica).



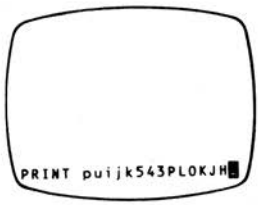
Manter premido **CAPS SHIFT**, Premir a tecla P.

O CAPS SHIFT... Letras Maiúsculas.
Observe que na parte inferior do teclado, na última fila de teclas, respectivamente à esquerda e à direita, estão duas teclas iguais, ambas identificadas por **CAPS SHIFT**. Mantendo uma dessas teclas premidas, toque outra vez na tecla P.



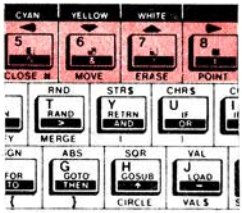
Obteve um P maiúsculo, não foi?

Experimente fazer o mesmo com outra teclas de letras. Por ora, não tente nenhuma experiência com teclas numéricas.



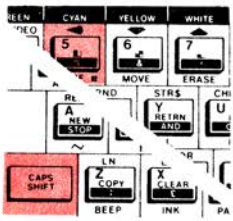
Vamos agora às teclas numéricas. As palavras e os símbolos escritos por cima das teclas numéricas (mas por baixo do nomes das cores) obtêm-se através da pressão simultânea de uma tecla **CAPS SHIFT** e da respectiva tecla numérica. Por exemplo:

Capítulo 2: Descubra o Teclado

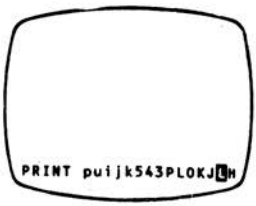


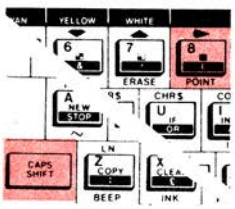
Prima CAPS SHIFT e as «teclas» 5, 6, 7 ou 8 para as setas.

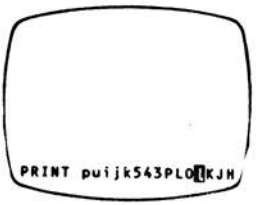
Comece com as teclas das setas: 5, 6, 7 e 8. Mantendo premida a tecla **CAPS SHIFT**, toque na tecla 5. Repare que o cursor se movimenta para a esquerda do ecrã, aparecendo no meio das letras. Experimente ainda mais algumas vezes. Depois mantenha premidas, ao mesmo tempo, as teclas **CAPS SHIFT** e 5 sem largar nenhumas delas.



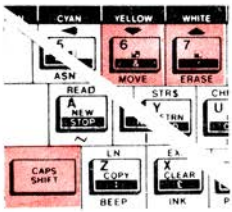
Experimente também isto: Mantenha **CAPS SHIFT** premida e toque na tecla 5 mantendo-a também premida. Agora, mantendo a tecla 5 premida largue a tecla **CAPS SHIFT**.

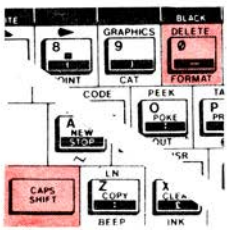






Proceda da mesma maneira em relação à tecla 8, fazendo deslocar o cursor para o lado direito. Depois mova o cursor, diversas vezes, para a direita ou para a esquerda.





Mantenha premida CAPS SHIFT, permir a tecla 0

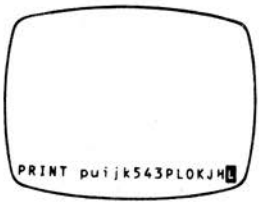
Agora experimente as teclas 6 e 7.

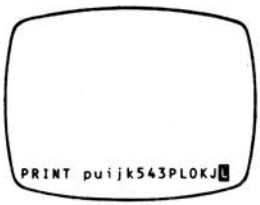
O ecrã piscará, mas o cursor não se desloca. Essas duas teclas são apenas utilizadas para originar movimentos entre as linhas, através de programação BASIC, mas isso será discutido mais tarde.

A tecla DELETE.

Coloque o cursor no fim da linha impressa no fundo do ecrã, utilizando CAPS SHIFT e a tecla 8 A seguir, mantendo premida a tecla CAPS SHIFT toque na tecla ZERO. (Não confunda com o O. O Zero «0» está a seguir ao 9; é a última tecla da primeira fila).

Pode assim fazer o apagamento de caracter a caracter, a começar do fim para o princípio! Se mantiver a tecla zero premida poderá apagar uma série de caracteres seguidos. (Note que os «comandos» são apagados só com uma pressão, tal como foram também impressos).

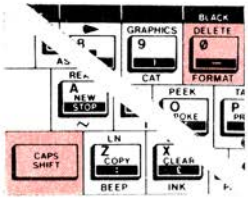




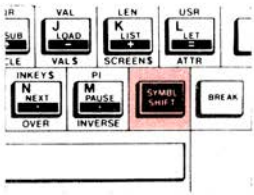
NOTA: Por agora note que, se utilizar a característica de repetição automática (mantendo a tecla premida) para apagar uma linha completa de texto, pode, eventualmente, regressar, ao cursor K.

Isto é muito importante. Agora já aprendeu como «apagar» alguma coisa que tenha no ecrã e não pretenda.

Experimente utilizar as teclas 5 e 8 para deslocar o cursor até uma qualquer letra específica no meio da linha e, então, apague-a. Lembre-se que o que se apaga é o caracter que fica à esquerda do cursor.



É agora uma boa altura para praticar sobre o dispositivo de auto-repetição. Mantenha uma tecla premida até que obtenha diversas impressões desse caracter no ecrã.
Seguidamente, mantendo o CAPS SHIFT e o DELETE premidos faça desaparecer todos os caracteres que imprimiu.



A tecla Symbol SHIFT

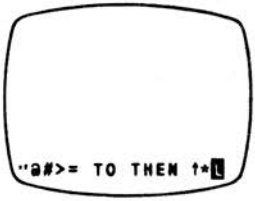
SYMBOL SHIFT... Palavra e Símbolos impressos nas teclas a branco sobre tarjas negras

Localizada quase ao fim da última fila de teclas (antepenúltima) do lado direito (que termina com CAPS SHIFT) está a tecla SYMBOL SHIFT. Na realidade está apenas identificada por uma abreviatura SYMBOL SHIFT, escrita a letras brancas sobre um fundo negro.

É capaz de adivinhar o que acontece no ecrã, quando premir a tecla SYMBOL SHIFT e ao mesmo tempo uma outra tecla?

A banda ou tarja negra é a chave do problema. Obterá, de facto,a palavra ou o símbolo que se encontra escrito em branco, sobre fundo negro, na respectiva tecla que premir.

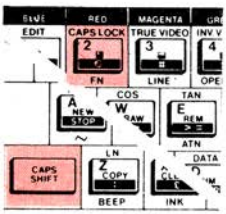
Experimente com algumas das teclas e depois apague tudo servindo-se da tecla DELETE.



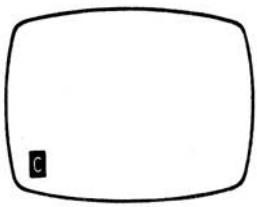
CAPS LOCK... o Cursor C

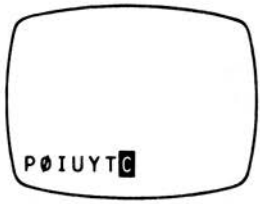
Mantendo a tecla CAPS SHIFT premida carregue na tecla 2. Repare que o cursor se alterou para um C intermitente. «Tecle», agora, algumas letras e alguns números.

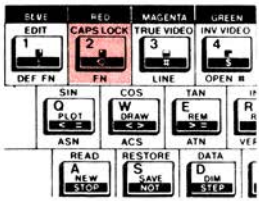
O CAPS LOCK é um dispositivo que comanda a impressão de letras maiúsculas, tal como acontece numa máquina de escrever, mas permite-lhe que continue a usar também as teclas dos números.



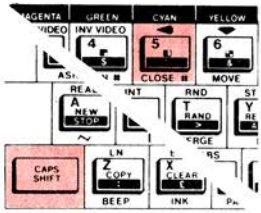
Premir CAPS SHIFT e 2



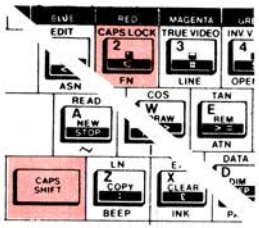




O comando CAPS LOCK imprime letras maiúsculas assim como os números



Premir CAPS SHIFT e simultaneamente as teclas 5 e 8



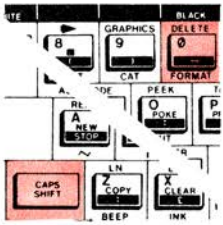
Poderá ainda ter acesso às palavras e símbolos que estão impressos acima das teclas dos números, se usar a tecla CAPS SHIFT. Experimente manter a tecla CAPS SHIFT premida e, ao mesmo tempo, premir também a tecla 5 ou 8. (Por agora não experimente com as teclas 1, 3, 4 ou 9 com a CAPS SHIFT premida. Voltaremos a elas brevemente.)

Experimente agora premir a tecla CAPS SHIFT e a seguir a tecla 2.

Repare que o cursor **L** regressou ao ecrã. Torne a premir a tecla 2 (ao mesmo tempo que CAPS SHIFT, claro). Observe que, alternadamente, passa de minúsculas para maiúsculas, como um interruptor que liga e desliga a tecla de maiúsculas ou CAPS SHIFT. (Ao contrário do que acontece numa máquina de escrever não basta premir apenas o CAPS SHIFT para soltar o CAPS LOCK).

Todas as letras que imprimir enquanto o cursor **C** estiver no ecrã serão maiúsculas.

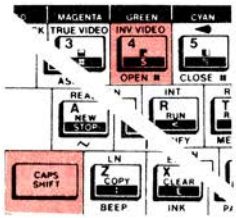
Provavelmente, nesta altura, é capaz de ter uma longa linha de caracteres, numa miscelânea sem nexo, impressa no fundo do ecrã. Pode ser que até tenha duas em vez de uma só linha. Pois bem, apague-os com DELETE.



Lembre-se que a tecla Ø apaga o caracter à esquerda do cursor

LEMBRE-SE: CAPS SHIFT e, ao mesmo tempo, a tecla Ø apagam o caracter ou o «comando» que se encontre à esquerda do cursor. CAPS SHIFT e, ao mesmo tempo, tecla 5 ou tecla 8 deslocam o cursor respectivamente para esquerda ou para a direita.

Agora vamos voltar ao cursor **K**. Faça mais algumas experiências se lhe apetecer. Poderá utilizar, sempre que o queira, DELETE para voltar ao princípio.

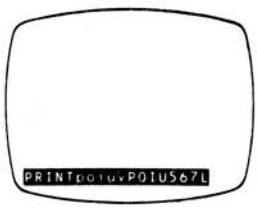


Premir CAPS SHIFT e a tecla 4

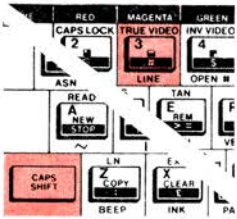
INVERSE VIDEO e TRUE VIDEO

Existem outras palavras escritas por cima das teclas da primeira fila do teclado que desejamos ainda investigar.

Mantenha premida a tecla CAPS SHIFT e toque na tecla 4, para conseguir a posição INV VIDEO (o cursor não se modificará). Depois tecle alguns caracteres.



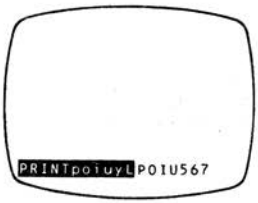
Este é o «modo» INVERSE VIDEO. Os caracteres são impressos em branco sobre fundo negro. Infelizmente não existe nenhum cursor para lhe indicar que estão neste «modo», se se esquecer disso. É óbvio, contudo, que poderá apagar, com DELETE, os caracteres que não desejar. Verifique aquilo que acontece com CAPS LOCK ligado ou desligado.



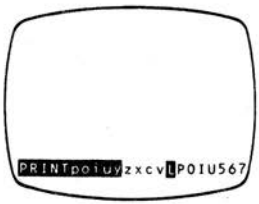
Premir **CAPS SHIFT** e a tecla 3 para **TRUE VIDEO**

Para poder sair do «modo» **INVERSE VIDEO** e regressar ao «modo» normal deve premir-se a tecla **CAPS SHIFT** ao mesmo tempo que a tecla 3: **TRUE VIDEO.**

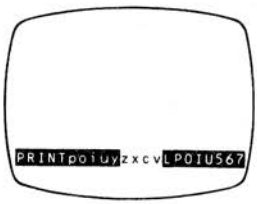
Algo de engraçado acontece quando posicionar o cursor ao meio de uma linha com caracteres impressos no «modo» **INVERSE VIDEO** e premir **CAPS SHIFT** e tecla 3, passando ao «modo» **TRUE VIDEO**: O resto da linha modifica-se! Se, contudo, voltar ao «modo» **INVERSE VIDEO** tudo regressa ao mesmo novamente.



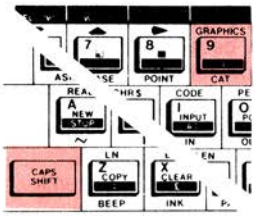
Para inserir caracteres em **TRUE VIDEO**, no meio de uma linha com caracteres **INVERSE VIDEO**, mova o cursor até ao local onde pretende fazer a inserção e passe então ao «modo» **TRUE VIDEO.**



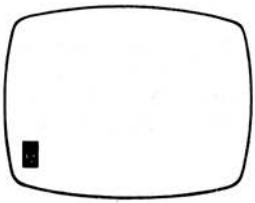
«Tecle» um a um os caracteres, passando depois novamente ao «modo» **INVERSE VIDEO** para que o resto da linha fique impressa a branco sobre negro.

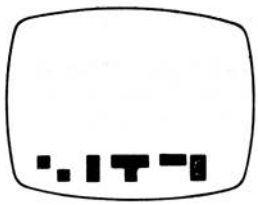


Naturalmente que terão de seguir-se operações idênticas quando se pretenda fazer a inserção de caracteres **INVERSE** no meio de uma sequência de caracteres em **TRUE.**



Premir CAPS SHIFT e 9 para «modo» gráfico







O cursor G... modo gráfico

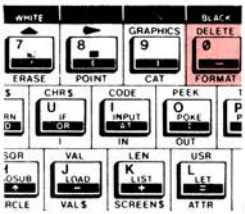
Na fila de teclas numéricas, no topo do teclado, encontram-se inscritos diversos símbolos gráficos. Para que possa imprimi-los no ecrã terá de passar ao «modo» gráfico. Mantendo premida a tecla CAPS SHIFT toque na tecla 9.

Repare que o cursor muda para um G intermitente.

Agora, se premir qualquer tecla onde exista um símbolo gráfico (teclas dos números 1 a 8) verá surgir esse símbolo no ecrã. Experimente com algumas delas.

NOTA: O símbolo que obterá será idêntico ao representado pela parte cinzenta (cor de tecla) e não pela parte preta que está desenhada na tecla. Poderá parecer estranho que a parte preta do desenho da tecla não corresponde àquela que o ecrã representa. Mais tarde, contudo, aperceber-se-á de que nem sempre os gráficos se apresentam a preto e branco mas também podem inscrever-se em diversas cores (INK). E nessa altura notará que a parte da tecla (do quadrado) que tem a cor de base corresponderá à cor determinada pela impressão da cor (INK) no ecrã.

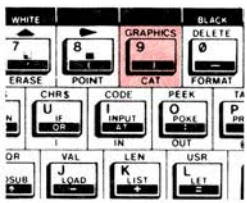
Faça algumas experiências.
É possível conseguir 16 gráficos diferentes a partir das oito teclas. Exactamente um gráfico com o inverso da imagem representada na tecla.
Para isso bastará premir CAPS SHIFT ou SYMBOL SHIFT (indiferentemente) enquanto se prime ao mesmo tempo a tecla respeitante ao gráfico pretendido. Notará mais facilmente a diferença se premir a tecla 3, primeiro sem SHIFT e seguidamente com SHIFT. (Se se encontrar no «modo» INVERSE VIDEO, uma tecla sem SHIFT dar-lhe-á a imagem correspondente à parte negra do quadrado).
As três filas de teclas mais abaixo — que não têm símbolos gráficos — dar-lhe-ão letras maísculas entre o A e o U e uma curiosa mistura de símbolos diversos a partir da letra V até Z.
Nada disto terá, por agora, muita importância, mas, mais tarde, no Capítulo 18, aprenderemos a utilizar as teclas das letras A a U para realizar os nossos próprios símbolos gráficos!



Experimente a tecla DELETE e verifique que não necessita ter a tecla SHIFT premida para fazer apagamentos no «modo» gráfico.

Para sair do «modo» gráfico e voltar ao cursor **L** torne a usar a tecla 9.

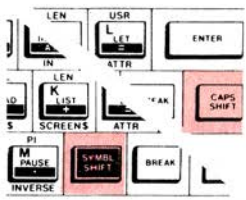
(Para reentrar em «modo» gráfico será necessário premir a tecla 9 enquanto se mantém premida a tecla CAPS SHIFT).



Premir a tecla 9 para abandonar o «modo» gráfico

A propósito, é importante notar que não poderá sair de INVERSE VIDEO para TRUE VIDEO (ou vice-versa) enquanto estiver no «modo» gráfico. Para poder alterar o «modo» VIDEO terá primeiro de abandonar o «modo» gráfico.

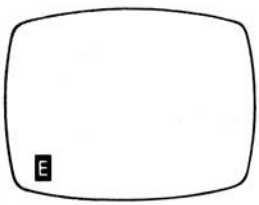
Enquanto o cursor **G** estiver no ecrã, poderá obter todos os gráficos inscritos nas teclas 1 a 8.



Prima CAPS SHIFT e SYMBOL SHIFT para «modo» extensivo

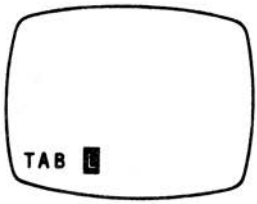
O cursor ... as palavras por cima das teclas.

Premir ao mesmo tempo as teclas de CAPS SHIFT e SYMBOL SHIFT fará mudar o cursor para um **E** intermitente. Significa que o computador fica no «modo» extensivo. Quando o cursor **E** estiver no ecrã, obter-se-ão as palavras que se encontram escritas não nas teclas mas acima delas.



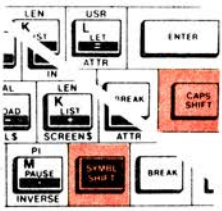
«modo» extensivo





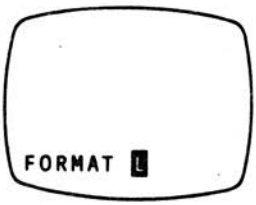
Dado que estes comandos são funções matemáticas, o E volta a mudar para L depois de se ter premido uma tecla. Tudo se processa de forma idêntica à do cursor K. Ora experimente!

A fila superior do teclado é um caso especial. Como pudemos ver, as palavras logo acima das teclas da primeira fila obtêm-se com o cursor L utilizando o CAPS SHIFT. O «modo» extensivo utiliza-se para seleccionar cores. É um método que funciona nas programações, como veremos mais tarde. Mas, se se entretiver com as teclas TRUE VIDEO, INVERSE VIDEO, CAPS e SYMBOL SHIFT, assim como com as teclas das diversas cores, aperceber-se-á de que o resultado obtido no «modo» imediato é relativamente insatisfatório e algumas vezes ilegível. Não se preocupe com isso agora.



SHIFT e cursor E ... palavras e símbolos abaixo das teclas

Coloque o computador no «modo» extensivo, premindo ao mesmo tempo as teclas CAPS SHIFT e SYMBOL SHIFT. Depois largue a tecla CAPS e mantenha premida a SYMBOL SHIFT enquanto toca noutra tecla. Poderá verificar que obteve a função ou a palavra de comando que está escrita abaixo da tecla em causa. Logo a seguir o cursor voltará a L.



(Muitas vezes poderá usar CAPS SHIFT em vez de SYMBOL SHIFT para conseguir as funções que estão escritas abaixo das teclas. Contudo, neste Manual, apenas nos referiremos ao SYMBOL SHIFT.)
A propósito, quando por equívoco tiver entrado no «modo» extensivo e deseje sair desse «modo» sem ter de tocar noutras teclas, basta que volte a premir, ao mesmo tempo, ambas as teclas CAPS SHIFT e SYMBOL SHIFT.

Quando o cursor E se encontrar no ecrã obterá as palavras que estão escritas acima das teclas ou, premindo simultaneamente a tecla de SYMBOL SHIFT, as palavras que estão abaixo delas.

Nesta altura já deve sentir-se relativamente à vontade quanto ao teclado. Existe uma

quantidade de comandos e de caracteres disponíveis nele mas, na maior parte das vezes, encontrar-se-á no modo **L**.



E Cursor
L Cursor (Y)
C Cursor ou **L** Cursor e CAPS SHIFT (Y)
K Cursor
K **L** ou **C** Cursor e SYMBOL SHIFT
E Cursor e SYMBOL SHIFT



Eis um diagrama que mostra claramente a que tipo de cursor e de teclas SHIFT se deve recorrer para conseguir obter os diversos símbolos produzidos por cada tecla.

Não nos referimos ainda a todos os símbolos e comandos do teclado. Alguns deles serão já seus conhecidos, das matemáticas, e outros conhecê-los-á eventualmente se já programou outros computadores em BASIC. Outros ainda são utilizados em relação a funções específicas do Timex Computer 2048 e dos periféricos que lhe pode ligar.

Repita este Capítulo sempre que lhe apetecer ou, se preferir, use o programa de treino sobre o teclado do TC 2048 que lhe entregámos.

Está no bom caminho!

Nota: O Apêndice F, Tabela de «comandos», pode ser-lhe muito útil quando estiver a programar (se quiser, dê-lhe agora uma «espreitadela»). É um índice alfabético de cada palavra ou função, com a indicação precisa de como se podem obter a partir do teclado do TC 2048.

A Tabela do Teclado pode ser-lhe útil se:

1. Pretender determinado «comando» e não souber localizá-lo no teclado.
2. Não estiver absolutamente seguro se determinada palavra, num programa, é ou não um «comando».
3. Tiver um problema na introdução de uma linha de programação, (o computador imprimiu um sinal de erro de sintaxe) ou durante a execução de um problema (o computador parou e imprimiu uma mensagem de erro relativa a determinada linha).
   Pode ser que tenha escrito uma palavra, letra por letra, quando deveria ter introduzido tal palavra como «comando».

É extremamente fácil cometer este erro com as palavras **AT** e **TO**. A Tabela de «comandos» ajudará imensamente a reconhecer e localizar o «comando».

Como Dar Ordens ao Computador 3

Síntese do Capítulo

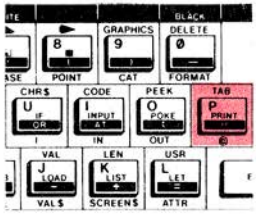
PRINT e ENTER ajudarão a dar as primeiras ordens ao computador TC 2048. Aprenda a forma de fazer imprimir letras, números ou palavras no ecrã. Falaremos de sinais gráficos, de «strings» e de como se utilizam funções como SQR.



Tal como dissemos antes, o computador é uma máquina versátil. Pode fazer inúmeras coisas desde que se lhe «diga» o que deve fazer. Mas tem de se lhe «dizer» isso em palavras que ele entenda e em passos curtos que ele possa executar.
Agora já pode utilizar o conhecimento do teclado para começar a dar instruções ao seu computador.
O Timex Computer 2048 foi construído para compreender ordens que lhe sejam transmitidas na linguagem BASIC (o que significa «Beginner's All-Purpose Symbolic Instruction Code»).
Desenvolvida no «Dartmouth College», a linguagem BASIC parece-se mais com o Inglês do que com outras linguagens de computadores e é extremamente fácil de utilizar.

Os «comandos», impressos nas teclas, são em Inglês.
Também são em BASIC, o que significa que cada um deles corresponde exactamente ao sentido da palavra.

Capítulo 3: Como Dar Ordens ao Computador



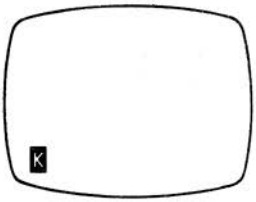
Por exemplo, o «comando» que provavelmente mais utilizará será **PRINT**. Em Inglês **PRINT** significa:

1. Escrever letras num papel com o auxílio de um lápis, caneta ou outro instrumento que escreva e mantenha na sua mão.
2. Transferir letras de uma imprensa para o papel
3. Publicar.

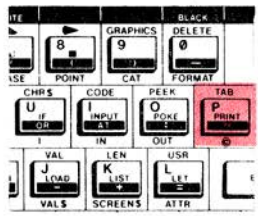
Existirão provavelmente outros significados ligeiramente diferentes destes.
Mas em BASIC a palavra **PRINT** tem um único significado:

1. Imprimir, «no ecrã», seja o que for que se siga à palavra **PRINT**.

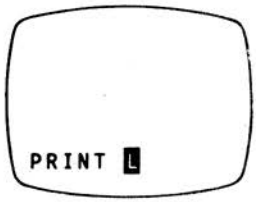
Pratiquemos um pouco mais, desta vez dando ao computador algumas ordens e observando como ele as cumpre.



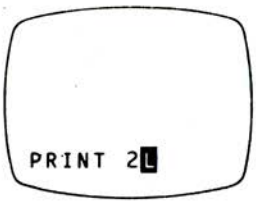
Comece por ligar o computador de forma a obter um ecrã limpo e apenas com o cursor **K**, como se vê na imagem.
Agora vamos fazer com que o computador determine quantos são 2 e 2 e nos apresente o resultado no ecrã.



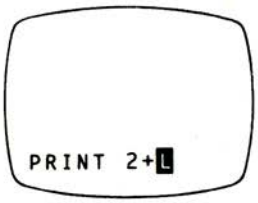
Primeiro faça **PRINT**. Basta premir a tecla P, como sabe, para que a palava apareça. Lembre-se sempre de que embora possa escrever a palavra letra a letra, P,R,I,N,T, as suas ordens ao computador devem ser sempre dadas através dos «comandos» escritos em cada tecla. Só assim o TC 2048 entenderá. Estes «comandos» aparecem, como vimos, quando o cursor **L** está no ecrã.
Para o caso de não ter ainda notado, repare que, neste Manual, imprimimos os «comandos» num TIPO DE LETRA DESTACADO.



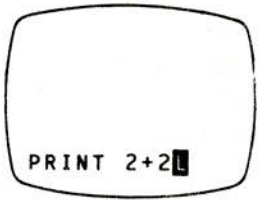
Além de ter surgido a palavra **PRINT** no ecrã, o cursor mudou de para



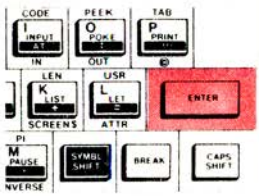
A seguir «tecle» 2. O 2 também aparecerá no ecrã e o cursor L deslocar-se-á para a sua direita. Note, a propósito, que ficou um espaço em branco entre a palavra **PRINT** e o 2.
Os «comandos» do BASIC do TC2048 já vêm com os seus próprios espaços em branco evitando-lhe o trabalho de os introduzir (embora nada prejudique o facto de colocar mais alguns espaços).



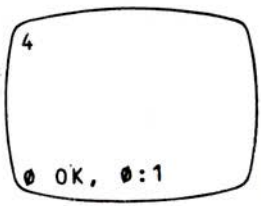
A terceira operação que terá a fazer consiste em «teclar» o sinal (+). Lembra-se de como se obtêm os caracteres que estão impressos em branco sobre bandas negras, nas teclas? Obtêm-se usando **SYMBOL SHIFT**. Mantenha **SYMBOL SHIFT** premida e ao mesmo tempo toque na tecla K.



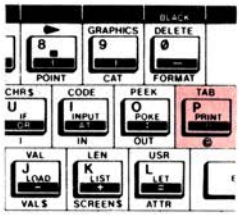
A seguir obtenha um segundo 2. O ecrã tomará o aspecto que pode apreciar na figura do lado.



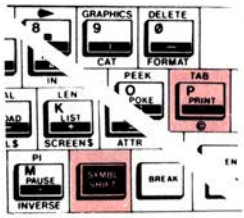
Finalmente deve premir a tecla **ENTER**. Sempre que tiver terminado uma linha de instruções ou um comando e desejar que o computador faça algo em relação a isso deve premir-se a tecla **ENTER**.



O computador exibirá enfim, a resposta no ecrã. É evidente que não precisa de um computador para cálculos desta natureza mas poderá utilizá-lo para operações matemáticas mais complexas (veja o Capítulo 11).



Tentemos nova experiência. A mensagem codificada que apareceu no fundo do ecrã (0 OK) está a ocultar um cursor K. Se premir qualquer tecla, enquanto a mensagem codificada estiver presente o ecrã obterá um «comando» da mesma forma que aconteceria se lá estivesse um cursor K. Toque na tecla P e veja que surgiu a palavra PRINT.



Prima SYMBOL SHIFT e P

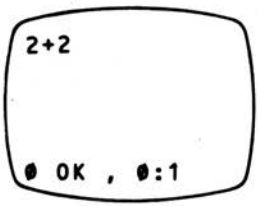
Não faz mal que a resposta do cálculo, que se efectuou há pouco, ainda permaneça no ecrã. Agora, utilizando SYMBOL SHIFT, obtenha o sinal gráfico de aspas premindo ao mesmo tempo a letra P.

Depois toque na tecla 2 e no sinal mais (+) (usando SYMBOL SHIFT). Em seguida «tecle» um outro 2.



Por fim, feche as aspas (SYMBOL SHIFT e P). O ecrã apresentar-se-á assim:

Agora toque na tecla ENTER.



Vejamos agora a diferença. Da primeira vez, quando compusemos

PRINT 2+2

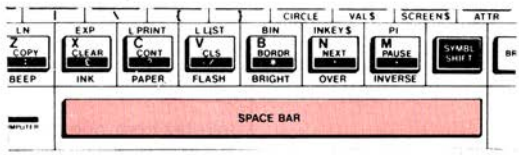
dissemos ao computador que avaliasse a expressão matemática e nos desse a resposta. Da segunda vez, quando escrevemos

PRINT "2+2"

dissémos ao computador: «Não execute quaisquer cálculos; limite-se a imprimir o que está entre aspas».

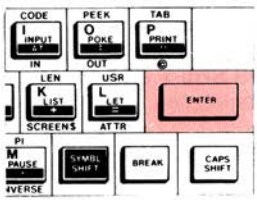
Capítulo 3: Como Dar Ordens ao Computador

Faça esta experiência (mas lembre-se de que, entre aspas, será forçado a colocar os espaços que pretender, utilizando a barra de espaços para esse fim — a barra de espaços esta na parte inferior do teclado —):





PRINT "Tudo quanto apareça entre aspas, tenha o comprimento que tiver".



Não se esqueça de premir a tecla **ENTER** quando acabar.



Aqui está um primeiro vocábulo da gíria da Informática para fixar: Tudo quanto apareça entre aspas denomina-se «string». À primeira vista pode parecer estranho, mas, na essência, todos os caracteres que se encontrem na «string», entre aspas, são tratados pelo computador como uma simples unidade.

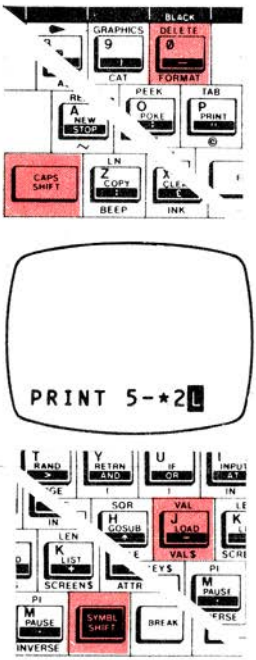
Agora «tecle» NEW ENTER

Isto faz com que a frase desapareça do ecrã. Sempre que pretender «limpar» o computador e começar de novo, como se tivesse acabado de o ligar, bastará premir a tecla **NEW** seguida de **ENTER**.
Pode também considerar que a mensagem de «copyright» é um cursor **K**, tal como acontece com as mensagens codificadas.

Que fazer quando se comete um erro qualquer?
Se tiver cometido um erro qualquer, poderá apagá-lo com **DELETE** (teclas **CAPS SHIFT** e **Ø**). Premindo **DELETE** apaga-se o caracter ou o «comando» que fica à esquerda do cursor.

Também lhe é possível deslocar o cursor para o local que entender, se se servir das teclas que têm as setas para a esquerda e para a direita (**CAPS SHIFT** e 5 ou **CAPS SHIFT** e 8) e fazer nesse local as eventuais correcções.

Componha o seguinte:
PRINT 5-*2

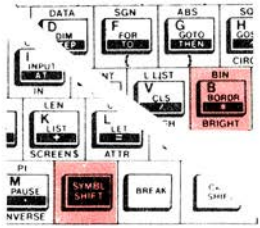


Premir SYMBOL SHIFT e J.

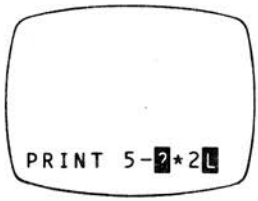
(Utilize **SYMBOL SHIFT** e J para o sinal menos (-) e **SYMBOL SHIFT** e B para obter o asterisco (*) que é o sinal de multiplicação no TC2048.

Depois prima a tecla **ENTER**.

Capítulo 3: Como Dar Ordens ao Computador



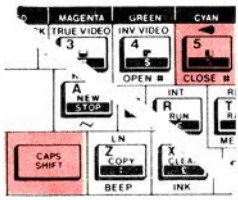
Premir SYMBOL SHIFT e B



O ecrã tomará o aspecto que pode observar na gravura ao lado.

O ? é o indicador de «erro de sintaxe». Significa que o TC 2048 não pode executar a ordem (não sabe se se pretende subtrair ou multiplicar).

Suponha-se que o que se pretendia era multiplicar PRINT 5*2. Procedamos à correcção, pelo método que aprendemos no Capítulo 2.

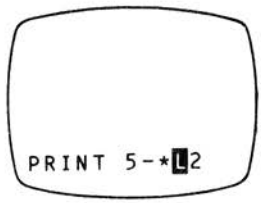


Prima CAPS SHIFT e 5

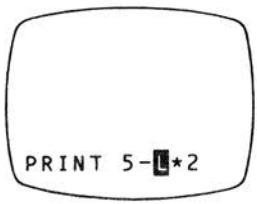
Antes de tudo, é preciso levar o cursor L até «ao local de erro de sintaxe».

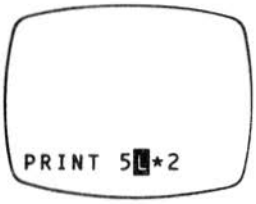
Mantendo a tecla CAPS SHIFT premida toque na tecla 5 que corresponde à seta que origina o movimento do cursor para a esquerda.

O cursor deslocar-se-á um espaço para a esquerda.

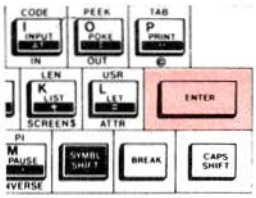


Matendo ainda o CAPS SHIFT premido, torne a tocar na tecla 5. Colocará o cursor na posição conveniente, à direita do sinal menos (–).



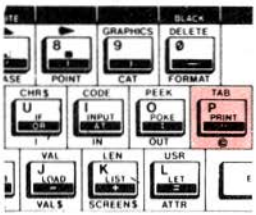


Pode premir, agora, **DELETE** (**CAPS SHIFT** e **0**) de forma a que o ecrã se apresente como se vê aqui ao lado.



Seguidamente poderá então premir novamente **ENTER**. Não se torna necessário deslocar o cursor **E** até ao fim da linha. Não faz qualquer diferença o local onde o cursor estiver, desde que o resto da linha esteja correcto. O cursor não é mais do que um indicador de posicionamento, só para os seus olhos verem.

Também poderá inserir novos caracteres, como verificámos no Capítulo 2, deslocando o cursor até ao local conveniente e premindo então a tecla (ou as teclas) dos caracteres que pretenda inserir.

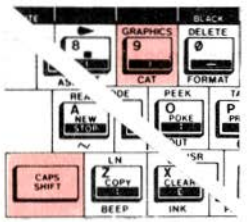


Vamos praticar:
Experimente compor **PRINT** "seguido de qualquer frase de que goste. Não se esqueça de fechar aspas quando terminar. Mas não carregue ainda na tecla **ENTER**.

Pratique, deslocando o cursor através da frase, para trás e para a frente, na linha composta por si e servindo-se das teclas das setas.

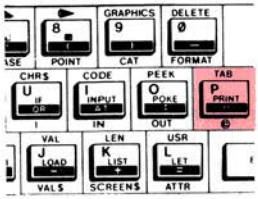
Proceda a eventuais alterações ou correcções do texto antes de premir a tecla **ENTER**. Além de corrigir erros, poderá ainda acrescentar novas palavras se as compuser depois de deslocar o cursor até ao local onde pretenda inseri-las.

Agora pode premir a tecla **ENTER**.

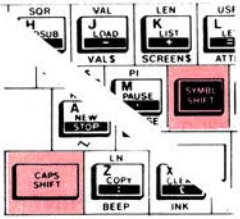


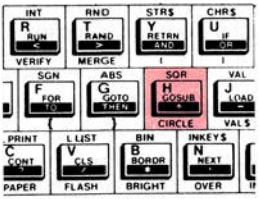
Pratique fazendo uma linha de símbolos gráficos, premindo a tecla **PRINT** e passando depois ao «modo» gráfico com **CAPS SHIFT** e a tecla 9 (GRAPHICS). Recorde-se de que para sair desse «modo» terá de voltar a premir **CAPS SHIFT** e a tecla 9, outra vez. Só então se torna possível fechar as aspas e dar entrada (**ENTER**) à linha produzida.

Capítulo 3: Como Dar Ordens ao Computador

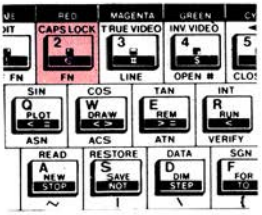


Se tiver inclinação para as matemáticas, aqui tem uma forma rápida de poder utilizar as funções. «Tecle» **PRINT** e depois passe ao «modo» extensivo através de pressão simultânea das teclas **CAPS SHIFT** e **SYMBOL SHIFT**.

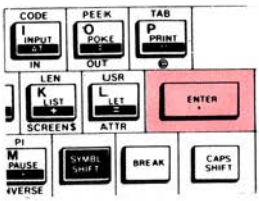




Com o cursor **E** no ecrã, toque na tecla H que fará imprimir **SQR** (Square Root — raiz quadrada).

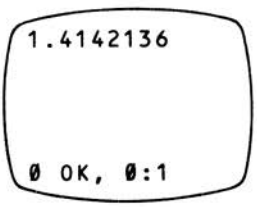


Agora toque nas teclas 2 e **ENTER**. (Aqui não precisa aspas).



Capítulo 3: Como Dar Ordens ao Computador





O computador, avaliada, a expressão matemática, imprime a raiz quadrada de 2 (ou seja 1.4142136) no ecrã.

Não vamos gastar muito tempo, neste Manual, com as outras funções, excepto para as mencionar no Capítulo 11 e para delas fazermos uma listagem no Apêndice A. Se for um matemático certamente que reconhecerá as abreviaturas que estão acima e abaixo das teclas. Se não for matemático, então não lhes ligue.

SUMÁRIO

1. PRINT («comando») localizada na tecla P «diz» ao computador para imprimir qualquer coisa no ecrã.

2. Se colocarmos um número a seguir a **PRINT** ele imprimir-se-á no ecrã; se uma expressão matemática como (2 + 2) se seguir a **PRINT**, a expressão será calculada e imprimir-se-á só o resultado. (Neste caso obter-se-ia um simples 4).

3. Tudo quanto estiver entre aspas, depois de **PRINT**, será impresso no ecrã exactamente como estiver na linha original. O conteúdo localizado entre aspas toma o nome de «string» porque de facto é uma cadeia de caracteres.

4. Quando premir a tecla **ENTER**, está a dizer ao computador que acabou de compor as suas ordens e deseja que ele as execute.

5. **NEW** limpa o computador para reiniciar trabalhos.

Como Usar Programas Pré-gravados 4

Síntese do Capítulo

Como utilizar os programas prontos a «correr» contidos em «cassettes» e também como gravar programas próprios através de LOAD e de SAVE. Também veremos, RUN, REM, LINE, MERGE e VERIFY.



Já tínhamos afirmado antes que poderá utilizar programas que foram escritos por outras pessoas para fazer funcionar o seu TC 2048. A Timex edita bastantes programas — de jogos, de aplicações domésticas, de assuntos relacionados com os negócios e também com a educação no lar — mas existem muitos outros programas editados por empresas especialistas de «Software».

Capítulo 4: Como Usar Programas Pré-gravados

Programas em «cassettes»
Poderá adquirir programas gravados em fita magnética (cassettes) e introduzí-los (**LOAD**) no computador através de um simples leitor/gravador de «cassettes».

Por vezes terá necessidade de introduzir um programa registado em «cassette», no seu computador. Quando tiver trabalhado com o programa e decidir desligar o computador, esse programa desaparece automaticamente da memória interna do TC 2048, mas, obviamente, continua a dispôr do programa registado na «cassette» e poderá voltar a «carregá-lo» quando muito bem entender.

Bastantes vezes sentirá a necessidade de introduzir um programa copiado de uma revista técnica ou de um livro, para o experimentar e, naturalmente, gravá-lo-á (**SAVE**) numa «cassette». Da próxima vez que tiver necessidade de o utilizar não será obrigado a introduzi-lo outra vez através do teclado; bastará carregá-lo a partir da «cassette» em que estiver gravado.

Pode também acontecer que introduza um programa no computador e decida acrescentar-lhe algumas informações, após o que desejará gravá-lo novamente numa outra «cassette» ou noutro local da «cassette» original, mas devidamente separados e identificados.

Vejamos como pode realizar-se cada uma destas tarefas.

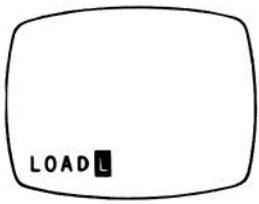
Como «carregar» um programa de uma «cassette».

Cada programa deve ter um nome e cada «cassette» que contenha mais do que um programa deverá ser munida de um índice relativo os nomes de cada um dos programas nela contidos.

Depois de ter todos os elementos componentes do seu sistema devidamente ligados entre si e à corrente eléctrica, de acordo com as informações que foram dadas no Capítulo 1, verifique se o seu gravador está a funcionar e se a «cassette» se encontra no início. Verifique também se tem o cursor K no ecrã.

Ligue a ficha do cabo de audio da tomada EAR do gravador para a tomada EAR do computador (ficha da mesma cor). Posicione o volume do som do gravador a cerca de 3/4 do volume máximo. Se o gravador tiver controlo de tonalidade, ajuste-o para o máximo de agudos e o mínimo de graves. (Se tiver um único comando de tonalidade coloque-o em alto «high» e obterá automaticamente o máximo de agudos e mínimo de graves.)

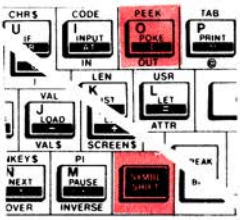
Depois disso «tecle».



LOAD

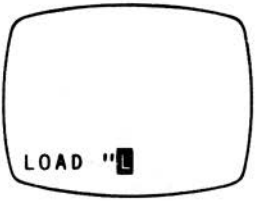
O que conseguirá quando premir a tecla J, se tiver o cursor K no ecrã. (Recorde-se de que, enquanto o cursor K estiver no ecrã se obtêm sempre os «comandos» inscritos em cada tecla)

Capítulo 4: Como Usar Programas Pré-gravados



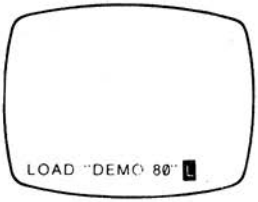
Notará que o cursor **K** se mudou para **L**. Isso significa que, a partir de agora, o computador imprimirá o caracter principal inscrito em cada tecla premida. Se, porém, premir **SYMBOL SHIFT** enquanto toca noutra tecla obterá os símbolos impressos em branco sobre tarja preta.

Recorde-se de que a tecla **SYMBOL SHIFT** está toda ela impressa negro, com letras a branco, exactamente para permitir que se lembre do efeito que produz quanto às outras teclas e a identifique com as bandas ou tarjas negras.



Terá necessidade de informar o computador quanto ao nome do programa que quer usar. Esse nome deve ser indicado entre aspas. Suponha que pretende introduzir um programa de um jogo que se chama DEMO 8Ø.

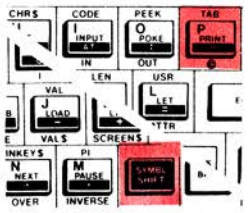
Mantenha premida a tecla **SYMBOL SHIFT** e ao mesmo tempo toque na tecla P para obter as primeiras aspas.



Depois, componha o nome do programa, verificando bem se está exactamente igual.

O nome de um programa poderá ter 1Ø caracteres, incluindo os espaços, entre nomes. Se houver espaços, o nome do programa terá também de incluí-los.

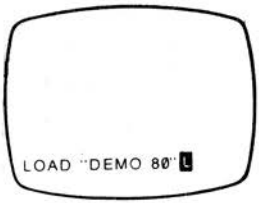




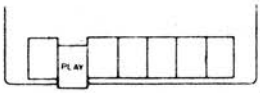
O computador faz completa distinção entre letras maiúsculas e letras minúsculas. procure, portanto, escrever o nome do programa com maiúsculas ou com minúsculas de acordo com a forma como o programa se encontra registado, isto é, só letras maiúsculas, só letras minúsculas ou os dois tipos de letra misturados.

Poderá então fechar as aspas com **SYMBOL SHIFT** e P.

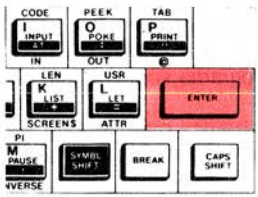
Capítulo 4: Como Usar Programas Pré-gravados



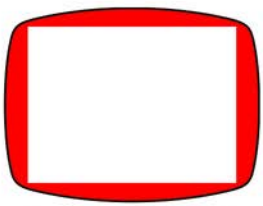
O ecrã do seu televisor apresentar-se-á assim:



Agora pode premir a tecla «PLAY» do seu gravador de «cassettes» e, logo a seguir, a tecla **ENTER** do computador.



(**LOAD** é um comando que informa o computador daquilo que você pretende dele, "DEMO 8Ø" é o nome do programa que o computador deverá «carregar» e **ENTER** é o sinal de que completou as suas instruções, pelo que o TC2048 deve iniciar o seu trabalho).



Pesquisa de programa

O bordo (**BORDER**) do ecrã do seu televisor alternará entre azul pálido (cyan) e vermelho (red), durante todo o tempo em que o computador estiver à procura do programa referenciado, na «cassette».



Programa encontrado

Quando, finalmente, o programa for detectado pelo computador, o bordo do ecrã apresentar-se-á com um padrão de tracejados das mesmas tonalidades (azul e vermelho) e, logo a seguir, surgirá o nome do programa.

Capítulo 4: Como Usar Programas Pré-gravados

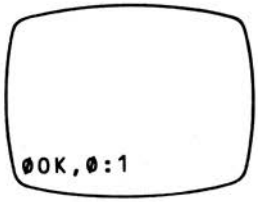


Programa a «carregar»

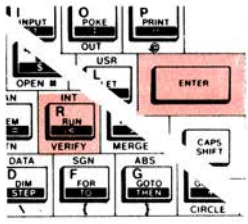
Depois, à medida que o programa vai sendo «carregado» no computador, o padrão das riscas coloridas tornar-se-á mais estreito e apresentará riscas em amarelo e azul escuro movendo-se mais rapidamente.

Quando o computador acabar de «carregar» o programa, acontece o seguinte:

1. Ou «arranca» automaticamente, normalmente com um «título em ecrã» ou com instruções para o operador, se se tratar de certos programas comerciais. Deverá parar imediatamente o gravador para que fique pronto a «carregar» o programa seguinte, se assim o desejar;
2. Ou o ecrã apresenta-se branco com uma mensagem no fundo, como se segue: **Ø OK,Ø:1**. Esta é uma mensagem ou «relatório codificado» e significa que o computador completou satisfatoriamente a leitura e «carregamento» do programa (há uma lista de relatórios codificados no fim deste Manual — Apêndice H. Os «relatórios codificados» são a forma pela qual o TC 2048 pode comunicar consigo para lhe transmitir que acabou o seu trabalho ou que encontrou alguma dificuldade ou problema).



Mensagem (relatório codificado)



Prima **RUN** e **ENTER**

Mais uma vez: Pare o gravador imediatamente. Depois, para dar execução ao programa deve premir as seguintes teclas:

RUN

e

ENTER

Isto fará com que o programa «arranque».

Capítulo 4: Como Usar Programas Pré-gravados

Se estiver a utilizar uma «cassette» que contenha mais do que um programa registado e desejar «carregar» um que não esteja gravado logo no início da fita, notará os padrões indicadores de pesquisa e de «carregamento» mais do que uma vez.
Cada programa que for passando na «cassette» originará um padrão de carregamento «loading» no ecrã, e o seu nome aparecerá também, embora o computador não esteja efectivamente a carregá-lo. O TC2048 só começará verdadeiramente a «carregar» quando detectar o nome do programa que lhe tenha indicado.
Se pretender saber que programas tem na «cassette» poderá dar a ordem

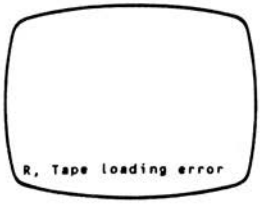


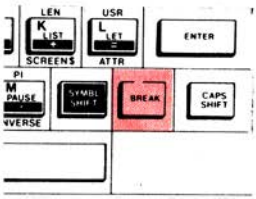
LOAD "JORGE" (ou qualquer outro nome)

desde que saiba que não existe qualquer programa com esse nome registado naquela «cassette».
Então, à medida que o computador for procurando o programa "JORGE", fará imprimir no ecrã os nomes dos programas que de facto estão gravados na referida «cassette».
Também poderá fazer um índice para cada «cassette», colocando o contador em zero antes de iniciar o processo que descrevemos. Depois, bastará anotar o número em que se encontra cada um dos programas, à medida que os vir impressos no ecrã.

Ocasionalmente pode acontecer que um determinado programa não «carregue» bem. Terá então de investigar as razões que motivaram essa anomalia ou experimentar novamente. Tomará conhecimento de que um **LOAD** falhou se:

1. O cursor K voltar a aparecer no ecrã.
2. O padrão de indicação de pesquisa de programa surgir novamente no ecrã, depois de já ter aparecido o padrão indicador de «carregamento» de programa. (Isto, naturalmente, se estiver seguro de que o programa cuja indicação de «carregamento» deu, é, efectivamente, aquele que deseja e não um outro qualquer).
3. Aparecer um relatório codificado dizendo: R, Tape loading error





O relatório «Tape loading error» significa que o TC 2048 encontrou o programa (foi capaz de ler o seu nome), mas não conseguiu «carregá-lo» por lhe ter detectado erros (por hipótese, uma interferência poderá ter originado um «bit» a mais ou a menos na leitura da fita, anulando totalmente a assimilação do programa).

Se o cursor **K** aparecer no ecrã, não há nada a fazer senão rebobinar a «cassette» e tentar de novo.

Se os padrões de traçados indicadores de pesquisa e «carregamento» de programa ainda continuam a apresentar-se no ecrã, será então necessário premir a tecla **BREAK** para que o computador pare. A seguir estará então em condições de enfrentar o problema.

O problema mais corrente é devido ao volume de saída do gravador estar muito alto ou muito baixo.

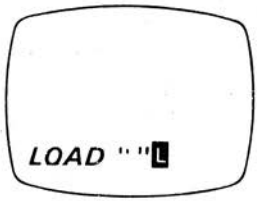
O melhor sistema de ajuste consiste em colocar o volume de som tão alto quanto possível sem que os «silêncios», indicadores de espaços, na fita magnética, produzam ruído de fundo; poderá verificar este aspecto se desligar a ficha que tem ligada ao terminal de saída de som do gravador, ouvindo a leitura da fita através do altifalante. Se os momentos de silêncio tiverem demasiado ruído, então é possível que tenha outros problemas:

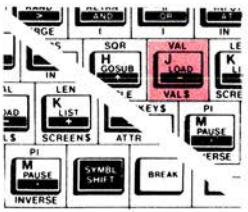
Alguns aparelhos gravam (por defeitos) o ruído de fundo da tensão alterna de 50 ciclos. Isto poderá evitar-se, se adoptar o sistema de trabalhar com o seu gravador a pilhas.

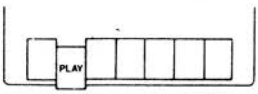
Outros — especialmente os gravadores antiquados ou com muito uso, são, já de si barulhentos (ruidosos) produzindo um ruído de fundo constante quando estão a fazer a reprodução das «cassettes». Neste caso terá de investir na compra de um novo gravador.

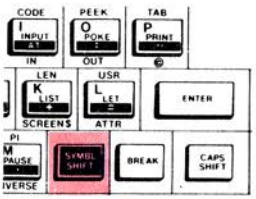
Poderá ter de ajustar melhor a ficha de saída de som, no gravador. Nalguns gravadores, o contacto deixa de ser perfeito sempre que a ficha fique um pouco mais dentro ou um pouco mais fora da posição ideal. Tente puxar ligeiramente a ficha até sentir que a colocou efectivamente numa oposição correcta e firme.

Capítulo 4: Como Usar Programas Pré-gravados









É ainda possível que tenha uma «cassette» com um programa que foi gravado num outro gravador cujas cabeças de gravação não estivessem devidamente alinhadas (azimutadas). Verificará isso se, utilizando tal gravador, o programa ficar bem «carregado» no computador.

Se tiver demasiados problemas com a leitura das «cassettes»e isso inclui as «cassettes» comerciais, é provável que sejam as cabeças do seu gravador que não estão bem alinhadas.
Poderá ter que mandar ajustá-las ou afiná-las.

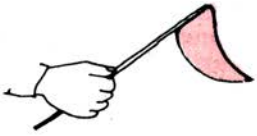
Pode acontecer que ambos os gravadores estejam ligeiramente desalinhados — sem que isso impeça que gravem ou «carreguem» (**SAVE** ou **LOAD**) as suas próprias «cassettes» ou as comerciais,mas o suficiente para que cada um dos gravadores não possa aceitar as «cassettes» que foram gravados no outro.

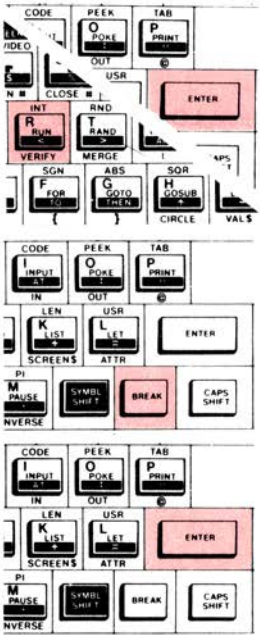
É possível «carregar» um programa sem indicar o seu nome. Se der a ordem **LOAD""**

(isto é, depois de premir a tecla **LOAD** (J), usar **SYMBOL SHIFT** e premir duas vezes a tecla P. — Nunca coloque um espaço entre as aspas). Ponha o seu gravador na posição PLAY com as ligações necessárias, e carregue na tecla **ENTER**. O TC 2048 «carregará», desta forma, o primeiro programa que encontrar. É uma ideia útil para o caso de apenas existir um programa na «cassette» ou mesmo quando desejar «carregar» o que vem a seguir, mas se esqueceu do nome dele.

Capítulo 4: Como Usar Programas Pré-gravados







Como introduzir um programa no computador através do teclado e gravá-lo depois em «cassette»

São muitos os pequenos programas publicados em livros, revistas e magazines da especialidade. Poderá utilizá-los se os introduzir através do teclado do seu TC 2048. Mas faça-o com toda a atenção e cuidado para que tudo corresponda à forma correcta, não se esquecendo da pontuação e dos espaços.

Depois de ter introduzido o programa, poderá comparar a listagem impressa com a do ecrã. Veja o Capítulo 2 para verificar como é fácil proceder a emendas.

Tenha cautela, contudo. Pode acontecer que o próprio programa listado que esteve a copiar, já contenha erros, originais ou tipográficos. Pode ter copiado bem e, ainda assim, ter problemas.

Quando tiver terminado a introdução do programa, faça-o «correr», através da pressão sucessiva das seguintes teclas:

RUN e ENTER

como já vimos anteriormente. Quando estiver disposto a acabar ou interromper o programa, tem duas soluções: ou espera que o programa chegue ao fim ou interrompe-o premindo a tecla BREAK simultaneamente com CAPS SHIFT.
Poderá obter a listagem do programa, no ecrã, se tornar a premir

ENTER

Então, depois de ter verificado (utilizando-o) que o programa resulta e que está introduzido correctamente, pode gravá-lo na fita de uma «cassette», para uso posterior. (Também é possível gravar um programa mesmo que contenha erros. É até uma boa política gravar o programa antes de o ensaiar, como medida de precaução. Se tiver procedido assim, pode sempre «carregá-lo» de novo e tentar descobrir as incongruências ou erros. E, se não tiver tempo no momento, poderá tentar mais tarde).

Como gravar um programa numa «cassette»
Como dissemos anteriormente, cada programa deve ter um nome. Até porque o TC 2048 não gravará nenhum programa sem que lhe tenha atribuído primeiro um nome. Poderá criar o nome para um programa que tenha copiado através do teclado, como acabámos de descrever, ou, até mesmo, trocar esse nome por um outro qualquer de que goste mais. Seja o que for que chame ao seu programa, quando o gravar, esse será também o nome que terá de usar quando, mais tarde, o voltar a «carregar» para o computador.

Lembre-se: O nome do programa deve ser constituído por um máximo de dez caracteres. Pode ter mais do que uma palavra, mas cada espaço conta também como uma unidade para o limite dos dez caracteres.

Nota: É uma boa ideia colocar o nome do programa no começo da sua própria listagem. Isso dar-lhe-á a possibilidade de verificar se, de facto, obteve o programa que desejava. A forma mais fácil de o conseguir é utilizar uma linha **REM** logo no início do programa.

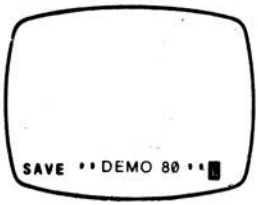
Dar-se-á eventualmente conta de que as linhas de grande parte dos programas estão numeradas em múltiplos de 1Ø, de maneira que, se um programa ainda não tiver uma linha **REM** com o nome respectivo, pode perfeitamente compor, no teclado

5 **REM** PROGRAMA — DEMO 8Ø

e, seguidamente, premir a tecla **ENTER**, utilizando assim um número de linha a seguir ao mais baixo já existente no programa.

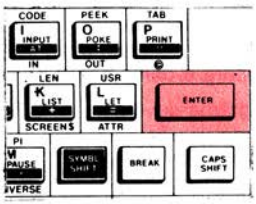
O computador encarrega-se de colocar, automaticamente, a nova linha logo no início da listagem.
Obviamente que o **REM** conterá o nome real do seu programa.
Uma linha de programação que se inicie com **REM** (o que quer dizer **REM**ark ou **REM**inder,

Capítulo 4: Como Usar Programas Pré-gravados



que é como quem diz «observação») é desprezada pelo computador no decorrer do programa. As linhas **REM** aparecem na listagem apenas para nossa orientação e não do computador.

Ligue o terminal MIC do computador à tomada MIC do seu gravador. Depois, posicione a «cassette» num local onde a fita esteja virgem ou gravada com programas que pretenda ver destruídos.
Escreva:



SAVE "DEMO 80"

utilizando a tecla **SAVE** que se encontra em S. Depois deve premir

ENTER



O computador imprimirá então, no ecrã, a seguinte frase:

«Start tape then press any key» o que significa «ligue o gravador e depois prima qualquer tecla».



Botão de Gravação (RECORD)

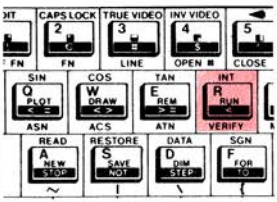
Ligue o gravador, na posição de GRAVAÇÃO, tocando depois em qualquer tecla do TC 2048.



Padrão de Gravação

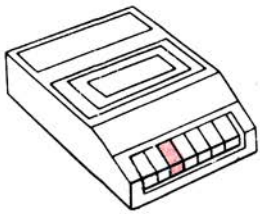
Olhe para o ecrã. Observará um padrão de linhas ou traços, semelhantes aos que descrevemos para o «carregamento» de programas (**LOAD**ing), seguido da mensagem **Ø OK,Ø:1** que, neste caso, significa: A GRAVAÇÃO FOI BEM SUCEDIDA.

Capítulo 4: Como Usar Programas Pré-Gravados



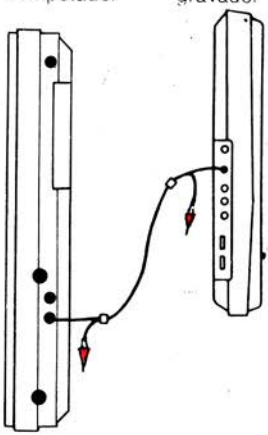
Como verificar (VERIFYing) uma gravação
Para obter um controlo mais efectivo de que a gravação foi de facto, perfeita, poderá utilizar a função VERIFY que está localizada por baixo da tecla R.

Antes de tudo, rebobine a fita da «cassette» até chegar ao ponto onde iniciou a gravação com o SAVE.



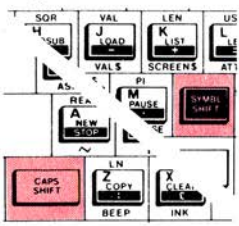
Botão de rebobinamento

computador gravador



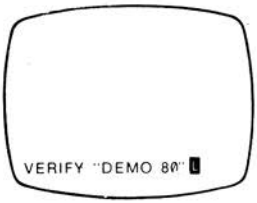
Ligar EAR com EAR

Em segundo lugar, verifique em se as fichas do EAR tanto do gravador como do computador, estão ligadas.



Em terceiro «tecle» VERIFY "DEMO 8Ø". (Para conseguir a função VERIFY, deve premir CAPS SHIFT e SYMBOL SHIFT ao mesmo tempo para obter o cursor **E** e, depois, premir as teclas SYMBOL SHIFT e R).

Capítulo 4: Como Usar Programas Pré-gravados





Botão de reprodução (PLAY)



Por fim, «arranque» com o leitor de «cassettes» — em PLAY (para reproduzir) — e pressione.

ENTER

O computador procede então a uma comparação entre o programa que está gravado na fita e aquele que já se encontra na sua memória. Se encontrar o título mostrará de seguida, no ecrã a mensagem seguinte:

Program: DEMO 80

após o que continuará a produzir o padrão de tracejados coloridos, como aconteceu com o **LOAD**. Se o programa for verificado e tudo estiver certo, surgirá, no canto inferior esquerdo do ecrã, o relatório codificado **0 OK.**

Se porém for encontrada qualquer anomalia entre a fita da «cassette» e o programa memorizado no computador o relatório codificado será, **R Tape loading error**, pelo que convirá fazer um novo **SAVE**.

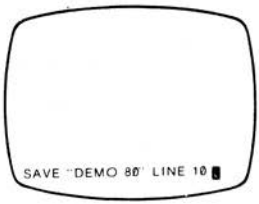
Se o nome do programa não aparecer no ecrã, então é porque o programa nem sequer chegou a ser gravado. É preciso verificar bem:

1. Se as fichas estão correctamente ligadas.
2. Se o volume de som do gravador estava correctamente ajustado.
3. Se, acaso, não tentou gravar nos extremos (não magnéticos) da fita de tracção que se encontram no início e no fim da «cassette».

4. Se, no seu gravador, a tecla que comanda a gravação (RECORD) estava, efectivamente, premida.

Finalmente, se o nome do programa apareceu, mas o computador terminou o seu trabalho sem produzir a mensagem **0 OK** depois de ter apresentado o padrão de **LOAD** no ecrã e, em vez disso, continua a procurar o programa, é muito possível que tenha feito um erro na composição do nome, quer no **SAVE** quer no **VERIFY** (Se acaso o nome não for repetido exactamente da mesma forma gráfica, o computador não aceita os programas como idênticos).

Gravação para Arranque Automático
Poderá gravar os programas de tal forma que, quando forem «carregados», tenham logo «auto--arranque». A única coisa que de que necessita é de juntar o **LINE** e o número da linha do programa onde se deseja que o computador comece (usualmente, mas nem sempre, é a primeira linha do programa).



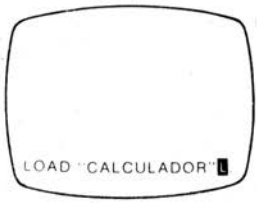
Por exemplo:

SAVE "DEMO 8Ø " **LINE** 1Ø

LINE é a função localizada sob a tecla 3. Para a conseguir terá de , antes de tudo, premir ao mesmo tempo as teclas **CAPS SHIFT** e **SYMBOL SHIFT**, entrando no «modo» extensivo.
Depois de ter obtido o cursor **E** no ecrã, deverá premir **SYMBOL SHIFT** e 3.
Quando «carregar» (**LOAD**) algum programa que gravou (**SAVE**d) desta forma, ele começará automaticamente a rodar, iniciando-se exactamente na linha que indicou.

Como gravar programas com «dados pessoais» introduzidos por **DATA**

Alguns programas são concebidos e realizados com a finalidade de lhe permitir a introdução de









elementos pessoais (DATA). É o caso de listas de telefones ou outras idênticas, com nomes, números e quaisquer outras indicações. Normalmente, tais programas são utilizados de forma idêntica e com os mesmos procedimentos que acabamos de examinar:

1. **LOAD** («carregue») o programa como explicamos.
2. **RUN** («corra») o programa e introduza os elementos pessoais que entender, à medida que o computador lhos solicite.
3. **SAVE** («grave») o programa que já contenha os seus elementos pessoais (**DATA**) acrescentados ao programa original, com outro nome. Se, por hipótese, «carregou» um programa que se chama «CALCULADOR» e a seguir lhe introduziu dados novos, (digamos, por exemplo, elementos relativos às suas finanças), pode desejar que o novo programa, já com tais indicações, passe a chamar-se «FINANÇAS».
4. **VERIFY** («verifique») o programa que acabou de gravar.

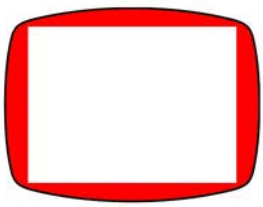
Como «carregar» Programas através de MERGE

O comando **MERGE** pode utilizar-se, em substituição de **LOAD**, se pretender combinar dois programas distintos num só. Enquanto **LOAD** faz limpar totalmente os dados inseridos na memória do computador antes de introduzir o novo programa, **MERGE** deixa que o anterior programa continue na memória enquanto está a «carregar» o novo.

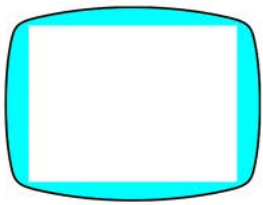
Contudo (muita atenção!) é necessário que as linhas de programação estejam em zonas separadas e sejam completamente distintas. Se houver duas linhas com o mesmo número no programa anterior e no que está a ser «carregado» através de **MERGE**, a nova linha toma prioridade e a do programa anterior é apagada.

Significa isto que se torna necessária uma cuidada planificação prévia, antes de proceder ao «carregamento» de um programa através de

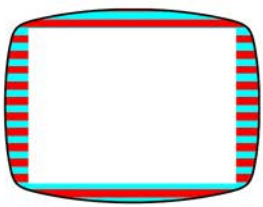
Capítulo 4: Como Usar Programas Pré-gravados



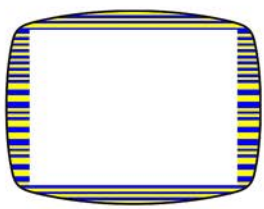
Pesquisa do programa



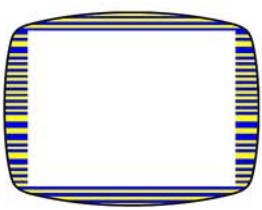
Pesquisa do programa



Programa encontrado



LOAD de um programa



SAVE de um programa

MERGE. (Se gravar um programa através de LINE e posteriormente o «carregar» com MERGE em vez de LOAD, pode acontecer que ele não salte para a linha com o número adequado, deixando assim de «arrancar» automaticamente).

Como se pode verificar, existem inúmeras formas de utilizar o seu Timex Computer 2048 sem que tenha de aprender a programar. Mas, se por acaso desejar dar uma olhadela sobre essa matéria, experimente ler os Capítulos que se seguem e decida por si próprio se gosta ou não.

SUMÁRIO

1. Existem muitos programas para o TC 2048 e não é preciso que saiba programar para poder utilizá-los.
2. O comando LOAD seguido do nome de um programa (entre aspas) faz com que o computador possa «carregar» esse programa numa «cassette».
3. LOAD seguido de duas aspas (sem espaços ou qualquer coisa entre elas — LOAD " " faz com que o TC 2048 «carregue» o primeiro programa que encontra na «cassette».
4. O comando SAVE seguido do nome de um programa (entre aspas) ordena ao computador que produza uma gravação desse programa numa «cassette», desde que o gravador esteja a trabalhar no modo de gravação (RECORD).
5. O programa carece absolutamente de um nome quando se pretenda fazer a sua gravação através de SAVE. Mas poderá escolher o nome que entender — desde que não tenha mais do que dez caracteres, incluindo os espaços em branco —. Também poderá mudar o nome de um programa, alterando-o para qualquer outro, de forma a que diversas versões de um programa sejam facilmente identificáveis (tal como acontece naqueles programas em que deverá introduzir dados com actualização periódica).
6. Se acrescentar LINE seguido de um número de linha de programação para gravar (SAVE) um programa, este programa, «arrancará»

automaticamente nessa linha, quando posteriormente o «carregar» (**LOAD**).

7. O comando **VERIFY** compara o programa que está gravado na fita da «cassette» com aquele que o computador contém na sua memória. Terá, assim, a certeza de que o programa foi bem gravado antes de o suprimir da memória do TC 2048.

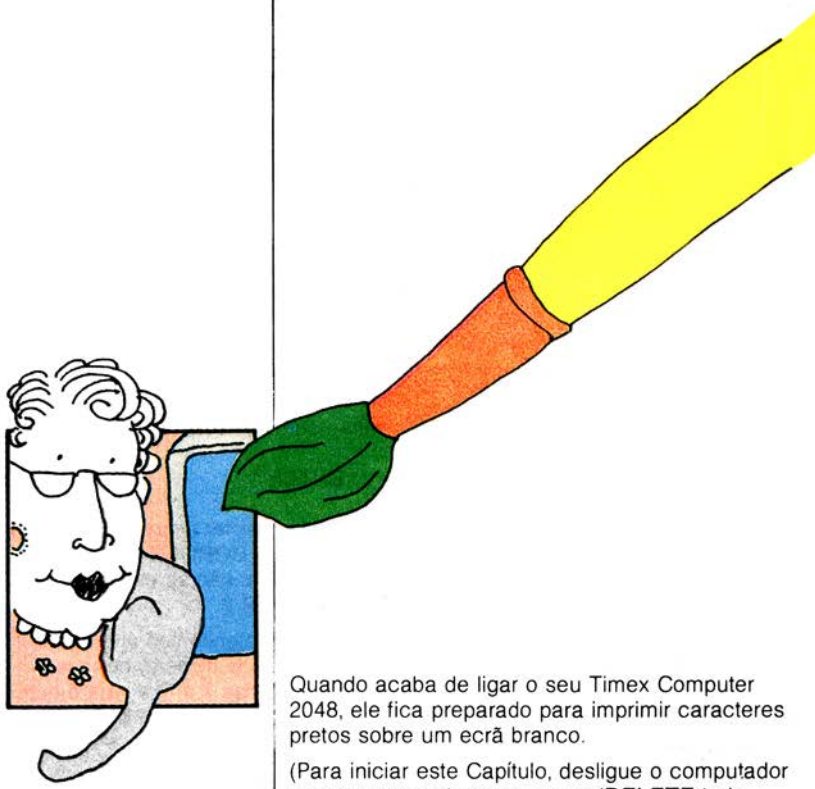
8. Quando se «carrega» um programa através de **MERGE** em vez de **LOAD**, o programa que já estiver no computador não será apagado. Contudo, se houver linhas com o mesmo número em ambos os programas, é dada prioridade à nova linha do programa que está a ser carregado, eliminando-se, automaticamente, aquela que existia no programa anterior.

   Convirá, portanto, que veja cuidadosamente se os programas estão ou não em condições de serem «carregados» através de **MERGE**.

Como Usar as Cores 5

Síntese do Capítulo

Aprenda a utilizar as instruções, INK, PAPER e BORDER, para obter efeitos de cor.

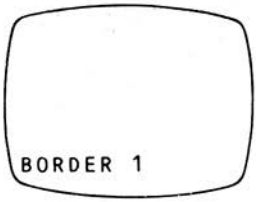


Quando acaba de ligar o seu Timex Computer 2048, ele fica preparado para imprimir caracteres pretos sobre um ecrã branco.

(Para iniciar este Capítulo, desligue o computador por uns momentos ou apague (**DELETE** tudo quanto esteja no ecrã e, a seguir, toque nas teclas **NEW** e **ENTER**. Isto dá-lhe a certeza de que começa com o computador limpo).

Poderá escolher entre oito cores — para o ecrã, para o bordo à volta dele e também para os caracteres a imprimir.

Toque na tecla **BORDER** (é a tecla do B) — desde que tenha o cursor **K** ou a mensagem de «copyright» no ecrã, claro — se acabou de ligar o computador.



Seguidamente, toque na tecla 1 de forma que o ecrã se apresente como na figura ao lado:

BORDER 1

(Na tecla está escrito **BORDR** (para caber na tecla) mas no ecrã aparece correctamente escrito **BORDER**).

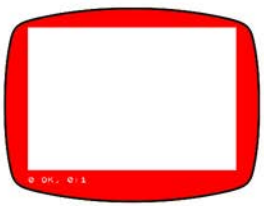
Agora toque na tecla **ENTER**. Não é uma modificação agradável do preto e branco que tínhamos? Deve ter obtido um bordo (**BORDER**) do ecrã na cor azul escuro, tal como está indicado por cima da tecla 1 que premiu.



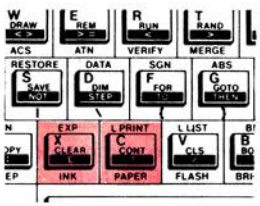
A propósito: não seria necessário dizer que só poderá observar as cores se o seu televisor for, de facto, um televisor a cores. Na eventualidade de o seu televisor ser a preto e branco (e treinar todo este Capítulo nele) à medida que o número da cor (entre Ø e 7) sobe, passa-se gradativamente, do negro, por tonalidades cinzentas mais claras até que se atinge o branco do 7.

Observemos outras cores: «Tecle»

BORDER 2 ENTER



e assim sucessivamente até ter atingido **BORDER** 7. Quando tiver chegado a **BORDER** 7 (branco), o bordo fica da cor do ecrã. E não se esqueça também do preto: **BORDER** Ø.

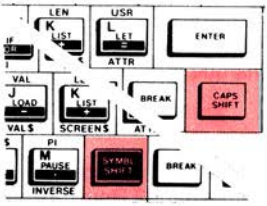


Suponhamos agora que pretendemos alterar a cor do ecrã, na sua área de comunicação connosco (**PAPER**). Olhe para as teclas X e C e veja o que está escrito abaixo delas. Estão lá os «comandos» **INK** e **PAPER**.

Chamamos **PAPER** ao rectângulo de comunicação, no ecrã, porque é aí que se escreve. A cor que usamos para aí escrever chama-se **INK**.

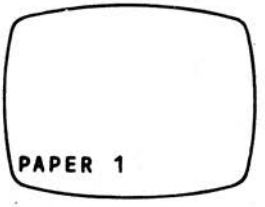
Comece por tentar mudar a cor do **PAPER**, antes de mais.

Ainda se lembra de como se obtêm os «comandos» situados abaixo das teclas?

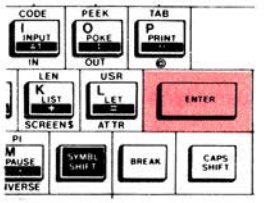


Premir simultaneamente CAPS SHIFT e SYMBOL SHIFT

Deverá premir, ao mesmo tempo, as teclas **CAPS SHIFT** e **SYMBOL SHIFT** para que surja no ecrã o cursor **E** . Depois disso mantenha premida a tecla **SYMBOL SHIFT** e pressione a tecla C: obterá a palavra **PAPER**. Depois «tecle» 1 de forma que o ecrã apresente



PAPER 1



Seguidamente deve premir a tecla **ENTER**. Agora volte a premi-la.

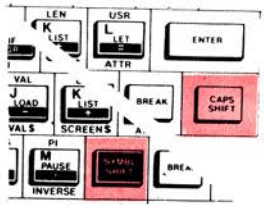
É preciso premir duas vezes a tecla **ENTER** para poder observar a cor que escolheu para o **PAPER**. No Capítulo 2Ø, voltaremos a falar deste assunto e explicaremos porquê.



Ensaie com as diversas cores, como fizemos com o **BORDER**.

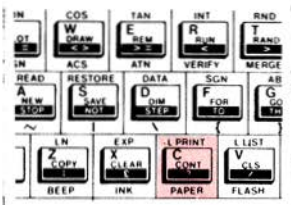
Quando chegar a Ø, obterá um ecrã totalmente negro.

Experimente, também, as combinações de cor que desejar. Pode escolher um **BORDER** com uma cor e um **PAPER** com outra qualquer ou modificar uma delas e não a outra.



Agora vamos voltar ao ecrã branco, com **PAPER** 7 (não tem qualquer importância a cor que ficar no **BORDER**).

Mantenha os dois **SHIFTS** premidos para obter o cursor **E** e, depois, deixe ficar o **SYMBOL SHIFT** premido e «tecle» X:



INK

Pressione a tecla 1, de forma que o ecrã mostre

INK 1

e a seguir prima a tecla **ENTER**. Não aconteceu nada de extraordinário, não é verdade? Bom, no próximo Capítulo voltaremos a examinar a palavra **INK**. É preciso que se tenha alguma coisa escrita no ecrã (no **PAPER**) para que a cor (**INK**) apareça.

A propósito, se tiver escolhido a mesma cor para **PAPER** e para **INK** é o mesmo que estar a escrever com tinta invisível, pois nada verá no ecrã! (Por vezes é conveniente).

Antes de passar ao próximo Capítulo, talvez seja bom voltar à posição de preto no branco original. Basta desligar o computador por alguns segundos ou então premir as seguintes teclas:

BORDER 7	ENTER
PAPER 7	ENTER (duas vezes)
INK Ø	ENTER

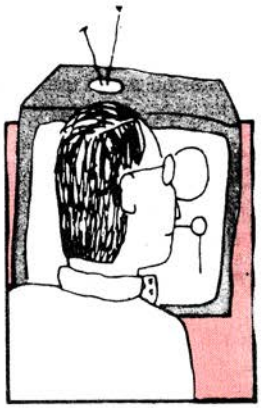
SUMÁRIO

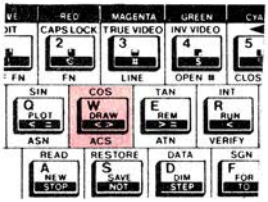
1. **BORDER** permite determinar a cor para o bordo do ecrã, através das cores que estão indicadas por cima das teclas da primeira fila.
2. **PAPER** que, através das mesmas teclas, se especifique a cor da área de comunicação na qual aparecerá o texto ou os gráficos produzidos pelo computador.
3. **INK** especifica a cor dos símbolos ou quaisquer outros caracteres que apareçam escritos no ecrã (no **PAPER**).

Como Desenhar Linhas e Círculos 6

Síntese do Capítulo

Este capítulo mostra-lhe como produzir, facilmente, trabalhos artísticos em qualquer ponto do ecrã, com as instruções PLOT, DRAW e CIRCLE.

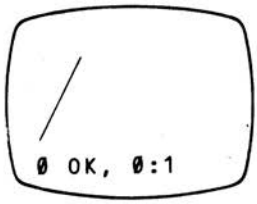




Uma das notáveis características do Timex Computer 2048 é a sua capacidade quanto aos gráficos de alta resolução. Mais tarde, neste mesmo Manual, desenvolveremos a matéria falando sobre um outro tópico acerca dos gráficos, mas, por agora, vamos ensaiar dois dos mais simples e mais poderosos comandos, **DRAW** e **CIRCLE**.

Antes de tudo, para nos orientarmos, é preciso saber que o **PAPER** (rectângulo de comunicação no ecrã) tem 256 posições no sentido da largura e 176 no da altura — ao todo 45Ø56 posições (256 × 176) nas quais pode fazer gráficos. Cada uma destas posições denomina-se «pixel», o que significa «Picture Element» ou elemento gráfico. (Veja a definição destes elementos na tabela impressa no Capítulo 17).

Vamos às coisas práticas, começando com o comando **DRAW**. Partindo do princípio de que no ecrã está o cursor **K** ou a mensagem de «copyright» (já vimos que consideramos que a mensagem de «copyright» está a ocultar um cursor pressione a tecla W e verá surgir a palavra **DRAW** no ecrã. «Tecle» então. 5Ø depois, uma vírgula (**SYMBOL SHIFT** e N)



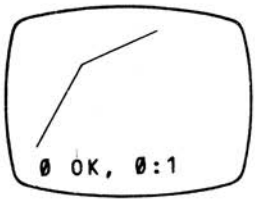
e por fim o número 100. No ecrã deverá aparecer o seguinte:

DRAW 50,100

Pressione **ENTER**. Surgiu uma linha, traçada a partir do último ponto assinalado (**PLOT**) que, quando se liga o computador, fica sempre situado no canto inferior esquerdo do **PAPER**).

A linha foi traçada, portanto, deslocando-se 50 posições (pixel) para a direita e de 100 posições para cima e originou uma diagonal a partir do canto inferior esquerdo do ecrã, que corresponde às coordenadas 0,0.

Deixe que a linha fique no ecrã e experimente outra coisa. Recorda-se como obter o comando **INK?** (Vem no Capítulo anterior).
«Tecle»



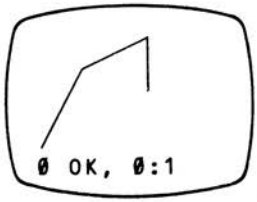
INK 2 ENTER

e a seguir

DRAW 100,50 ENTER

Desta vez a linha partiu do ponto onde tinha terminado a outra e deslocou-se 100 posições (pixel) para a frente e 50 posições para cima — e apareceu desenhada na cor 2 (Vermelho).

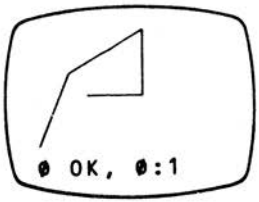
Deixe ficar, no ecrã, aquilo que desenhou. Durante todo este Capítulo, aliás, não deve limpar o ecrã, nem «teclar» **NEW**, a não ser que lho digamos claramente. (Alguns dos comandos que utilizamos como exemplos sairão fora do ecrã se não tivermos o cuidado de os fazer começar no local apropriado).



Vamos experimentar outra vez:

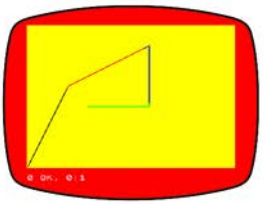
INK 1 ENTER
DRAW 0,-75 ENTER

A linha que agora surgiu foi traçada a azul; o zero (0) significa que não houve deslocação no sentido horizontal, mantendo-se a mesma posição e o -75 (menos 75) comandou o desenho do traço para 75 posições abaixo.



Como é que desenharia um traço para a esquerda... horizontal.. e em verde?

INK 4 ENTER
DRAW — 75,0 ENTER

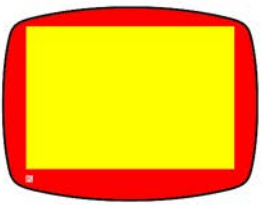


Vamos ver se conseguimos mudar a cor do ecrã. Primeiro componha:

BORDER 2 ENTER

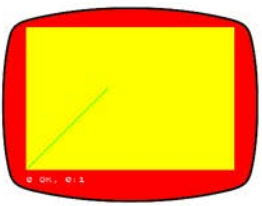
Óptimo! Um bordo encarnado. Agora

PAPER 6 ENTER ENTER



Eia! Temos um PAPER amarelo, sim senhor, mas apagaram-se todos os desenhos.

Veremos mais alguns aspectos deste assunto, como dissemos, mas no Capítulo 8. Por agora lembremo-nos apenas de que não é possível mudar a cor do PAPER sem limpar o que lá estiver escrito ou desenhado.



Agora experimente:

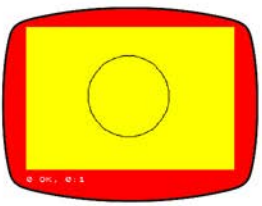
DRAW 100,100 ENTER

A linha traçada surgiu na cor verde dado que o verde é, ainda, a cor que foi escolhida em último lugar. Torne a premir a tecla ENTER.

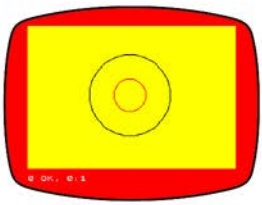
É uma outra forma de apagar! Há detalhes com que é necessário ser-se cauteloso.

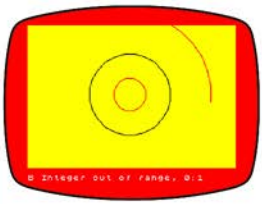
Mude outra vez a cor:

INK 0 ENTER



Agora vai traçar um círculo, premindo a tecla H com SYMBOL SHIFT (depois de ter obtido o cursor **E** no ecrã). Mas precisa de três números — os dois primeiros (coordenadas) para localizar o centro do círculo no PAPER, a começar da esquerda e de baixo para cima, e o terceiro número para indicar o comprimento do raio da circunferência.





A posição 125 é quase metade da largura do **PAPER** e a posição 90 está perto de metade da altura. 50 parece ser um número razoável para desenhar um círculo. Assim:

CIRCLE 125,90,50 ENTER

E se desenhasse um círculo mais pequeno, noutra cor e com o mesmo centro?

INK2 ENTER

CIRCLE 125,90,20 ENTER

E agora, um bastante maior:

CIRCLE 125,90,100 ENTER

Quando pretender traçar um círculo que saia dos limites do **PAPER**, o computador dar-lhe-á a seguinte mensagem:

B Integer out of range (número inteiro fora dos limites).

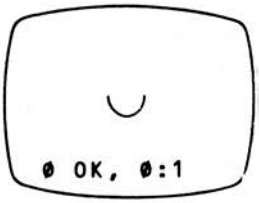
É uma boa altura para limpar tudo do computador, teclando

NEW ENTER

e recomeçar a praticar qualquer tipo de desenho que deseje, através dos comandos **DRAW**, **CIRCLE**, **INK**, **PAPER** e **BORDER**.

Pode ainda experimentar mais duas coisas:

1. Poderá definir o local onde deseja começar a traçar (**DRAW**) uma linha. Use o comando **PLOT**.
   PLOT 50,100 colocará um pequeno ponto no ecrã, 50 posições (pixels) para a direita e 100 posições para cima, a partir do canto inferior esquerdo original, ou a partir da última posição atingida por um **DRAW** anterior, ou, ainda, a partir do «pixel» mais à direita, obtido por um **CIRCLE** anterior.



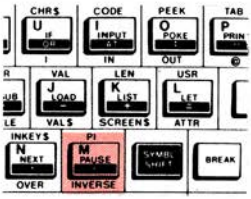
PLOT 127,87 ENTER

colocará um ponto no centro do ecrã. Comece aí o próximo exercício.

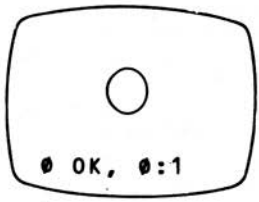
2. É possível traçar um arco de círculo se adicionarmos um terceiro número ao comando **DRAW**. Uma vez que tenha determinado o ponto de partida, os dois números que estão em primeiro lugar, a seguir ao **DRAW**, continuarão a seleccionar o ponto onde se termina, e terceiro número escolherá uma parte de um círculo (descrevendo o ângulo coberto por esse arco em radianos).

DRAW 5Ø ,Ø, PI ENTER

(PI é o «comando» que está localizado acima da tecla M e obtém-se com o cursor). A ordem que acabou de dar fará traçar metade de um círculo numa posição afastada 5Ø pontos na horizontal (lado direito do ecrã).



Porquê a metade inferior de um círculo e não a metade superior? — Porque todos os círculos são traçados no sentido inverso dos ponteiros de um relógio, no TC 2048.

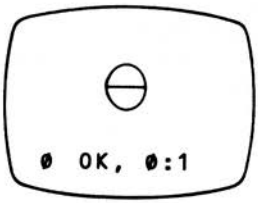


O que aconteceria se tivesse composto

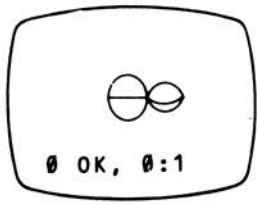
DRAW – 5Ø ,Ø ,PI ENTER

Tente e veja o que acontece. Depois tente também

DRAW 5Ø ,Ø ENTER

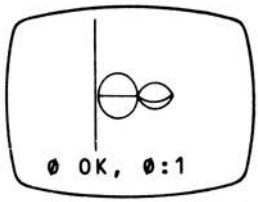


Um círculo completo poderia ter-se traçado se tivesse composto o terceiro factor como 2*PI. Também poderia traçar um quarto de círculo com PI/2 ou 5*PI. Dado que os pontos inicial e final são determinados previamente, uma porção mais pequena do círculo produziria um arco de um círculo maior. Faça algumas experiências para se treinar:



«Tecle»

```
INK 1                ENTER
PLOT 125,0           ENTER
DRAW 0,175           ENTER
```




```
DRAW 50,0,.8*PI      ENTER
DRAW -50,0,.5*PI     ENTER
DRAW 50,0,.3*PI      ENTER
```

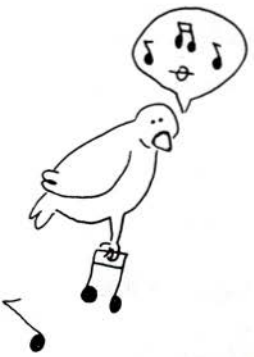
Verifique por si mesmo. Continue a experimentar diversas coisas ao mesmo tempo que vai seguindo este Manual. Não dedicaremos muito tempo ao aspecto das cores quando começarmos a tratar da programação em BASIC.
Deveria também tentar acrescentar aos programas que lhe sugerimos alguns **INK**, **PAPER** e **BORDER** para melhorar eventualmente o aspecto gráfico dos resultados.
(Aquilo que deverá obter com as ordens que deu ao computador, há pouco, é uma linha vertical, azul, quase no centro do ecrã).

Sumário

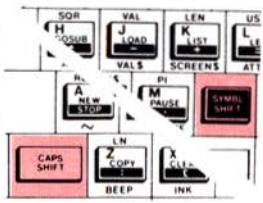
1. **PLOT** coloca um ponto no ecrã (**PAPER**) numa localização que se determina por dois números (coordenadas): o primeiro deles escolhe a posição quanto à largura do ecrã e vai de zero (totalmente à esquerda) até 255 (totalmente à direita) e o segundo número, separado por uma vírgula, fixa a altura e vai de zero (totalmente em baixo) a 175 (no topo). Quando o computador acaba de ser ligado ou depois de **NEW** seguido de **ENTER**, a posição de **PLOT** é sempre (0,0) e 175 (no topo). Quando o computador acaba de ser ligado ou depois de **NEW** seguido de **ENTER**, a posição de **PLOT** é sempre (0,0) e corresponde ao canto inferior esquerdo do **PAPER**.
2. **DRAW** traça uma linha a partir do último ponto onde se situa o **PLOT** para um novo canto do ecrã especificado por dois números, ou coordenadas, respectivamente ao lado e para cima ou para baixo dessa posição inicial. Se acrescentar um terceiro número, poderá desenhar um arco de círculo. (A posição do **PLOT** mover-se-á para novo local, depois do **DRAW**).
3. **CIRCLE** traça um círculo num local especificado poir dois números, identicamente ao que se viu em **PLOT**, mas carece de um terceiro número que indique (em número de «pixels») o respectivo raio.

Síntese do Capítulo

Neste Capítulo, aprenderá a compor e a tocar música com a instrução BEEP



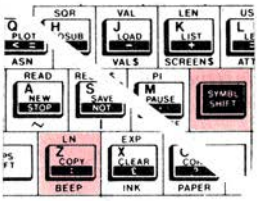




Prima CAPS SHIFT e SYMBOL SHIFT

O seu Timex Computer 2048 dispõe de um meio simples de produzir sons através do «comando» BEEP.

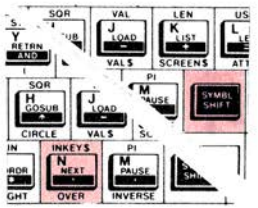
Pressione ao mesmo tempo as teclas CAPS SHIFT e SYMBOL SHIFT para conseguir entrar em «modo» extensivo com o cursor E.



Premir SYMBOL SHIFT e Z

Em seguida, ao mesmo tempo que mantém premida a tecla **SYMBOL SHIFT**, toque na tecla Z. Obterá o comando que está indicado abaixo da tecla:

BEEP

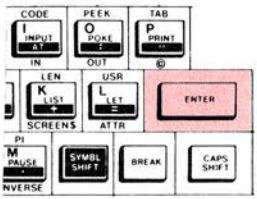


Prima SYMBOL SHIFT e N

Depois toque na tecla 1. Logo depois, uma vírgula — com **SYMBOL SHIFT** e a tecla N. Finalmente, tem de premir a tecla Ø . O ecrã deve apresentar-se assim:

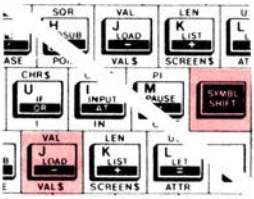
BEEP 1,Ø

BEEP é um comando que necessita de dois números, separados por uma vírgula, para poder funcionar. O primeiro número indica o tempo ou duração da nota (em segundos) e o segundo número a frequência. Zero (Ø) é a frequência que corresponde à nota DÓ CENTRAL (ou C em notação musical clássica).

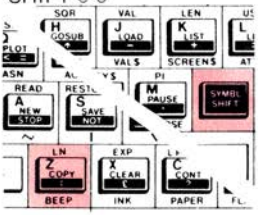


Se premir a tecla **ENTER** obterá um DÓ CENTRAL durante 1 segundo.

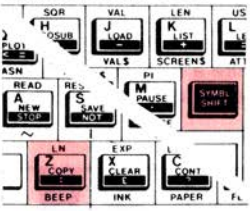
Breve lição de música: A escala musical do ocidente, de oito notas, é constituída na base de 12 meios-tons (acreditem) com dois intervalos de meio tom entre cada nota à excepção de MI para FÁ (4-5) e de SI para DÓ (11-12) onde há apenas um intervalo.

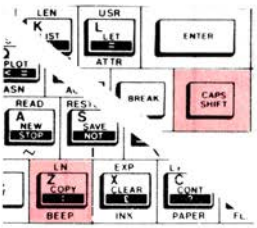


Para notas abaixo do Dó central prima **SYMBOL SHIFT** e J



Para dois pontos prima **SYMBOL SHIFT** e Z





Se pretender tocar notas acima do DÓ central deve contar meio tom por cada número. As notas abaixo do DÓ central são contadas como números negativos (usando o símbolo menos (–) que se obtém com **SYMBOL SHIFT** e J).

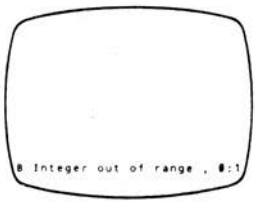
Também poderá utilizar os dois pontos (**SYMBOL SHIFT** e Z) para colocar uma série de comandos juntos. Desta forma pode conseguir tocar toda a escala natural (DÓ, RÉ, MI, FÁ, SOL, LÁ, SI, DÓ) se escrever

BEEP 1,0:**BEEP** 1,2:**BEEP** 1,4:**BEEP** 1,5:
BEEP 1,7:**BEEP** 1,9:**BEEP** 1,11:**BEEP** 1,12

(Verifique agora que, de facto, há dois meios tons entre todas as notas com excepção de MI a FÁ (de quatro passa para cinco) e de SI a DÓ (de 11 passa para 12), em que só há meio tom de intervalo.
(É possível apressar a introdução destes dados no teclado se se aperceber de que, depois do segundo número de cada par, poderá manter premida a tecla **SYMBOL SHIFT** e tocar no Z, **CAPS SHIFT** e Z de novo, para conseguir os dois pontos (:) e logo a seguir o **BEEP**).

Depois poderá fazer «tocar» o TC 2048 bastando premir **ENTER**.

Se desejar . pode tentar várias tonalidades e várias durações; pode usar pontos decimais para tocar durante fracções de segundo até fracções de tons (no caso de apreciar música oriental ou indiana). Experimente



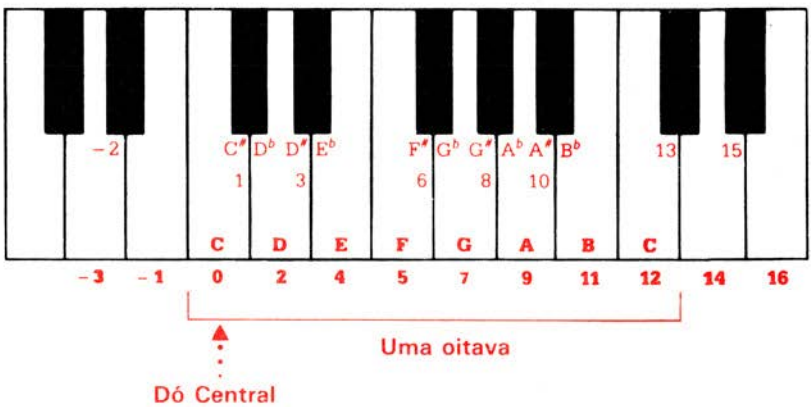
BEEP .5,9.77
BEEP 1.89,14

Virá a notar que 10 segundos é a duração máxima de qualquer nota tocada através de **BEEP**. Do mesmo modo pode verificar que a nota mais baixa que pode tocar é — 60 enquanto a mais elevada se situa a 69.
Se tentar números não compreendidos nestes limites receberá uma mensagem

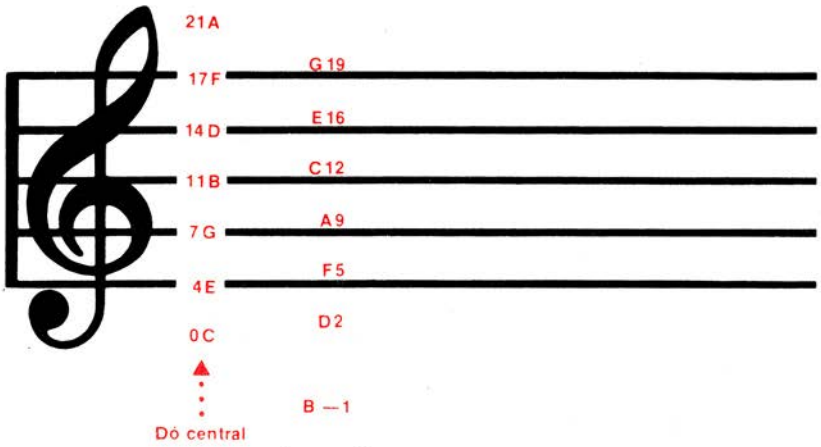
B Integer out of range 0:1 (B — Número inteiro fora dos limites).

Os sons musicais tocados através de **BEEP** podem ser escutados não apenas através do altifalante interno do TC 2048, mas também através de um outro sistema reprodutor de som que receba os sinais a partir da tomada MIC do computador. Poderá ligar um amplificador de som ao MIC e reproduzir as músicas através de grandes altifalantes externos.

Aqui estão dois diagramas que mostram o relacionamento existente entre a frequência e as teclas de um piano ou de outro instrumento musical de teclado.



Acrescente 12 a cada um dos valores quando pretender tocar uma oitava acima ou deduza 12 quando desejar uma oitava abaixo.



Deve acrescentar-se 1 à nota que pretenda tocar-se em posição de «sustenido»(#) e subtrair quando se pretenda um «bemol» (b).

Exemplo: **BEEP** 1,Ø **ENTER** (produz um DÓ NATURAL);
BEEP 1,1 **ENTER** (produz um DÓ SUSTENIDO);
BEEP 1,-1 **ENTER** (produz um DÓ BEMOL).

SUMÁRIO

1. **BEEP** seguido de dois números separados por uma vírgula produz uma nota musical. O primeiro número indica a duração da nota em segundos e o segundo número a sua posição na escala musical. Podem utilizar-se números decimais. O zero (Ø) produz o DÓ central (C da notação clássica). Por unidade positiva adicionada a zero aumenta-se meio tom acima do DÓ natural e por uma unidade deduzida dá-se o inverso, reduzindo-lhe meio tom.

Como Escrever Um Programa 8

Síntese do Capítulo

Este capítulo vai ensiná-lo a escrever o seu primeiro programa, socorrendo-se de NEW para começar, GOTO, para repetir, BREAK para parar e CONT para continuar.

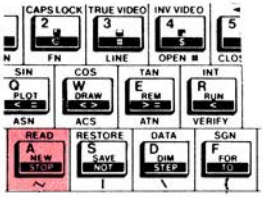


Estamos agora em condições de escrever o nosso primeiro programa para computador.

No último Capítulo exercitámo-nos usando o computador em «modo imediato», o que significa que o computador executa cada um dos comandos imediatamente (depois de premir a tecla ENTER).

Quando escrevemos programas, damos um certo número de ordens e o computador executa-as, umas após outras, no momento em que lhe dissermos para o fazer.

O facto de começarmos por um pequeno programa não significa que não haja programas com algumas centenas de linhas de programação que levam os computadores a executar complexas e longas tarefas. A força do computador reside na capacidade que tem de receber, armazenar e tratar diferentes e complexos programas.



Prima **NEW** para limpar o ecrã e a memória do computador.

Vamos começar. Se acabou de ligar o seu computador para iniciar este Capítulo, já tem um cursor **K** ou a mensagem de «copyright» no ecrã — a mensagem que, como vimos, encobre o cursor.

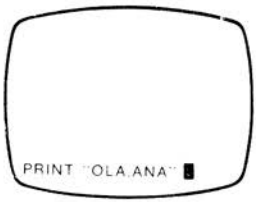
Se, contudo, manteve o computador ligado desde o último Capítulo que estivemos a ver, então «tecle»

NEW

(a tecla da letra A com o cursor **K** ou qualquer mensagem também ocultam o cursor **K**) e

ENTER

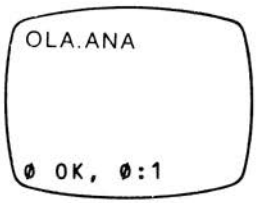
Isto limpará tanto o ecrã como a memória do computador, de forma a colocá-lo em posição de iniciar um novo (**NEW**) programa.



«Tecle»

PRINT "OLA, ANA" (ou qualquer outro nome que prefira).

Depois toque na tecla **ENTER**. Tal como no Capítulo anterior, o computador executa a ordem imediatamente. (A propósito, pode até escrever o seu próprio nome se por acaso não fôr ANA...).

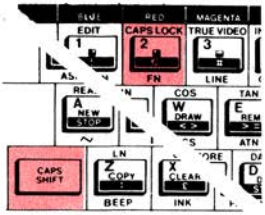


Agora escreva

1Ø **PRINT** "OLA, ANA"



Repare que o cursor **K** não mudou enquanto premiu a tecla 1 e o zero (Ø). É esta a razão por que não existem «comandos» nas teclas numéricas para lhe permitir numerar as suas linhas de programação. O cursor só mudará para **L** depois de premir a tecla P para obter o comando **PRINT**.

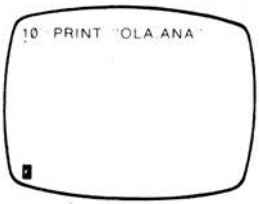


Prima **CAPS SHIFT** e 2 para obter **CAPS LOCK**

Se por acaso souber dactilografia, recomendamos-lhe muito cuidado com a tecla 1. Ao contrário do que acontece com muitas máquinas de escrever, que usam l (l minúsculo) para escrever a número 1, no computador não é assim. Há uma tecla para o número 1 e outra para o L. Cada caracter ou cada «comando» tem unicamente um significado para o TC 2048.

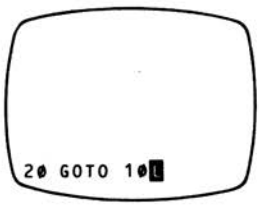
De facto, é melhor que escrevamos todo o nosso programa com letras maiúsculas. Mantenha premida a tecla **CAPS SHIFT** e depois toque na tecla 2 para passar a **CAPS LOCK** (tecla de maiúsculas) e deixe ficar nessa posição enquanto escrever programas. Continuará a ter acesso aos números e poderá continuar a utilizar **CAPS SHIFT** e **SYMBOL SHIFT** para obter outros símbolos e sinais de pontuação. Como dissemos no Capítulo 2, no «modo» **CAPS LOCK** todas as letras serão maiúsculas.

Tenha também cuidado com o numeral zero (Ø) e com a letra O. Os zeros são representados por um símbolo semelhante ao O mas cortado por uma barra oblíqua (Ø). É importante distingui-los e nunca usar o O por um zero. (É um facto comum na linguagem Informática).
Agora faça **ENTER** e observe que toda a linha de programação passou para o topo do **PAPER**, no ecrã... em vez de simplesmente as palavras OLÁ, ANA.

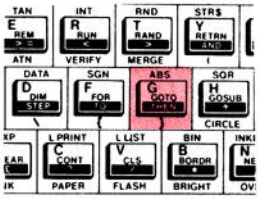


Notará também que aparece um símbolo novo entre o número 1Ø da linha e a palavra **PRINT**. É o «cursor do programa» e tem a forma de um V no sentido horizontal (>). Coloca-se automaticamente na linha que mais recentemente entrou no programa.

A única diferença entre a primeira fila que compusemos no teclado do computador e a segunda que agora produzimos é que, nesta última, há um «número de linha». Quando colocamos um número antes de um comando, ele torna-se numa «linha de programação» e não se executa imediatamente.
O cursor esta de novo pronto, no fundo do ecrã. «Tecle» o seguinte:



2Ø GOTO 1Ø ENTER




```
10 PRINT "OLA ANA"
20 GOTO 10
```

```
OLA ANA
OLA ANA
OLA ANA
OLA ANA
OLA ANA
OLA ANA
OLA ANA
OLA ANA
scroll?
```

```
D BREAK--CONT repeats, 10:1
```

Observe que **GOTO** é uma «comando» que se encontra na tecla G e que, portanto, não deve ser escrita letra a letra. (De facto ela surge no ecrã logo que toque na tecla G). Repare também que o cursor aparece agora na linha 20 .

Agora já dispõe de um curto, mas completo... programa.
O comando da linha 20 diz simplesmente ao computador para regressar à linha 10 onde tudo começa de novo. Pode fazer uma ideia do que vai acontecer quando «correr» o programa?

RUN
(«comando» na tecla R) e
ENTER

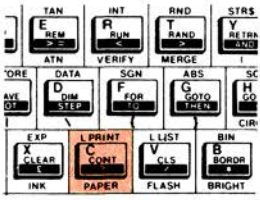
Que tal? Há imenso material escrito no ecrã se o compararmos com um programa tão pequeno. Como lhe dissemos o computador é rápido, preciso e infatigável.

Um dos comandos mais poderosos do BASIC é o **GOTO** que dirige o computador directamente para uma linha determinada do programa, que não a seguinte. O **GOTO** é muitas vezes utilizado para fazer com que o computador regresse a uma linha anterior do programa e repita um determinado processo de tratamento (Iteração em Informática).

A pergunta "**scroll?**" que está impressa no fundo do ecrã informa-o de que o ecrã esta cheio. Se não desejar continuar a imprimir texto basta que responda tocando na tecla N (não) ou **BREAK** e o computador parará a execução do programa com a mensagem.

D BREAK — CONT repeats, 10 : 1

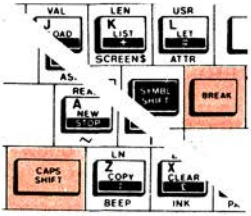
Se a seguir premir a tecla C (**CONT**inue), para continuar, notará que depois de um ligeiro bruxulear na linha do fundo do ecrã, o



computador torna a parar com a mesma pergunta "**scroll?**"
O que realmente sucede é que se imprimem mais 22 linhas com a frase "OLA, ANA" no ecrã, para cima.

Isso é o que acontecerá, também, se premir qualquer tecla que não seja o N (de não) ou **BREAK** em resposta à pergunta "**scroll?**".

(A propósito, esta característica torna-se muito mais útil num programa onde a informação se modifica — tal como num programa de contagem onde cada uma das 22 linhas é de maior valor — do que quando a informação é simplesmente repetida).



Prima **CAPS SHIFT** e **BREAK**

Outra coisa ainda. Pare o programa com **BREAK** e depois recomece a «corrê-lo» através de **RUN** e **ENTER**. Enquanto o computador estiver a imprimir "OLA, ANA" de cima para baixo, no ecrã, toque na tecla **BREAK** (deve premir ao mesmo tempo **CAPS SHIFT**) e repare que a listagem da frase pára, seja em que ponto estiver, quando premir a tecla **BREAK**.

Terá de ser rápido. Experimente premir **RUN** e depois manter premida a tecla **CAPS SHIFT** enquanto faz **ENTER** e, logo a seguir, rapidamente, **BREAK**.

O computador verifica se, depois de ter executado as ordens de cada linha de programação, alguém premiu a tecla **BREAK**. Se isso tiver acontecido ele pára o programa.

(Com a tecla **BREAK** também é possível fazer parar um programa de iteração contínua, — como por hipótese um «ciclo (loop) sem fim» àcerca do qual falaremos mais tarde — ou um programa que falhou no **LOAD** da «cassette». Se o **BREAK** não funcionar, poderá eventualmente ter de recorrer ao interruptor da corrente, desligando-o e tornando a ligá-lo, mas, naturalmente, sabendo desde logo que, nessa eventualidade, perderá toda a informação que o computador tiver recebido).

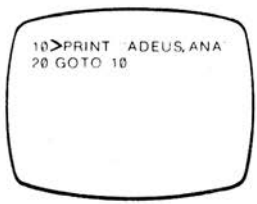
Note bem que quando pretender fazer **BREAK** terá de premir ao mesmo tempo a tecla **CAPS SHIFT**.

CONTinue permitir-lhe-á CONTinuar quando o ecrã estiver cheio, quanto tiver interrompido o programa através de BREAK ou STOP, ou, ainda, quando o programa se interromper por si próprio com a mensagem STOP. (O comando STOP — SYMBOL SHIFT e A — utiliza-se na programação, em vez do BREAK, ou quando o computador estiver a aguardar uma entrada de dados. Voltaremos a falar de STOP, mais tarde. BREAK usa-se quando o programa está a «correr»).

Muito bem. Quando tiver praticado suficientemente com BREAK e CONT, deixe que o programa pare com o ecrã cheio, toque no BREAK e depois faça outra vez ENTER.

Cá tem outra vez o seu programa no ecrã. Naturalmente que pode tornar a «corrê-lo». (Vamos a isso: RUN e ENTER).

Também poderá deixar que o programa fique na memória do computador e ir fazer qualquer outra coisa. Ele ficará aí até que o apague ou que desligue a máquina.



10 PRINT "ADEUS, ANA"

e torne a premir a tecla ENTER.

A nova linha 10 substituirá a linha 10 que anteriormente estava no programa. (E o cursor do programa indicar-lhe-á que a linha 10 foi a última que deu entrada, embora não seja a última em relação à ordem numérica na listagem que está no ecrã).
Não poderá existir mais do que uma linha 10 no programa e sempre que entrar uma nova linha com o mesmo número perderá a antiga.
Isto significa que, se deixar ficar um programa no computador TC2048 (em vez de recomeçar de novo com NEW), correrá o risco de fazer com que o novo programa apague o anterior. Ou, pior do que isso, obter um entrelaçado de linhas antigas com linhas novas.

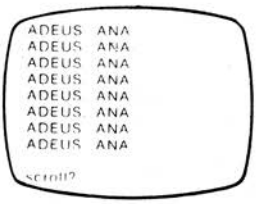
Experimente isto:

5 **REM** PROGRAMA — ADEUS

e faça **ENTER**

A linha 5 foi inserida no programa no local que efectivamente lhe pertence, segundo a ordem numérica. É esta a razão pela qual costumamos programar com linhas numeradas de dez em dez: isso garante-nos que fique espaço livre para inserir novas linhas, se acaso precisarmos de o fazer.

Agora «tecle» **RUN** e **ENTER**.



A linha 5 do programa, começando com o «comando» **REM** não «faz» absolutamente nada no programa. Qualquer linha que comece por **REM** — **REM** (significa **Rem**arck ou **REM**inder, e quer dizer observação) — é colocada na listagem do programa apenas para o recordar de algo importante ou ajudá-lo a entender a programação.

Lembre-se de que as letras maiúsculas são diferentes das minúsculas para o TC 2048; é aconselhável que utilize nos **REM**s letras da mesma espécie que usa no nome do programa quando faz o seu «carregamento» (**LOAD**) ou quando o grava (**SAVE**).



Um outro exemplo para treinar:

15 **PRINT** "ATE A VISTA"

e faça **ENTER**.

Depois «corra» o programa (**RUN**) e torne a premir a tecla **BREAK** para o parar. Faça novamente **ENTER** para que a listagem do programa surja no ecrã. Compreende agora a razão por que o programa origina aquilo que viu?

Nesta altura é muito possível que já queira escrever o seu próprio programa, ainda que simples, utilizando quer linhas de "strings" (cadeia de carácteres para imprimir, quer cálculos de matemática. Poderá até misturar os dois tipos de actividade num só programa. Lembre-se de que o computador fará exactamente aquilo que lhe disser para fazer, mesmo que o resultado pareça sem qualquer nexo).

SUMÁRIO

1. **NEW** apaga tudo aquilo que possa ter entrado no computador e dará espaço livre para novo programa.
2. Quando coloca números antes dos «comandos», estes tornam-se em declarações de programação e não sã executados imediatamente. Em vez disso são guardados por ordem do número de linha e executados posteriormente em resposta à ordem **RUN**.
3. **GOTO** é um «comando» extremamente poderoso do BASIC. Dirige o computador para uma determinada linha de programação em vez de seguir para a linha seguinte na ordem numérica. Permite também que o computador itere (repita sectores de programação, uma ou diversas vezes seguidas) realizando uma determinada tarefa.
4. **BREAK** faz parar o programa no ponto em que estiver a ser executado.
5. **CONT**inue faz recomeçar o programa, a partir do sítio onde tiver parado anteriormente (seja porque se parou com **BREAK**, seja porque o ecrã estava cheio, por hipótese).
6. «**scroll?**» é uma pergunta que nos é posta pelo computador quando já não pode escrever mais nada no ecrã, por este estar cheio. Cabe-lhe a si decidir se deve parar («teclando N ou **BREAK**) ou seguir com a execução do programa, premindo outra qualquer tecla.

Como Organizar o ecrã 9

Síntese do Capítulo

Este Capítulo mostra-lhe como poderá usar EDIT, AT, TAB, vírgula e ponto e vírgula, para movimentar o texto no ecrã. Também utilizaremos as setas para cima e para baixo.



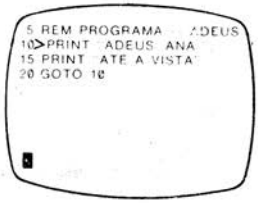


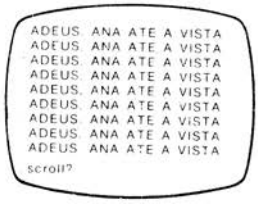
Pode fazer maravilhas com os sinais de pontuação num programa de BASIC para o TC 2048. Por hipótese, comecemos com o programa que escrevemos no último Capítulo.

```
 5 REM PROGRAMA — ADEUS
10 PRINT "ADEUS, ANA"
15 PRINT "ATE A VISTA"
20>GOTO 10
```

O Cursor do Programa e o comando EDIT

Vamos juntar alguma pontuação ao programa.
Se acaso tornou a escrever o programa para esta lição, poderá ver que o «cursor do programa», que parece um V no sentido horizontal (>), está na linha 20 visto ser esta última linha que introduziu.
Mas carregue na tecla da seta que aponta para cima (CAPS SHIFT e 7) e verá que o cursor se desloca para a linha 15.
Se voltar a premir a tecla da seta referida, o «cursor do programa» sobe para a linha 10.





(Mas, se por acaso está a começar este Capítulo exactamente após ter lido o anterior e ainda tem TC 2048 como estava, então o cursor estará na linha 15. (Com a seta, leve-o até à linha 10).
Agora (com **CAPS SHIFT** e tecla 1) faça **EDIT** e verá surgir a linha 10 no fundo do ecrã, no espaço reservado às linhas de trabalho.

A Vírgula
Mova o cursor até ao fim da linha, premindo repetidamente a tecla da seta que aponta para o lado direito (**CAPS SHIFT** e 8). Quando estiver no fim da linha introduza uma vírgula (**SYMBOL SHIFT** e N) e faça **ENTER**. Não parece muito diferente. Mas faça **RUN** e **ENTER**.
O TC 2048 tem um ecrã (**PAPER**) que comporta 32 caracteres a toda a largura (mas que são identificados de 0 a 31 e não de 1 a 32).
Acontece que a vírgula ordena a impressão do texto para o princípio da segunda metade do ecrã. Visto que juntou uma vírgula ao fim da linha 10, o texto da linha 15 passa a ser impresso na segunda metade do ecrã.
Mas se acrescentar também uma vírgula no fim da linha 15, da mesma forma que vimos, a apresentação das frases, no ecrã, não muda.
Uma declaração **PRINT** sem qualquer pontuação, desloca o ponto de impressão para o início da próxima linha.
Uma vírgula, no fim de um comando **PRINT**, move o ponto de impressão ou para a coluna 16 ou para a coluna 0 da seguinte linha conforme se encontrar mais próximo de uma ou de outra posição.

Observação: Uma vírgula entre aspas é impressa como vírgula. Quando estiver fora das aspas faz deslocar o ponto de impressão para o início da próximo metade do ecrã.

O Ponto e Vírgula
Utilizando a mesma técnica que acabámos de ver (setas, cursores, tecla **EDIT**) e também a tecla **DELETE**, substitua a vírgula, no fim da linha 10, por um ponto e vírgula:

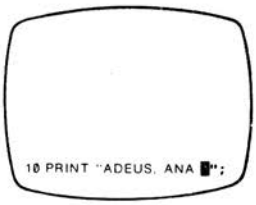
```
5 REM PROGRAMA — ADEUS
10 PRINT "ADEUS, ANA";
15 PRINT "ATE A VISTA"
20 GOTO 10
```

Mova o cursor de programa para a linha 1Ø com a seta (**CAPS SHIFT** 6 ou 7).
Traga a linha 1Ø para a zona de trabalho, no fundo do ecrã, com a tecla **EDIT** (**CAPS SHIFT** e 1).
Mova o cursor até ao fim da linha, servindo-se da seta para o lado direito (**CAPS SHIFT** e 8).
Apague a vírgula com **DELETE** (**CAPS SHIFT** e Ø).
Introduza um ponto e vírgula (**SYMBOL SHIFT** e O. — Note que é a letra O e não o zero (Ø).
Faça **ENTER**.
Agora «corra» (**RUN**) o programa.

```
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
ADEUS, ANAATE A VISTA
scroll?
```

Hummmm. O que é que temos aqui?
O ponto e vírgula move a posição de **PRINT** para o espaço logo a seguir à última impressão.
Temos, portanto, de encontrar um meio de deixar um espaço entre duas frases.
A forma como pode consegui-lo é colocar um espaço em branco entre as aspas. Faça como já sabe:
Premir **ENTER** para obter a listagem do programa.
Mover o cursor do programa para a linha 1Ø.
Trazer a linha cá para baixo, servindo-se de **EDIT**.
Deslocar o cursor, com o auxílio da seta da tecla 8, para a posição imediatamente anterior às aspas que fecham a frase.

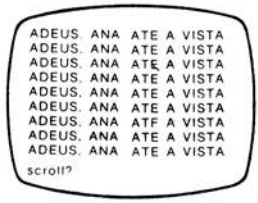
```
10 PRINT "ADEUS,
```



Agora pressionou a tecla **SPACE** para produzir o desejado espaço em branco que vai ficar inserido no local onde estava o cursor.



Faça **ENTER**



E agora «corra» o programa, **RUN** e **ENTER**.
Nota: Se, por qualquer razão, pretender separar dois números, terá de colocar espaços «entre aspas», separados por ponto e vírgula.

```
PRINT 1234          imprime  1234
PRINT 12  34        imprime  1234
PRINT 12; " ";34    imprime  12  34
```

Não se esqueça do ponto e vírgula.

O Apóstrofo

Existe ainda um outro sinal de pontuação que também actua como «controlador de caracteres» para fazer deslocar o ponto de impressão. Trata-se do apóstrofo (**SYMBOL SHIFT** e 7) que move a posição de impressão para a próxima linha. Assim, em vez de compor:

```
10 PRINT "OLA"
20 PRINT "ALI"
```

Poderia escrever, numa única linha,

```
10 PRINT "OLA" ' "ALI"
```

para obter exactamente o mesmo efeito.

Mas há outras formas de posicionar a impressão de caracteres no ecrã. Vamos experimentá-las.

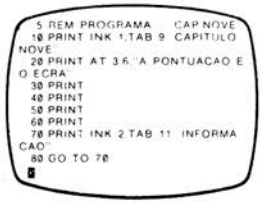
TAB e AT

Antes de tudo limpe o computador com **NEW** e **ENTER**.
Depois, introduza o seguinte programa:

```
 5 REM PROGRAMA — CAP. NOVE
10 PRINT INK 1;TAB 9;"CAPÍTULO NOVE"
20 PRINT AT 3,6;"A PONTUAÇÃO E O ECRÃ"
30 PRINT
40 PRINT
50 PRINT
60 PRINT
70 PRINT INK 2;TAB 11;"INFORMAÇÃO"
80 GO TO 70
```



Agora faça **RUN** e **ENTER**. Linda página, hein?
Observação: Quando premir a tecla N (para dizer Não) ou fizer **BREAK**, em resposta à pergunta do computador «**scroll?**» que aparece no fundo do ecrã, este não «rola» e poderá fazer, portanto, qualquer outra coisa.
Se premir qualquer outra tecla o programa fará «rolar» o ecrã para imprimir as 22 linhas seguintes Tem de parar o movimento (scroll) com a tecla N (ou com **BREAK**) antes de poder fazer qualquer outra coisa com o computador.



Examinemos agora o programa, linha a linha.
Faça **ENTER** para poder observá-lo no ecrã.
A linha 5 é apenas o nosso **REM** que contém o nome do programa (que é o mesmo que utilizámos para o gravar ou «carregar» **SAVE** ou **LOAD**).
REM não é absolutamente necessário mas é um hábito que convém adquirir.
A linha 10, com a mensagem CAPÍTULO NOVE começou a imprimir-se a partir da coluna 9 por determinação de **TAB** 9. **TAB** (Tabulador) funciona como numa máquina de escrever com uma diferença: indica o número exacto da coluna. Lembre-se de que para obter **TAB** tem que entrar no «modo» extensivo

```
5 REM PROGRAMA - CAP NOVE
10 PRINT INK 1;TAB 9; CAPITULO
NOVE
20 PRINT AT 3,2; A PONTUACAO E
O ECRA
30 PRINT
40 PRINT
50 PRINT
60 PRINT
70 PRINT AT 10,11;"INFORMACAO"
80 GO TO 70
```

(premindo as duas teclas de **SHIFT** e depois, isoladamente, a tecla P). Não deve escrever a palavra **TAB** letra a letra.

Repare que o ponto e vírgula, depois de **TAB** 9, coloca a posição de impressão na coluna 9. Se, em seu lugar, estivesse uma vírgula, a posição de impressão passaria automaticamente para 16 e anularia o efeito do **TAB** 9. Se lá não existisse nem vírgula nem ponto e vírgula, então, a linha de programação seria rejeitada pelo computador que daria a mensagem de «erro de sintaxe».

Exactamente por essa razão se faz seguir o comando **INK** de um ponto e vírgula. **PRINT**, "RUI" (com uma vírgula antes de "RUI"). (Pode colocar-se a vírgula à frente do **PRINT**).

A linha 2Ø orienta a impressão para um local determinado do **PAPER**, através de **AT** (em). A definição do local faz-se através de dois números: o primeiro indica a linha e o segundo, depois da vírgula, a coluna onde deve imprimir-se o texto. (Neste caso indicou-se 3 linhas a contar de cima e 6 colunas a contar da esquerda). **AT** também é um «comando» pelo que não deve ser escrita com um A seguido de um T mas sim através da tecla I com **SYMBOL SHIFT**. Repare, novamente, na existência do ponto e vírgula.

Dado que o **INK** 1, na linha 1Ø, se encontra inserido numa linha de programação, após um **PRINT**, a sua função é apenas especificar a cor com que se pretende imprimir aquele texto (e só aquele).

Na linha 2Ø retoma-se a posição normal de **INK** que é o Ø (preto) na ausência de outra indicação.

As linhas de 3Ø a 6Ø reproduzem uma impressão a branco e, ocupando a respectivas quatro linhas, colocam a posição de impressão na linha anterior à 7Ø.

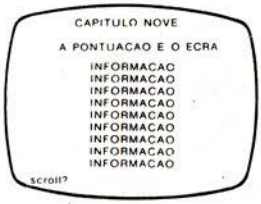
A linha 7Ø imprimiu-se na coluna indicada por **TAB** e a cor dos caracteres desse texto é o encarnado, por determinação de **INK** 2.

A linha 8Ø faz com que o programa repita os comandos da linha 7Ø até que o ecrã se encha e o computador pare.

Será capaz de modificar a linha 7Ø para que passe a ser: **PRINT AT** 1Ø,11;"INFORMACAO"




```
 5 REM PROGRAMA - CAP.NOVE
 10 PRINT INK 1; TAB 9;"CAPITULO
NOVE"
 20 PRINT AT 3,6;"A PONTUACAO E
O ECRA"
 30 PRINT
 40 PRINT
 50 PRINT
 60 PRINT
 70 PRINT AT 10,11;"INFORMACAO"
 75 PRINT INK 2; TAB 11;"INFORMA
CAO"
 80 GO TO 75
```



Experimente isso e depois faça **RUN**.

Sugestão: Terá de utilizar o **BREAK** para conseguir parar o programa. Isto porque as linhas 70 e 80 formam um «ciclo sem fim», visto que a linha 70 manda que se imprima sempre no mesmo local (linha 10 do ecrã).
Haverá alguma alteração se juntar uma vírgula no final da linha 70? Porquê?
E se acrescentar uma linha 75, idêntica à linha 70, alternando o **GOTO** da linha 80, para ficar assim:

```
70 PRINT AT 10,11;"INFORMAÇÃO"
75 PRINT INK 2; TAB 11;"INFORMAÇÃO"
80 GOTO 75
```

Experimente. Faça **RUN** e **ENTER**.

Como imprimir aspas

Poderá ter levado algum tempo a pensar sobre a questão de como imprimir aspas, se elas no computador, servem para indicar onde começa e onde acaba uma «string».
Se introduzir duas aspas juntas depois de ter iniciado um «string» com as aspas usuais, então o computador imprimi-las-á.
Eliminemos agora a linha 75, mas por um novo processo. Basta compor 75 no teclado e premir a tecla **ENTER**.
Agora modifique outra vez a linha 80 para 80 **GOTO** 70.
(Basta que componha uma nova linha 80 e faça **ENTER** para que ela substitua a antiga). Depois



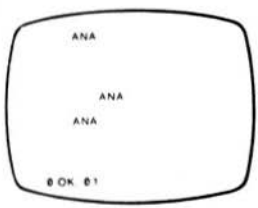
modifique a linha 7Ø para 7Ø **PRINT TAB** 11, """"INFORMAÇÃO"""" (depois de **TAB** 11, componha três vezes aspas com **SYMBOL SHIFT** e P e, depois de ter escrito INFORMAÇÃO, novamente três vezes **SYMBOL SHIFT** e P para obter mais 3 aspas.) Conseguirá um arranjo curioso na listagem do programa, mas obterá a impressão correcta quando o programa «rodar», imprimindo o texto.



Aqui está mais uma coisa que gostará de experimentar: — pode fazer a impressão de múltiplos textos apenas com um **PRINT**. Primeiro vamos ter de retirar o programa CAP. NOVE do computador. Faça **NEW** e **ENTER**. (É esta a última vez que o lembraremos para o fazer...). Depois, muito cuidadosamente, faça o seguinte programa de uma só linha!:

1Ø **PRINT TAB** 5;"ANA"; **AT** 5,1Ø;"ANA",,,**TAB** 5;"ANA"

Agora «corra» o programa (**RUN** e **ENTER**). Obterá um ecrã com impressão idêntica à que pode observar na imagem do lado. **TAB** 5 fez imprimir na primeira linha e na coluna 5. Depois o **AT** 5,1Ø fez imprimir na linha 5, coluna 1Ø. As quatro vírgulas fazem mover a posição de impressão primeiro para a linha 5, coluna 16, depois para linha 6, coluna Ø, a seguir para a coluna 16 da linha 6 e finalmente para a linha 7 coluna Ø. Quanto ao **TAB** 5, move a posição para a coluna 5 mas ainda na linha 7.



Experimente fazer por si próprio alguns arranjos deste tipo.

Linhas com declarações múltiplas: os dois pontos

Também é possível introduzir mais do que uma declaração numa única linha de programação, desde que se separem por dois pontos. Já fizemos isso quando usámos comandos (sem números de linha) logo no início do Manual.

Geralmente, mantendo cada declaração numa linha de programação isolada, o prorama fica mais compreensível e mais fácil de alterar, se for o caso.

Mas, por vezes, é necessário economizar a memória do computador e pode fazer-se isso, sem dificultar a compreensão do programa, se se combinarem declarações inter-relacionadas numa linha apenas. Faça **NEW** e **ENTER** e depois:



10 PRINT "ANA": GOTO 10

ENTER

RUN e ENTER

SUMÁRIO

1. Depois de uma ordem **PRINT**, não seguida de pontuação, a posição de impressão para o próximo **PRINT** será na linha seguinte.

   PRINT "ANA".

2. Uma «vírgula» depois de um **PRINT** move a posição do próximo **PRINT** para a segunda metade do ecrã (coluna 16) ou eventualmente, para o princípio da próxima linha (dependendo isso de o texto, acabado de imprimir no ecrã, ter ou não ultrapassado a primeira metade). Pode utilizar as vírgulas ara mover a posição de impressão até tão longe quanto deseje, correspondendo meia linha a cada vírgula.

   PRINT "ANA",,,

3. O «ponto e vírgula» move a posição de impressão apenas um carácter para a direita.

   PRINT "ANA";

4. O «apóstrofo» move a posição de impressão para o princípio da linha seguinte.

5. **TAB** coloca a posição do **PRINT** na coluna fixada pelo próprio **TAB**. Recorde-se, contudo, de que a primeira coluna tem o número zero, de que a coluna 9 corresponde à 1Ø e assim sucessivamente.

   PRINT TAB 1Ø;"ANA"

6. **AT** coloca a posição do **PRINT** nas coordenadas fixadas por dois números que se lhe seguem, separados por uma vírgula: o primeiro número indica a linha de impressão (zero a 21 contando de cima para baixo) e o segundo número, depois da vírgula, indica a coluna dessa linha (zero a 31 a começar da esquerda para a direita).

   PRINT AT 5,15;"ANA"

7. **TAB** e **AT** originam a deslocação do **PRINT** para o local determinado, «antes» da impressão do texto indicado para a respectiva linha; a «vírgula» e o «ponto e vírgula» movem a posição de impressão só «depois» de terem impresso o texto da linha, preparando a posição para o que se lhe seguir. Assim podemos programar os dois juntos:

   PRINT AT 5,15;"ANA",,

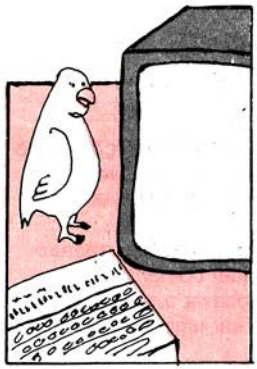
8. Poderá imprimir aspas dentro de uma «string» desde que introduza duas aspas para conseguir imprimir uma delas.

9. Os «dois pontos» (:) permitem introduzir múltiplas delcarações numa só linha de programação; depis de dois pontos o cursor **K** regressa, permitindo-lhe recomeçar uma nova instrução com um «comando».

Poupe Tempo e Espaço Usando Variáveis 10

Síntese do Capítulo

Neste Capítulo vai aprender a utilizar o comando LET para identificar números, palavras ou frases. Verá como «limpar o ecrã» com CLS e a «memória» com CLEAR. Além disso, aprenderá a maneira de introduzir «variáveis».



```
10 LET = 5328
20 LET A$ = "LUÍS"
30 PRINT A
40 PRINT A$
```

Escreva este pequeno programa. Recorde-se de que necessita da tecla **SYMBOL SHIFT** para conseguir $,=,e ". Depois faça **RUN** e **ENTER**. Reparou no que aconteceu?

```
5328
LUIS


0 OK, 40:1
```

Torne a premir **ENTER** e a listagem do programa aparecerá novamente no ecrã..

Capítulo 10: Poupe Tempo e Espaço Usando Variáveis

```
10 LET A=5328
20 LET A$="LUIS"
30 PRINT A
40 PRINT A$

K
```

As linhas 1Ø e 3Ø cumprem função idêntica à que conseguiria se se limitasse a escrever **PRINT** 5328. A letra A, quando precedida de **LET** e seguida de = torna-se uma «variável». No programa acima é evidente que o método consome mais tempo à memória para produzir o resultado que se pretende. Mas se tivesse um programa em que fosse necessário repetir o número diversas vezes, poderia usar a «variável» A em vez de todo o número 5328.
Seria mais fácil introduzir as instruções e economizar-se-ia um precioso espaço na memória do computador.

Nalguns programas, como veremos posteriormente, o número A pode sofrer alterações no decorrer do programa; por isso se chama «variável» ao A.

A declaração de programação:

LET A = 5328

denomina-se uma «declaração de atribuição» visto que atribui um valor (5328) a uma «variável». A letra A denominar-se-á nome da variável.

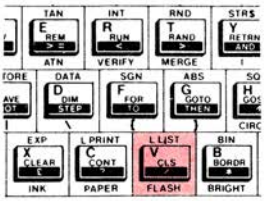
O nome de uma variável não tem de ser apenas uma só letra. Pode ter qualquer comprimento e conter letras ou números. Todavia o primeiro caracter da variável tem de ser uma letra.

A declaração poderia ter-se escrito:

LET NUMERO = 5328 ou
LET ESTE NUMERO = 5328 ou, ainda,
LET A5328 = 5328.
e assim sucessivamente.

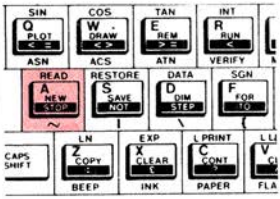
As linhas 2Ø e 4Ø fazem a mesma coisa que as linhas 1Ø e 3Ø com a diferença de que A$ é uma variável de cadeia («string»). (A propósito, diz-se «A string» em vez de «A dólar»).

As «string» podem ter qualquer comprimento — tudo quanto se encontre entre aspas — mas o nome de uma variável «string» só pode conter uma letra seguida de $. Poderá escrever:



LET A$="OLÁ"
LET B$="DESCENDO O RIO TEJO"
LET C$="5328"

Agora que tem o programa no ecrã, carregue na tecla V. No «modo KEYword» — ou modo de «comando», indicado pelo cursor , ele dará um CLS. Agora faça **ENTER**.

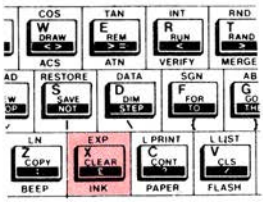


CLS quer dizer «CLear Screen», que é como quem diz, em português: limpe o ecrã. O programa desapareceu. Mas se tornar a fazer **ENTER** voltará e aparecer. Foi limpo do ecrã, mas continua guardado na memória do computador. Carregue depois na tecla A, para obter **NEW**, e faça **ENTER**. Depois volte a premir a tecla **ENTER**. Desta vez o programa desapareceu mesmo, não só do ecrã mas também da memória. **NEW** apaga tudo, ecrã e memória, ficando assim o TC 2048 pronto para receber nova programação. Vamos voltar ao «modo imediato» e tentar outra coisa. Escreva:

LET A = 5 ENTER

Depois torne a premir a tecla **ENTER**. Não há programa. Não está nada na memória do computador? Vamos ver: Faça

PRINT A ENTER



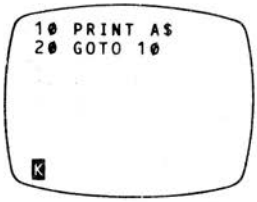
Bom. E esta? O computador grava as variáveis na sua memória! Claro que isso pode originar uma certa confusão, no fim de algum tempo, razão pela qual existe um comando para limpar a memória de variáveis nela contida. Faça

CLEAR ENTER e, depois,
PRINT A ENTER

O computador envia a mensagem

2, Variable not found (2, Variável não encontrada). Isto acontece porque, entretanto, limpámos a variável A, e o seu valor, da memória do computador, utilizando **CLEAR**.

Capítulo 10: Poupe Tempo e Espaço Usando Variáveis

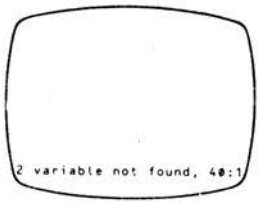


Façamos nova experiência. Introduza o seguinte programa:

10 PRINT A$
20 GOTO 10

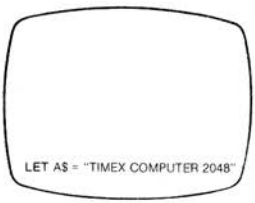
recordando-se, evidentemente, de que deve premir a tecla **ENTER** no fim de cada linha programada.

A partir de agora não vamos tornar a falar de **ENTER**! Já deve saber que tem de premir a tecla **ENTER** depois de qualquer comando directo ou no fim de cada linha de programação.



Faz ideia daquilo que acontece quando «correr» este programa? — Ora experimente.

(Não se esqueceu do **ENTER**?)

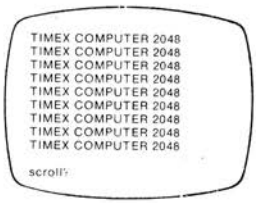


Uma mensagem codificada «2, Variable not found» novamente. Pois claro. Ainda não definiu a variável A$. Toque novamente na tecla **ENTER** desta vez para fazer surgir o programa no ecrã.

Agora escreva, sem qualquer número de linha:

LET A$ = "TIMEX COMPUTER 2048"

(podendo escrever qualquer outro texto entre aspas, se o preferir). Depois faça **ENTER** para poder observar o programa no ecrã, de novo. Verifique que o texto atribuído à variável A$ não apareceu no ecrã. (Claro, não deu qualquer número à linha quando fez a atribuição).



Agora «tecle» **GOTO** 10

GOTO 10 inicia o programa, fazendo-o «correr» e, desta vez, usou a variável A$ que, como sabemos, estava armazenada na memória.
Porque não usámos **GOTO** 10 em vez de fazermos **RUN** como de costume? Ora experimente fazer **RUN**.

Quando se usa **RUN** para executar um programa, está efectivamente a dizer-se ao computador para limpar (**CLEAR**) a memória de quaisquer variáveis e seguir para a linha 1 — princípio do programa.

Utiliza-se este método para evitar que alguma variável, esquecida na memória do computador venha a perturbar o novo programa.

Se, porém, quiser de facto utilizar variáveis que introduziu em «modo directo», nada impede que o faça desde que nunca utilize **RUN**, mas sim, **GOTO**.

Dê entrada de uma nova «string»A$, usando a declaração **LET**.

Faça «correr» o programa novamente, mas, desta vez,usando **GOTO**.
Depois toque na tecla **CLEAR**.

Faça surgir o programa outra vez no ecrã, com **ENTER**.
Torne a correr o programa outra vez no ecrã, com **ENTER**.
Torne a correr o programa outra vez no ecrã com **GOTO** de novo.

CLEAR elimina as variáveis, deixando, contudo, o programa na memória do TC 2048.

SUMÁRIO

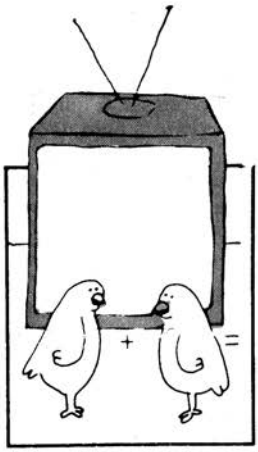
1. **LET** permite atribuir valores ou dados às variáveis.
2. Os nomes das variáveis numéricas devem começar por uma letra, mas podem ter qualquer comprimento.
3. Os nomes das variáveis «string» são formados por uma única letra e $.
4. Os nomes das variáveis «string» poderão ter qualquer comprimento desde que o seu conteúdo esteja entre aspas.
5. **CLS** faz limpar o ecrã.
6. **CLEAR** limpa qualquer variável, da memória do computador.
7. **RUN** dá início ao programa, mas primeiro limpa as variáveis.
8. **GOTO** inicia um programa (na linha que lhe seja indicada), mas não limpa as variáveis da memória do computador.
9. **NEW** limpa o computador completamente.

Matemáticas Com o TC 2048

11

Síntese do Capítulo

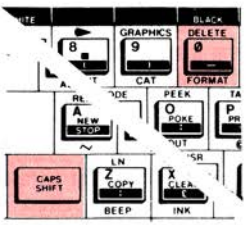
Este capítulo mostra-lhe a forma de somar, subtrair, multiplicar, dividir e utilizar funções já inseridas no computador tais como RND e INT.



Se já usou uma calculadora de bolso está habituado a proceder mais ou menos assim:

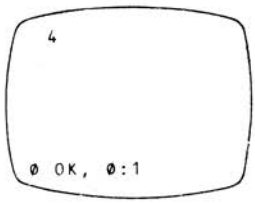
2 + 2 =

e a obter imediatamente a resposta. Experimente fazer isso no seu TC 2048. Não acontece nada. Se premir a tecla **ENTER** surge um erro de sintaxe como mensagem.



Prima CAPS SHIFT e DELETE

Isto não está a parecer-nos muito prometedor. É melhor premir **CAPS SHIFT** e Ø (**DELETE**) e livrar-se disso tudo.



A verdade é que pode usar o seu computador como calculadora, desde que saiba como fazê-lo. Experimente o seguinte:

PRINT 2 + 2 ENTER

Correcto! Pode efectuar qualquer operação matemática de modo semelhante. Os símbolos são os seguintes:

+	«tecle»	SYMBOL	SHIFT	e	K	— Adição
–	»	»	»	»	J	— Subtracção
*	»	»	»	»	B	— Multiplicação
/	»	»	»	»	V	— Divisão
↑	»	»	»	»	H	— Potenciação

Os sinais de adição e subtracção são os normalmente utilizados. A divisão usa um sinal, que provavelmente já conhece (/), para substituir o sinal ÷ que o computador não possui. E um asterisco (*) substitui o sinal × (vezes) para não se confundir com a letra X.
A potenciação é um caso especial. Com o TC2048 não se pode escrever um expoente, nem ele o entenderia. Então utiliza-se o símbolo ↑.

3^2 = 3 ao quadrado	= 3↑2
3^3 = 3 ao cubo	= 3↑3
3^4 = 3 à quarta potência	= 3↑4
3^{10} = 3 à décima potência	= 3↑10

Todas estas operações matemáticas podem fazer parte de programas.

Prioridades e Parêntesis
Num programa que contenha expressões matemáticas que envolvam várias operações, o TIMEX COMPUTER 2048 executá-las-á pela seguinte ordem:

Primeiro, calcula as potências da esquerda para a direita;
Segundo, executa todas as multiplicações e divisões, de novo da esquerda para a direita;

Finalmente, fará todo as adições e subtracções, também da esquerda para a direita.
A isto chamamos prioridades e fazem parte das especificações do computador.
Tal como o conjunto de caracteres inclui mais do que as letras do alfabeto, também as letras do alfabeto, também as prioridades vão além das operações matemáticas elementares. No Apêndice A encontrará uma tabela detalhada.

Pode, no entanto, querer efectuar as operações numa ordem diferente daquela que o computador considera. Conseguirá isto usando parêntesis! Tudo aquilo que esteja entre parêntesis é executado em primeiro lugar (da esquerda para a direita) e o resultado tratado como um só número.

Por exemplo:

3 • 4 + 3 = 15

Porque 3 • 4 = 12 (multiplicação antes da adição) e 12 + 3 = 15. Mas

3•(4 + 3) = 21

porque 4 + 3 = 7 (primeiro os parêntesis) e depois 3 • 7 = 21

Pode ir mais longe, colocando parêntesis dentro de parêntesis. Os parêntesis interiores serão executados em primeiro lugar e assim sucessivamente.

Notação Científica
O TC2048 utiliza a notação científica, quando um número tem mais de 14 caracteres de comprimento. Esta notação consiste num número com um dígito à esquerda de um vírgula decimal, outros à direita desta e um E (de expoente) seguido de um sinal + (ou —) e um número que representa o expoente de uma potência de 1Ø.

Por exemplo:

2.34E + 14

Significa 2.34 vezes 10 elevado a 14 isto é:

2.34×10^{14}.

Tente «teclar» um número com mais de 14 dígitos (depois da instrução **PRINT**) como por exemplo:

PRINT 2345678923456789

e veja o que acontece!

Pode também introduzir números em notação científica, e o computador convertê-los-á em notação normal. Quando o número tiver mais de 14 dígitos a notação científica mantém-se. Experimente.

PRINT 2.34E0 **ENTER**
PRINT 2.34E1 **ENTER**
PRINT 2.34E2 **ENTER**

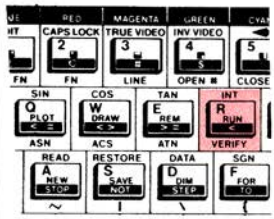
e assim sucessivamente (não necessita de introduzir o sinal +). Em que altura é que o computador volta à notação científica?

Erros por arredondamento
Todos os computadores têm problemas com os erros, devido aos arredondamentos — certas respostas são ligeiramente incorrectas devido a esse facto.

Trata-se de um fenómeno inerente à operação de conversão de binário em decimal. Para corrigir estes desvios a maior parte dos computadores profissionais possui «rotinas» já incorporadas.

Se «teclar»

```
100 LET A = 1.01 - 1
110 PRINT A
```

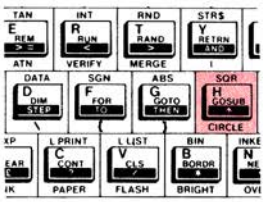


o computador responder-lhe-á com Ø.ØØ99999998, enquanto que a resposta correcta é, evidentemente, .Ø1.

Isto pode ser corrigido acrescentando

1Ø5 **LET** A = **INT** (A * 1ØØ + .5)/1ØØ

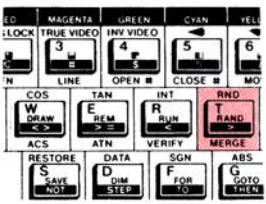
(**INT** é a função localizada acima da tecla R, e é obtida com o Cursor ■).



Funções
Muitas funções matemáticas estão já definidas no «SINCLAIR BASIC». Por exemplo, para calcular a raiz quadrada deverá «teclar»

PRINT SQR 9

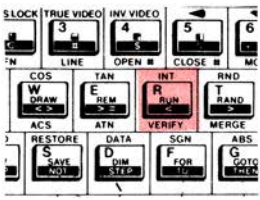
utilizando a função **SQR** inscrita acima da tecla H. Por agora não falaremos mais em funções; encontrá-las-á definidas no Apêndice A. Como dissemos no Capítulo 3, os matemáticos com certeza que as sabem e nós não precisamos delas...



Gerador de números aleatórios
há uma função que queremos discutir um pouco melhor, porque é muito útil em programas didácticos e em jogos. É o gerador de números aleatórios.

A função **RND**, localizada acima da tecla T, gera um número entre Ø e 1. Tente alguns:

PRINT RND



Neste momento isto pode não parecer ter grande utilidade, no entanto permite-lhe obter todos os números que desejar. Por exemplo, se quiser que o computador escolha um número inteiro entre 1 e 6 «tecle»:

PRINT INT (RND * 6) + 1

INT (de inteiro) é a função localizada acima da tecla R, perto da tecla **RND**, com convém. Necessitará de passar duas vezes para o «modo» extensivo. Eis como tudo se vai passar:

1. **RND*** 6 gera um número decimal entre Ø e 1, e multiplica-o por 6.
2. **INT** considera apenas a parte inteira do número. Se ele for 3.Ø9345622, ficará 3. Se for Ø.97888545, ficará Ø . E se for 5.876Ø ficará 5.
3. Teremos assim um número inteiro entre Ø e 5, mas como o que se pretendia era entre 1 e 6, deveremos adicionar uma unidade (+ 1).

Este passo é necessário porque **INT** arredonda por defeito e não por excesso.

Será bom que escreva aquela fórmula em qualquer parte, pois vai necessitar dela muitas vezes.

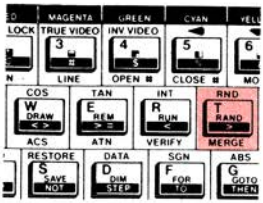
«Escolher um número inteiro entre 1 e 6» simula um lançamento de um dado. Execute-o duas vezes e simulará o lançamento de dois dados.

O número gerado pela função **RND** pode ser atribuído a uma variável.

LET A = INT (RND*32)

e ser depois utilizado para muitos fins, incluindo decidir em que local do ecrã um dado símbolo é colocado!

Para acabar: **RND** não é um verdadeiro gerador de números aleatórios, é apenas um «pseudo--gerador». Com efeito esta função fornece-lhe números de uma longa tabela gerada aleatoriamente. A tabela é enorme e não terá, portanto, possibilidade de a memorizar. No entanto, pode fixar os primeiros números, que serão sempre os mesmos, todas as vezes que se desliga o computador.



Prove-o. Desligue o computador e volte a ligá-lo. Em seguida, «tecle».

PRINT RND

várias vezes. Repita a operação. A sequência é sempre a mesma!

Para não obter os primeiros números da tabela, pode utilizar o «comando» **RAND**, localizada na tecla T. Quando ligar o computador ou mesmo no meio de um programa, e antes de qualquer função **RND**, «tecle»

RAND Ø

Quando se usa Ø a função **RAND** (do inglês randomize), escolhe um ponto da tabela baseado no tempo que decorreu desde que o computador foi ligado (quantas imagens foram enviadas para a TV), o que já é suficientemente aleatório.

Por outro lado, se usar um número diferente de zero, **RAND** escolhe um local da tabela baseado nesse número, por exemplo:

RAND 5Ø

fará com que **RND** se inicie sempre com o mesmo número.
Experimente!

SUMÁRIO

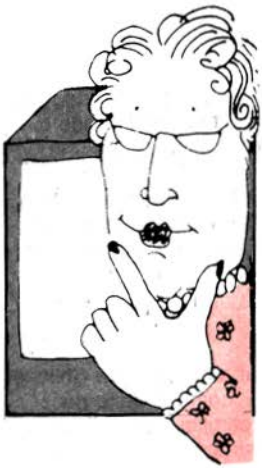
1. O TC 2048 executa as operações matemáticas + (adição), — (subtracção), * (multiplicação), / (divisão) e ↑ (potenciação).
2. Para um programa pode fazê-lo sempre que quiser; no «modo» imediato, necessita de usar o comando **PRINT** para visualizar o resultado: **PRINT** 2 + 2
3. As operações matemáticas são efectuadas atendendo às prioridades; estas podem-se alterar usando parêntesis. As operações entre parêntesis são efectuadas primeiro.

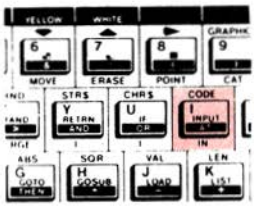
4. O TIMEX COMPUTER 2048 entende a notação científica e utiliza-a para grandes números.
5. O computador tem um número de funções matemáticas pré-definidas, às quais se pode aceder com uma só tecla e no «modo função».
6. A função **RND** gera números «pseudo--aleatórios»

Programas que pedem informação 12

Síntese do Capítulo

Neste capítulo veremos como introduzir informação através do comando INPUT e como os comandos READ, DATA e RESTORE permitem ao Computador reconhecê-la.






```
10 REM PROGRAMA-TABUADA
20 INPUT INK 1; "OLA. COMO TE
CHAMAS?"; A$
30 PRINT INK 2; "DA-ME UM NUMERO"; A$,
"E DAR-TE-EI A TABUADA."...
40 INPUT A
50 PRINT 2; " VEZES "; A; " IGUAL A "; 2*A
60 PRINT 3; " VEZES "; A; " IGUAL A "; 3*A
70 PRINT 4; " VEZES "; A; " IGUAL A "; 4*A
80 PRINT 5; " VEZES "; A; " IGUAL A "; 5*A
```

Um programa pode ser concebido de forma a parar para pedir informação. Uma das formas de lhe responder é através da instrução **INPUT**.

Capítulo 12: Programas Que Pedem Informação



Introduza o programa indicado acima, mas, antes de o fazer «correr» (**RUN**), vamos estudá-lo, linha por linha;

Linha 1Ø — Instrução **REM** «standard», explicando-nos o programa.

Linha 2Ø — Instrução **INPUT**. Se simplesmente «teclar» **INPUT** A

o programa, quando atingir essa linha, pára aguardando nova ordem. Essa indicação é dada pelo cursor intermitente (**L** ou **C**, consoante o estado de **CAPS LOCK**) na parte inferior do ecrã.

Se introduzir a seguinte linha de programa

INPUT A$

o computador indicará que espera uma «string» de caracteres (pode ser só uma letra) mostrando o cursor **L** ou **C** entre aspas.

Para explicar o tipo de dados pretendidos, poderá introduzir, juntamente com a instrução, uma nota explicativa entre aspas, entre **INPUT** e a variável que lhe é atribuída:

INPUT "NOTA"; A

Finalmente, pode especificar a cor de tinta (**INK**) e do **PAPER**, a atribuir à nota, como foi feito no programa em análise. Mas certifique-se de que cada uma das partes da instrução é seguida por ponto e vírgula.

2Ø **INPUT INK** 1; "NOTA"; A$

Linha 3Ø — Forma alternativa de juntar uma explicação à instrução **INPUT** que se segue e aparece no topo superior do ecrã. Note que, para o seu nome — variável A$ — ser escrito na segunda metade do ecrã, deve colocar a seguir uma vírgula em vez de um ponto e vírgula, tendo presente que a próxima instrução **PRINT** começa na linha seguinte. (A propósito: Como a instrução **PRINT** tem exactamente 32 caracteres — um ecrã completo — utilizaram-se três vírgulas no fim da linha para posicionar correctamente a tabuada (TABUADA).

Linha 4Ø — Instrução **INPUT**, pedindo um número.

Linhas 5Ø a 8Ø — Impressão da tabuada. Note os espaços inseridos dentro dos parêntesis, antes e depois das palavras.

Pode utilizar **EDIT** para duplicar as linhas de 5Ø a 8Ø. Depois de introduzir a linha 5Ø, prima **CAPS SHIFT** e 1 (**EDIT**) e a linha 5Ø aparecerá na parte inferior do ecrã. Apague (**DELETE**) o número de linha e escreva 6Ø. Em seguida, use as setas e **DELETE** para fazer as modificações necessárias e «tecle» **ENTER**.

```
DA-ME UM NUMERO, TONI
E DAR-TE-EI A TABUADA

2 VEZES 7 IGUAL A 14
3 VEZES 7 IGUAL A 21
4 VEZES 7 IGUAL A 28
5 VEZES 7 IGUAL A 35

Ø OK, 8Ø:1
```

«Corra» (**RUN**) o programa. Verifique como as notas explicativas que acompanham a instrução **INPUT** o ajudam nas suas respostas.

As instruções **PRINT** nas linhas 5Ø-8Ø, imprimem a preto, já que a cor da tinta não foi explicitada, como o tinha sido nas linhas 2Ø e 3Ø. Experimente agora: apague **INK** 1 da linha 2Ø e insira a seguinte linha:

15 **INK** 1

«Corra» o programa novamente e veja o que sucede nas linhas 5Ø-8Ø. **INK** como uma instrução separada, modifica a cor normal da tinta, que é aquela que aparece sempre, após **PRINT**, a não ser que a instrução **PRINT** tenha o seu próprio comando **INK**.

Capítulo 12: Programas Que Pedem Informação



(Excepção: A nota que acompanha a instrução **INPUT** é sempre escrita em branco ou preto — aquela que fornece maior contraste — a não ser que uma cor **INK**-seja especificada na instrução **INPUT**).

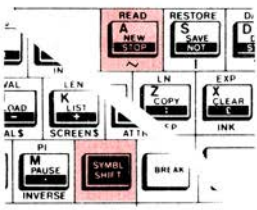
Eis outro exemplo de **INPUT**, pedindo um número sempre que o ciclo se repete:

```
10 REM PROGRAMA-TREZE
20 PRINT "MULTIPLICAREI CADA NUMERO
QUE", "ME DER POR 13"
30 INPUT A
40 CLS
50 PRINT A; " VEZES 13 E IGUAL A "; A*13
60 GO TO 30
```

Com este programa, à introdução de um número, o computador responderá com outro.

Se introduzir uma letra, quando o computador espera um número, o programa pára com relatório 2, variable not found (a não se que por sorte, o computador tenha em memória uma variável do mesmo nome da introduzida...)

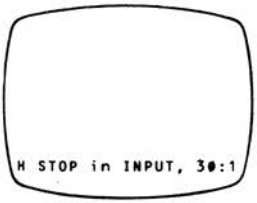
Como parar um programa que já não nos interessa?



Para para um programa prima **SYMBOL SHIFT** e A.

Com o cursor na parte inferior do ecrã, à espera de **INPUT**, prima **STOP** (**SYMBOL SHIFT** e A) em vez de um número.

Capítulo 12: Programas Que Pedem Informação



Porque razão não se pôs **CLS** antes de **INPUT**? Logicamente, parece que se pretendeu juntar as perguntas por um lado e as respostas por outro. Mas troque a linha 40 pela linha 25 e veja o que acontece.

Pretende-se manter a nota explicativa no ecrã até ser lida, pelo que este só se limpará depois de introduzir o **INPUT**.
Experimente tirar por completo a instrução **CLS**. «Tecle».

25 **ENTER**

e veja o que acontece quando «corre» (**RUN**) o programa.
Pode utilizar a instrução **INPUT** para controlar a velocidade com que as coisas acontecem. Neste caso, as entradas são desprovidas de conteúdo e desprezadas pelo computador.

```
200 PRINT "PRIMA ENTER PARA
CONTINUAR"
210 INPUT A$
 .
 .
 .
 .
 .
220 (Próxima linha)
 .
 .
 .
 .
 .
```

Este programa espera que a tecla **ENTER** seja premida antes de prosseguir.

READ, DATA e RESTORE
Há outro modo de um programa incluir informações que não tinha.

Capítulo 12: Programas Que Pedem Informação



A instrução **DATA** é um lugar onde se armazenam valores — quer sejam números ou «string» de caracteres, ou ambos interligados — separados por vírgulas.
A instrução **READ** introduz esses valores no programa, um de cada vez, até terminar.

```
10 REM PROGRAMA-CAPITAIS
20 READ A$
30 PRINT "QUAL E A CAPITAL DA/DOS", A$;"?"
40 READ B$
50 INPUT C$
60 PRINT "A SUA RESPOSTA FOI", C$
70 PRINT "A RESPOSTA CORRECTA E", B$
80 GO TO 20
90 DATA "HOLANDA", "HAIA", "BELGICA", "BRUXELAS", "ESTADOS UNIDOS DA AMERICA DO NORTE", "WASHINGTON"
```



De novo, as vírgulas múltiplas no final da instrução **PRINT** servem para separar as saídas, no ecrã.

«Corra» (**RUN**) o programa. Parará quanto tiver lido todos os dados (na instrução **DATA**).

O que é útil neste programa é que pode alterá-lo ou acrescentar-lhe dados, o que o transformará num verdadeiro teste. (Pode até trocar de pergunta!).

Num programa podem existir várias instruções **DATA**, mas serão consideradas como uma só lista. Assim, pode acrescentar ao programa:

```
100 DATA "PORTUGAL", "LISBOA",
"BRASIL", "BRASÍLIA", "CANADÁ", "OTAVA",
'DINAMARCA", "COPENHAGA"
110 DATA "ESPANHA", "MADRID", "GRÉCIA",
"ATENAS"
```

ou ainda

```
120 DATA "INGLATERRA", "LONDRES",
"FRANÇA", "PARIS", "ALEMANHA", "BONA",
"NORUEGA", "OSLO"
```

Uma instrução **DATA** pode ter qualquer número de elementos. É mais simples realizar um programa se não estiverem todos numa instrução só. Podem ser números, que o programa lê através de

```
READ A
```

ou «string» de caracteres (uma ou mais letras entre aspas), lido por

```
READ A$
```

Claro que os elementos não têm de ser aos pares; podem ser lidos individualmente como números ou «strings» ou em grupos conforme o programa. Pode querer utilizar um conjunto de dados para um certo número de operações, e periodicamente repôr a lista de dados desde o princípio (isto é, fazer com que a próxima instrução **READ** comece de novo no primeiro elemento da instrução **DATA**).
Assim, no lugar certo, inserirá uma linha como

200 RESTORE

Também pode repôr apenas a lista de dados duma linha específica (não necessariamente todos os dados) utilizando a instrução **RESTORE** com um número de linha.

Por exemplo, se tiver acrescentado as linhas 100, 110 e 120 como se indicou acima, poderia utilizar

RESTORE 100

para que a próxima instrução **READ** comece com "PORTUGAL" em vez de "HOLANDA". (Eis outra razão para utilizar mais do que uma linha de **DATA**).

Se quiser para (**STOP**) um programa que peça para introduzir «strings» — o cursor na parte inferior do ecrã encontra-se entre aspas — terá que apagar as aspas da esquerda antes de premir **STOP**. De outro modo o programa considerará a palavra **STOP** como um **INPUT**.
Pode também introduzir, no programa, a linha

55 IF C$ = "STOP" **THEN STOP**

e se introduzir as letras STOP, em resposta ao cursor entre aspas, o programa parará.

SUMÁRIO

1. **INPUT** pára o programa e espera pela entrada de informação.

   INPUT atribui um número à variável; um cursor na parte inferior do ecrã sinaliza que o computador espera a introdução de dados.

   **INPUT A$** atribui uma «string» à variável A$; um cursor entre aspas, sinaliza a espera de uma «string» de caracteres.

2. **STOP**, como resposta ao cursor **INPUT**, pára o programa. Se o cursor está entre aspas, as aspas da esquerda têm de ser apagadas antes de introduzir **STOP**

3. Instruções **DATA** contêm números e/ou «strings» separadas por vírgulas, para uso em programas.

   Podem existir várias linhas de **DATA**, mas o computador interpretá-las-á como uma única lista de dados.

4. A instrução **READ** introduz elementos a partir da lista de dados um de cada vez, para uso num programa.

   READ A (ou **READ** A$) atribui o próximo elemento da lista de dados à variável A (ou A$).

   (Se os dados não correspondem à instrução **READ**, resultará um erro — por exemplo, se **READ** A encontra uma «string»).

   READ A, B, C lê os próximos 3 elementos da lista de dados e atribui-os às variáveis A, B e C.

5. A instrução **RESTORE** dirige a instrução **READ** seguinte para o primeiro elemento da lista de dados.

Programas Que se Repetem: Ciclos 13

Síntese do Capítulo

Veja neste capítulo como os trabalhos repetitivos são simples com FOR, TO, NEXT e STEP. Também se usa LIST para que a listagem do programa apareça no ecrã.



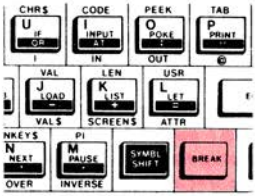

```
NUMERO  1
NUMERO  2
NUMERO  3
NUMERO  4
NUMERO  5
NUMERO  6
NUMERO  7
NUMERO  8
NUMERO  9
NUMERO  10

0 OK, 3:1
```

```
1Ø FOR I = 1 TO 1Ø
2Ø PRINT "NUMERO", I
3Ø NEXT I
```

Introduza este programa no seu computador TIMEX COMPUTER 2048. Na lista 1Ø não escreva **TO** letra a letra, prima simultaneamente **SYMBOL SHIFT** e F. «Tecle» **RUN**.
Já anteriormente se disse que **GOTO** é em BASIC uma instrução muito poderosa. Usamo-la para que um dado programa se repita. «Teclando» **GOTO** fazemos com que o computador volte para uma das linhas anteriores do programa e execute novamente as mesmas operações.

Capítulo 13: Programas Que se Repetem: Ciclos

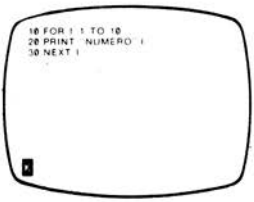


Ao processo que consiste em repetir as mesmas operações várias vezes dá-se o nome de «ciclo» (loop). Quando apenas utilizamos **GOTO**, não podemos controlar o número de vezes que o programa se repete. De facto, ele não pára (a não ser quando o ecrã já estiver totalmente preenchido, ou quando premimos **BREAK**).

A isto chamamos «ciclo sem fim», e não é de muita utilidade.

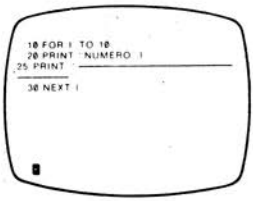
O ciclo FOR/NEXT

Na verdade aquilo de que necessitamos é de «ciclos» (loops) que consigamos controlar. No SINCLAIR BASIC, a melhor maneira de o conseguir é com o o ciclo **FOR/NEXT**. O programa apresentado acima é um exemplo desse ciclo:



A linha 1Ø contém os «comandos» **FOR** e **TO**, dois números que indicam quantas repetições se pretende, e uma variável de controlo. Uma variável de controlo num ciclo **FOR/NEXT** deve ser representada por uma única letra: Os peritos normalmente utilizam I, no entanto qualquer letra serve.

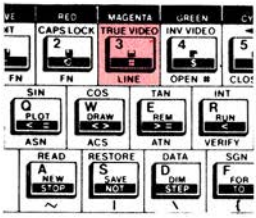
A linha 2Ø contém a instrução que irá ser repetida durante o ciclo. Podem existir dentro do mesmo ciclo, várias linhas como esta.



A linha 3Ø é necessária para fechar o ciclo. Diz ao computador para parar e voltar para a linha **FOR**.

Tente o seguinte: para obter novamente a listagem do programa «tecle» **ENTER** e depois acrescente-lhe

25 PRINT" _____
_____"

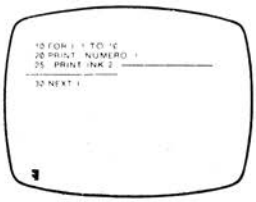


Entre aspas está o símbolo gráfico indicado na tecla 3. Recorda-se de como o pode obter?



Depois de SYMBOL SHIFT e P para obter as aspas «tecle» CAPS SHIFT e 9 para entrar no «modo» gráfico. Com o cursor **G** no ecrã pressione a tecla 3, trinta e duas vezes, que é o número de colunas do ecrã. Volte a «teclar» CAPS SHIFT e 9 para abandonar o «modo» gráfico e, já com o cursor **L**, feche aspas com SYMBOL SHIFT e P. Prima ENTER e RUN e agora cada uma das soluções anteriores ficará separada por uma linha.

A propósito, uma maneira de verificar se «teclou», entre as aspas, os 32 caracteres — uma linha do ecrã — é ver se as segundas aspas estão um espaço à frente das primeiras (mas na linha seguinte).



Prefere uma solução colorida? Tente, então, juntar INK 2 à linha 25:

25 PRINT INK 2;" _____ "



O comando LIST

«Tecle» ENTER para obter novamente a listagem do programa. O cursor do programa encontra-se na linha 25 — a última a ser introduzida. Neste momento queremos chamar a linha 10. Em vez de usar as teclas 6 e 7, vamos recorrer a outro processo.

Ao contrário do que acontece com o TC2048, na maioria dos computadores, «teclando» ENTER não se obtém a listagem do programa. Nesses casos deve inserir a instrução LIST acompanhada do número da linha onde deseja iniciar a listagem.

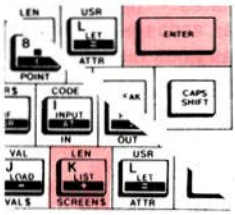
Capítulo 13: Programas Que se Repetem: Ciclos

Neste computador TIMEX pode usar **LIST**. **No** entanto, e como já notou, se premir

ENTER

depois de um programa ter parado, a sua listagem aparece no ecrã, desde o início, ocupando tantas linhas quantas as que couberem no ecrã.
O cursor do programa encontra-se na última linha introduzida.

Se fizer



LIST ENTER

aparecerá a listagem também desde o início, mas com duas diferenças:

1. O cursor do programa estará na primeira linha;
2. A pergunta "**scroll?**" aparecerá na parte inferior do ecrã. Se premir N (de Não), **BREAK** ou **STOP** (**SYMBOL SHIFT** e A), o resto da listagem não aparecerá. Se premir qualquer outra tecla, aparecerão as 22 linhas seguintes. O processo pode ser repetido até que toda a listagem tenha «corrido».

Se fizer

LIST 9Ø ENTER

verificará que a listagem do programa se inicia na linha 9Ø, estando aí situado, também, o cursor. Isto é útil, pois permite chamar uma linha que se encontre a meio de um programa muito longo.

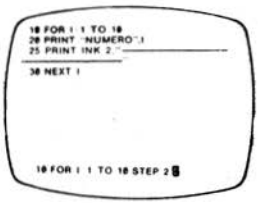


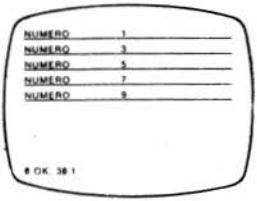
Portanto, para chamar a linha 1Ø, em lugar de usar as setas de movimento «tecle».

LIST 1Ø ENTER

(é evidente que neste caso poderia garantir apenas **LIST**) e responder com N (não) à pergunta "scroll?".

Capítulo 13: Programas Que se Repetem: Ciclos





Como usar STEP no ciclo FOR/NEXT
Neste momento já sabemos colocar o cursor do programa onde quisermos. «Tecle» **EDIT** e chame a linha 10 para a parte inferior do ecrã. Seguidamente coloque o cursor no fim da linha e acrescente **STEP** 2 («Tecle» **SYMBOL SHIFT** e D, e não letra a letra):

10 FOR I = 1 TO 10 STEP 2

«Tecle» **ENTER** e seguidamente «corra» o programa (**RUN**).

O resultado de **STEP** é aquele que se esperava: incrementa a variável de controlo de um valor igual ao indicado pelo **STEP**. Se se desejar que o programa conte 2,4,6,8,10 em vez de 1,3,5,7,9, como acha que se deveria fazer? Tente.

Usando **STEP** também é possível uma contagem descendente. Modifique a linha 10 para:

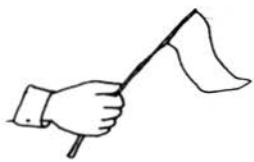
10 FOR I = 10 TO 1 STEP − 2

e veja o que acontece!
Se quiser que o computador faça uma contagem decrescente de um em um e «teclar»

10 FOR I = 10 TO 1

não obterá resposta.

É necessário acrescentar, à linha anterior, **STEP-1**.
Tente e verá.



No programa "TABUADA" apresentado no Capítulo anterior, tente substituir as linhas 50-80 por um ciclo **FOR/NEXT**. Repare que é possível sem grande dificuldade conseguir-se uma tabuada tão grande quanto se quiser.

Sugestão:

FOR I = 1 TO 10
PRINT...I*A
NEXT I

Ciclo contendo outro ciclo
Será possível ter-se um ciclo («loop») dentro de outro ciclo?
Claro que sim! Muitas vezes, desejamos repetir uma dada sequência que por sua vez contém outras sequências que também queremos repetir. Lembre-se que a variável de controlo num ciclo FOR/NEXT é representado por uma única letra. Portanto, podemos ter no máximo 26 ciclos (as 26 letras do alfabeto).

Ou seja, poderá fazê-lo desde que os ciclos estejam correctamente contidos uns nos outros. Tente o seguinte:

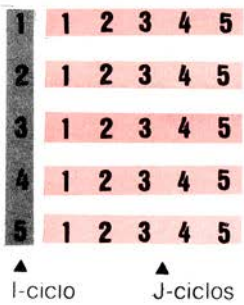
```
1Ø FOR I = 1 TO 5
2Ø FOR J = 1 TO 5
3Ø PRINT I; ":";J;"   ";
4Ø NEXT J
5Ø PRINT
6Ø NEXT I
```

Atenção à linha 3Ø: introduza-a correctamente e deixe um espaço entre as segundas aspas. Repare também que para além do I, foi necessária uma segunda letra (J) para que se efectue a contagem no segundo ciclo.

«Corra» o programa.

```
1:1  1:2  1:3  1:4  1:5
2:1  2:2  2:3  2:4  2:5
3:1  3:2  3:3  3:4  3:5
4:1  4:2  4:3  4:4  4:5
5:1  5:2  5:3  5:4  5:5

Ø OK, 6Ø:1
```



Com efeito as filas devem-se ao «ciclo I» e as colunas ao «ciclo J»

A instrução PRINT, na linha 5Ø, faz com que o computador imprima a fila seguinte para que assim se inicie um novo «ciclo I».

Para utilizar a cor — e, talvez, pra melhor compreender os ciclos — tente juntar INK I ou INK J às linhas 3Ø, como por exemplo

```
3Ø PRINT INK I;I;":";J;" "; ou
3Ø PRINT I;":";INK J;J; " "; ou
3Ø PRINT INK I;I;":"; INK J;J;"   ";
```

E, claro pode ainda acrescentar

15 BORDER 6

Os ciclos estão correctamente contidos um no outro, uma vez que o ciclo J é integralmente englobado pelo ciclo I. Terá problemas se os ciclos se entrecruzarem. Umas vezes aparecerão mensagens de erro, outras obterá resultados incompreensíveis. Modifique as linhas 40 e 60 de modo a que o programa fique com o seguinte aspecto:

```
1:1  2:1  3:1  4:1  5:1
6:2
7:3
8:4
9:5

0 OK, 60:1
```

```
10 FOR I=1 TO 5
20 FOR J=1 TO 5
30 PRINT=I;":";J;" ";
40 NEXT I
50 PRINT
60 NEXT J
```

«Corra» (**RUN**) o programa e tente compreender o que obteve!

Quando fizer uso de ciclos **FOR/NEXT** deve prestar muita atenção ao seguinte: não é possível usar a instrução **GOTO** para «saltar» para o interior de um ciclo. Pois quando chegar à instrução **NEXT**, sem que anteriormente tenha lido **FOR**, começam os problemas.

SUMÁRIO

1. É possível obter repetições através do ciclo **FOR/NEXT** utilizando os comandos **FOR TO** e **NEXT** além de uma variável de controlo.
2. As variáveis de controlo são representadas por uma única letra.
3. Num ciclo **FOR/NEXT** a instrução **STEP** é utilizada para incrementar ou decrementar a variável de controlo de um valor diferente de um, ou de um valor negativo.
4. Ciclos sucessivos devem estar contidos uns nos outros e não entrecruzados.
5. A instrução **LIST** faz com que a listagem do programa apareça no ecrã;
   LIST seguido do número de uma linha faz com que a listagem se inicie nesse número e que o cursor do programa se coloque nessa linha.

Programas Que Decidem: Ramificação 14

Síntese do Capítulo

Para se tomarem certas decisões usa-se IF e THEN, juntamente com as relações matemáticas =, <, >, <=, >=, e <>. Para se conjugarem as relações usa-se AND, OR e NOT.



Falamos já em três das quatro razões que tornam o computador um instrumento com tanto poder e utilidade:

1. Trabalha com rapidez.
2. Consegue memorizar inúmeras informações, incluindo as suas próprias instruções (programas).
3. Pode repetir certas operações consecutivamente e sem se cansar (ciclos).

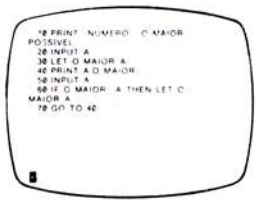
Falemos agora da quarta razão:

4. O computador pode tomar decisões.

Um programa pode conter um certo número de instruções, cuja execução depende de condições muito precisas. A um tal programa chama-se PROGRAMA RAMIFICADO, isto é, a sua execução pode prosseguir por um ramo ou por outro, consoante as condições que lhe forem dadas.

O comando que torna tudo isto possível é **IF**.

Capítulo 14: Programas Que Decidem: Ramificação



Intreduza o programa indicado abaixo, mas **antes** de o fazer «correr», falemos acerca dele — linha a linha. Assim, ao mesmo tempo que revemos matérias já estudadas, analisamos questões novas. Para obter **THEN** «tecle» **SYMBOL SHIFT** e G.

```
10 PRINT "NUMERO", "O MAIOR POSSIVEL"
20 INPUT A
30 LET O MAIOR=A
40 PRINT A, O MAIOR
50 INPUT A
60 IF O MAIOR<A THEN LET O MAIOR=A
70 GO TO 40
```

A linha 10 é uma simples instrução **PRINT**; a vírgula entre as duas «strings» faz com que sejam escritas no ecrã nas posições 0 e 16 — início e meio da zona de escrita.

A linha 20 é uma instrução de **INPUT**. Pede que lhe seja fornecido um número e atribui o seu valor à variável A. Quando «correr» o programa, o cursor **L**, na parte inferior do ecrã, assinala que o TC2048 aguarda a instrução desse valor

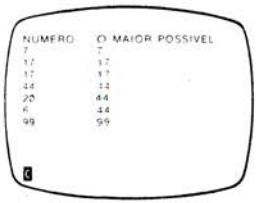
A linha 30 indica que a variável O MAIOR (em inglês Largest significa «o maior»; assim será mais fácil recordá-la) é igual à variável A

A linha 40 faz com que os valores destas duas variáveis surjam no ecrã; debaixo da palavra "NUMERO" e debaixo de "O MAIOR POSSIVEL". (Repare novamente no efeito da vírgula).

A linha 50 pede que o utilizador lhe forneça um outro número cujo valor atribuirá, novamente, a A. Este número substitui o anterior valor de A.

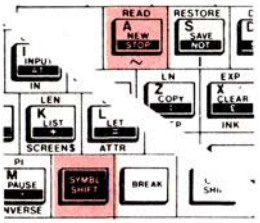
Na linha 60 é tomada a decisão. Se o número a que chamámos "O MAIOR" for menor (<) do que o novo valor de A, o programa torna-o igual a A. Se "O MAIOR" for realmente (>) ou igual a A, o seu valor não é alterado.

Capítulo 14: Programas Que Decidem: Ramificação



A linha 7Ø, provocando um retorno à linha 4Ø, origina um ciclo.
Assim, a parte do programa entre as linhas 4Ø e 7Ø repete-se sempre que for introduzido um novo número.

Pronto! Agora «corra» o programa, atribua os valores que quiser à variável, e veja como ele funciona.



Prima SYMBOL SHIFT e STOP

Este programa contém um «ciclo sem fim». Felizmente, neste caso, isto não causa qualquer problema uma vez que o programa pára periodicamente e espera que lhe seja fornecido um novo valor.
Em vez de introduzir um novo número, pode parar o programa respondendo ao pedido do **INPUT** — desde que o cursor **L** esteja na parte inferior esquerda do ecrã — com **STOP** (**SYMBOL SHIFT** e A).

Também se consegue interromper o programa «teclando» **BREAK**, mas é necessário ser-se rápido — isto só funciona quando o programa não estiver à espera de um novo **INPUT**. Tente! Deve «teclar» **BREAK** do cursor **L** surgir no ecrã.

Por vezes, pode estar interessado em que lhe surja, no ecrã, a indicação (nota explicativa) de que deve atribuir um valor à variável, mas tal indicação não aparece.

Eis uma maneira de o conseguir: Introduza mais algumas linhas de modo a que o programa fique com o seguinte aspecto (nas linhas 15, 4Ø e 45 não escreve **AT** letra a letra, «tecle» simplesmente **SYMBOL SHIFT** e I):

```
 5 LET X=3
1Ø PRINT "NUMERO", "O MAIOR POSSÍVEL"
15 PRINT AT X+5,1; "ESCREVA UM NUMERO"
2Ø INPUT A
25 PRINT AT X+5,1; "                    "
3Ø LET O MAIOR=A
4Ø PRINT AT X,Ø;A, O MAIOR
45 PRINT AT X+5,1; "ESCREVA UM NUMERO"
5Ø INPUT A
55 PRINT AT X+5,1;                      "
6Ø IF O MAIOR<A THEN LET O MAIOR=A
65 LET X=X+1
7Ø GO TO 4Ø
```

A linha 5 faz X igual a 3; a linha 4Ø mostrará os valores de A e de "O MAIOR"; a linha 65 vai adicionar uma unidade X em cada ciclo; os números aparecerão escritos no ecrã na fila 4, e depois na 5 e assim sucessivamente.
As linhas 15 e 45 fazem com que, cinco filas abaixo de X, surja a indicação de que um novo número deve ser introduzido.
O deslocamento para baixo da frase "ESCREVA O NUMERO" não deixará que se esqueça de atribuir um novo valor à variável.
(Para apagas estas linhas basta simplesmente imprimir uma linha com espaços em branco. Substituirá, assim, as palavras da nota explicativa).
Foi necessário modificar a linha 4Ø para assim se especificar o local onde irá ser escrito o próximo par de números, uma vez que o comando **PRINT AT** nas linhas 15 e 45 mudam a posição de impressão para a linha 2Ø.
Se não tivessemos modificado a linha 4Ø, o próximo par de números apareceria escrito na linha 21.
A linha 65 não lhe parece estranha? X=X+1, que igualdade matemática é esta? Bem, claro que isto não é matemática; é uma instrução de atribuição de valores a variáveis. Significa: Faça X igual ao anterior valor de X, mais uma unidade.
Esta é outra maneira de obter um ciclo. No entanto este é um método menos expedito que o ciclo **FOR/NEXT**, e por isso raramente é utilizado. A única vantagem que apresenta é a de se poder usar um nome mais descritivo para definir a variável e não apenas uma letra X.
Pode utilizar.

```
 5 LET LINHA = 3
65 LET LINHA = LINHA + 1
```

enquanto que, como com certeza se recorda, no ciclo **FOR/NEXT** as variáveis só podiam ser representados por uma única letra.

Retire agora a linha 65 e vejo o que acontece. O valor X não é incrementado, e volta à linha 3 de cada vez que se repete o ciclo.

Consegue introduzir os dados das linhas 15 e 45 em letras inversas?

A instrução IF

A instrução **IF**, verifica «se» uma dada condição é verdadeira: se for, toda a linha é executada; se não for, o resto da linha é ignorado e o programa passa para a linha seguinte.

```
40 IF A = 5 THEN GOTO 100
```

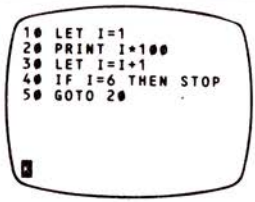
que significa que, se a variável A é igual a 5, o programa avança para a linha **100** e continua a «correr» a partir dessa linha.
Se A não é igual a 5, a próxima linha a ser executada é a que vem logo a seguir à **40** (provavelmente a **50**, certo?).

A propósito, no Sinclair BASIC, sempre que se tiver **IF** e **GOTO** é necessário incluir também **THEN** (noutros pode-se omitir tal indicação). Entre outras coisas **THEN** faz com que o cursor **K** volte ao ecrã, sendo então possível usar um «comando», como por exemplo **GOTO**; caso contrário não seria possível ordenar ao computador a execução de qualquer comando

```
IF A = 5
```

Por vezes — como no exemplo dado no início do Capítulo — fornece-se, de imediato, o valor em relação ao qual a decisão é tomada. Mas em geral esse valor resulta de cálculos efectuados ao longo do programa.
Podemos mesmo usar **IF** para terminar um ciclo:

Capítulo 14: Programas Que Decidem: Ramificação

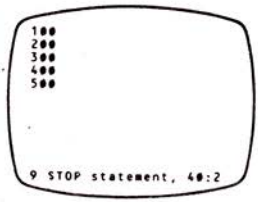


10 LET I = 1
20 PRINT I*100
30 LET I = I + 1
40 IF I = 6 THEN STOP
50 GOTO 20

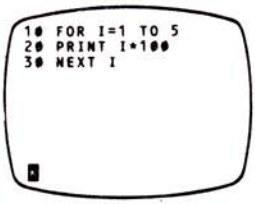
Pode-se poupar uma linha modificando a 40 se se «teclar»

40 IF I < 6 THEN GOTO 20

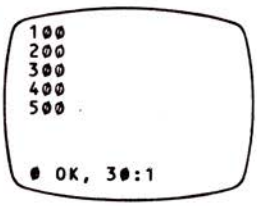
e eliminar em seguida a linha 50.



Como é que se poderia escrever o programa acima usando o ciclo **FOR/NEXT**?



10 FOR I = 1 TO 5
20 PRINT I * 100
30 NEXT I



A instrução **IF** compara valores e utiliza os seguintes símbolos matemáticos:

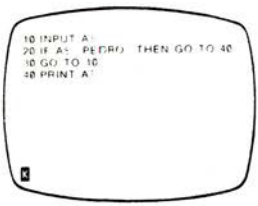
		SYMBOL	SHIFT	e	L
=	é igual a	»	»	»	R
<	é menor que	»	»	»	T
>	é maior que	»	»	»	Q
<=	é menor ou igual a	»	»	»	E
>=	é maior ou igual a	»	»	»	W
<>	não é igual a	»	»		

=, <, >, <=, >=, <>

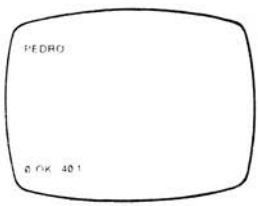
Para obter <= (menor ou igual a) não «tecle» SYMBOL SHIFT e L. Deve usar a tecla Q. Proceda de modo idêntico para obter >= e <>.

Comparação de «strings»

É também possível usar os símbolos anteriores para comparar «strings». Normalmente utilizamos = ou <> para verificar se uma dada palavra introduzida coincide ou não com outra previamente escolhida. Introduza o programa seguinte em que PEDRO quer que descubramos o seu nome.




```
10 INPUT A$
20 IF A$ ="PEDRO" THEN GOTO 40
30 GOTO 10
40 PRINT A$
```



«Corra» o programa e repare que o cursor **L** na parte inferior do ecrã, está entre aspas, indicando que deverá introduzir uma «string».

Dê algumas respostas erradas e, seguidamente, introduza a resposta correcta.

Tente agora o seguinte: adicione uma nova linha.

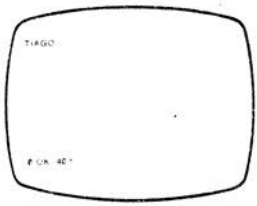
5 LET B$ = "PEDRO"

e modifique a linha 20 de modo a ficar

20 IF A$ <=B$ THEN GOTO 40

Pode utilizar as setas de movimento, EDIT e DELETE para a substituição ou pode simplesmente introduzir uma nova linha. Substitui "PEDRO" por B$ e o sinal = (igual) por <= (menor ou igual a). (Não esqueça: «tecle» SYMBOL SHIFT e Q, e não SYMBOL SHIFT e R seguidos de SYMBOL SHIFT e L).

Capítulo 14: Programas Que Decidem: Ramificação



Depois experimente os seguintes nomes: AVA, MARIANO, TIAGO e JOÃO.
O que acontece?

Numa «string» os símbolos matemáticos estabelecem uma comparação com a primeira letra dessa «string». Se a primeira letra de A$ for, no alfabeto, anterior à primeira letra B$, então considera-se que A$ é menor que B$. Se a primeira letra de cada uma das «strings» foi igual, faz-se a comparação entre as segundas letras, e assim sucessivamente.
Portanto com «strings»:

anterior, o alfabeto, é o mesmo que menor que;
posterior, no alfabeto, é o mesmo que maior que.

De facto, o que o computador na realidade faz é comparar o código de números dos caracteres dados na tabela de caracteres do TC 2048.
Se analisar o Apêndice intitulado «Conjunto de caracteres», verificará que A > 9. Na tabela de caracteres as letras do alfabeto vêm depois dos numerais, portanto, qualquer letra é sempre maior que qualquer número.

Faça um teste no programa do PEDRO. Introduza um número — por exemplo 25 — como resposta ao pedido de instrução de um nome.

De que modo é que se poderiam colocar no programa do PEDRO expressões como "ADIVINHE O MEU NOME" e «Errado». Tente de novo?

Sugestão: pode-se modificar a linha 4Ø de modo a ficar

4Ø PRINT "CERTO. O MEU NOME É"; A$
Mas só no caso de, na linha 2Ø, estar o sinal =, e não < =.

AND, OR, NOT
Damos o nome de relação a =, <, >, < =, > = e <>.

Estas podem ser combinadas através das relações lógicas **AND**, **OR** e **NOT**.

```
IF A = B AND C = D THEN GOTO 100
```

significa que, se as duas relações se verificarem, o programa vai para a linha 100. Se A for igual a B, mas C não for igual a D, o programa segue simplesmente para a linha seguinte, ignorando a ordem de avançar para a linha 100.

```
IF A < B OR C > D THEN GOTO 100
```

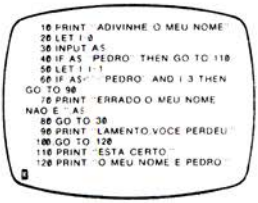
coloca o programa na linha 100 se ambas as relações A < B ou se C > D forem verdadeiras.

```
IF NOT A = B THEN GOTO 10 0
```
é o mesmo que
```
IF NOT <> B THEN GOTO 100
```

No Sinclair BASIC, ao contrário do que acontece com outras linguagens, é necessário incluir, em linhas semelhantes à anterior, simultaneamente **THEN** e **GOTO**, para que se possam utilizar «comandos».

Mas, por outro lado, a sua inclusão torna o programa mais claro e mais simples de compreender.

Adicionemos mais algumas linhas ao programa PEDRO:




```
 10 PRINT "ADIVINHE O MEU NOME"
 20 LET I=0
 30 INPUT A$
 40 IF A$="PEDRO" THEN GO TO 110
 50 LET I=I+1
 60 IF A$<>"PEDRO" AND I=3 THEN
GO TO 90
 70 PRINT "ERRADO. O MEU NOME
NAO E ";A$
 80 GO TO 30
 90 PRINT "LAMENTO, VOCE PERDEU."
100 GO TO 120
110 PRINT "ESTA CERTO."
120 PRINT "O MEU NOME E PEDRO."
```

Capítulo 14: Programa Que Decidem: Ramificação





«Corra» (**RUN**) o programa e dê algumas respostas. A combinação de relação na linha 60 faz com que o jogo pare ao fim de três tentativas. Analise cuidadosamente a lógica do programa e tente compreender porquê e como é que ele se desloca de uma linha para a outra.

Tente «organizar» o ecrã adicionando

15 **PRINT**

e

75 **PRINT**

Temos agora uma pergunta a fazer. Descobriu, quando analisou o programa, que a linha 60 não necessitava das duas relações?

Uma vez que a linha 40 transfere o programa para a linha 110, com A$ = "PEDRO", logicamente, se estamos na 60, A$ é diferente de "PEDRO". Portanto a linha 60 pode apenas

60 IF I=3 **THEN GOTO** 90

Tente e veja que assim é. A lição a tirar daqui, é a de que se deve estudar sempre cuidadosamente um programa e analisar bem a sua lógica de modo a fazê-lo da forma mais directa possível. A procura da solução mais simples para um dado programa é o que faz com que uma programação seja divertida e estimulante!

SUMÁRIO:

1. **IF/THEN** analisa uma condição. Se a condição se verificar, programa executa o que lhe é ordenado a seguir a **THEN** (normalmente para se deslocar — **GOTO** — para outro local do programa). Se a condição não se verificar, será executada a próxima linha do programa.

2. IF analisa valores matemáticos usando as relações

=	igual a
<	menor que
>	maior que
<>	não é igual a
<=	menor ou igual a
>=	maior ou igual a

3. Pode-se combinar mais do que uma relação matemática, usando

   AND (ambas verdadeiras)
   OR (pelo menos uma delas é verdadeira)
   NOT (a relação não é verdadeira)

=, <, >, <>, <=, >=, AND, OR, NOT

Programas Dentro de Programas: Subrotinas 15

Síntese do Capítulo

Como repetir determinadas partes de um programa, utilizando GOSUB e RETURN



Se habitualmente lida com programadores profissionais, provavelmente ouviu já falar em «PROGRAMAÇÃO ESTRUTURADA». Mas se perguntar a dez desses profissionais o que é que isso quer dizer, é provável que receba dez respostas diferentes.

Uma resposta com sentido será que todos os bons programas são estruturados; isto é, planeados e bem organizados.

Outra resposta repetidamente ouvida será que a programação estruturada envolve módulos em programas extensos. Uma tarefa é analisada e dividida em sub-tarefas dentro do programa principal.
Isto traz-nos um certo número de benefícios:

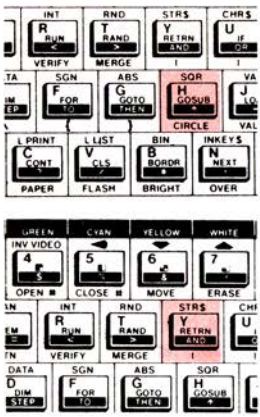
1. O programa torna-se de mais fácil compreensão, quanto tiver sido revisto pela pessoa que o escreveu ou por qualquer outra.

Capítulo 15: Programas Dentro de Programas: Subrotinas

2. O programa é mais simples de modificar, ou de manter.
3. Quando o programa está a ser preparado, se for muito extenso, pode ser dividido em módulos e cada um destes módulos atribuído a um programador. Isto torna mais rápido o desenvolvimento de «Software».
4. Finalmente, o processo de subdivisão de uma questão em sub-tarefas é uma ajuda ao processo de programação e de solução de problemas.

Todos estes benefícios, excepto o apontado em 3, podem ser úteis no trabalho com o TC2048. Utilizando a primeira definição apontada acima, e o conceito de «subrotinas», podemos fazer os nossos programas «estruturados».

Uma «subrotina» é um «mini-programa» que pode ser «chamado» pelo programa principal (ou por outras «subrotinas», ou até por ela própria, para o que terá de recorrer a outros livros), tantas vezes quantas as desejadas. A «subrotina» cumpre a sua função quando «chamada», após o que o programa principal continua a desenvolver-se.



GOSUB e RETURN

Há dois comandos muito simples utilizados numa «subrotina», **GOSUB** e **RETURN**. **GOSUB**, com um número de linha, é inserido no programa principal, onde a «subrotina» é necessária ; o número de linha é o do início de «subrotina». **RETURN** (que se encontra abreviada para **RETRN** na tecla Y) é inserido no final da própria «subrotina», e envia a execução do programa para a linha imediatamente a seguir àquela que continha o **GOSUB**.

Qual seria o problema se utilizasse **GOTO** em vez de **GOSUB** e **RETURN**?

Eis um exemplo:

```
10 REM PROGRAMA PRINCIPAL
20
30
40 GOTO 1000
50
60
70

1000 REM A SUBROTINA
1010
1020
1030
1040 GOTO 50
```

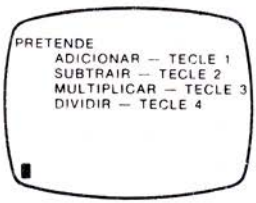
Suponha que queria «chamar» esta «subrotina» a partir de pontos diferentes do programa... E não queria voltar (**GOTO**) à linha 50 cada vez que a «subrotina» fosse executada. É para isto que serve **GOSUB** e **RETURN**. **GOSUB** 1000 conduz o programa à linha 1000 tal como **GOTO** 1000. Mas com **GOSUB** o computador «lembra-se» donde partiu e comandado por **RETURN** dirige-se para a linha a seguir à do comando **GOSUB**. Por exemplo:

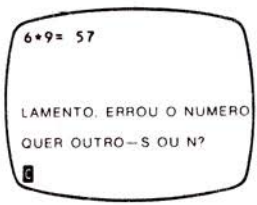
```
10 REM PROGRAMA PRINCIPAL
20
30
40 GOSUB 1000
50
60
70 GOSUB 1000
80
90

1000 REM SUBROTINA
1010
1020
1020
1040 RETURN
```

Capítulo 15: Programas Dentro de Programas: Subrotinas

Neste tipo de programa, o comando **RETURN**, que termina a «subrotina», dirige o computador à linha 5Ø depois da primeira execução (chamada pela linha 4Ø) e à linha 8Ø depois da segunda volta (chamada pela linha 7Ø).
GOSUB e **RETURN** pode-lhe poupar trabalho e espaço na memória do computador Mas, mais importante ainda, organiza os seus programas de modo que, quando outras pessoas os utilizarem — ou mesmo você, quando mais tarde voltar a ele — compreendam como funcionam.






```
    10 REM PROGRAMA-MAT.
    20 LET A=INT (RND*9)+1
    30 LET B=INT (RND*9)+1
    40 PRINT "PRETENDE"
    50 PRINT TAB 10;"ADICIONAR-TECLE 1"
    60 PRINT TAB 10;"SUBTRAIR-TECLE 2"
    70 PRINT TAB 10;"MULTIPLICAR-TECLE
3"
    80 PRINT TAB 10;"DIVIDIR-TECLE 4"
    90 INPUT D
   100 IF D<1 OR D>4 THEN GO TO 40
   110 CLS
   120 GO SUB D*1000
   130 INPUT E
   140 PRINT AT 10,15;E
   150 IF E=C THEN PRINT AT
15,10;"CORRECTO"
   160 IF E<>C THEN PRINT AT 15,10;
"LAMENTO, ERROU O NUMERO"
   170 PRINT AT 17,10; "QUER OUTROS S OU
N?"
   180 INPUT A$
   190 CLS
   200 IF A$ <>"S" THEN STOP
   210 GO TO 20
  1000 LET C=A+B
  1010 PRINT AT 10,10;A;"+";B;"=   ?"
  1020 RETURN
  2000 LET C=A-B
  2010 PRINT AT 10,10;A;"-";B; "=   ?"
  2020 RETURN
```

```
3000 LET C = A * B
3010 PRINT AT 10, 10; A; "*"; B; "=  ?"
3020 RETURN
4000 LET  C = A/B
4010 PRINT AT 10, 10; A; "/"; B; "=  ?"
4020 RETURN
```

Na linha 100, use o «comando» **OR** (**SYMBOL SHIFT** e U), mas na linha 170, «tecle» a palavra OR letra a letra.
Este programa exemplo inclui um número de conceitos que discutimos em Capítulos anteriores. Também ilustra como o uso de «subrotinas» serve para tornar a estrutura de um programa, mais simples de seguir. Na verdade, este programa não requer o uso de «subrotinas».
Porque não? (Responder-lhe-emos, depois de lhe chamar a atenção para outras especificações do programa).

1. As linhas 20 e 30 usam o gerador de números aleatórios.

2. As linhas 50-80 usam **TAB** para formar uma coluna no ecrã.

3. As linhas 90, 130 e 180 usam **INPUT**.

4. A linha 100 faz a detecção de erro, repetindo a pergunta se introduzir um número que não seja aquele com que o programa pode operar.

5. A linha 120, em vez de utilizar 4 linhas diferentes para separar os endereços **GOSUB**, usa a multiplicação para seleccionar a «subrotina». Esta técnica pode também utilizar-se com **GOTO**.

6. A linha 140 usa **PRINT AT**.

7. As linhas 150 e 160 são ambas necessárias, como no exemplo apresentado no Capítulo 14. Porquê?

8. A linha 200 pára (**STOP**) o programa se se premir qualquer tecla que não S (de sim), embora no ecrã se veja a pergunta "S ou N?".

Isto porque, de outro modo, se se introduzisse algo diferente de S ou N, surgiria um erro, provocando uma falha no programa. Seria capaz de criar um detector de erro, tal como o utilizado na linha 100?

9. Cada «subrotina» faz uma operação matemática diferente, mas trabalha com o número gerado aleatoriamente, e dá uma resposta que vem testar a sua.

10. Erros por arredondamento podem causar-lhe problemas na «rotina» de divisão que começa na linha 4000. Será capaz de usar o processo de correcção dado no Capítulo 11 para remediar este facto?
A resposta à nossa quetão é que, realmente, pode utilizar **GOTO** D * 1000 para cada «subrotina» e **GOTO** 130, em vez de **RETURN**, no final de cada uma.

Como fazer «correr» mais rapidamente os programas BASIC

Eis um modo de fazer «correr» mais rapidamente os seus programas BASIC; apresentamo-lo aqui pois tem algo a ver com «subrotinas». Só necessitará disto quando começar a preparar longos programas, mas, então, esta experiência ser-lhe-á extremamente útil.

Capítulo 15: Programas Dentro de Programas: Subrotinas

Ao longo deste Manual discutimos a construção de programas, naquilo que considerámos ser a ordem lógica: primeiro estabelecemos (iniciamos) as variáveis (**LET** A = 1, etc.), a seguir, fazemos o programa principal, e por último completamos as «subrotinas».
Isto está correcto para os compiladores, mas uma vez que o TC 2048 usa um interpretador, a lógica de programação é diferente. Daqui resulta que o computador procure um número de linha indicado por um **GO TO** ou **GOSUB**, verificando cada número de linha desde o início do programa.

Isto significa que se a linha 10 é **LET** A = 1 (por exemplo e nunca mais é utilizada no programa, transforma-se num elemento extra a ser examinado por cada **GOTO** e **GOSUB**.
Logicamente, deve, então, ser afastado para o fim do programa... como uma «subrotina».
Assim, a sua primeira linha de programa deverá ser

```
10 GOSUB 9000
```

e todas as suas iniciais — variáveis definidas, gráficos, etc. — devem ser colocadas nesta «subrotina». Assim o programa nunca mais volta a apresentar essas instruções depois de executar o primeiro **GOSUB**, e como cada linha só é vista uma vez, o programa vai à procura de um novo número de linha.
Uma excepção a esta regra será pôr todas as instruções **DEF FN** no início do programa, pois com esta disposição, tais instruções serão procuradas todas as vezes que o programa necessita delas.

De igual modo, as «subrotinas» chamadas frequentemente deverão estar no início do programa — não no final.

A seguir, virá o programa principal, e depois as «subrotinas» menos utilizadas. O esqueleto do programa deverá parecer-se com:

```
  10  REM TÍTULO
  20  GOSUB 9000
 100  SUBROTINA A
 200  SUBROTINA B
 300  SUBROTINA C
1000  PROGRAMA PRINCIPAL
5000  SUBROTINA D
5100  SUBROTINA E
9000  SUBROTINAS DE INICIAÇÃO
9900  FUNÇÕES FINAIS
```

Verifique se cada módulo termina com uma direcção (normalmente um **GOTO**) se não quiser que o programa prossiga a execução na linha seguinte (por exemplo, necessitará de **GOTO** 9900 no final do programa principal para alcançar as funções finais).

SUMÁRIO

1. As «subrotinas» ajudam-no a utilizar as técnicas de programação estruturada, em BASIC, que tornam os programas de mais fácil compreensão.

2. **GOSUB** dirige o programa para uma linha específica, tal como o **GOTO**, mas guarda a localização de linha do programa que contém o **GOSUB**.

3. **RETURN**, a última linha da «subrotina», dirige o programa para a linha a seguir a **GOSUB**.

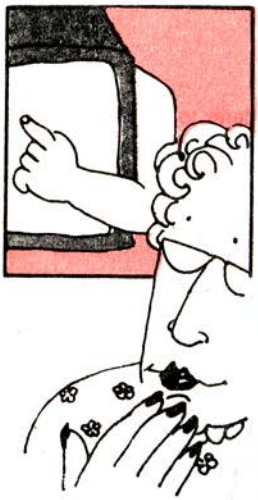
4. **GOSUB** e **RETURN** devem ser usados juntos, tal como **FOR** e **NEXT**.

5. Programas extensos «correrão» mais rapidamente se os blocos do programa utilizados menos frequentemente forem colocados depois daqueles que se utilizam mais vezes.

Matrizes 16

Síntese do Capítulo:

Organize os dados e poupe espaço com «matrizes», utilizando DIM, subscritos e cortes de «string».
Usando juntos SAVE e DATA, armazene «matrizes» em fita.



Uma matriz (array) é uma das formas de estruturar um número de valores ficando a conhecer a sua localização. Cada um dos valores é denominado um elemento da matriz. Pode pensar numa matriz como se se tratasse da folha mensal de um calendário.

DOM	SEG	TER	QUA	QUI	SEX	SAB
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

A Matriz

Um elemento

Poderá atribuir quaisquer valores numéricos aos elementos de uma matriz em vez de atribuir cada um dos valores a nomes de variáveis independentes. Este método pode utilizar-se não apenas para poupar memória mas também porque, por vezes, existe uma relação entre esses valores.

Suponha que tem uma fila de números

12 5 7 22 14

É evidente que pode atribuir estes valores a variáveis independentes, como

```
LET A = 12
LET B =  5
LET C =  7
LET D = 22
LET E = 14
```

mas também poderá considerar toda a fila de números como uma matriz. Assim:

```
LET A(1) = 12
LET A(2) =  5
LET A(3) =  7
LET A(4) = 22
LET A(5) = 14
```

Matriz A

(1)	(2)	(3)	(4)	(5)
12	5	7	22	14

O número que está entre parêntesis, a seguir à variável, denomina-se subscrito.

Imagine a matriz como na figura ao lado:

DIM... A declaração de DIMENSÃO

Antes de atribuir valores a cada elemento de uma matriz, terá de reservar, (na memória do computador) espaço para essa matriz. Conseguirá fazê-lo através de uma declaração **DIM** (dimensionamento).

Para preparar a matriz que vimos acima, teríamos de dar entrada a

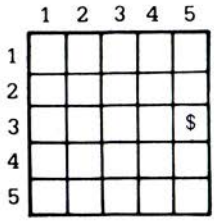
DIM A (5)

Existem algumas regras relacionadas com as variáveis da matriz.

1. O nome de cada variável de uma matriz não pode ter mais do que uma letra (como acontece com as variáveis de controlo num «ciclo **FOR/NEXT**»).
2. Uma variável de matriz pode ter o mesmo nome que outra variável simples. (Significa isso que, por hipótese pode ter, ao mesmo tempo no TC 2048 memorizadas duas variáveis com o nome A — uma delas será a variável simples A e a outra será a variável A da matriz, dimensionada para o número de elementos que tiver decidido). Uma não interfere com a outra pois os elementos da variável da matriz são sempre referidos através do seu subscrito.
3. Uma declaração **LET** apaga a variável do mesmo nome que eventualmente já exista na memória do computador.
   Da mesma forma, quando cria uma matriz através de um **DIM**, apaga qualquer outra matriz, do mesmo nome que já exista.
   Contudo, embora seja possível construir uma nova variável simples a partir de uma antiga (como em **LET** A=A+1), perderia toda a informação da antiga matriz pelo simples facto de **DIM**ensionar uma nova matriz com o mesmo nome.
   Mas não deixa de ser possível alterar um elemento da matriz, se o desejar, usando

   LET A(1)=A(1)+1

DIM A(5,5)



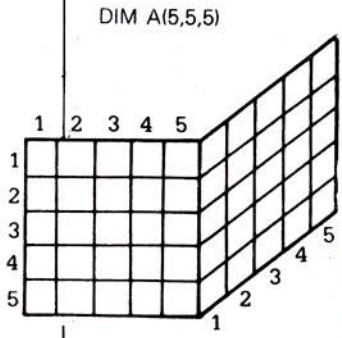
Matrizes de mais de uma Dimensão

Pode obter matrizes com as dimensões que desejar, desde que as dimensione convenientemente no início:

DIM A(5,5)

dimensiona uma matriz do género da que está desenhada aqui ao lado. Pode supôr que o primeiro número indica a fila e o segundo a coluna. Assim, na figura, a localização do $ será A(3,5).

Que valor indica a posição (3,5) na nossa folha de calendário?



Também é possível criar qualquer matriz em três dimensões. Pense nisso, reparando no aspecto da figura abaixo:

DIM A(5,5,5)

Nesta ordem de ideias, também poderia criar matrizes com quatro ou mais dimensões. Mas se for capaz de as representar graficamente, então é porque está, também, em condições de explicar a teoria da relatividade!

Matrizes de caracteres
Pode atribuir matrizes de caracteres, como

DIM A$(5,5)

mas há também regras especiais, neste caso.

1. Quando **DIM**ensionar uma matriz de caracteres, apaga não somente qualquer outra com o mesmo nome mas também «strings» de variáveis do mesmo nome, ainda que simples.
2. Em duas dimensões, poderá pensar no primeiro número como sendo o que identifica cada uma das «strings» ou «palavras» e o segundo como aquele que fixa a quantidade de caracteres contidos em cada uma das palavras.
3. A atribuição aos elementos, da variável de cadeia, é Procrusteana, que é como dizer obrigatória para a ocupação de todos os caracteres **DIM**ensionados, desde o carácter n.º 1 até ao último fixado pelo segundo subscrito. Assim:
   a. se indicar um número maior de caracteres do que os designados pelo segundo subscrito, quando atribuir a «string», os caracteres excedentes serão automaticamente perdidos;
   b. se a «string» for demasiado curta, não atingindo o número de caracteres indicados pelo segundo subscrito, os restantes espaços serão preenchidos com vazios.

Matriz A$

	1	2	3	4	5
1	T	E	R	R	A
2	C	O	R	D	A
3	D	I	A	M	A
4	C	L	U	B	
5	J	O	K	E	R

Porquê Procrusteana? O método é referido por analogia com uma velha lenda da mitologia grega. Procruste teria sido um antigo salteador (outros dizem estalajadeiro) de Ática. Apanhava os viajantes e deitava-os numa cama de ferro. Se os viajantes fossem curtos e não chegassem aos pés da cama, estirava-os, com o auxílio de um martelo e cordas. Se fossem demasiado compridos e saíssem fora da cama, então cortava-lhes os pés. Foi por este modo que Teseu acabou, afinal, por matar Procrustes. (Também se diz cama de Procruste).

No TC 2048 é assim mesmo. O comprimento da «string» **DIM**ensionada tem sempre o número de caracteres rigorosamente exacto encarregando-se o computador de eliminar os excedentes ou acrescentar espaços vazios.

Neste tipo de «strings», o acesso é obtido por utilização limitada apenas ao primeiro subscrito, quando se trate de palavras, mas dos dois subscritos quando se pretenda uma letra determinada.

No exemplo que inserimos ao lado, **PRINT** A$(3) faz imprimir DIAMA (o NTE da palavra DIAMANTE foi automaticamente eliminado), e **PRINT** A$(4) faz imprimir CLUB
(incluindo um espaço em branco depois do B).

Também **PRINT** A$(3,2) fará imprimir a letra I (ou seja a segunda letra da linha 3 — A$(3).

PRINT A$(4,5) dar-nos-á um espaço em branco.

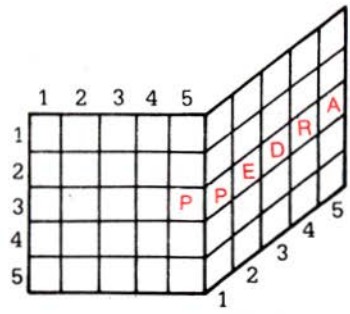
Também poderá utilizar o método de «cortar a string» (já vamos ver o que é, daqui a pouco) para obter apenas uma parte da «string»:

PRINT A$(4,2 **TO** 4) fará imprimir LUB
PRINT A$(3, **TO** 3) fará imprimir DIA

Matrizes «alfanuméricas» — em três ou mais dimensões

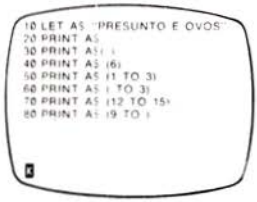
Pode criar matrizes de «string» («string arrays») de variáveis em quantas dimensões desejar. Contudo, nessas dimensões, sempre separadas por vírgulas, há que ter presente que o último número indica a quantidade de caracteres de cada «string» enquanto os números anteriores especificam a «string» pela sua localização. Em três dimensões:

PRINT A$(3,5,5) para um «string» de variável do tipo que se desenhou aqui abaixo. Fará imprimir a palavra PEDRA.



Novamente dizemos que não é muito fácil configurar (em diagrama) matrizes de «string» para mais do que três dimensões. Lembre-se sempre de que o último subscrito especifica o número de caracteres de cada «string» na matriz, enquanto os outros números indicam as coordenadas ou localização da «string».

```
10 LET A$="PRESUNTO E OVOS"
20 PRINT A$
30 PRINT A$( )
40 PRINT A$ (6)
50 PRINT A$ (1 TO 3)
60 PRINT A$ ( TO 3)
70 PRINT A$ (12 TO 15)
80 PRINT A$ (9 TO )
```



Como cortar «strings»

Se introduzir o programa abaixo listado obterá uma demonstração exacta do que é o corte de «strings».

A linha 20 imprime a «string» na sua totalidade.

A linha 30 demonstra que quando se colocam os parênteses (à frente da variável de cadeia), sem qualquer conteúdo, o computador entende que a «string» completa corresponde à própria «substring».

A linha 40 selecciona, como «substring», apenas um carácter (o sexto). Não se esqueça de contar também com os espaços.

A linha 50 faz a impressão dos caracteres 1 a 3.

A linha 60 prova que pode emitir o primeiro dígito, caso em que o primeiro caracter da «string» é automaticamente considerado.

A linha 70 demonstra como é possível imprimir apenas os últimos quatro caracteres.

A linha 80 mostra que pode omitir o último dígito, caso em que se considera todo o resto da «string» (até ao fim).

Como é que imprimiria «E» fora daquela string?

A partir de agora poderemos utilizar «o corte de strings» para poupar espaço de memória, num programa. Atribui-se a uma variável de cadeia um longo conjunto de caracteres; depois, é só cortar pedaços dela em vez de estar a atribuir uma quantidade de variáveis.

Como gravar e «carregar» matrizes em fita de «cassettes»

Poderá gravar (SAVE) uma matriz em fita magnética, utilizando os comandos SAVE e DATA:

SAVE "QUADRO" DATA A ( )

registaria magneticamente, sob o nome de "QUADRO", uma matriz numérica que tivesse criado e a que desse o nome A. Entre outras coisas, isso dá-lhe a possibilidade de armazenar matrizes de dados sob uma designação mais descritiva do que a designada apenas com uma letra. Além disso, pode armazenar e localizar mais do que 26 matrizes (dado que poderá voltar a utilizar as letras do alfabeto).

É preciso incluir os parêntesis, ainda que não tenha necessidade de indicar os números que fazem parte da matriz original.

Poderá «recarregar» a matriz gravada através de

LOAD "QUADRO" DATA A ( )

Terá, normalmente, de utilizar este método em programas que já estejam no computador e que trabalhem na base de DATA; LOAD com DATA não apagam o que já está na memória do computador (a menos que exista lá uma outra matriz que tenha extactametne o mesmo nome).

As matrizes de «string» são tratadas da mesma forma, mas não esqueça que é necessário utilizar o $ no nome da matriz.

Capítulo 16: Matrizes

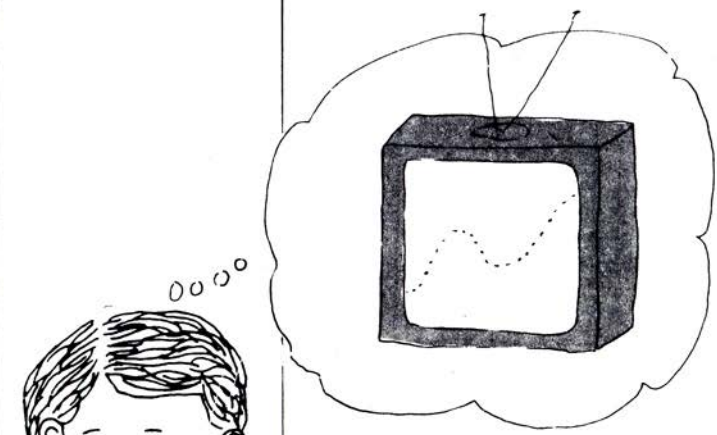
SUMÁRIO

1. Uma matriz é constituída por um determinado número de elementos, todos eles com o mesmo nome de variável e que se distinguem, entre si, pelos subscritos.
2. O nome de uma matriz numérica deve ser uma letra simples; é possível, contudo, que exista paralelamente uma variável simples com o nome dessa mesma letra.
3. O nome de uma matriz «alafanumérica» deve ser obrigatoriamente composta por uma única letra à qual se junta um $; não é possível a existência simultânea na memória do computador de uma variável de cadeia com o mesmo nome.
4. Antes de atribuir quaisquer valores aos elementos de uma matriz, torna-se indispensável reservar o respectivo espaço na memória do computador, utilizando a declaração **DIM**.
5. Podem «cortar-se as strings» e obter «dessa forma» «substrings» através da utilização de **TO**.
6. A informação (**DATA**) relativa a matrizes (numéricas ou «alfanuméricos» pode ser gravada ou «carregada» através dos comandos habituais, mas incluindo **DATA** como declaração.

Gráficos 17

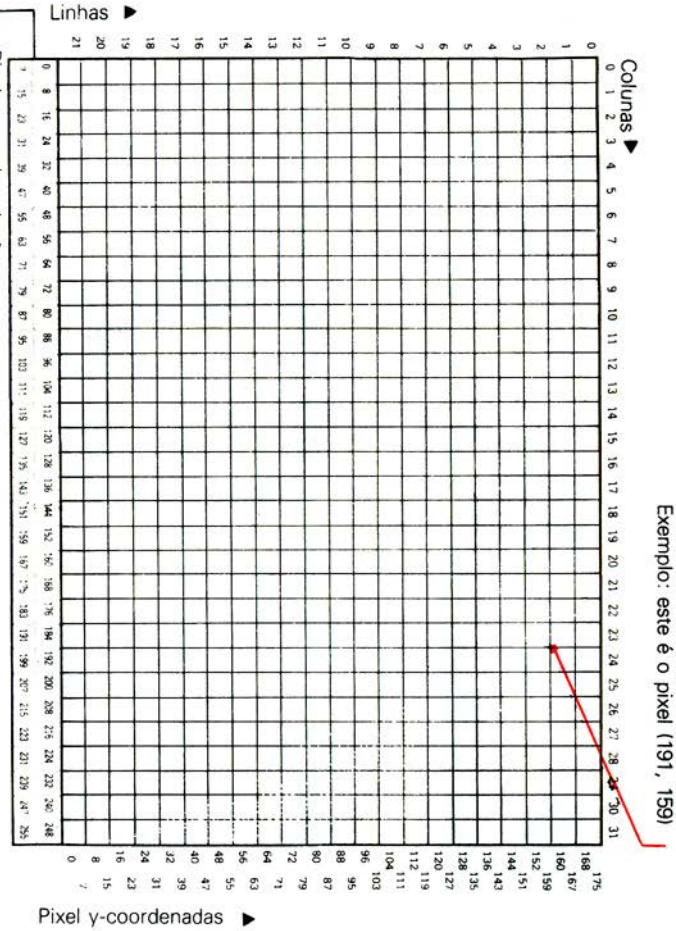
Síntese do Capítulo

Este Capítulo oferece-lhe uma observação completa da técnica de produção de imagens no ecrã, utilizando PRINT e PLOT. Verá também como se podem gravar essas imagens com SCREEN$



Existem diversas formas de produzir gráficos com o seu Timex Computer 2048. Quatro desses métodos, extremamente fáceis de aprender, serão discutidos neste Manual:

1. Utilizando o «modo» gráfico (cursor **G**) poderão imprimir-se combinações dos caracteres gráficos existentes nas teclas numéricas
2. Poderá criar os seus próprios gráficos e gravá-los em memória, podendo utilizá-los por simples comando de uma só tecla, no «modo» gráfico. Trataremos desse aspecto no próximo Capítulo.
3. Como já vimos no Capítulo 4, podemos utilizar as funções CIRCLE e DRAW.
4. Pode utilizar-se a função PLOT, introduza também no Capítulo 4 e que exploraremos melhor neste.


Exemplo: este é o pixel (191, 159)
Colunas ▶
Linhas ▶
Pixel x-coordenadas ▶
Pixel y-coordenadas ▶
Não pode fazer PRINT ou PLOT nestas duas linhas.

Existem ainda mais três métodos de produzir gráficos que podem efectivamente explorar as capacidade do TC 2048. São métodos que estão ao alcance de programadores avançados (ou de alguém que realmente conheça gráficos) utilizando código máquina.

Primeiro, podem expandir o «display» de 32×24 para um de 64×24 caracteres (ou até mais de 64, se formar os seus próprios caracteres) utilizando 512 «pixels» como largura do ecrã.

Segundo, poderá utilizar duas vezes o «display» de 32×24 caracteres, movimentando-os alternadamente para conseguir efeitos grandiosos em animação de imagens.

Finalmente TC 2048 permite, no «modo extensivo de cor», até oito cores para cada um dos caracteres. — Cada fila de 1×8 «pixel» pode receber uma cor diferente. Neste «modo» pode criar efeitos coloridos de ultra-alta resolução para conseguir gráficos tanto para distracção como para fins didácticos ou comerciais. Veja o Apêndice C.

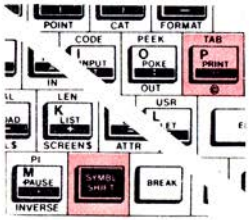
Como utilizar símbolos gráficos com declarações PRINT

Os símbolos gráficos das teclas numéricas fazem parte do conjunto de caracteres básicos do TC 2048 — veja o Apêndice B. São impressos utilizando **PRINT** e um ecrã baseado em coordenadas de 32×24 caracteres.

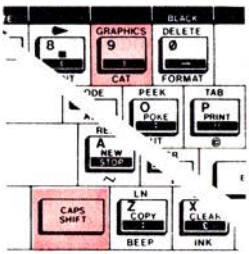



```
  5 REM PROGRAMA — GRAF.BARRAS
 10 FOR I=1 TO 3
 20 PRINT AT 5,0;"INTRODUZA NOME (10
CARACTERES) #";I: INPUT A$
 30 PRINT AT 1,10*I-10;A$
 40 NEXT I
 50 PRINT AT 5,0;"
 60 FOR I=1 TO 3
 70 PRINT AT 20,0;"INTRODUZA O VALOR (0 A
15)#";I: INPUT A
 80 FOR B=19 TO 19-A STEP -1
 90 PRINT AT B,10*I-7;"■"
100 NEXT B
110 NEXT I
120 PRINT AT 20,0;"
```

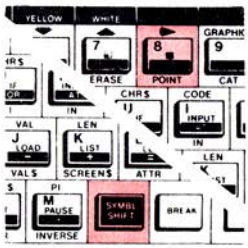
Introduza o programa listado na página anterior. Serve para imprimir um mapa de barras. O símbolo gráfico da linha 9Ø é o «inverso» do quadrado preto que está marcado na tecla 8. Para o obter, depois de ter colocado o «ponto e vírgula» (;), faça o seguinte:



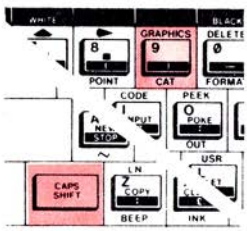
1. SYMBOL SHIFT e P para as primeiras aspas.



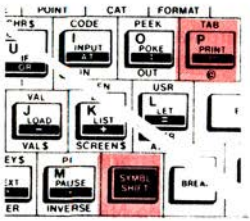
2. CAPS SHIFT e 9 para entrar em «modo» gráfico.



3. SYMBOL SHIFT e 8 para obter o quadrado preto.



4. CAPS SHIFT e 9 para abandonar o «modo» gráfico.



5. Premir **SYMBOL SHIFT** e P para fechar aspas.

Outras observações sobre o programa:

1. O 1.º **INPUT** que aparece na linha 2Ø é escrito letra a letra, enquanto que o 2.º da mesma linha e o da linha 7Ø é o «comando» que se encontra na tecla I.

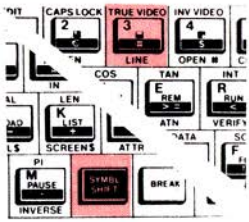
2. O número de espaços entre aspas, na linha 5Ø é igual a 3Ø ou seja, o número de caracteres — incluindo espaços — que se encontram no disposto na linha 2Ø, mais um caracter para a variável I.
   Na linha 12Ø, o número de espaços deverá ser achado do mesmo modo, para apagar a linha 7Ø. Conte-os!
   Com este programa tipo pode generalizar e obter um gráfico para qualquer fim. No nosso caso programamos os nomes dos meses e atribuimos valores que representassem algo que tivese sucedido em cada mês — vendas, despesas, idas ao jardim zoológico. Utilize os seus próprios dados, para executar o programa.
   (No final deste Capítulo, veremos como grava (**SAVE**) um ecrã em «cassette», e no Capítulo 22 veremos como passá-lo para a impressora).
   Imagina como os números especificados nas instruções **PRINT AT**, das linhas 3Ø e 9Ø, colocam os títulos e as barras?
   Verifique como as frases introduzidas em 2Ø e 7Ø são apagadas pelas 5Ø e 12Ø.
   Repare na existência de dois ciclos **FOR/NEXT**, um dentro do outro.
   Veja como a linha 8Ø imprime em barras verticais e, efectivamente, de baixo para cima.
   Consegue modificar o programa para obter barras coloridas?



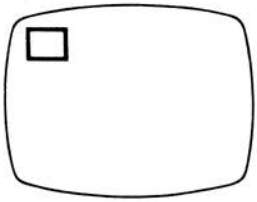
Capítulo 17: Gráficos

Faz uma ideia de como pode conseguir barras com o dobro ou o triplo da largura?
Como poderia imprimir barras horizontais, em vez de verticais?
É fácil...
Poderá utilizar os vários símbolos gráficos do teclado, em combinação com a declaração **PRINT**, para desenhar figuras no ecrã.
Experimente (mas não se esqueça do **NEW** antes de tudo, para apagar o programa anterior):

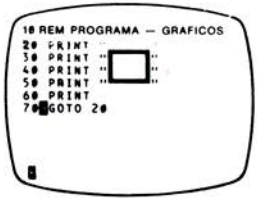
```
10 REM      PROGRAMA — GRÁFICOS
20 PRINT    "          "
30 PRINT    "          "
40 PRINT    "          "
50 PRINT    "          "
```



Sugestão: Na linha 20, depois das primeiras aspas, entre em «modo» gráfico. Depois, faça **SYMBOL SHIFT** e 3 (para obter o inverso do caracter que está gravado na tecla 3). Faça isso por cinco vezes e depois saia do «modo» gráfico e feche as aspas.



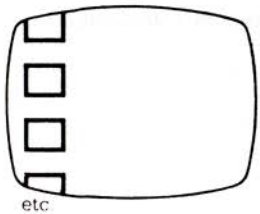
Na linha 30 entre novamente em «modo» gráfico e use o **SYMBOL SHIFT** e a tecla 5 para obter o gráfico do lado esquerdo. Depois faça três espaços em branco. Finalmente, para obter o símbolo gráfico do lado direito deverá premir a tecla 5, mas, desta vez, sem **SYMBOL SHIFT**.
A linha 40 repete as instruções da linha 30 e a 50 as da 20, mas, atenção, sem usar **SYMBOL SHIFT**.



Também é possível utilizar desenhos gráficos por mais de uma vez num mesmo programa.
Acrescente ao programa de cima mais as seguintes linhas:

```
60 PRINT
70 GOTO 20
```

e faça «correr» o programa (**RUN**... **ENTER**).



O que produziria o programa sem a linha 60?
E o que aconteceria se fizesse **GOTO** 30 em vez de **GOTO** 20?

Experimente agora traçar outras figuras mais elaboradas, utilizando os símbolos gráficos. Repare como é possível fazer com que haja coincidências entre eles.

Experimente também o programa seguinte. (Tenha cuidado e verifique se está em «modo G». Se assim não for, o programa deixa de funcionar). Será capaz de modificar o programa de maneira a que possa escolher a cor em vez de a obter por selecção do **RND**?




```
 5 REM PROGRAMA — DESENHO A CORES
10 PRINT AT 20,0;"Q=CIMA A=BAIXO"..
"O=ESQUERDA P=DIREITA"
20 LET A=0: LET B=0
30 PRINT AT A,B; INK INT (RND*7)+1;"■"
40 IF INKEY$="Q" THEN LET A=A+1
50 IF INKEY$="A" THEN LET A=A-1
60 IF INKEY$="O" THEN LET B=B-1
70 IF INKEY$="P" THEN LET B=B+1
80 GO TO 30
```

Evite sair fora do **PAPER** em qualquer das direcções. Se isso acontecer, não só o programa pára, como poderá «carregar» a parte inferior do ecrã com uma «string» de caracteres (por virtude de «auto-repetição»).
Há uma forma de evitar isso: Junte as seguintes linhas ao programa, para que não possa ultrapassar os limites do **PAPER** no ecrã:

```
45 IF A<0 THEN LET A=0
55 IF A>21 THEN LET A=21
65 IF B<0 THEN LET B=0
75 IF B>31 THEN LET B=31
```

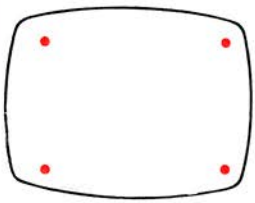
(INKEY$ — tecla N no «modo E» — explica-se completamente no Capítulo 19).

Experimente o programa depois de ter mudado a cor do PAPER. Use também diferentes cores para o BORDER.

Gráficos de alta resolução com a declaração PLOT

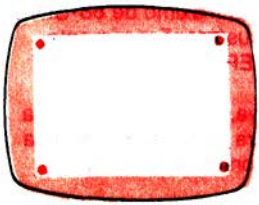
O formato da declaração PLOT é o seguinte:
PLOT x,y
em que x é um número entre Ø e 255 (da esquerda para a direita do ecrã) e y um número entre Ø e 175 (de baixo para cima).

O comando PLOT enche o «pixel» (picture element-«elemento da imagem») localizado pelas duas coordenadas x, y com uma pequeníssima caixa, tão diminuta que parece um simples ponto. Eis um programa que delimita o ecrã para o comando PLOT, colocando um ponto em cada canto do PAPER:



```
1Ø PLOT Ø,Ø
2Ø INPUT A$
3Ø PLOT Ø,175
4Ø INPUT A$
5Ø PLOT 255,Ø
6Ø INPUT A$
7Ø PLOT 255,175
```

Depois de ter introduzido o programa e feito RUN e ENTER, terá uma certa dificuldade em encontrar o ponto. Mas, ele está lá, no canto inferior esquerdo. O programa espera, agora, por um INPUT antes de produzir cada um dos outros pontos, nos restantes cantos — o computador nada fará. Este é um método que serve perfeitamente como dispositivo para parar o computador e esperar pela nossa decisão para continuar o seu trabalho. Bastará premir qualquer tecla. Poderá responder com a tecla ENTER perante o INPUT presente.

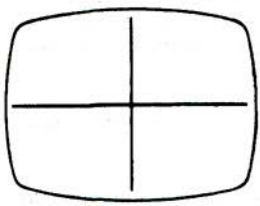


Verifique agora a posição dos quatro pontos em relação ao **BORDER**, premindo

BORDER 4 **ENTER**

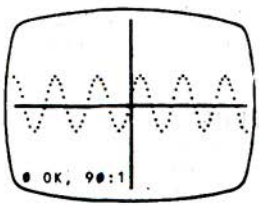
Pode localizar os eixos x e y no ecrã com este pequeno programa:

```
10 REM PROGRAMA — EIXOS X/Y
20 FOR X=0 TO 255
30 PLOT X,87
40 NEXT X
50 FOR Y=0 TO 175
60 PLOT 127,Y
70 NEXT Y
```



Dado que os eixos se cruzam no meio do ecrã, na posição 127,87, torna-se um pouco complexo fazer o «plot» de gráficos matemáticos (nos quais o ponto onde os eixos se cruzam é definido por 0,0 de forma a permitir números negativos abaixo do eixo x e para a esquerda do eixo y).

Eis agora um programa que traça uma onda sinusoidal. Note que integrámos as linhas deste programa no anterior para tracar os eixos.




```
 5 REM PROGRAMA — SINUSOIDE
10 FOR X=0 TO 255
20 PLOT X,87
30 NEXT X
40 FOR Y=0 TO 175
50 PLOT 127,Y
60 NEXT Y
70 FOR I= -20 TO 20 STEP .03
80 PLOT I*6+127,87+20*SIN I
90 NEXT I
```

A linha 70 dimensiona o gráfico de forma a caber no **PAPER**. Experimentámos até atingir o limite que é de −20 a +20 e incluimos o **STEP** para colocar os pontos próximos uns dos outros de forma a constituirem uma linha contínua. Se tentar **STEP** .08 apressa o traçado da sinusóide mas os pontos afastam-se uns dos outros.

Se alterar o +2Ø•SIN I até um máximo de 88•SIN I aumenta a amplitude da sinusóide. Experimente, jogando com o valor do **STEP**.

Quando desenhar uma curva, deve ter cuidado na escolha dos eixos x e y, para obter tanto uma boa definição dos respectivos pontos como uma melhor imagem no ecrã.

O 6 na linha 8Ø torna mais simples a visualização da curva. Deve ter a certeza que a utilização desse número não distorce o gráfico. Felizmente, pode dimensionar o eixo dos x, de acordo com o período de uma onda sinusoidal que é a distância entre dois pontos similares em ciclos sucessivos. Isto é 2. Deve marcar o eixo dos x de acordo. A amplitude é 2Ø, já que uma onda de amplitude 1 tem a forma y=**SIN**x. isto refere-se ao 2Ø na linha 8Ø e será o guia para o dimensionamento do eixo dos y.

O 127 e o 87 na linha 8Ø servem, claro está, para centrar no ecrã a origem dos eixos.

Pode-se, evidentemente, representar graficamente parábolas, linhas rectas e todo o tipo de funções matemáticas.

Mantenha a onda sunusoidal no ecrã; utilizá-la--emos para praticar a gravação de uma imagem em fita.

Gravar ecrãs com SCREEN$

Sempre que tenha uma imagem no ecrã e a quiser guardar — o que normalmente acontece com um desenho artístico ou com um gráfico — poderá utilizar o comando **SCREEN$**.

Para gravar o gráfico da sinusóide, «tecle» **SAVE** "SINUSOIDE" **SCREEN$** utilizando o «comando» **SAVE** da tecla S, «teclando» o nome da imagem — pode dar-lhe o nome que quiser — entre aspas, e adicionando, por fim, a função **SCREEN$**, (Com o cursor ◘, premir simultaneamente a tecla K e **SYMBOL SHIFT**).

Para voltar a chamar a imagem ao ecrã, teriade fazer:

LOAD "SINUSOIDE" SCREEN$

Com excepção para o facto de se acrescentar **SCREEN$**, o processo é exactamente igual àquele que já foi demonstrado no Capítulo 4 para gravar ou «carregar» programas.
LOADing **SCREEN$** (isto é, o «carregamento» da imagem de um ecrã) não faz apagar o programa que está no computador.
Note-se que **VERIFY** não trabalha para verificar **SCREEN$**.

SUMÁRIO

1. Os símbolos gráficos das teclas numéricas (que duplicam o seu número se se usar **TRUE VIDEO** e **INVERSE VIDEO**) inscrevem-se no ecrã através do comando **PRINT**, ocupando o espaço de um carácter no **PAPER** que contém 32 × 22 caracteres.

   PRINT AT 1Ø,1Ø; " "

2. **PLOT** coloca um ponto — preto ou noutra cor (**INK**) — num «pixel» (elemento da imagem) que se define por duas coordenadas, separadas por uma vírgula:
   O primeiro número ou coordenada horizontal vai de Ø a 255, contando da esquerda para a direita;
   O segundo ou coordenada vertical, vai de Ø a 175, contando-se de baixo para cima.

   PLOT 255, 175 produz um ponto no canto superior direito.

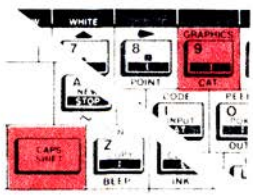
3. SCREEN$ deve acrescentar-se ao **SAVE** ou ao LOAD, respectivamente, quando se pretende gravar ou «carregar» o conteúdo de um ecrã de informação ou gráfico.
   O ecrã (SCREEN$) pode ser carregado por si só ou para utilização dentro de um programa.

Gráficos a Defenir Pelo Utilizador 18

Síntese do Capítulo ***Pode definir os seus próprios gráficos, símbolos ou caracteres, utilizando BIN.***
Também pode colocá-los na memória do TC 2048 usando POKE e USR e voltar a chamá-los, posteriormente, através do «modo» gráfico.

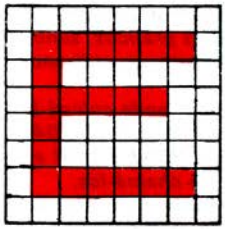


Além dos gráficos das teclas numéricas de que já falámos, poderá definir ou criar os seus próprios gráficos. Utilizará para isso as letras de A a U que servirão para os guardar em memória.
Posteriormente pode ter acesso a esses gráficos se carregar em cada uma das respectivas teclas, mas no «modo» gráfico.



Com o cursor em **K** faça CAPS SHIFT e 9 para obter o cursor **G** do «modo» gráfico.
Agora toque em qualquer das teclas 1 a 8.
Se premir, a seguir, qualquer das teclas A a U, tem exactamente o mesmo resultado que teria se estivesse um «modo» **C**, porque se imprimem letras maiúsculas. Mas acontece que lhe é possível alterar os caracteres nas posições de memória representadas por essas letras.

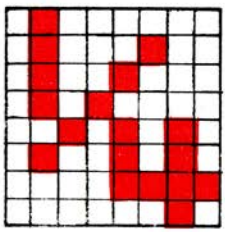
Capítulo 18: Gráficos a Definir Pelo Utilizador



Cada um dos caracteres — letras, números ou símbolos — é feito de «pixels» (quadradinhos em miniatura) numa matriz do oito por oito. Por exemplo, a letra E é feita assim:

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	1	1	1	1	1	Ø
Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	1	1	1	1	Ø	Ø
Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	1	1	1	1	1	Ø
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Pense que as áreas enegrecidas, na matriz, são uns (1) e que as áreas livres ou brancas são zeros (Ø). Veja aqui ao lado como se representaria um E.

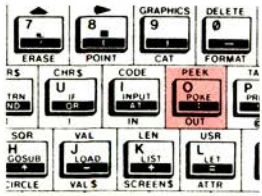


Suponha agora que pretendia um novo caracter (1/4) que o computador fizesse imprimir como se vê na imagem do lado.
O melhor seria começar por desenhá-lo em papel quadriculado.

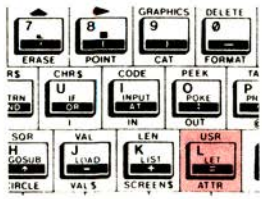
Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	Ø	Ø	Ø	1	Ø	Ø
Ø	1	Ø	Ø	1	Ø	Ø	Ø
Ø	1	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	1	Ø	1	Ø	1	Ø
Ø	1	Ø	Ø	1	Ø	1	Ø
Ø	Ø	Ø	Ø	1	1	1	1
Ø	Ø	Ø	Ø	Ø	Ø	1	Ø

Depois, traduziria os quadradrinhos negros por uns (como se viu acima para o E) e os quadrados brancos por zeros, tal como os representamos aqui.

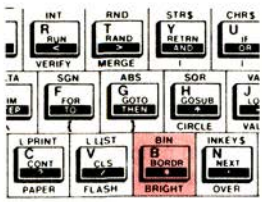
Cada linha de oito dígitos («bits», da matriz ou grelha) ocupa um lugar específico na memória. E é possível memorizar valores deste tipo, utilizando a declaração **POKE**.



POKE, o «comando» que está na letra O, é um comando normalmente seguido de dois números, sendo o primeiro deles relativo ao endereço na memória e o segundo um número que deve ser colocado nesse mesmo endereço.



USR é a função que está situada acima da tecla L. Supondo que pretendemos formar um novo caracter na tecla E, utilizamos **USR** "E" para indicarmos ao computador exactamente o início da memória (endereço) onde está localizado o UDG (Gráfico Definido pelo Utilizador).



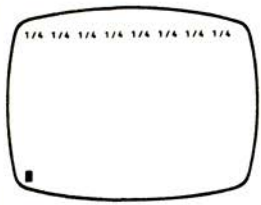
BIN (que quer dizer Binário) é uma função que está acima da tecla B e serve simplesmente para assinalar ao computador que deve esperar por um «byte» de dígitos binários e não por um número decimal.

POKE USR "E", **BIN** Ø1ØØØØØØ fará armazenar a primeira linha do nosso caracter (1/4) na primeira linha do UDG E.

Claro que para formar todo o novo caracter, vamos ter que dar entrada a mais sete linhas. A segunda linha entrará assim:

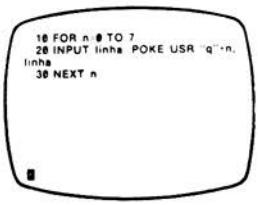
POKE USR "E" + 1, **BIN** Ø1ØØØ1ØØ (compare com o desenho).

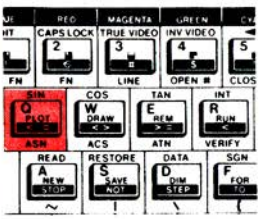
Já deve ter percebido que as outras linhas entram de maneira idêntica, mas, com **POKE USR** "E" + 2... até **USR** «E» + 7 e (depois da vírgula) os valores **BIN** a indicar corresponderão aos do desenho.



Quando tiver introduzido as oito linhas do novo caracter, faça **CAPS SHIFT** e 9 para obter o «modo» gráfico (**G**) e, depois, toque na tecla 3. Lá está o novo caracter 1/4!

Capítulo 18: Gráficos a Definir Pelo Utilizador





Uma vez definido um caracter gráfico a partir de uma letra, obtê-lo-á sempre que «teclar» essa letra em «modo» gráfico, até que

a) defina outro caracter, ou
b) desligue o computador (**NEW** ou **CLEAR** não o apagará).

Pode tornar mais simples a definição de caracteres gráficos se «teclar» o problema seguinte:

```
10 FOR n = 0 TO 7
20 INPUT linha: POKE USR "q" + n, linha
30 NEXT n
```

«Corra» (**RUN**) o programa e cada vez que ele peça os dados (**INPUT**), introduza uma linha. Neste caso, use

BIN 00010000

para cada uma das oito linhas.

Quando o programa deixar de pedir dados, obtenha o cursor **G** e «tecle» Q. Adivinhe o que obterá!

Desenhe outros gráficos e utilize o programa para os obter. Terá de usar outras letras para além do "q", cada vez que queira obter um novo gráfico.

Guarde uma lista dos gráficos correspondentes a cada tecla.

Como os gráficos definidos pelo utilizador desaparecem sempre que se desliga o computador, é natural que queira gravar um programa que contenha todos esses gráficos e que os reponha automaticamente. Será capaz de o fazer?

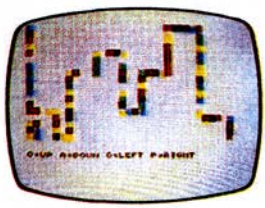
SUGESTÃO: Experimente utilizar **READ** e **DATA**.

Capítulo 18: Gráficos a Definir Pelo Utilizador

Com um tal programa é possível criar e memorizar alfabetos alternativos, assim como elementos gráficos para utilizar em jogos ou, até, símbolos de que possa carecer e não estejam incluídos no conjunto de caracteres do TC2048, tal como o 1/4 que acabámos de formar.

SUMÁRIO

1. **POKE** serve para introduzir novas informações na memória do computador; é seguido de dois números separados por uma vírgula. O primeiro número representa o endereço na memória e o segundo a informação (compreendido entre 0 e 255).
2. **BIN** quando seguido de oito dígitos binários (uns ou zeros) cria uma das oito linhas necessárias para formar um símbolo gráfico através dos UDG.
3. **POKE USR** "A", **BIN** 11111111 produz uma linha preta (que também poderá ser de outra cor) feita através de oito pontos, na primeira das oito linhas do UDG A. Para introduzir uma linha branca, ou de espaços, na segunda linha do caracter UDG A, a definir, seria necessário fazer **POKE USR** "A" + 1, **BIN** 00000000.



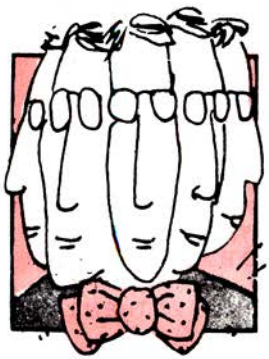
Este é o desenho do programa COLORSKETCH da pág. 155

Tempo e Movimento

19

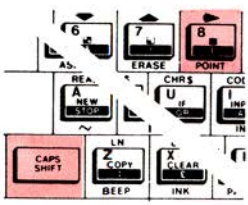
Síntese do Capítulo

Este Capítulo diz respeito aos meios que permitem colocar as imagens em movimento (INKEY$). PAUSE faz com que tudo pare por momentos.



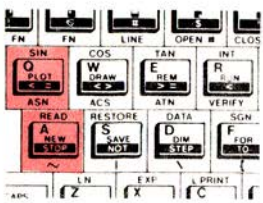
O comando **INKEY$** é parecido com o **INPUT**, exceptuando o facto de que não espera por si. **INKEY$** pesquisa o teclado, para detectar se alguma tecla foi premida — e qual delas —, para que o computador tome as medidas adequadas de acordo com a programação.

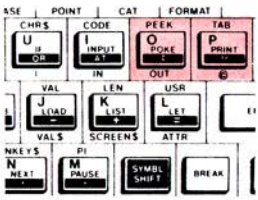
O programa que se segue é de um jogo infantil — acertar no alvo — que produz, (linhas 2Ø a 7Ø) um alvo quadrado no centro do ecrã. Para que possa conseguir-se o quadrado colorido da linha 7Ø, é necessário entrar em «modo» gráfico e fazer **CAPS SHIFT** e 8; mas terá de sair desse «modo» para conseguir fechar as aspas.
Depois, é só controlar o quadrado axadrezado (gráfico 6) que surgirá no canto superior esquerdo do ecrã.
As linhas 12Ø a 16Ø utilizam um gerador de número aleatórios (ao acaso) para fazer deslocar o quadrado preto, dois espaços de cada vez em qualquer direcção.



```
  5 REM PROGRAMA — ALVO
 10 BORDER 0: PAPER 0: CLS
 20 LET X=15
 30 LET Y=11
 40 LET A=0
 50 LET B=0
 60 PRINT AT A,B;" "
 70 PRINT AT Y,X; INK INT (RND*7)+1;"■"
 80 FOR N=1 TO 17
 90 NEXT N
100 PRINT AT Y,X; INK 0;" "
110 PRINT AT A,B; INK 0;" "
120 LET C=INT (RND*4)+1
130 IF C=1 AND X<=29 THEN LET X=X+2
140 IF C=2 AND X>=2 THEN LET X=X-2
150 IF C=3 AND Y<=19 THEN LET Y=Y+2
160 IF C=4 AND Y>=2 THEN LET Y=Y-2
170 IF INKEY$="O" AND B>=1 THEN LET
B=B-1
180 IF INKEY$="A" AND A<=20 THEN LET
A=A+1
190 IF INKEY$="Q" AND A>=1 THEN LET
A=A-1
200 IF INKEY$="P" AND B<=30 THEN LET
B=B+1
210 IF A=Y AND B=X THEN GOTO 900
220 PRINT AT Y,X; BRIGHT 1; INK INT (RND*6)
+2;"■"
230 PRINT AT A,B; INK 7;" "
240 GOTO 80
900 FOR N=1 TO 50: PRINT AT 0,0; PAPER 7;
INK 2;"VOCE ACERTOU!"
910 BORDER 0
920 PRINT AT A,B; BRIGHT 1; INK INT (RND*6)
+2; FLASH 1;" "
930 PRINT AT A,B; INK 7; FLASH 1;" "
940 BORDER 2
950 NEXT N
960 PAUSE 1000: STOP
```

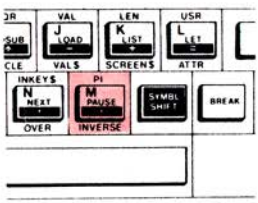
As linhas 170 a 200 movimentam o quadrado axadrezado deslocando-o um espaço de cada vez de acordo com as indicações que fornecer ao computador ao premir as teclas A e Q (do lado esquerdo do teclado) ou as teclas O e P (do lado direito).
A segunda parte de cada uma das linhas 130 a 200 evita que o seu quadrado possa sair fora dos limites do ecrã (**PAPER**).
As linhas 220 e 230 imprimem os quadrados nas posições calculadas depois de as linhas 100 e 110 os terem retirado das suas últimas posições (imprimindo nessas posições um espaço negro que, por ser da mesma cor do **PAPER**, fazer desaparecer os quadrados.)
As linhas 80 e 90 introduzem uma curta pausa — **PAUSE** — nas actividades do computador, entretendo-o com uma contagem comandada através de um «ciclo» vazio. Conta muito rapidamente entre 1 a 17. É um meio que poderá utilizar para fins semelhantes.
A linha 210 e a ROTINA DE PARABÉNS, na linha 900, executam-se se (e quando) acertar no alvo, isto é, quando for capaz de fazer coincidir o quadrado preto com o seu quadrado axadrezado, o que significa que as suas coordenadas são as mesmas.
Quando estiver a jogar este jogo lembre-se de que **INKEY$** verifica o teclado para «ver» se há alguma tecla premida (e qual), no momento em que passar por esse ponto (linha) do programa. Assim, não se limite a tocar na tecla, mas mantenha premida aquela que diz respeito ao sentido e direcção para onde deseja mover-se para ficar mais perto do quadrado negro.
A tecla A desloca o quadrado para baixo e a Q para cima.
O desloca-o para a esquerda e P para a direita. (Note que lhe será possível designar quaisquer outras 4 teclas, à sua escolha, para cumprirem esta missão, se se sentir mais à vontade com elas. É nas linhas 170 a 200 que se definem as teclas a usar. Pode, igualmente, escolher as teclas das setas ou qualquer outro conjunto).





Nunca deixe duas teclas premidas, ao mesmo tempo, (por exemplo para baixo e à esquerda) porque **INKEY$** só permite que se leia uma delas de cada vez.
Dado que as letras definidas nas linhas 17Ø a 2ØØ são maiúsculas, antes de jogar deve colocar o computador no «modo» C (tecla de maiúsculas).
Eis um pequeno programa que lhe mostra que **INKEY$** não espera por ninguém. Se não tocar em nenhuma tecla, o computador imprimirá um espaço em branco (linha 1Ø) e, como a linha 2Ø o remete novamente para a linha 1Ø, torna a imprimir outro espaço em branco. O computador permanece assim, preenchendo o ecrã de espaços brancos. Experimente tocar uma ou outra tecla, leve e rapidamente, e verá surgir o símbolo respectivo no movimento ascendente que o computador está a fazer no ecrã. Se premir uma tecla por mais tempo o seu caracter imprime-se enquanto tiver a tecla premida. Para parar basta que faça **BREAK**.

```
1Ø PRINT INKEY$
2Ø GOTO 1Ø
```

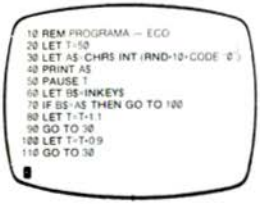


PAUSA
O comando **PAUSE** faz exactamente aquilo que poderia esperar dele: uma pausa. Mas essa pausa pode ser controlada por si, tornando-a maior ou menor de acordo com o número que indicar a seguir ao comando **PAUSE**.

PAUSE 5Ø corresponde a cerca de um segundo. Um número maior dará ocasião a uma pausa maior e, obviamente, um número menor, a uma pausa menor.
Acrescente, ao programa que acabou de introduzir há pouco, mais esta linha:

```
15 PAUSE 6Ø
```

e notará que será muito mais fácil controlar o «scrolling».
Um comando **PAUSE** termina automaticamente quando se premir qualquer tecla. Repare que o programa movimentar-se-á com a mesma rapidez com que premiu as teclas.



Imite uma máquina de escrever, pela simples adição de um ponto e vírgula no fim da linha 10.

Tem aqui um outro programa, verdadeiramente diabólico, que altera o tempo da **PAUSE** de acordo com o sucesso que alcance no jogo. Quanto a este, é muito simples:

O computador selecciona um número ao acaso (entre zero e nove). Quando o computador imprimir esse número, deve responder-lhe premindo imediatamente a tecla respectiva, antes que o computador tenha tempo para imprimir outro número qualquer.

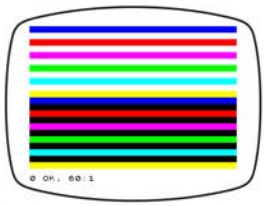
A situação torna-se intrigante porque se não acertar, o computador aumenta o tempo, concedendo-lhe uma maior **PAUSE**, mas, se acertar, o computador reduz a **PAUSE** e aumenta a velocidade de impressão dos números sorteados!

```
 10 REM PROGRAMA — ECO
 20 LET T=50
 30 LET A$=CHR$ INT (RND*10+CODE "0")
 40 PRINT A$
 50 PAUSE T
 60 LET B$=INKEY$
 70 IF B$=A$ THEN GO TO 100
 80 LET T=T*1.1
 90 GO TO 30
100 LET T=T*0.9
110 GO TO 30
```

INKEY$
Também é possível fazer movimentar (aparentemente, claro) as figuras que tenha definido através do UDG (GRÁFICO DEFINIDO PELO UTILIZADOR) alternando duas figuras em posições ligeiramente diferentes e que se deslocam no ecrã através de **INKEY$** ou de declarações programadas.

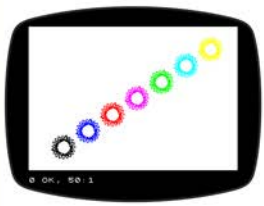
Sumário

1. **INKEY$** lê o teclado e faz o «input», na forma de «string», do caracter relativo à tecla premida. Se nenhuma tecla for premida, o **INKEY$** lê um espaço vazio que introduz na «string». Por isso o **PRINT INKEY$** actua só por si como se fosse apenas um **PRINT** (no caso de não haver teclas premidas) fazendo imprimir uma linha em branco.
2. **PAUSE** faz com que o computador espere, por determinado tempo, antes de continuar a execução do programa (50=1 segundo) na linha que se segue. Logo que uma tecla seja premida a **PAUSE** cessa.



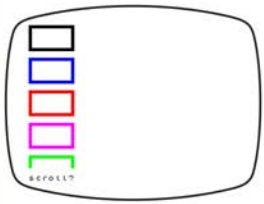
As cores disponíveis no TC 2048 podem observar-se através do seguinte programa:

```
1Ø FOR I = Ø TO 21:READ A
2Ø FOR J= Ø TO 31
3Ø PRINT INK A; AT I, J; "■"
4Ø NEXT J
5Ø NEXT I
6Ø DATA 1,7,2,7,3,7,4,7,5,7,6,1,Ø,2,Ø,3,Ø,4,Ø,5,Ø,6
```



```
1Ø REM PROGRAMA-ESTRELAS
2Ø PAPER 7:BRIGHT 1:BORDER Ø
3Ø FOR I = Ø TO 6: INK I
4Ø PLOT 3Ø + 3Ø*I,2Ø +2Ø*I: DRAW 2Ø,2Ø,5ØØ
5Ø NEXT I
```

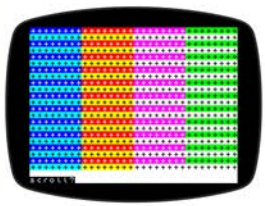
Experimente este programa com diferentes valores a substituir o 500 da linha 40.



Acrescente cor ao programa de GRÁFICOS da pág. 155.



```
 5 REM PROGRAMA — CÍRCULOS COLORIDOS
1Ø FOR I = 1 TO 5
2Ø FOR J = Ø TO 6
3Ø BORDER 1:INK J:CIRCLE 127,87, (5+1)*J
4Ø NEXT J
5Ø NEXT I
```

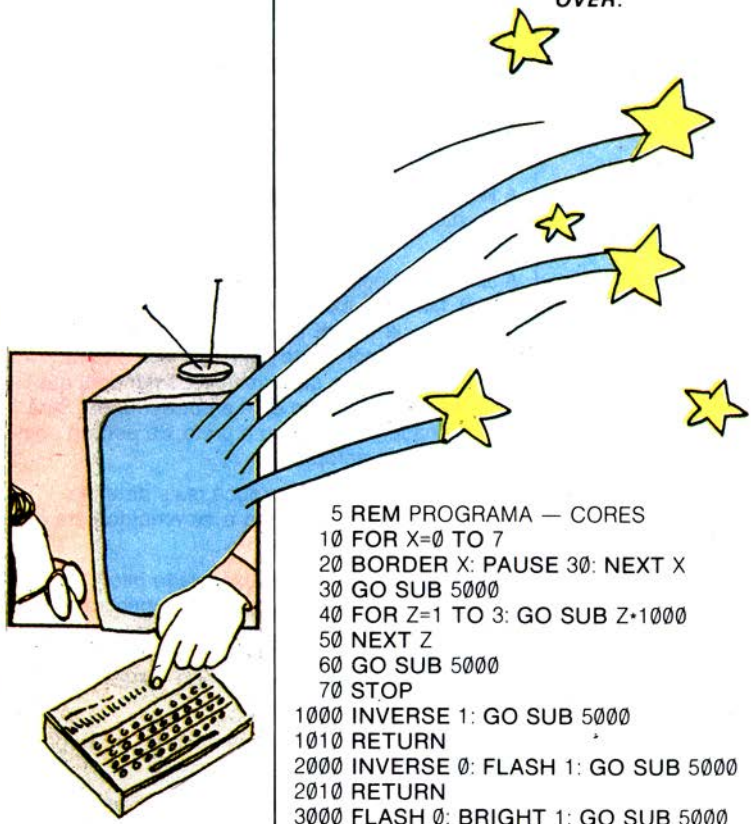


```
  5 REM PROGRAMA — TIRAS COLORIDAS
 1Ø FOR N = 0 TO 7
 2Ø BORDER N: PRINT PAPER N + 1; INK 9;
"++++++++";: PAUSE 3Ø
 3Ø NEXT N
4Ø GOTO 1Ø
```

A Cor 20

Síntese do Capítulo

Este capítulo retoma o estudo de BORDER, INK e PAPER e mostra como é possível realçar as cores através de BRIGHT, FLASH e INVERSE. Introduz, também o comando OVER.



```
  5 REM PROGRAMA — CORES
 10 FOR X=0 TO 7
 20 BORDER X: PAUSE 30: NEXT X
 30 GO SUB 5000
 40 FOR Z=1 TO 3: GO SUB Z*1000
 50 NEXT Z
 60 GO SUB 5000
 70 STOP
1000 INVERSE 1: GO SUB 5000
1010 RETURN
2000 INVERSE 0: FLASH 1: GO SUB 5000
2010 RETURN
3000 FLASH 0: BRIGHT 1: GO SUB 5000
3010 BRIGHT 0: RETURN
5000 FOR Y=0 TO 7: PAPER Y
5010 PRINT INK 9;"TIMEX COMPUTER 2048"
5020 PAUSE 30: NEXT Y
5030 PRINT
5040 RETURN
```

Já no Capítulo 3 introduzimos os comandos relacionados com a cor, **BORDER**, **PAPER** e **INK**. Faremos agora uma revisão dessa matéria e exploraremos algumas outras características do TC 2048 que na altura não foram discutidas.

Introduza o programa da página anterior e faça-o «correr» (**RUN**).

Façamos a sua análise:

Na linha 20 apenas se mudam as cores do **BORDER**, fazendo com que cada uma delas se mantenha por um pouco mais de meio segundo.

A linha 30 dirige o programa para a «subrotina» que se inicia na linha 5000.

A linha 5000 determina a cor do **PAPER** para as diferentes cores que são fixadas pelo ciclo («loop»).

A linha 5010 comanda a impressão do texto em **INK** 9. Mas não existe nenhuma cor indicada por cima da tecla 9! **INK** é uma instrução para escolher uma cor (tanto para **INK** como para **PAPER**, pois serve para os dois comandos) que dê o máximo contraste com a cor de fundo. Será seleccionado o branco ou o preto, de acordo com a que for mais apropriada.

A linha 5020 espera um pouco mais de meio segundo e depois comanda o movimento para o próximo número do ciclo.

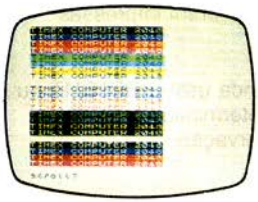
Depois de o programa ter executado oito impressões do texto, a linha 5030 imprimirá uma linha em branco, para separar este grupo das oito anteriores. A linha 5040 termina a «subrotina». Fazendo regressar (**RETURN**) o programa à linha 40.

A linha 40 / comanda a chamada a um grupo de «subrotinas». São elas:

Linha 1000 que coloca o computador em **INVERSE** 1 e seguidamente faz repetir a «subrotina» 5000.

Linha 2000 que anula o **INVERSE** e liga o **FLASH**, chamando depois a «subrotina» 5000.

Linha 3000 que desliga o **FLASH** e liga o **BRIGHT**. Note que a linha 3010 desliga o **BRIGHT** antes de terminar a «subrotina».





Após a iinha 5Ø ter completado o ciclo, a linha 6Ø envia novamente o programa para a «subrotina» 5ØØØ e a linha 7Ø pára o programa com **STOP**.

Quando «correr» o programa (**RUN**), as primeiras impressões que aparecem no ecrã ilustram o seguinte:

1. O **PAPER**, quando chamado num programa, actua unicamente nas áreas onde se fizer a impressão.
2. **INK** 9 funciona assim: para as primeiras quatro impressões é seleccionada, automaticamente, a cor branca. Para as últimas quatro, a tinta preta é a escolhida para imprimir sobre as cores mais claras do **PAPER**.

As seguintes oito impressões ilustram o comando **INVERSE**.
Dentro do computador as cores do **INK** e do **PAPER** mantêm-se as mesmas, mas os pontinhos de cada um dos elementos que formam o caracter (na grelha de 8 × 8) são invertidos. Os que eram **INK** passam para **PAPER** e vice-versa.

INVERSE 1 liga a função **INVERSE** e **INVERSE** Ø (veja a linha 2ØØØ) desliga-a.

O terceiro grupo das oito impressões seguintes, as primeiras quatro das quais surgem primeiro no ecrã, ilustram a função do **FLASH**. Como pode verificar, essa função não é mais do que uma rápida comutação fazendo alternar o **INVERSE** 1 com o **INVERSE** Ø.

FLASH 1 liga a função **FLASH**.
FLASH Ø desliga-a.

Toque na tecla Y, ou **ENTER**, ou qualquer outra que não **BREAK** ou N, quando surgir o "scroll?" e observe depois o resto dos trabalhos produzidos pelo programa.

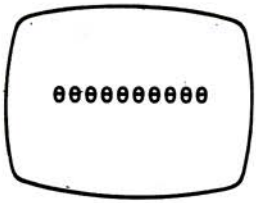
Primeiro surge o resto dos artigos impressos em **FLASH**.
As oito impressões seguintes ilustram a função **BRIGHT**, sendo possível que tenha de olhar com mais atenção para ver o efeito. (Repare

sobretudo nas letas brancas impressas sobre o fundo, para ver melhor a diferença).
BRIGHT 1 liga a função **BRIGHT** e **BRIGHT** Ø desliga-a.

As últimas oito impressões repetem novamente as oito primeiras quej á tínhamos visto, para que as possa comparar com as que foram impressas com o mando **BRIGHT**.

É muito possível que pretenda usar o **BRIGHT** ou o **FLASH** para destacar determinado texto do ecrã — um letreiro ou observação.

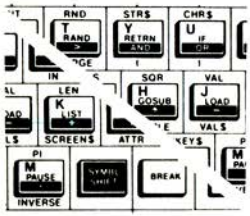
OVER
Uma outra função que pode ser ligada com 1 e desligada com Ø é **OVER**. Quando o **OVER** está ligado, pode imprimir um carácter juntamente com outra (impressão sobreposta). Introduza o seguinte programa:




```
1Ø PRINT AT 5,5;"OOOOOOOOOO"
2Ø OVER 1: PAUSE 3Ø
3Ø PRINT AT 5,5;"----------"
4Ø OVER Ø:PAUSE 3Ø
5Ø PRINT AT 5,5;"**********"
```

Na linha 1Ø utilize a letra O maiúscula e não o numeral zero (Ø).
Na linha 3Ø utilize o traço de união (**SYMBOL SHIFT** e J).

A linha 1Ø faz imprimir os caracteres «OO...» e a linha 2Ø liga o comando **OVER**, após o que faz uma pausa de pouco mais de meio segundo. A linha 3Ø imprime o traço-de-união sobre os «OO...», resultando daí a oitava letra do alfabeto grego (chamada «theta») repetida dez vezes.



A linha4Ø desliga o **OVER** e produz nova **PAUSE**. Quando a linha 5Ø comandar a impressão dos asteriscos já não há sobreposição e são estes que surgem em vez da impressão anterior. Imprimindo um número de caracteres suficiente, em sobreposição (**OVER** 1), acabaria por originar um quadrado preto.



Também poderá utilizar o **OVER** conjuntamente com o **CHR$** 8, que é um



retrocesso, para originar caracteres compostos. Faça um O com um «umlaut», da seguinte forma:

```
10 OVER 1: PRINT "o"; CHR$ 8; """"
```

(Quanto às aspas lembre-se que tem de introduzir as de abertura, mais duas aspas para imprimir uma só e, depois, fecha como abriu.)
Voltaremos novamente ao assunto dos **CHR$**, no Capítulo 22. Por agora, se der uma espreitadela ao Apêndice B, dar-se-á conta de que o **CHR$** 8 (ou código #8) é definido como retrocesso «cursor left» (cursor/esquerda). Pode chamar, para qualquer acção ou caracter, seja que elemento for do conjunto de caracteres do TC 2048, utilizando **CHR$**. Em vez de comandar **PRINT** "$" pode perfeitamente comandar **PRINT CHR$** 36).

Alguns Apontamentos Sobre BORDER, PAPER e INK

O **BORDER** tanto pode ser especificado como um comando, através de uma linha de programação, como numa linha em «modo» imediato. A cor seleccionada pelo **BORDER** permanecerá até que se seleccione nova cor ou que se desligue o computador.
Quando, numa linha de comando, se especifica **PAPER** de uma determinada cor, todo o centro do ecrã (**PAPER**) muda para essa cor, mas apenas depois de premir (duas vezes seguidas) a tecla **ENTER**. A cor permanecerá atá que seja alterada, — como no **BORDER**.)
Numa linha de programação o **PAPER** só produz efeito após o primeiro **PRINT** e cobre apenas a área relativa a essa impressão. Experimente:

```
10 PAPER 2
20 PRINT INK 7; "VEJA ISTO"
```

Depois de premir **RUN** e **ENTER**, observe que a cor do **PAPER** encarnado só aparece sob as letras brancas.
A cor do **PAPER** só se torna extensiva a toda a zona central do ecrã apenas após um comando **CLS**... ou depois de um **ENTER** para chamar a listagem do programa ao ecrã (porque isso, de

facto, limpa o ecrã antes de imprimir tal listagem).

Faça **ENTER** outra vez. Daqui faça

PAPER 7 ENTER ENTER

e cá estamos novamente em «normal».
Acrescente ao programa a seguinte linha:

15 CLS

e veja agora o que acontece quando «correr» (**RUN**) o programa.

INK como linha de comando, altera a cor de impressão. Mas não poderá aperceber-se dessa alteração até que imprima (**PRINT**) qualquer coisa. Experimente (depois de **NEW** e **ENTER**)

INK 2 ENTER

e seguidamente

PRINT "VEJA ISTO"

Numa linha de programação, o **INK** também afectará a cor de impressão, seleccionado a que tiver escolhido e mantendo-a até que seja alterada por uma linha de programação subsequente, ou que se desligue o computador voltará a imprimir na cor previamente determinada ou na falta de especificação, a preto e branco.
Recorda-se de como a **INK** se modificou para preto quanto premiu duas vezes **ENTER** e colocou a cor encarnada em toda a zona central do ecrã? Isso aconteceu porque se havia especificado **INK** 7 na declaração **PRINT** (linha 20) do pequeno programa que utilizámos.

SUMÁRIO

1. **INVERSE** inverte os pontos de impressão de **INK** e de **PAPER** permitindo a impressão de caracteres invertidos.

   INVERSE 1 liga-o.
   INVERSE Ø desliga-o.

2. **FLASH** faz com que o caracter comece a

   «piscar» originando uma rápida comutação entre true e inverse video.

   FLASH 1 liga-o.
   FLAHSH Ø desliga-o.

3. **BRIGHT** torna os caracteres mais brilhantes, no ecrã.

   BRIGHT 1 liga-o.
   BRIGHT Ø desliga-o.

4. **OVER** sobrepõe uma impressão de caracteres sobre quaisquer outros que já estejam impressos na mesma posição, não os apagando.

   OVER 1 liga-o.
   OVERØ desliga-o.

Como Obter Informações Especiais do TC 2048 21

Síntese do Capítulo

Este capítulo desenvolve alguns dos meios que permitem obter informações, as mais diversas, se se fizer as perguntas adequadas. Trata-se das funções POINT, ATTR, CODE e CHR$.

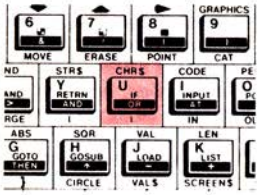


CHR$ e CODE

Já atrás falámos acerca do alfabeto alongado do TC 2048 no qual todas as letras e números (e até mesmo as palavras de comandos) estão incorporados numa listagem de 256 caracteres.

Cada caracter ou «comando» possui um código próprio (um número situado entre zero e 255), como pode verificar-se no Apêndice B, que os lista integralmente. Por outro lado, a cada um destes códigos numéricos corresponde um caracter, a que se deu o nome de **CHR$** ou «character string» (uma «string» extremamente pequena...).

Capítulo 21: Como Obter Informações Especiais do TC 2048





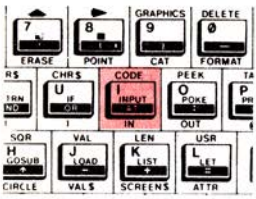
CHR$** (que se obtém através da função **CHR$ localizada acima da tecla U aplicado a um número situado entre zero e 255 dá-nos a «string» simples de um caracter correspondente ao código desse mesmo número. Eis um programa que permite localizar o caracter quando se lhe conhece o respectivo código:

```
1Ø REM PROGRAMA — CARACTERES
2Ø PRINT "ESCREVA UM NÚMERO DE 32 ATÉ 255"
3Ø PRINT
4Ø INPUT A
5Ø PRINT CHR$ A;
6Ø GO TO 4Ø
```

Indicámos 32 a 255, na linha 2Ø, dado que muitos dos caracteres abaixo de 32 correspondem a indicações para as quais o TC 2048 não possui nenhum sinal gráfico que possa imprimir no ecrã. (Veja-se o Apêndice do Conjunto de Caracteres). Além disso, alguns dos códigos referidos aos comandos das cores fariam parar o programa se os incluíssimos.

De cada vez que der entrada a um número entre 32 e 255, imediatamente o computador fará imprimir, no ecrã, o caracter correspondente a esse código numérico. Em muitos casos, todavia, o computador responde com um ponto de interrogação porque não dispõe de outro símbolo adequado para reproduzir melhor tal código. No exemplo «teclámos» os seguintes números: 5Ø, 1ØØ, 15Ø, 2ØØ e 25Ø.

Repare-se que o ponto e vírgula, no fim da linha 5Ø, faz com que os caracteres fiquem colados uns aos outros, no ecrã. Poderá substituir-se por uma vírgula, para obter os caracteres separados de meio ecrã, ou por simples anulação, o que daria cada caracter numa linha.



Aqui está um outro programa que funciona de forma inversa. Dê entrada do caracter através do teclado e receba a informação de qual é o seu código. A função **CODE** da linha 5Ø obtém-se através da tecla I.

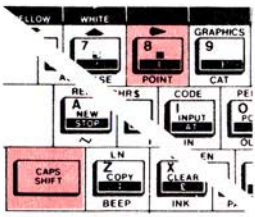
```
10 REM PROGRAMA — CÓDIGOS
20 PRINT "PRIMA QUALQUER TECLA"
30 PRINT
40 INPUT A$
50 PRINT CODE A$
60 GO TO 40
```

No exemplo que aqui figuramos ao lado, obtivémos os códigos para as seguintes teclas que foram premidas:

Y	y minúsculo
CAPS SHIFT Y	Y maiúsculo
SYMBOL SHIFT Y	AND
GRAPHICS Y	«Para utilização em gráficos a definir por si mesmo (UDG)»
Modo Extensivo, Y	STR$
Modo Extensivo, SHIFT Y	Chaveta esquerda

```
PRIMA QUALQUER TECLA

121
89
198
166
195
91
```



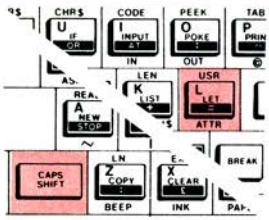
POINT

A função **POINT** (localizada abaixo da tecla 8) pode obter-se com **SHIFT** e 9 no «modo» extensivo. Deve ser seguida de dois números entre parênteses e separados por uma vírgula. Os números corresponderão à posição de **PLOT** do ponto em questão.

PRINT POINT (255,175)

A resposta do TC 2048 será

1, no caso do ponto especificado ter a cor de **INK**
ou
0 se esse mesmo «pixel» tiver a cor do **PAPER**.



ATTR

A função ATTRibuto, localizada abaixo da tecla L (e conseguida com SHIFT e L em «modo» extensivo), corresponde um número que codifica os atributos da posição especificada.

PRINT ATTR (15,1Ø)

Em binário, o «bit» 7 é 1 se estiver «a piscar» e Ø se não estiver. O «bit» 6 será 1 se a posição corresponder a **BRIGHT** e Ø (zero) se estiver normal. Os «bits» 3 a 5 definem a posição do **PAPER** relativamente à sua cor, tal como os «bits» Ø a 2 definem a cor da tinta de impressão (**INK**).
Convertidos para decimal poderemos descodificar o número da seguinte forma:

1. Se ele contiver (se for maior ou igual a) 128, a posição está a «piscar». Se o número for menor que 128 não estará.
2. Se lhe subtrair (sendo possível) 128 e o número resultante contiver 64, então a posição será brilhante (**BRIGHT** e 1). Se assim não acontecer a posição será normal.
3. Se ainda for possível, subtráia-lhe 64. **INK** e **PAPER** podem determinar-se em relação às suas cores, através da tabela abaixo. (Por exemplo, se o resto for zero, tanto o **PAPER** como a **INK** serão pretas; se for 21 o **PAPER** tem de ser vermelho e a **INK** de cor turqueza (cyan), (pois nenhuma outra combinação dará esse número).

COR	INK	PAPER
PRETO	Ø	Ø
AZUL	1	8
ENCARNADO	2	16
MAGENTA	3	24
VERDE	4	32
TURQUEZA (CYAN)	5	4Ø
AMARELO	6	48
BRANCO	7	56

SUMÁRIO

1. CHR$ aplica-se a um número e devolve o caracter que representa o seu respectivo código numérico.

   PRINT CHR$ 22Ø

2. **CODE** aplica-se a um caracter e restitui-lhe o respectivo código.

   PRINT CODE "Z"
3. **POINT** informa-o se o ponto (ou «pixel») especificado pelas respectivas coordenadas tem a cor **PAPER** ou **INK**. Será 1 se a posição tiver cor **INK** ou será Ø se a posição tiver a cor **PAPER**.

   PRINT POINT (255,175)

4. **ATTR** responde com um número decimal (entre Ø e 255) que poderá ser seccionado para revelar a cor do **PAPER** e do **INK** (para a posição de impressão especificada) a ainda se tal impressão tem brilho e/ou esta a «piscar».

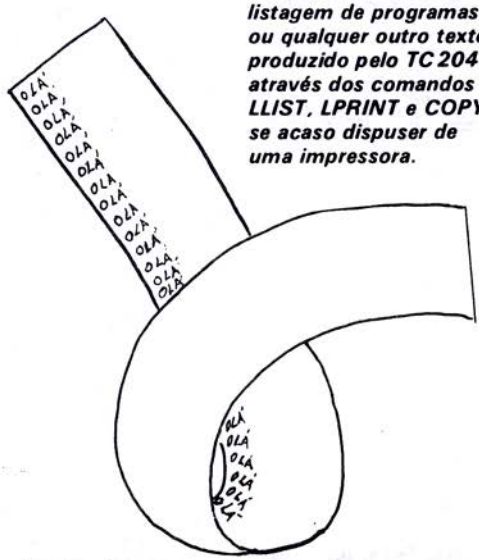
   PRINT ATTR (15,10)

Como Usar a Impressora 22



Síntese do Capítulo

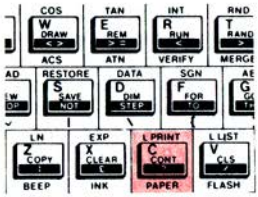
Este capítulo mostra-lhe como poderá obter a listagem de programas ou qualquer outro texto produzido pelo TC 2048 através dos comandos LLIST, LPRINT e COPY, se acaso dispuser de uma impressora.



```
10 REM PROGRAMA - IMPRESSORA
20 LPRINT "ESTE PROGRAMA",,,,
30 LLIST
40 LPRINT
50 LPRINT "ESCREVE O CONJUNTO DE
CARACTERES.",,,,
60 FOR N = 32 TO 255
70 LPRINT CHR$ N;
80 NEXT N
```

Poderá obter cópias dos seus programas (listagens) e também os seus resultados, em textos ou em gráficos produzidos pelo computador directamente sobre papel, ligando a impressora Timex Printer 2040 ao seu TC 2048.
Este tipo de material impresso denomina-se «hard copy» dado que se mantém, mesmo depois do computador se ter desligado, o que não acontece com o texto que vemos no ecrã.

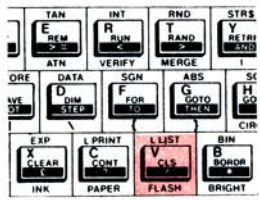
A impressora Timex 2040 é um dispositivo extremamente económico que se liga directamente ao computador TC 2048 (ao «port» colocado na parte de trás) e que se opera apenas com três comandos.



LPRINT

LPRINT (tecla C em «modo» extensivo) obtém-se com o cursor **L** e corresponde à função **PRINT**, só que o material a imprimir é enviado para a impressora e não para o ecrã. O "L" significa «Line Printer» que corresponde, afina, ao que é a impressora Timex 2040 — imprime uma linha completa de cada vez —, ainda que o comando se use agora para qualquer tipo de impressora.

Quando se inventou a linguagem BASIC, o dispositivo normal de impressão era uma espécie de máquina de escrever eléctrica e não um monitor de video ou, como acontece hoje, um simples televisor, pelo que **PRINT** significa efectivamente imprimir. Correspondendo ao desejo de mais trabalho em menos tempo uma impressora do tipo Line Pinter (uma linha de cada vez) é muito mais eficaz do que a máquina de escrever (que imprime caracter a caracter).

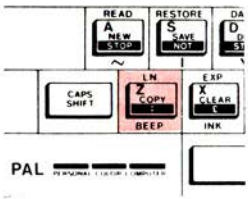


LLIST

Da mesma forma, LLIST (premir a tecla V com o cursor **L** fará listar o programa que estiver na memória do computador, directamente através da impressora e não no ecrã. Com **LLIST** obtém uma listagem sem ter sido preciso indicar a linha de programação a seguir ao comando. Por outras palavras, se deseja uma listagem do programa que em determinado momento se encontra na memória do TC2048, limite-se a premir **LLIST** e **ENTER** e obtê-la-á. Se incluir uma linha de programação a seguir ao **LLIST** (como por hipótese **LLIST 9Ø**), a listagem do programa far-se-á apenas a partir dessa linha.

LLIST poderá, também ser usado dentro de uma programação, (como no exemplo que aqui inserimos). É agora um bom momento para experimentar, se é que ainda não o fez.
A propósito, o programa que inserimos não imprime, através da impressora, o conjunto de caracteres completo, estritamente definido, (que vai de zero a 255), porque não é possível imprimir os comandos das cores (códigos entre Ø e 31).

(LLPRINT pode, obviamente, utilizar-se dentro de uma linha de programação — LLPRINT qualquer coisa — mas é mais normal servir-se dele fora da programação para obter cópias e poder examiná-las mais à vontade).



COPY
O terceiro comando da impressora é **COPY** (na tecla Z). Este comando origina simplesmente uma cópia de tudo (texto ou gráficos) que nesse momento estiver no ecrã. Pode utilizar o comando **COPY** quer numa linha de programação que no «modo» directo, como comando imediato, em qualquer momento que deseje obter um cópia do que se encontra no ecrã.

```
10 REM PROGRAMA — GRAFICO
20 PRINT "VENDAS 1 TRIM"
30 PRINT
40 PRINT "JANEIRO";TAB 9;"FEVEREIRO";
TAB 20;"MARÇO"
50 LET X = 10: LET N = 3
60 FOR A = 15 TO X STEP — 1
70 PRINT AT A,N;"■"
80 NEXT A
90 LET X=X—3: LET N=N + 10
100 IN N>23 THEN GOTO 120
110 GOTO 60
120 COPY
130 STOP
```



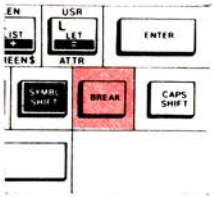


Introduza e experimente o programa inserido abaixo, que fará imprimir imaginários gráficos de vendas:

Perguntas:
Conseguirá obter o mesmo efeito se eliminar a linha 120 e depois premir as teclas **COPY** e **ENTER** após o programa ter parado?
É capaz de determinar o que cada linha do programa fez?

Sugestão: É preciso entrar no «modo» gratico e utilizar SYMBOL SHIFT e 8 para obter o quadrado negro.

Conseguirá premindo **LLIST** e **ENTER** o mesmo efeito que se obtém ao listar o programa no ecrã e depois premir as teclas **COPY** e **ENTER**? Poderá copiar-se a listagem de um programa com **COPY**?



Poderá copiar o gráfico de barras do programa com que trabalhou no Capítulo 17?

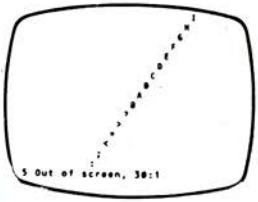
Parar a Impressora com BREAK

Quando a Impressora está a funcionar, pode pará-la com **BREAK**.
Pode desejar poupar papel se, por exemplo, estiver a utilizar **COPY** para imprimir as cinco linhas do programa listado acima. O comando fará «correr» 24 linhas, desperdiçando papel, a não ser que prima **BREAK** quando a linha 130 aparecer.



10 **REM** PROGRAMA — LETRAS
20 **FOR** N=31 **TO** 0 **STEP** — 1
30 **PRINT AT** 31 — N,N; **CHR$** (**CODE**"O" + N);
40 **NEXT** N

Se executar qualquer dos três comandos para a impressora, sem esta estar ligada, o programa normalmente executará a linha seguinte sem que nada fique impresso. Outras vezes, contudo, o computador ficará «pendurado» e terá de premir **BREAK** para o libertar.



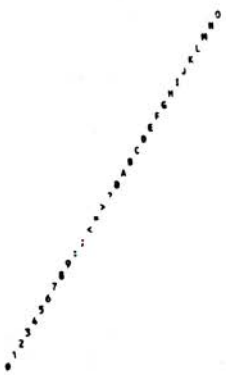
Formato da Impressão utilizando a Impressora

Todos os comandos para o formato da impressão, excepto um, funcionam com **LPRINT**.
A vírgula, o ponto e vírgula e **TAB** podem ser utilizados, mas **AT** não funcionará. Para ilustrar isto, experimente:
«Corra» (**RUN**) o programa. Verá uma lista de letras em diagonal atravessando o ecrã, até parar com a mensagem 5: **out of screen**.
Em seguida, mude **AT** 31-N, N na linha 30 para **TAB** N.
«Corra-o» e obterá o mesmo resultado, mas à paragem corresponderá "**scroll?**".

O.K. faça-o continuar.
Seja paciente e lá chegará.

```
10 REM PROGRAMA - LETRAS
20 FOR N=31 TO 0 STEP -1
30 LPRINT TAB N;CHR$ (CODE
0 + N);
40 NEXT N
```

Agora altere o **PRINT** da linha 30 para **LPRINT**. O padrão continuará por mais 10 linhas, depois de «teclar» **RUN**, porque a impressora não pode «encher-se» ao contrário do que acontece com cada ecrã. Não terá portanto qualquer mensagem ou indicação de **scroll**?.



```
10 REM PROGRAMA - LETRAS
20 FOR N=31 TO 0 STEP -1
30 LPRINT AT 21-N,N;CHR$ (CODE
0 + N);
40 NEXT N
```

Finalmente, altere o **TAB** N para **AT** 21-N,N. e volte novamente a «teclar» **RUN**.

Desta vez a impressora apresenta uma única fila de caracteres! Isto acontece apenas porque o comando **AT** não envia uma entrelinha (line feed) para a impressora. A posição de impressão movimenta-se uma coluna, mas não muda de linha.

A impressora imprimirá uma linha:

1. Depois de um comando **LPRINT** que não termine numa vírgula ou num ponto e vírgula.
2. Quando uma vírgula ou uma declaração **TAB** requer uma nova linha para iniciar o texto seguinte.
3. No fim de um programa, se ainda ficou alguma coisa para ser impressa.

4. Sempre que o »buffer» está cheio. Trata-se da área onde os caracteres a imprimir são armazenados até serem impressos. O «buffer» tem precisamente uma linha (32 caracteres) de comprimento, e, a não ser que um dos factos relatados acima ocorra primeiro, a impressora só imprimirá uma linha quando dispuser dela completa. Entre outras coisa necessita de desocupar o «buffer» sempre que deseje armazenar novos caracteres.

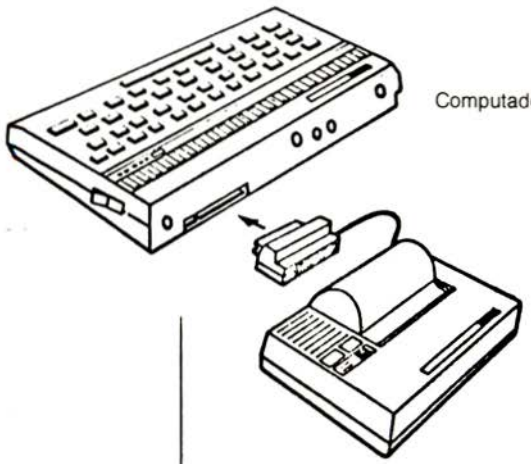
Lembra-se de como a impressora, no primeiro programa do Capítulo, parecia hesitar antes de imprimir cada uma das linhas do conjunto de caracteres? Estava a «encher» o BUFFER durante cada pausa.

SUMÁRIO

1. **LPRINT** imprime na impressora tal como **PRINT** imprime no ecrã.
2. **LLIST** envia a listagem de um programa para a impressora tal como **LIST** a envia para o ecrã.
3. **COPY** faz a duplicação para a impressora de tudo o que se encontra no ecrã.
4. **BREAK** pára a impressão ou interrompe os comandos da impressora quando surgem problemas.
5. **TAB**, Vírgula e Ponto e Vírgula podem ser utilizados para dar forma a instruções **LPRINT**.

Capítulo 22: Como Usar a Impressora

Diagrama do computador TC 2048
ligação à impressora



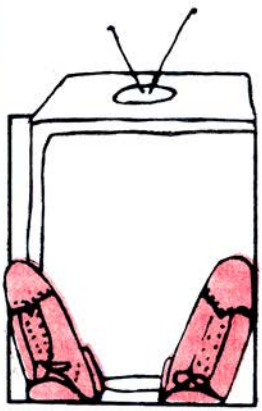
Computador TC 2048

Impressora

«Entradas» e «Saídas» (INPUT/OUTPUT) 23

Síntese do Capítulo

Este Capítulo convida-o a dar uma vista de olhos aos comandos que controlam o INPUT (ENTRADA) e o OUTPUT (SAÍDA) de elementos, incluindo PEEK, POKE, IN, OUT, OPEN, CLOSE, FORMAT, ERASE, CAT, MOVE.



Seja o que for que der entrada no computador é (como é lógico) denominado entrada (INPUT). E, naturalmente, que tudo que sai do computador chama-se saída (OUTPUT).

Através deste Manual observámos diversas formas de introduzir elementos no computador, quer através do teclado quer pelas «cassettes» de fita de gravação, assim como de produzir saídas através do ecrã do televisor, da impressora ou do gravador de «cassettes».

Vejamos, brevemente, outros aspectos das entradas e saídas (INPUT e OUTPUT).

PEEK e POKE

Quando introduz no computador

LET A=150

o computador toma esse número e armazena-o numa localização específica da memória. Não lhe interessa a si saber onde fica essa localização mas o TC 2048 pode localizá-la e fá-lo-á desde que lho ordene.

PRINT A

Poderá contudo trabalhar directamente com a memória do computador em localizações específicas, se utilizar os comandos **PEEK** e **POKE**. **POKE** coloca qualquer número (entre 0 e 255) em qualquer lugar da memória, situado entre 0 e 65535, e **PEEK** permite-lhe ver qual o número que está armazenado numa determinada localização. Por exemplo:

PRINT PEEK 20000
dar-lhe-á a resposta 0. Mas se a seguir comandar

POKE 20000, 150
e voltar a repetir o comando **PEEK**, verificará que, desta vez, o computador lhe responde com 150, dado que é esse valor que efectivamente está agora arquivado nesse local da memória.

Tem de ter muito cuidado quando usa o comando **POKE** (embora possa sempre utilizar **PEEK** sem qualquer perigo). Não fará qualquer dano ao computador, mas poderá destruir algum programa que nesse momento esteja em curso alterando um «byte» de informação num qualquer sítio do programa.

Aqui está um programa que lhe permite verificar os números armazenados em várias localizações da memória:

```
10 FOR I=23500 TO 24500 STEP 10
20 PRINT PEEK I,"";
30 NEXT I
```

O espaço na linha 20 é unicamente para dizer onde o número termina e o seguinte começa. Pode explorar o total de endereços a partir de 0 até 65535 fazendo modificações na linha 10.

Pode verificar o programa fazendo o **POKE**ing de um número num endereço (Para isto os melhores endereços são os situados acima de 24000) e estando incluídos nos limites dos endereços da linha 10.

Comece o programa com **GOTO** 10 em vez de **RUN** afim de ter a certeza que não apaga nada do que fez **POKE** na área de armazenamento de variáveis.

O que os números nos vários endereços de **ROM** significam para o computador tem a ver com o código máquina e o «sistema operativo» — o programa que controla o TC 2048 — é assunto para outra altura...

IN e OUT

IN e **OUT** são usados para ler e escrever «port de endereços» externo à memória do TC 2048 (isto inclui o teclado bem como os pontos para o presente e futuros periféricos).

PRINT IN 49150

Diz-lhe o que «chega» a partir do port de endereços.

OUT 49150, 150

«Escreverá» 150 no periférico ligado a esse port de endereços. Note que quando experimentar isto com o endereço 49150 o 150 não «entra». Neste caso, é porque não pode escrever para esse endereço: 49150 «contém» uma quantidade que diz ao computador que na fila do meio a partir de H até **ENTER** qualquer tecla está a ser pressionada.

Se conseguir ser suficientemente rápido nesta operação, pode até divertir-se. Premindo **ENTER** firmemente verá surgir 30, o que significa que a tecla **ENTER** está a ser premida. Se porém «picar» a tecla **ENTER** com golpes secos e rápidos, de forma a que não esteja de facto premida quando o TC 2048 recebe e executa o comando, então o número impresso será 31 (o que significa que nenhuma tecla da parte inferior do teclado está a ser premida). Se conseguir ser mais rápido e «picar» a tecla **ENTER** enquanto pressiona ao mesmo tempo,uma outra qualquer tecla da metade inferior do teclado, poderá observar o código dessa outra tecla!

(Se estiver interessado em fazer crescer o seu sistema Timex Computer 2048, adicionando-lhe «hardware» (equipamento) ou »software» (programação), contacte com a Timex-Portugal.

Comandos para os Periféricos Futuros

Um número de palavras referentes a «comandos» para utilizar com outros periféricos encontra-se já à disposição no teclado do Timex Computer 2048 Entre eles podemos destacar **IN** e **OUT** e alguns outros como **LOAD, SAVE, MERGE, VERIFY, INPUT** e **RESET** que também poderão ser utilizados com tais periféricos.

Alguns comandos a usar apenas com esses periféricos, denominados «file manipulation» (manipulação de ficheiros) destinam-se a equipamento de armazenamento de dados diferentes dos gravadores de «cassettes»:

FORMAT preparará um «disco» ou «diskette» ou outro meio de armazenamento de dados para trabalhar com o TC 2048.

OPEN abrirá um ficheiro para ser «lido» ou «escrito».

CLOSE, obviamente, fechará esse ficheiro; procedendo assim teremos a certeza de que outras informações sem interesse ficarão de fora

MOVE transferirá ou mudará o nome a um ficheiro.

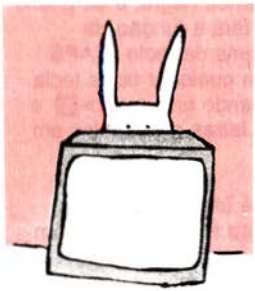
CAT — de «catálogo» — mostrará o menú, ou lista, dos ficheiros armazenados num dado periférico.

ERASE, como poderá supôr, fará desaparecer (apagando-o) qualquer ficheiro especificado.

O TIMEX COMPUTER 2048 pode aceitar até dois periféricos no seu terminal de ligações localizado na parte de trás do computador. Por hipótese, a impressora T/S 2040 pode ser ligada simultaneamente com o MODEM T/S 2050 que lhe permite fazer gravações permanentes de elementos (DATA) a que tenha acesso pelo serviço de telecomunicações.

O gravador de «cassettes», o televisor, o Monitor de Alta Resolução, o FDD e os «joysticks» podem estar ligados em qualquer momento, independentemente de haver ou não outros periféricos já ligados.

Apêndice A: Revisão do BASIC do TC 2048





O Teclado

O conjunto de caracteres do TC2048 compreende não só os símbolos simples (letras, dígitos, etc.), mas também os compostos («comando», nome de funções, etc.), que se introduzem em bloco e não letra a letra.

Assim, para que isto seja possível, cada tecla tem cinco ou mais significados diferentes. Estes conseguem-se ou premindo seja **CAPS SHIFT** seja **SYMBOL SHIFT** simultaneamente com a tecla requerida, ou colocando a máquina em «modos» diferentes.

O «modo» é indicado por um cursor, letra intermitente que nos indica onde será impresso o caracter seguinte.

«Modo» **K** (de «Keywords») substitui automaticamente o modo **L** quando o computador espera por um comando ou linha de um programa.

Isto acontece no princípio de uma linha, ou imediatamente a seguir a **THEN**. Se não se estiver a premir nenhuma das teclas **SHIFT**, à tecla premida corresponderá um «modo» (escrita na tecla) ou um dígito.

«Modo» **L** (de letra) aparece, habitualmente, em todos os outros casos. Sem premir qualquer das teclas **SHIFT**, obter-se-á o símbolo principal impresso na tecla, em minúsculas se se tratar de uma letra.

Quando, nos «modos» **K** ou **L**, premir **SYMBOL SHIFT** e uma tecla, obterá o caracter

impresso na tecla sobre banda negra, e se premir CAPS SHIFT e um dígito terá a função de controlo escrita a preto acima da tecla. CAPS SHIFT conjuntamente com qualquer outra tecla não afecta os «modos» quando em «modo» K e no «modo» L converte as letras minúsculas em maiúsculas.

«Modo» C (de «capitais») é uma variante do «modo» L obtendo-se então todas as letras em maiúsculas. CAPS LOCK provoca a passagem do «modo» L para o C e vice-versa.

«Modo» E (de extensivo) obtém-se premindo simultaneamente as teclas SHIFT e não se mantém depois de qualquer tecla ter sido pressionada. Neste «modo», quando premir uma tecla não numérica obterá o símbolo impresso acima da tecla e se premir simultaneamente qualquer das teclas SHIFT obterá o símbolo impresso debaixo da tecla.

Uma tecla numérica dar-lhe-á um símbolo se premida simultaneamente com SYMBOL SHIFT; senão obterá um código de cor.

«Modo» G (de gráficos) ocorre quando premir CAPS SHIFT e 9 (GRAPHICS) e mantém-se até que seja novamente «teclado». Premindo uma tecla numérica obterá um gráfico, abandonará o «modo» gráfico ou DELETE e com cada tecla não numérica, à excepção de V. W, X, Y e Z, obterá um gráfico definido pelo utilizador (UDG).

Se qualquer tecla for premida por mais de um segundo, repetir-se-á.

As instruções dadas a partir do teclado aparecerão na parte do ecrã, tal como estiverem a ser «tecladas».
O cursor pode mover-se para a esquerda ou direita com CAPS SHIFT e, respectivamente, 5 ou 8. O caracter impresso antes do cursor pode apagar-se com DELETE (CAPS SHIFT e Ø).

(Nota: Toda a linha pode ser apagada premindo EDIT (CAPS SHIFT e 1) seguido de ENTER.

Quando premir a **ENTER**, a linha é executada e, ou se integra num programa ou é utilizada como **INPUT** de dados, a não ser que contenha um erro de síntaxe. Neste caso um **?** intermitente aparecerá junto ao erro.

À medida que as linhas de um programa vão sendo introduzidas, uma listagem é apresentada no topo superior do ecrã. O modo como tal listagem se produz é complicado, mas, no Capítulo 2, pode inteirar-se de mais pormenores.
A última linha produzida é a linha corrente e vem indicada pelo símbolo >, mas isto pode ser alterado utilizando as teclas ↓ (**CAPS SHIFT** e 6) e ↑ (**CAPS SHIFT** e 7). Se **EDIT** (**CAPS SHIFT** e 1) for premido, a linha corrente aparecerá na parte inferior do ecrã e pode ser modificada.

Quando um comando é executado ou um programa «corre», as saídas são apresentadas no topo superior do ecrã e aí permanecerão até que nova linha seja introduzida, ou se prima **ENTER**, ↓ ou ↑. Na parte inferior do ecrã aparece um código de relatório cujo significado encontrará no Apêndice H. Este relatório contém o número de linha da última instrução executada (ou 0 se for um comando) e a posição da instrução dentro da linha. O relatório manter-se-á no ecrã até que uma tecla seja premida (indicando o «modo» **K**).

Por vezes, **CAPS SHIFT** e **BREAK** actuam como **BREAK**, parando o computador com relatório D ou L. Isto acontece:

1. no fim de uma instrução, quando um programa está a «correr» ou

2. quando o computador está ligado ao gravador ou impressora.

O ecrã da televisão
Tem 24 linhas, cada uma tem 32 caracteres e está dividido em duas partes. A parte superior tem no máximo 22 linhas e apresenta-nos quer a listagem de um programa quer as saídas deste.

Quando esta parte estiver completamente preenchida, a continuação da impressão implicará a movimentação de todo o conjunto, uma linha a cima. Como isto envolve a «perda» da primeira linha, sem que tenha tido oportunidade de a ver, o computador parará com a mensagem **scroll?**. Premindo N, **BREAK** ou **STOP** fará parar o programa com relatório **D BREAK-CONT repeats**; qualquer outra tecla fará com que o «movimento ascendente» (**SCROLL**) continue. A parte inferior é utilizada para introdução de comandos, linhas de programas, e entrada de dados e ainda exposição de relatórios.
Esta parte começa com 2 linhas mas pode ser expandida a fim de acomodar o que quer que seja «teclado».

A cada posição de caracter correspondem atributos que especificam a cor (**INK** e **PAPER**), o nível de brilho, e se está intermitente (**FLASH**) ou não.

As cores possíveis são preto, azul, vermelho, magenta, verde, turquesa, amarelo e branco.

Os bordos ou margens do ecrã podem ter qualquer destas cores, usando a instrução **BORDER**.

Uma posição de caracter está dividida em 8 × 8 «PIXELS». Assim podem obter-se gráficos de alta resolução se se atribuir a cada «pixel» uma cor jogando com **INK** ou **PAPER**.

Os atributos de posição de um caracter são ajustados sempre que um caracter ou «pixel» aí for impresso. Tais ajustes conseguem-se através dos seguintes parâmetros (uns permanentes, outros temporários): **PAPER**, **INK**, **FLASH**, **BRIGHT**, **INVERSE** e **OVER**. Os parâmetros permanentes, no topo superior são estabelecidos por **PAPER**, **INK**, etc., e mantêm-se até que nova ordem seja dada. (Inicialmente tem-se tinta preta, papel branco, brilho normal, sem piscar, vídeo normal e sem impressão sobreposta). Na parte inferior os parâmetros permanentes tomam para **PAPER** a cor do **BORDER** e par **INK** uma cor

preto ou branco, contraste, brilho normal, sem piscar, vídeo normal e sem impressão sobreposta.

Os parâmetros temporários são estabelecidos por **PAPER INK**, etc., incluidos nas instruções **PRINT, LPRINT, INPUT, PLOT, DRAW** e **CIRCLE** e também por caracteres de controlo **PAPER, INK**, etc., que, quando escritos no ecrã, são seguidos por mais um «byte» para especificar o valor do parâmetro.

Os parâmetros temporários mantêm-se durante a execução da instrução (**PRINT, INPUT**, etc.) e serão depois substituídos pelos permanentes.

PAPER e **INK** podem tomar valores entre Ø e 9. De Ø a 7 são os valores com que o caracter será impresso:

Ø PRETO
1 AZUL
2 ENCARNADO
3 MAGENTA
4 VERDE
5 TURQUEZA
6 AMARELO
7 BRANCO

O valor 8 (transparente) especifica que a cor no ecrã deverá manter-se quando um caracter for impresso.

O valor 9 (contraste) especifica que a cor do **PAPER** ou **INK** (consoante a que se estiver a utilizar) deverá ser branco ou preto para se obter um maior contraste em relação à outra cor.

FLASH (piscar) e **BRIGHT** (brilho) podem tomar os valores Ø, 1 e 8 com os seguintes significados: Ø desligados, 1 activados e 8 inalterados.

OVER e **INVERSE** tomam os valores Ø ou 1:
OVER Ø — caracteres novos substituem os anteriores;
OVER 1 — o novo caracter é combinado com o anterior usando uma operação de «ou exclusivo» (impressão sobreposta);

INVERSE Ø — caracteres impressos na cor de **INK** sobre cor do **PAPER** (Vídeo normal);

INVERSE 1 — caracteres impressos na cor do **PAPER** sobre cor de **INK** (Vídeo inverso).

Depois de receber um caracter de controlo **TAB**, esperam-se mais dois «bytes» para especificar o ponto n de paragem (primeiro o «byte» menos significativo). Tal ponto é reduzido a módulo 32 (divide-se por 32 e utiliza-se somente o resto) para n_0 e então a impressão será feita na coluna n_0 deixando em branco o número de espaços suficientes.

Quando o caracter de controlo — vírgula — é recebido, são deixados espaços em branco em número suficiente para que a impressão seja feita na coluna Ø ou na coluna 16.

Quando se tratar de um outro caracter de controlo — apóstrofo ou **ENTER** —,a impressão é feita na linha seguinte.

A Impressora

A saída para a Impressora é feita via «buffer» (tampão) em linha de 32 caracteres de comprimento. Uma linha é enviada para a impressora:

1. Quando a impressão passa de uma linha para outra;
2. Quando se recebe um caracter **ENTER**;
3. Quando o programa termina, havendo ainda algo por imprimir;
4. Quando um controlo **TAB** ou uma vírgula fazem mudar a posição da impressão para uma nova linha.

Um controlo **TAB** ou vírgula, deixa espaços em branco tal como acontece no ecrã.
Um controlo **AT** muda a posição de impressão para a coluna indicada ignorando o número de linha.
A impressora, tal como o ecrã, considera o controlo **INVERSE** e **OVER**, mas ignora **PAPER**, **INK**, **FLASH** ou **BRIGHT**.

A impressora pára com erro B, se se premir **BREAK**.
Se a Impressora não estiver ligada à saída do computador, a mensagem será simplesmente perdida.

O BASIC
Os números são armazenados com uma exactidão de 9 ou 10 dígitos. O maior número que se pode obter é 10^{38} e o menor (positivo) $4*10^{-39}$.

Um número é armazenado no TC 2048 em binário de ponto flutuante com um «byte» para o expoente 'e' ($1 <= e <= 255$), e quatro «bytes» para a mantissa 'm' ($1/2 <= m < 1$). Assim o número será representado por $m*2^{e-128}$.

Uma vez que $1/2 <= m < 1$, o «bit» mais significativo da mantissa 'm' será sempre 1. Então poderemos substituí-lo por outro que designe o sinal — 0 para números positivos, 1 para os negativos.

Inteiros pequenos têm uma representação especial, na qual o primeiro «byte» é 0, o segundo é um «byte» de sinal (0 ou FFh) e o terceiro e quarto são os inteiros na forma complemento de dois, sendo o «byte» menos significativo o primeiro deles.

As variáveis numéricas têm nomes de comprimento arbitrário, começando por uma letra seguida de letras e dígitos. Controlo de espaços e cores são ignorados e todas as letras são convertidas para minúsculas.

Variáveis de controlo de ciclos **FOR/NEXT** têm nomes de uma só letra.

O nome das matrizes numéricas tem uma só letra que pode ser o mesmo de uma variável simples. Podem ter um número de dimensões arbitrário. Os índices começam em 1.
O comprimento de uma «string» (cadeia de caracteres) é perfeitamente flexível. O nome consiste numa só letra seguida de $. O

comprimento de uma matriz de «strings» é arbitrário assim como a sua dimensão. O nome é só uma letra seguida de $ e não pode ser o mesmo de uma «string».
Todas as «strings» de uma matriz têm um comprimento fixo, que é especificado por uma dimensão extra, no final duma instrução **DIM**. Os índices começam em 1.
«Corte»: subcadeias de caracteres podem ser especificadas por «elementos de corte». Um «elemento de corte» pode ser:

1. Vazio;
2. Uma expressão numérica;
3. Uma expressão numérica até outra (**TO**) expressão numérico;

e usa-se para exprimir uma subcadeia quer por
a) expressão alfanumérica («elemento de corte»), quer por
b) matriz alfanumérica (índice, ..., índice, «elemento de corte» que é o mesmo que matriz alfanumérica (índice, ..., índice) (corte).
Suponha que em (a) a expressão alfanumérica tem o valor de s$. Se o «elemento de corte» for vazio, s$ é considerado como uma subcadeia dela própria.

Se o «elemento de corte» for uma expressão numérica com o valor m, a subcadeia será formada por um só caracter, o m-ésimo caracter de s$.

Se o corte tiver a forma indicada em (3), suponha que a primeira expressão numérica tem o valor m (se faltar, o computador considera o valor 1) e a segunda expressão o valor n (por defeito, teríamos o comprimento de s$).

Se $1 <= m <= n <=$ comprimento de s$, então a «substring» de s$ começaria no caracter m e terminaria com o n-ésimo.

Se $0 <= n < m$, então teríamos uma «string» vazia.

Qualquer outra situação, resultaria em erro 3.
A operação de «corte» duma »string» em »subtrings» é efectuada antes de qualquer função ou operação, a não ser que existam parêntesis indicando outra forma de proceder.

As subcadeias podem assumir valores próprios (veja **LET**).
Para que uma cadeia inclua aspas, terá que as utilizar aos pares.

Funções
O argumento de uma função não necessita estar entre parêntesis se se trata de uma constante ou de uma variável (possivelmente subscrita ou subdividida).

Função	Tipo de argumento	Resultado
	(X)	
ABS	número	valor absoluto
ACS	número	Arco-coseno em radianos. Erro A se X não estiver entre −1 e +1.
AND	Operação binária, operando da direita é sempre um número.	
	Operando esquerdo, numérico:	A **AND** B= { A se B <> Ø ; Ø se B = Ø
	Operando esquerdo, cadeia:	A$ **AND** B= { A$ se B <> Ø ; " " se B = Ø
ASN	número	Arco-seno em radianos. Erro A se X não estiver entre −1 a +1.
ATN	número	Arco tangente em radianos.
ATTR	dois argumentos, x e y, ambos números entre parêntesis	Um número cuja forma binária codifica os atributos da linha X, coluna Y no ecrã. O «bit» 7 (o mais significativo) é 1 para **FLASH** e Ø para não-**FLASH**. O «bit» 6 é 1 para **BRIGHT** e Ø para brilho normal. Os «bits» 5 a 3 são a cor do **PAPER** e os «bits» 2 a Ø , a do **INK**. Erro B, a não ser que Ø <= X <= 23 e Ø <= Y = 31

Função	Tipo de argumento	Resultado
BIN		Não se trata bem de uma função, mas de uma notação alternativa para números: **BIN** seguido por uma sequência de zeros e uns é um número representado em binário.
CHR$	número	Caracter com código x, arredondado para o inteiro mais próximo.
CODE	cadeia	Código do primeiro caracter em x (Ø se x for uma «string» vazia).
COS	número (em radianos)	Coseno x
EXP	número	e^x
FN		FN seguido por uma letra chama uma função definida pelo utilizador (veja **DEF**). Os argumentos devem estar entre parênteses e mesmo que não haja argumento os parênteses devem aparecer.
IN	número	O resultado de uma entrada ao nível do processador no port x (Ø < = X < = FFFFh) carrega o par de registos bc com x e faz a instrução em linguagem «ASSEMBLER», **IN** a(c).
INKEY$	nenhum	Lê o teclado. O resultado é um caracter representativo (em «modo» **L** ou **C**) da tecla premida, se houver alguma, ou, então, uma «string» vazia.
INT	número	Parte inteira (sempre arredondada por defeito)
LEN	cadeia	Comprimento
LN	número	Logarítmo natural (base e) Erro A se x < = Ø
NOT	número	Ø se x < > Ø, 1 se x = Ø. NOT tem prioridade 4.

Função	Tipo de argumento	Resultado
OR	operação binária, ambos operadores numéricos	a OR b = { 1 se b <> Ø ; a se b = Ø } Or tem prioridade 2.
PEEK	número	Valor do «byte» guardado na memória, no endereço x, (arredondado para o inteiro mais próximo). Erro B se x não estiver entre Ø e 65535
PI	nenhum	π (3.14159265...)
POINT	Dois argumentos x e y, ambos numéricos e dentro de parênteses	1 se o «pixel» de (x, y) tiver a cor (**INK**). Ø se tiver a cor do **PAPER**. Erro B, a não ser que Ø < = x < = 255 e Ø< = y < = 175.
RND	nenhum	O próximo número pseudo-aleatório numa sequência gerada por potências de 75 módulo 65537, subraindo 1 e dividindo por 65536. Ø ⩽ Y < 1.
SCREEN$	Dois argumentos x e y, ambos numéricos e dentro de parênteses	Caracter que aparece, quer normal quer invertido, no ecrã na linha x, coluna y. Dá uma «string» vazia, se o caracter não for reconhecido. Erro B, a não ser que Ø < = x < = 23 e Ø < = y < = 31.
SGN	número	Sinal de x (−1 para negativos, Ø para zero ou +1 para positivos).
SIN	número (em radianos)	Seno de x
SQR	número	Raiz quadrada. Erro A se x < Ø

Função	Tipo de argumento	Resultado
STR$	número	A «string» que seria mostrada se x fosse introduzido.
TAN	número (em radianos)	Tangente.
USR	número	Chama a subrotina em código máquina cujo primeiro endereço é x. Como resultado dá-nos o conteúdo do par de «registos» bc.
USR	«string»	Endereço do «bit — padrão» para o caracter gráfico definido pelo utilizador correspondente a x. Erro A se x não for uma letra entre a e u ou um UDG.
VAL	«string»	Calcula x (sem aspas) como uma expressão numérica. Erro C se x contiver um erro de sintaxe ou resultar numa «string» de valores. Outros erros possíveis dependentes da expressão.
VAL$	«string»	Calcula x (sem aspas) como uma expressão literal. Erro C se x contiver um erro de sintaxe ou resultar dum valor numérico. Outros erros possíveis como em VAL.
–(MENOS)	número	Negação.

Função	Resultado
Seguem-se as operações Binárias:	
+	Adição (com números) ou concatenação (de cadeias)
–	Subtracção
*	Multiplicação
/	Divisão
↑	Potenciação. Error B se o operando esquerdo for negativo.
=	Igual
>	Maior que
<	Menor que
<=	Menor ou igual a
>=	Maior ou igual a
<>	Diferente de

Ambos os operandos têm de ser do mesmo tipo. O resultado é 1, se a comparação se verifica e Ø, se tal não acontece.

Funções e operações têm a seguinte prioridade:

Operação	Prioridade
Corte e sub-cadeias	12
Todas as funções excepto **NOT** e	
Menos «unitário»	11
↑	10
Menos («unitário») (negação de algo)	9
*,/	8
+, – (menos utilizado para subtrair dois números)	6
=, >, <, <=, >=, <>	5
NOT	4
AND	3
OR	2

Notações

Na listagem seguinte:

α	representa uma única letra
v	representa uma variável
x, y, z	representam expressões numéricas
m, n	representam expressões numéricas arredondadas para o inteiro mais próximo
e	representa uma expressão
f	representa o valor de uma expressão de «string»
s	representa uma sequência de instruções separadas por vírgulas

Função	Resultado
	c representa uma sequência de elementos de cor cada um dos quais terminado por vírgula ou por ponto e vírgula; um elemento de cor tem a forma de PAPER, INK, FLASH, BRIGHT, INVERSE ou OVER.
	Note que é permitido o uso de expressões arbitrárias (excepto como número de linha no início da instrução).
	Todas as instruções, excepto **INPUT**, **DEF** e **DATA**, podem ser utilizadas quer como comandos directos quer em programas (embora sejam mais indicados para um do que para o outro). Um comando ou linha de programa pode conter várias instruções separadas por dois pontos(:). Não há qualquer restrição sobre a posição de dada instrução dentro de uma linha, no entanto veja **IF** e **REM**.
BEEPx,y	Através do altifalante ouve-se uma nota musical por x segundos com uma frequência y de semi-tons acima do dó-central (ou abaixo se for negativo).
BORDER m	Atribui uma cor ao bordo do ecrã e também ao **PAPER** da parte inferior do ecrã. Erro K se m não se encontrar entre Ø e 7.
BRIGHT	Atribui brilho aos caracteres impressos a seguir. n = Ø — brilho normal; 1 — brilho acentuado; 8 — transparência. Erro K se n não for Ø,1 ou 8.
CAT	Utilizado com periféricos.
CIRCLE x,y,z	Desenha um arco de circunferência, centrado em (x,y) e de raio z.
CLEAR	Apaga todas as variáveis, libertando o espaço que elas ocupavam. Faz o **RESTORE** e **CLS**, repõe a posição de **PLOT** no canto inferior esquerdo e limpa o «stack» de **GOSUB**.
CLEAR n	Tal como **CLEAR**, modifica, se possível, o RAMTOP do sistema de variáveis para n, pondo lá de novo o «stack» de **GOSUB**.
CLOSE #	Para uso com periféricos.

Função	Resultado
CLS	(Clear Screen). Limpa o display file (ficheiro de imagem).
CONTINUE	Continua um programa, começando onde tinha parado pela última vez com mensagem diferente de Ø. Se a mensagem for 9 ou L, continua com a instrução seguinte (preservando os saltos), senão repete a instrução onde o erro ocorreu. Se a última mensagem de erro ocorreu num comando directo, então **CONTINUE** tentará continuá-lo e prosseguirá se o erro tiver sido Ø:1; dará mensagem de erro Ø se tiver ocorrido em Ø :2; ou dará erro N se tiver ocorrido em Ø :3 ou superior. **CONTINUE** aparece no teclado como **CONT**.
COPY	Quando ligado, envia para a impressora as 22 linhas do topo do ecrã. Caso contrário nada faz. Note que **COPY** não pode ser utilizado para imprimir automaticamente as listagens que aparecem no ecrã. Relatório D se premir **BREAK**.
DATA e_1, e_2, e_3,...	Parte da lista de dados. Usado em programas.
DEF FN $\alpha(\alpha_1,..., \alpha_k)=e$	Função definida pelo utilizador. Utilizada num programa. Cada α e α_1 a α_k, pode ser uma letra ou uma letra seguida por «$». Se não tiver argumento toma a forma **DEF FN**α ( ) = e.
DIM α $(n_1,...,n_k)$	Apaga qualquer matriz com nome α, e estabelece uma matriz numérica α com dimensão K $(n_1,...,n_k)$. Inicia todos os valores em Ø.
DIM α\$ $(n_1,...,n_k)$	Apaga qualquer matriz ou cadeia de caracteres cujo nome seja α\$, e estabelece uma matriz de caracteres com dimensões K. inicia todos os valores até " ". Isto pode ser considerado como uma matriz de caracteres de comprimento fixo n_k, com K-1 dimensões n_1,... n_{k-1}.

Função	Resultado
	Erro 4 ocorre se não houver espaço para a matriz. Uma matriz é definida até ser dimensionada por uma instrução **DIM**.
DRAW x,y	DRAW x,y,Ø.
DRAW x,y,z	Desenha uma linha a partir da posição **PLOT** correspondente, mudando x na horizontal e y na vertical relativamente a essa posição, varrendo um ângulo de valor z. Erro B se sair do ecrã.
ERASE	Para uso com periféricos.
FLASH	Define se os caracteres são intermitentes ou não. n=Ø para não intermitente, n = 1 para intermitente, n = 8 sem alteração.
FOR α = x **TO** y	FOR α= x TO y STEP 1
FOR α = x **TO** y **STEP** z	Apaga qualquer variável α e cria uma variável de controlo com o valor x, limite y, incremento z, e um endereço de ciclo que se refere à instrução depois de **FOR**. Verifica se o valor inicial é superior (se o STEP>=Ø) ou inferior (se o STEP<Ø) ao limite e nesse caso executa a instrução **NEXT** α, dando erro 1 se não houver nenhum. Veja **NEXT**. Erro 4 se não houver espaço para a variável de controlo.
FORMAT f	Para uso com periféricos.
FREE	Fornece o número de «bytes» disponíveis na RAM para programas em BASIC e variáveis.
GO SUB n	Empurra o número de linha da instrução **GOSUB** para um «stack»; de resto funciona como GO TO n. Erro 4 ocorre se não existirem os **RETURN**s correspondentes.
GO TO n	Salta para a linha n (ou se não existir nenhuma linha com esse número, para a primeira que se lhe segue).
IF x **THEN** s	Se x for verdadeiro (não-zero) então s é executado. Note que s compreende todas as instruções até ao fim da linha. A forma «IF x THEN número de linha» não é permitida.

Função	Resultado
INK n	Estabelece a cor da tinta dos caracteres impressos a seguir. n toma valores de Ø a 7 para cor, de 8 para transparência ou de 9 para contraste. Erro K se n não estiver entre Ø e 9.
INPUT...	O "..." é uma sequência de elementos de INPUT, separados, como na instrução PRINT, por vírgulas, ponto e vírgula ou apóstrofos. Um elemento de INPUT pode ser: 1. Qualquer elemento de PRINT que não comece por uma letra; 2. O nome de uma variável; 3. LINE, e depois o nome de uma variável de «string». Os elementos PRINT e separadores referidos em (1) são executados exactamente como numa instrução PRINT, a não ser que a impressão seja feita na parte inferior do ecrã. Na situação (2) o computador pára para admitir uma expressão a partir do teclado; o valor será atribuído à variável. A entrada da forma comum e erros de sintaxe são assinalados pelo intermitente. Para expressões de «string» o «input buffer» é iniciado de forma a conter duas aspas (apagáveis se necessário). Se o 1.º caracter de entrada for STOP, o programa pára com erro H. Quanto à situação (3), funciona como a (2), mas as entradas são tratadas como «strings» literais sem aspas e a instrução STOP não funciona, devendo premir para parar o programa.
INVERSE n	Controla a inversão de caracteres impressos a seguir. Se n=Ø, os caracteres serão impressos em vídeo normal, com a cor INK sobre a do PAPER. Se n = 1, os caracteres serão impressos em inverse vídeo, isto é, com a cor do PAPER sobre a cor INK. Erro K se n não for Ø ou 1.
LET v = e	Atribui o valor de 'e' à variável v. A palavra LET não pode ser omitida. Uma variável simples só fica definida quando atribuída por uma instrução LET, READ ou INPUT. Se v é uma «string» de variáveis indexadas, ou uma variável de «string» cortada («substring»), então a atribuição é procrusteana (comprimento fixo): o valor da «string» 'e' é truncado ou acrescentado com espaços vazios, de modo a ter o comprimento de v.

Função	Resultado
LIST	LIST Ø.
LIST n	Lista o programa a partir do topo superior do ecrã, começando na linha cujo número é maior ou igual a n e fazendo desta a linha corrente.
LLIST	LLIST Ø.
LLIST n	Como LIST, mas utilizando a impressora.
LOAD f	«Carrega» programa e variáveis.
LOAD f **DATA** ( )	«Carrega» uma matriz numérica.
LOAD f **DATA$** ( )	«Carrega» uma matriz «alfanumérica».
LOAD f **CODE** m,n	«Carrega» no máximo n «bytes», a partir do endereço m.
LOAD f **CODE** m	«Carrega» «bytes» a partir do endereço m.
LOAD f **CODE**	«Carrega» «bytes» a partir do endereço, do qual foram gravados.
LOAF f **SCREEN$**	LOAD f CODE 16384, 6912. Procura um ficheiro do tipo correcto, na «casette» e carrega-o, apagando as anteriores versões em memória. Ver o Capítulo 2Ø.
LPRINT	Como **PRINT** mas utilizando a impressora e variáveis.
MERGE f	Como LOAD f, mas não apaga linhas do programa anterior a não ser que tenham o mesmo número ou nome.
MOVE f_1, f_2	Para uso com periféricos.
NEW	Reinicia o sistema BASIC, apagando programas e variáveis, usando a memória até ao endereço cujo «byte» se encontra no RAMBOT preservando as variáveis no sistema UDG, P RAMT, RASP e PIP.

Apêndice A: Revisão do BASIC do TC 2048

Função	Resultado
NEXT :	(1) Encontra a variável do controlo :; (2) Acrescenta-lhe o incremento (STEP); (3) Se a STEP >=Ø e o valor > o limite, ou se o STEP < o limite, então salta para fora do ciclo. Error 2 se não existe nenhuma variável :; Error 1, se houver erro distinto da variável de controlo :.
OPEN$	Para uso com periféricos.
OUT m,n	Dá saída ao «byte» n, no porto m, ao nível do processador. (Carrega o par de registos bc com o valor m, o registo a com n, e executa a instrução em «Assembler»: Out (c), a.) Error B, se não tivermos Ø <m<=65535, -255 <=n<=255.
OVER n	Controla a impressão sobreposta dos caracteres seguintes. Se Ø n=, os caracteres apagam os anteriores nessa posição. Se n=1, os novos caracteres são misturados com os anteriores, dando os INK, se um deles (mas não ambos) a tiver, e cor PAPER dos locais onde ambos tivessem cor da tinta ou cor do papel. Erro K se n não for Ø ou 1.

PAPER n	Como **INK**, mas para controlar a cor do **PAPER** (fundo).
PAUSE n	Pausa no programa. Apresenta o «display file», (ficheiro do ecrã) durante n «imagens» (ao ritmo de 50 por segundo) até que uma tecla seja premida. 0 >=n<65535, senão ocorre erro B. Se n=0 , então a pausa não é temporizada, durando até que uma tecla seja premida.
PLOT c,m,n	Imprime um ponto de tinta (sujeitos a OVER e INVERSE) no «pixel» (|m|, |n|); muda a posição do **PLOT**. A não ser que o elemento de cor especifique outro «modo» a cor **INK** do «pixel» é mudada para a cor **INK** corrente, e os outros atributos (cor de **PAPER**, **BRIGHT**, **FLASH**) permanecem inalteráveis. 0 <=|m|<=255,0<=|n|<=175, senão ocorre B.
POKE m,n	Atribui o valor n ao «byte» armazenado no endereço m. 0 <=m<=65535, — 255<=n<=255, senão ocorre Erro B.
PRINT...	O '...' é uma sequência de elementos de **PRINT**, separados por vírgula, ponto e vírgula ou apóstrofo, armazenados no «display file» para serem impressos no ecrã. Um ponto e vírgula (;) entre dois elementos não tem qualquer efeito; serve unicamente para separar os elementos. Uma vírgula (,) envia o caracter de controlo de vírgula, e um apóstrofo (') envia o caracter **ENTER**. No final de uma instrução **PRINT**, se esta não terminar num ponto e vírgula, vírgula ou apóstrofo, é enviado um caracter **ENTER**.
	(1) Vazio, isto é, nada.
	(2) Uma expressão numérica. Primeiro é impresso um sinal menos se o número for negativo. Seja x o módulo do valor. Se $x<10^{-5}$ ou $x>=10^{13}$ então é impresso em notação científica. A mantissa pode ter até oito dígitos e o ponto decimal.

Função	Resultado
	(inexistente se for só um dígito) aparece depois do primeiro. O expoente é E, seguido de + ou – e um ou dois dígitos. Se não, x é impresso na notação decimal corrente, com um máximo de oito algarismos significativos e sem zeros à direita do ponto decimal. Um ponto decimal mesmo no princípio do número é acompanhado por um zero, de modo que .Ø3 e Ø.3 são escritos desta forma. Ø é impresso com o único dígito Ø. (3) Expressão de «string». Os símbolos na «string» são expandidos, possivelmente com um espaço antes ou depois. Caracteres de controlo têm o seu efeito de controlo. Caracteres irreconhecidos são impressos como ?. (4) **AT** m,n Saída de um caracter de controlo **AT** seguido por «byte» para um m (número de linha) e um «byte» para n (número de coluna). (5) **TAB** n Saída de um caracter de controlo **TAB** seguido por dois «bytes» para n (primeiro o «byte» menos significativo), e o fim do **TAB**. (6) Elemento de cor, sob a forma de instrução **PAPER, INK, FLASH, BRIGHT, INVERSE** ou **OVER**.
RANDOMIZE	RANDOMIZE Ø.
RANDOMIZE n	Inicia a variável de sistema chamada SEED utilizada para gerar o próximo valor de **RND**. Se n <> Ø, SEED toma um valor n; Se n = Ø então toma o valor doutra variável de sistema chamada FRAMES, que conta o número de imagens emitidas até então, no ecrã e toma o processo verdadeiramente aleatório. **RANDOMIZE** aparece no teclado como **RAND**. Erro B se n não estiver entre Ø e 65535.

Função	Resultado
READ v_1, v_2, ... v_K	Dá valores às variáveis usando valores sucessivos da lista **DATA**. Erro C se uma das expressões for de tipo errado. Erro E se houver ainda variáveis por ler quando a lista **DATA** se esgotar.
REM ...	'...' pode ser qualquer sequência de caracteres excepto **ENTER**. Não é possível inserir na mesma linha, outras instruções,
RESTORE	RESTORE Ø.
RESTORE n	Repõe o ponteiro da lista **DATA** na primeira instrução de **DATA** cujo número de linha seja maior ou igual a n: a próxima instrução **READ** começará a ler ali.
RETURN	Toma como referência uma instrução retirada da pilha **GOSUB** e salta para a próxima linha. Erro 7 se não existir qualquer referência na pilha **GOSUB**. Houve algum erro no programa; **GOSUBs** não totalmente contrabalançados por **RETURNs**.
RUN	RUN Ø.
RUN n	CLEAR, seguido de **GOTO** n.
SAVE f	Grava programa e variáveis.
SAVE f **LINE** m	Grava programas e variáveis e assim que o programa estiver carregado (**LOAD**) haverá um salto automático para a linha m.
SAVE f **DATA** ( )	Grava uma matriz numérica.
SAVE f **DATA$** ( )	Grava uma matriz «alfanumérica» $.
SAVE f **CODE** m,n	Grava n «bytes» a partir do endereço m.

Função	Resultado
SAVE f **SCREEN$**	SAVE f CODE 16384,6912 Grava informação em «cassette», dando-lhe o nome f. Erro F se f for vazio ou tiver 11 ou mais caracteres. Veja Capítulo 20.
STOP	Pára o programa com relatório 9. **CONTINUE** fará continuar o programa na instrução seguinte.
VERIFY	O mesmo de **LOAD** à excepção dos dados que são carregados para a **RAM**, mas comparados com os que já existem. Erro R se a comparação mostrar «bytes» diferentes.

Apêndice B: O Conjunto de Caracteres



É este o conjunto completo de caracteres do TC2048, com códigos em decimal e hexadecimal. Se se considerar os códigos como sendo as instruções de código máquina do Z8Ø, então as colunas da direita dão as correspondentes da mnemónica em linguagem «Assembler».

Como provavelmente já se apercebeu, algumas instruções do Z8Ø são compostas começando com CBh ou EDh, como se mostra nas colunas da direita.

Código	Carácter
Ø	Não usado
1	Não usado
2	Não usado
3	Não usado
4	Não usado
5	Não usado
6	PRINT vírgula
7	EDIT

Hex	Z8Ø assembler	depois de CB	depois de EB
ØØ	nop	rlc b	
Ø1	ld bc,NN	rlc c	
Ø2	ld (bc), a	rlc d	
Ø3	inc bc	rlc e	
Ø4	inc b	rlc h	
Ø5	dec b	rlc l	
Ø6	ld b,N	rlc (hl)	
Ø7	rlca	rlc a	

Apêndice B: O Conjunto de Caracteres

Código	Caracter
8	Cursor p/ Esquerda
9	Cursor p/ Direita
10	Cursor p/ Baixo
11	Cursor p/ Cima
12	DELETE
13	ENTER
14	Número
15	Não usado
16	Controlo de INK
17	Controlo de PAPER
18	Controlo de FLASH
19	Controlo de BRIGHT
20	Controlo de INVERSE
21	Controlo de OVER
22	Controlo de AT
23	Controlo de TAB
24	Não usado
25	Não usado
26	Não usado
27	Não usado
28	Não usado
29	Não usado
30	Não usado
31	Não usado
32	Space (Espaço)
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	–
46	.
47	/
48	0
49	1

Hex	Z80 assembler	depois de CB	depois de ED
08	ex af,af'	rrc b	
09	add hl,bc	rrc c	
0A	ld a,(bc)	rrc d	
0B	dec bc	rrc e	
0C	inc c	rrc h	
0D	dec c	rrc l	
0E	ld c,N	rrc (hl)	
0F	rrca	rrc a	
10	djnz DIS	rl b	
11	ld de,NN	rl c	
12	ld (de),a	rl d	
13	inc de	rl e	
14	inc d	rl h	
15	dec d	rl l	
16	ld d,N	rl (hl)	
17	rla	rl a	
18	jr DIS	rr b	
19	add hl,de	rr c	
1A	ld a,(de)	rr d	
1B	dec de	rr e	
1C	inc e	rr h	
1D	dec e	rr l	
1E	ld e,N	rr (hl)	
1F	rra	rr a	
20	jr nz,DIS	sla b	
21	ld hl,NN	sla c	
22	ld (NN),hl	sla d	
23	inc hl	sla e	
24	inc h	sla h	
25	dec h	sla l	
26	ld h,N	sla (hl)	
27	daa	sla a	
28	jr z,DIS	sra b	
29	add hl,hl	sra c	
2A	ld hl,(NN)	sra d	
2B	dec hl	sra e	
2C	inc l	sra h	
2D	dec l	sra l	
2E	ld, l,N	sra (hl)	
2F	cpl	sra a	
30	jr nc,DIS		
31	ld sp,NN		

Apêndice B: O Conjunto de Caracteres

Código	Caracter
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	/
93	]

Hex	Z80 assembler	depois de CB	depois de ED
32	ld (NN),a		
33	inc sp		
34	inc (hl)		
35	dec (hl)		
36	ld (hl),N		
37	scf		
38	jr c,DIS	srl b	
39	add hl,sp	srl c	
3A	ld a,(NN)	srl d	
3B	dec sp	srl e	
3C	inc a	srl h	
3D	dec a	srl l	
3E	ld a,N	srl (hl)	
3F	ccf	srl a	
40	ld b,b	bit 0,b	in b,(c)
41	ld b,c	bit 0,c	out(c),b
42	ld b,d	bit 0,d	sbc hl,bc
43	ld b,e	bit 0,e	ld (NN),bc
44	ld b,h	bit 0,h	neg
45	ld, b,l	bit 0,l	retn
46	ld b,(hl)	bit 0,(hl)	im 0
47	ld b,a	bit 0,a	ld i,a
48	ld c,b	bit 1,b	in c,(c)
49	ld c,c	bit 1,c	out(c),c
4A	ld c,d	bit 1,d	adc hl,bc
4B	ld c,e	bit 1,e	ld bc,(NN)
4C	ld c,h	bit 1,h	
4D	ld c,l	bit 1,l	reti
4E	ld c,(hl)	bit 1,(hl)	
4F	ld c,a	bit 1,a	ld r,a
50	ld d,b	bit 2,b	in d,(c)
51	ld d,c	bit 2,c	out(c),d
52	ld d,d	bit 2,d	sbc hl,de
53	ld d,e	bit 2,e	ld (NN),de
54	ld d,h	bit 2,h	
55	ld d,l	bit 2,l	
56	ld d,(hl)	bit 2,(hl)	im 1
57	ld d,a	bit 2,a	ld a,i
58	ld e,b	bit 3,b	in e,(c)
59	ld e,c	bit 3,c	out(c),e
5A	ld e,d	bit 3,d	adc hl,de
5B	ld e,e	bit 3,e	ld de,(NN)
5C	ld e,h	bit 3,h	
5D	ld e,l	bit 3,l	

Apêndice B: O Conjunto de Caracteres

Código	Caracter
94	↑
95	_
96	£
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	©
128	□
129	◹
130	◸
131	▀
132	◿
133	◨
134	▚
135	▛
136	◺
137	▞

Hex	Z80 assembler	depois de CB	depois de ED
5E	ld e,(hl)	bit 3,(hl)	im 2
5F	ld e,a	bit 3,a	ld a,r
60	ld h,b	bit 4,b	in h,(c)
61	ld h,c	bit 4,c	out(c),h
62	ld h,d	bit 4,d	sbc hl,hl
63	ld h,e	bit 4,e	ld (NN),hl
64	ld h,h	bit 4,h	
65	ld h,l	bit 4,l	
66	ld h,(hl)	bit 4,(hl)	
67	ld h,a	bit 4,a	rrd
68	ld l,b	bit 5,b	in l,(c)
69	ld l,c	bit 5,c	out(c),l
6A	ld l,d	bit 5,d	adc hl,hl
6B	ld l,e	bit 5,e	ld hl,(NN)
6C	ld l,h	bit 5,h	
6D	ld l,l	bit 5,l	
6E	ld l,(hl)	bit 5,(hl)	
6F	ld, l,a	bit 5,a	rld
70	ld (hl),b	bit 6,b	in f, (c)
71	ld (hl),c	bit 6,c	
72	ld (hl),d	bit 6,d	sbc hl,sp
73	ld (hl),e	bit 6,e	ld (NN),sp
74	ld (hl),h	bit 6,h	
75	ld (hl),l	bit 6,l	
76	halt	bit 6,(hl)	
77	ld (hl),a	bit 6,a	
78	ld a,b	bit 7,b	inc a,(c)
79	ld a,c	bit 7,c	out(c),a
7A	ld a,d	bit 7,d	adc hl,sp
7B	ld a,e	bit 7,e	ld sp,(NN)
7C	ld a,h	bit 7,h	
7D	ld a,l	bit 7,l	
7E	ld a,(hl)	bit 7,(hl)	
7F	ld a,a	bit 7,a	
80	add a,b	res 0,b	
81	add a,c	res 0,c	
82	add a,d	res 0,d	
83	add a,e	res 0,e	
84	add a,h	res 0,h	
85	add a,l	res 0,l	
86	add a,(hl)	res 0,(hl)	
87	add a,a	res 0,a	
88	adc a,b	res 1,b	
89	adc a,c	res 1,c	

Apêndice B: O Conjunto de Caracteres

Código	Caracter	Hex	Z80 assembler	depois de CB	depois de ED
138	◧	8A	adc a,d	res 1,d	
139	◨	8B	adc a,e	res 1,e	
140	◩	8C	adc a,h	res 1,h	
141	◪	8D	adc a,l	res 1,l	
142	◫	8E	adc a,(hl)	res 1,(hl)	
143	■	8F	adc a, a	res 1, a	
144	(a)	90	sub b	res 2,b	
145	(b)	91	sub c	res 2,c	
146	(c)	92	sub d	res 2,d	
147	(d)	93	sub e	res 2,e	
148	(e)	94	sub h	res 2,h	
149	(f)	95	sub l	res 2,l	
150	(g)	96	sub (hl)	res 2,(hl)	
151	(h)	97	sub a	res 2,a	
152	(i)	98	sbc a,b	res 3,b	
153	(j)	99	sbc a,c	res 3,c	
154	(k)	9A	sbc a,d	res 3,d	
155	(l)	9B	sbc a,e	res 3,e	
156	(m)	9C	sbc a,h	res 3,h	
157	(n)	9D	sbc a,l	res 3,l	
158	(o)	9E	sbc a,(hl)	res 3,(hl)	
159	(p)	9F	sbc a,a	res 3,a	
160	(q)	A0	and b	res 4,b	ldi
161	(r)	A1	and c	res 4,c	cpi
162	(s)	A2	and d	res 4,d	ini
163	(t)	A3	and e	res 4,e	outi
164	(u)	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	B0	or b	res 6,b	ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	

Código	Caracter	Hex	Z80 assembler	depois de CB	depois de ED
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	C0	ret nz	set 0,b	
193	STR$	C1	pop bc	set 0,c	
194	CHR$	C2	jp nz,NN	set 0,d	
195	NOT	C3	jp NN	set 0,e	
196	BIN	C4	call nz,NN	set 0,h	
197	OR	C5	push bc	set 0,l	
198	AND	C6	add a,N	set 0,(hl)	
199	<=	C7	rst 0	set 0,a	
200	>=	C8	ret z	set 1,b	
201	<>	C9	ret	set 1,c	
202	LINE	CA	jp z,NN	set 1,d	
203	THEN	CB	-----	set 1,e	
204	TO	CC	call z,NN	set 1,h	
205	STEP	CD	call NN	set 1,l	
206	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
208	FORMAT	D0	ret nc	set 2,b	
209	MOVE	D1	pop de	set 2,c	
210	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN#	D3	out (N),a	set 2,e	
212	CLOSE#	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	BEEP	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	in a,N	set 3,e	
220	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	prefixes instructions using ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(hl)	
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	E0	ret po	set 4,b	

Apêndice B: O Conjunto de Caracteres

Código	Caracter
225	LLIST
226	STOP
227	READ
228	DATA
229	RESTORE
230	NEW
231	BORDER
232	CONTINUE
233	DIM
234	REM
235	FOR
236	GO TO
237	GO SUB
238	INPUT
239	LOAD
240	LIST
241	LET
242	PAUSE
243	NEXT
244	POKE
245	PRINT
246	PLOT
247	RUN
248	SAVE
249	RANDOMIZE
250	IF
251	CLS
252	DRAW
253	CLEAR
254	RETURN
255	COPY

Hex	Z80 assembler	depois de CB	depois de ED
E1	pop hl	set 4,c	
E2	jp po,NN	set 4,d	
E3	ex (sp),hl	set 4,e	
E4	call po,NN	set 4,h	
E5	push hl	set 4,l	
E6	and N	set 4,(hl)	
E7	rst 32	set 4,a	
E8	ret pe	set 5,b	
E9	jp (hl)	set 5,c	
EA	jp pe,NN	set 5,d	
EB	ex de,hl	set 5,e	
EC	call pe,NN	set 5,h	
ED	- - - -	set 5,l	
EE	xor N	set 5,(hl)	
EF	rst 40	set 5,a	
F0	ret p	set 6,b	
F1	pop af	set 6,c	
F2	jp p,NN	set 6,d	
F3	di	set 6,e	
F4	call p,NN	set 6,h	
F5	push af	set 6,l	
F6	or N	set 6,(hl)	
F7	rst 48	set 6,a	
F8	ret m	set 7,b	
F9	ld sp,hl	set 7,c	
FA	jp m,NN	set 7,d	
FB	ei	set 7,e	
FC	call m,NN	set 7,h	
FD	prefixes instructions using iy	set 7,l	
FE	cp N	set 7,(hl)	
FF	rst 56	set 7,a	

Apêndice C: Modos de Apresentação do Ecrã e Memória



Apêndice C: Modos de apresentação do ecrã e memória
Além do ecrã normal de 32 colunas, o TC2048 permite várias melhorias no modo de representação, para uso de programadores de Software mais avançados. A explicação pormenorizada destas capacidades está fora deste Manual.

De qualquer modo, pode recorrer aos programas TIMEX que se servem destas destas capacidades no processamento de texto, na apresentação de dados financeiros, nos programas de divertimento e nos educativos.

Modos de apresentação do ecrã
Utilize este comando para estabelecer uma largura máxima (64 caracteres — ou mais se redefinir o conjunto de caracteres) e seleccione **INK** branca. A cor complementar própria do **PAPER** (neste caso, o preto) é seleccionada automaticamente:

OUT 255,62

Outras selecções dos valores de **INK** são apresentadas na tabela seguinte:

VALOR	FUNÇÃO	INK	PAPER
6	Modo de largura máxima	Preto	Branco
8+6		Azul	Amarelo
16+6		Vermelho	Turquesa
24+6		Magenta	Verde
32+6		Verde	Magenta
40+6		Turquesa	Vermelho
48+6		Amarelo	Azul
56+6		Branco	Preto

Use o seguinte comando para seleccionar outro «display file», («modo» duplo-ecrã):

OUT 255,1

O «modo» extensivo de cor, que utiliza ambas as áreas do «display file», é seleccionado por:

OUT 255,2

Pode sempre regressar ao «modo» normal-ecrã simples — de 32 colunas através de:

OUT 255,0

Em qualquer dos «modos de representação» adicionais (para além do «normal» de 32 colunas) necessita estabelecer os caracteres e atributos na área de memória reservada para esses «displays«. (Veja a tabela abaixo). Pode fazê-lo utilizando «rotinas» em linguagem máquina combinadas com «Software» do sistema.

Endereços do display

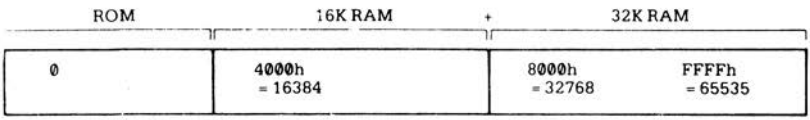
Endereço	Hexadecimal	Decimal
Display File 1	4000-54FF	16384-22527
Attribute File 1	5800-5AFF	22528-23295
Display File 2	6000-77FF	24576-30719
Attribute File 2	7800-7AFF	30720-31487

As instruções **POKE** devem utilizar endereços decimais.

Apêndice C: Modos de Apresentação do Ecrã e Memória

A memória

No interior do computador, tudo é armazenado em «bytes», ou seja, númros entre Ø e 255.
Pode pensar que armazenou o preço da lã ou a morada do fornecedor de fertilizantes, mas, na verdade, tudo isso foi convertido em «bytes».
O lugar onde cada «byte» é armazenado tem um endereço que é um número entre Ø e FFFFh (podendo então um endereço ser armazenado com dois «bytes»). Imagine a memória como uma longa lista de «caixas» numeradas, cada uma contendo um «byte«. Nem todas as «caixas» são iguais. Nas máquinas «standard» de 48 K RAM, as «caixas« de 4ØØØ a FFFFh são «caixas» RAM, o que significa que pode abri-las e modificar o conteúdo e as de Ø a 3FFFh são «caixas» ROM, que têm uma tampa de vidro e não podem ser abertas. Só pode ler o que quer que nestas últimas tenha sido posto, quando do fabrico do computador.



Para inspeccionar o conteúdo de uma «caixa«, utiliza-se a função **PEEK**; o argumento é o endereço da «caixa» e o resultado é o conteúdo. Por exemplo, o programa seguinte dá-nos os primeiros 21 «bytés» da ROM (e seus endereços):

```
1Ø PRINT "Endereço";TAB8;"Byte"
2Ø FOR a= ØTO 2Ø
3Ø PRINT a; TAB 8; PEEK a
4Ø NEXT a
```

Todos esses «bytes» não terão, provavelmente, qualquer significado para si, mas o microprocessador interpretá-los-á como instruções a executar. Para alterar o conteúdo de uma caixa (se for RAM), usa-se instrução **POKE**, cuja forma é:
POKE endereço, novo conteúdo onde

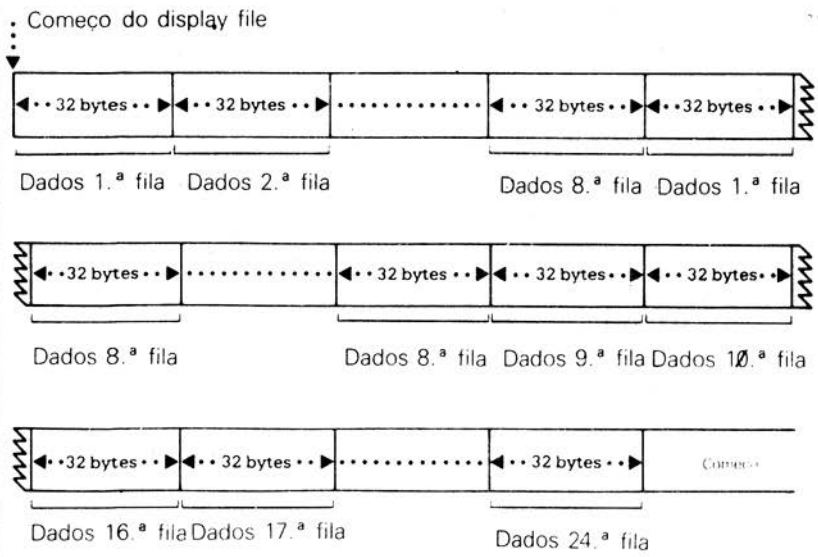
«endereço» e »novo conteúdo» serão expressões numéricas (note que o «endereço» deverá ser decimal e não hexadecimal). Por exemplo, se tiver:

POKE 31000,57

ao «byte» no «endereço» 31000 foi dado o valor 57 — «Tecle **PRINT PEEK** 31000, para o verificar. (Experimente fazer um **POKE** com outros valores, para se certificar de que não houve batota!). O valor novo deve estar compreendido entre — 255 e 255 e se for negativo então adiciona-se-lhe 256. As potencialidades de **POKE** dão imenso poder ao computador, se souber como fazê-lo, e imensas possibilidades de destruição se o não souber. Pode perder vastos programas que demoraram horas a «teclar», se o **POKE** for mal endereçado. Mas, felizmente, não causará qualquer dano ao computador.
Agora, vamos aprofundar conhecimentos sobre a utilização da RAM, mas não leia isto a não ser que esteja interessado.
A memória está dividida em áreas diferentes (mostradas no diagrama grande) para armazenamento de diversos tipos de informação. Cada área tem apenas dimensão suficiente para a informação que contém em dado momento, e se inserir mais informação em determinado ponto (por exemplo adicionado uma linha de programa ou uma variável) o espaço é obtido, deslocando tudo acima desse ponto.
Reciprocamente, se apagar informação, tudo o que lhe está acima é deslocado para baixo.

Organização dos dados de «pixel»
Os dados de «pixel» representam a localização dos »pontos» no ecrã. Os pontos são usados para construir caracteres e desenhar linhas no ecrã, através dos comandos **PLOT** e **DRAW**.
Estes dados são armazenados na memória, da seguinte forma:

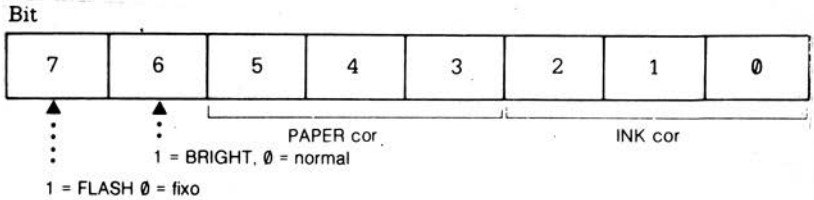
Apêndice C: Modos de Apresentação do Ecrã e Memória



Esta organização é aplicada a ambos os «display files» quando em vídeo normal, largura máxima ou ecrã duplo.
O «display file» armazena a imagem da televisão. A disposição é bastante curiosa, pelo que provavelmente não quererá fazer **PEEK** ou **POKE**. À posição de cada caracter, no ecrã, corresponde um quadrado de pontos de 8×8, e cada ponto pode ser ou **PAPER** 0 ou **INK** 1; e utilizando notação binária pode armazená-lo como 8 «bytes», um por cada fila.
Contudo, estes 8 «bytes» não são armazenados juntos. As filas correspondentes aos 32 caracteres de uma linha são armazenadas como um «varrimento« de 32 «bytes«, pois isto é o que o feixe de electrões de televisão necessita quando «varre» o ecrã da esquerda para a direita. Como uma imagem completa tem 24 linhas de 8 «varrimentos» cada, se espera um total de 172, armazenados uns após os outros, engan-se. Primeiro faz-se o das linhas 0 7, a seguir o das linhas 0 a 7 seguintes e assim por diante, até ao último varrimento das linhas de 0 a 7; depois o mesmo para as linhas de 8 a 15; a

seguir o mesmo para as linhas 16 e 23. A conclusão disto tudo é que se está habituado a um computador que utiliza **PEEK** e **POKE** do ecrã, terá que usar em vez dessas instruções. **SCREEN$** e **PRINT AT**, ou **PLOT** e **POINT**. Os atribuos são as cores, etc. da posição de cada caracter, usando a forma de **ATTR**. Estes são armazenados linha a linha pela ordem esperada.

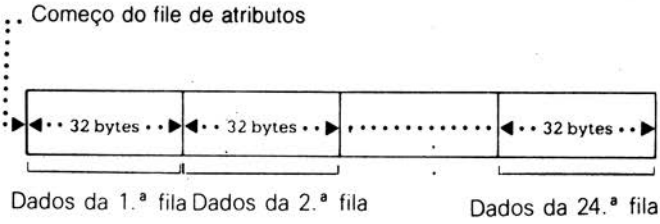
Formação do BYTE de atributo



Valor	Cor
7 111	Branco
6 110	Amarelo
5 101	Turquesa
4 100	Verde
3 011	Magenta
2 010	Vermelho
1 001	Azul
0 000	Preto

Organização dos dados de atributo

Os dados de atributo descrevem a cor do **PAPER** e **INK**, interminente (flashing) ou fixo, brilho intenso (bright) ou normal. Para cada caracter em «modo» video há um «byte» de «dados de atributo». Estes dados são armazenados na memória, assim:

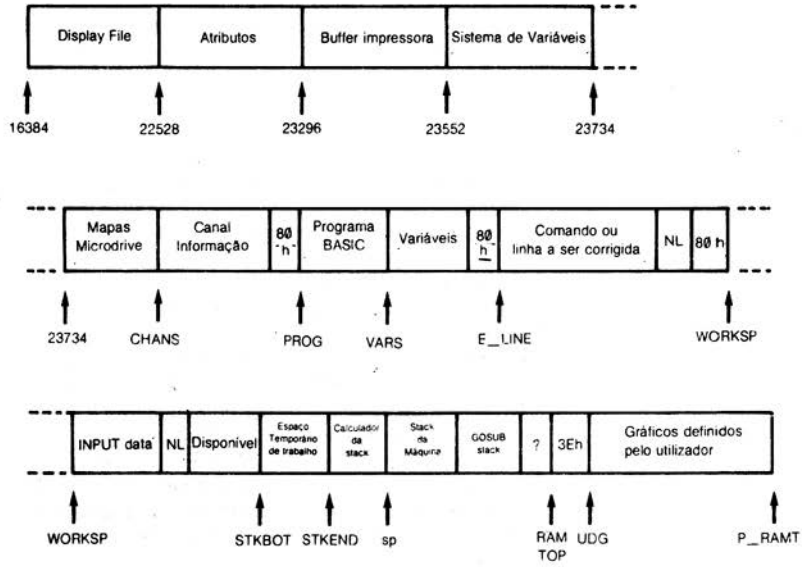


Em «modo extensivo» de cor, a organização de atributos (que residem na memória a partir de 6000h) é a mesma que para «os dados de pixel».

O «printer buffer» (buffer da impressora) armazena os caracteres destinados à impressora.

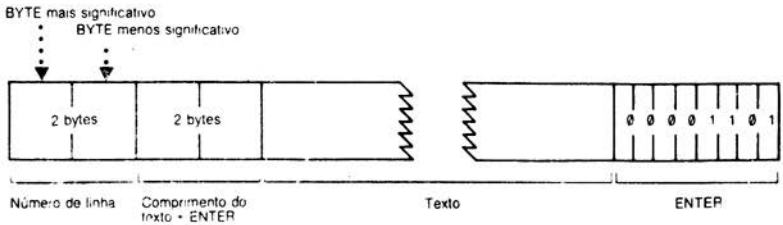
O sistema de variáveis contém informação para o computador saber em que estado se encontra. No Apêndice seguinte dá-se uma listagem completa, mas por agora, convém notar que algumas (**CHANS, PROG, VARS, E — LINE,** etc.) contém os endereços das fronteiras entre as várias áreas da memória. Estas não são variáveis BASIC e o compútador não as reconhecerá pelo nome.

Figura C-1 — Estrutura de memória



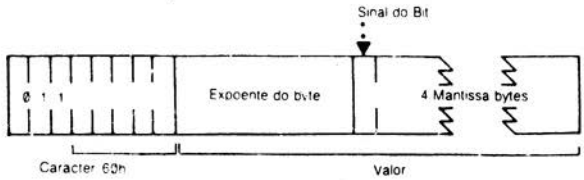
Cada linha de um programa BASIC tem a forma ilustrada na Figura C-2.

Figura C-2 — Estrutura da linha BASIC



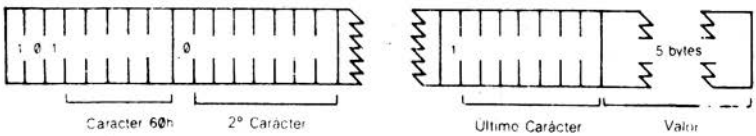
Note que, ao contrário de todo os outros casos, aqui o número de «bytes» é armazenado pondo em primeiro lugar o «byte» mais significativo: isto é, pela ordem que os escrevia.
No programa, uma constante numérica é seguida pela forma de ponto flutuante, utilizando o CHR$14 seguido de cinco «bytes» para o próprio número.
As variáveis têm formatos diferentes consoante a sua natureza. No nome, as letras são concebidas como começando por minúsculas. Isto está ilustrado na Fig. C-3.

Figura C-3 — Número cujo nome é só uma letra



A Figura C-4 ilustra o caso de um número cujo nome é maior que uma letra.

Figura C-4. Estrutura de um nome maior que uma letra.

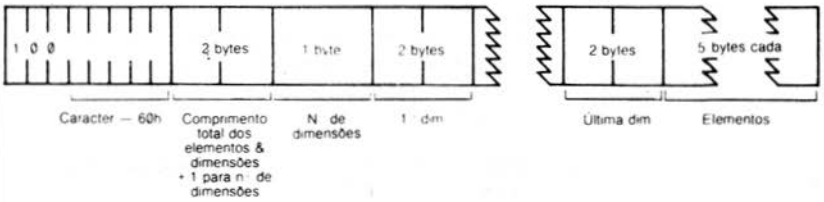


Uma matriz numérica é ilustrada na figura C-5.
A ordenação dos elementos é:
Primeiro, os elementos cujo primeiro índice é 1;
Depois, os elementos cujo primeiro índice é 3; e assim sucessivamente para todo os valores possíveis do primeiro índice.

Os elementos com um dado índice são ordenados da mesma maneira em relação ao segundo índice e assim por diante até ao último.

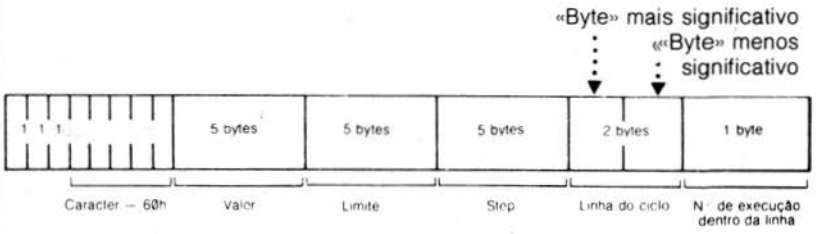
Como exemplo, teremos que os elementos de uma matriz b de 3•6 são armazenados pela seguinte ordem:
b(1,1) b(1,2) b(1,3) b(1,4) b(1,5) b(1,6),
b(2,1) b(2,2)... b(2,6) b(3,2)...b(3,6)

Figura C-5. Estrutura duma matriz numérica



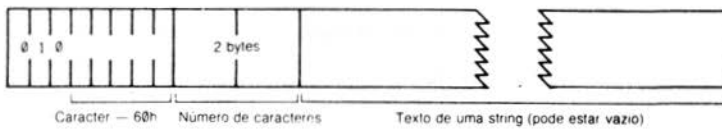
A estrutura de uma variável de controlo para um ciclo («loops») FOR/NEXT é ilustrada na Fig. C-6.

Figura C-6. Estrutura de um ciclo FOR/NEXT



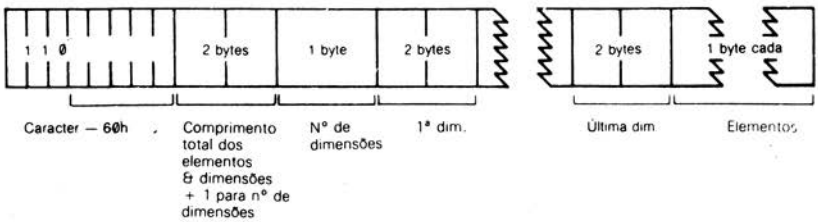
A estrutura de uma «string» é a ilustrada na Figura C-7;

Figura C-7. Estrutura de uma «string» (cadeia)



A Figura C-8 ilustra a estrutura de uma matriz de caracteres:

Figura C-8. Estrutura de uma matriz de caracteres



O calculador é a parte do sistema BASIC que trabalha com a aritmética, e os números com que está a trabalhar são retidos na «stack» do calculador.
A parte que resta contém o espaço ainda não usado.
A «stack» da máquina é a usada pelo circuito Z-80 para reter endereços de retorno, etc.
Qualquer número (à excepção de Ø) pode ser representado por + m* 2

onde

+ é o sinal
m é a mantissa e toma valores entre 1/2 e 1 (não pode ser 1)

e é o expoente, um número inteiro (possivelmente negativo).

Suponha que escreve m na escala binária. Como se trata de uma fracção, terá um ponto binário (como o ponto decimal na escala decimal) e a seguir uma fracção binária (tal como uma decimal). Assim, em binário.

um meio é representado por .1
um quarto é representado por .Ø1
três quartos é representado por .11
um décimo é representado por
ØØØ11ØØ11ØØ11ØØ11ØØ11 ... etc.

No nosso número m que é menor que 1, não há qualquer «bit» antes do ponto binário, e como é pelo menos 1/2, o «bit» imediatamente a seguir é 1.
Para armazenar o número no computador, utilizam-se cinco «bytes», Assim:

1. Escreva os primeiros oito »bits» da mantissa, no segundo «byte» (sabe-se que o primeiro é 1), os segundos oito no terceiro, os terceiros no quarto e os quartos oito «bits» no quinto «byte».

2. Substitua o primeiro «bit» no segundo «byte» — que se saber ser 1 — pelos valores: Ø para + (mais) e 1 para — (menos).

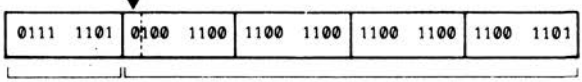
3. Escreva o expoente + 128 no primeiro «byte». Suponha, por exemplo, que o nosso número é 1/1Ø.

$$1/1Ø = \frac{4}{5} \times 2^{-3}$$

Então a mantissa m é.
11ØØ11ØØØØ11ØØ11ØØ11ØØ11ØØ1ØØ
em binário (uma vez que o 33.º «bit» é 1, deve-se arredondar o 32.º de Ø para 1) e o expoente é — 3. Aplicando as três regras dadas acima, obtêm-se os cinco «bytes» que a Figura C-9 ilustra.

Figura C-9. Zero representada sinal +

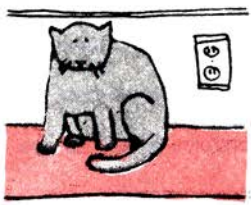
Ø zero representa o sinal +



Há uma forma alternativa de armazenar números entre — 65535 e + 65535:

1. O primeiro «byte» é Ø.
2. O segundo «byte» é Ø se for um número positivo. FFH se for negativo.
3. O terceiro e quarto «bytes» são respectivamente o menos e mais significativo «bytes» do número (ou o número + 131Ø72 se for negativo).
4. O quinto «byte» é Ø.

Apêndice D
As variáveis do sistema



Os «bytes» de memória entre 23552 e 23732 são guardados para uso específico do sistema. Pode-se espreitar (**PEEK**) para dentro deles para descobrir algo sobre o sistema, e alguns podem ter grande utilidade quando se lhes altera o valor (através **POKE**). São aqui listados com suas utilizações. Estas variáveis chamam-se variáveis do sistema, e têm nomes, mas não as confunda com as variáveis usadas em BASIC. O computador não reconhece os nomes como referidos às variáveis do sistema. Só são memórias, para nós.

As abreviaturas na coluna 1 têm o seguinte significado:

X — As variáveis não devem ser alteradas por **POKE** porque o sistema pode ficar inoperacional.

N — Fazer um **POKE** a essa variável não terá efeito duradoiro.

O número na coluna 1 é o número de «bytes» da variável. Se tiver dois «bytes», o primeiro é menos significativo — ao invés do que seria de esperar. Assim, para introduzir o valor v numa variável de dois «bytes» com endereço n «tecle»:

POKE n, v — 256 * **INT** (v/256)
POKE n + 1, **INT** (v/256)

e para ver o conteúdo use a expressão
PEEK n + 256 * **PEEK** (n + 1)

Nota	Endereço	Nome	Conteúdo
N18	23552	KSTATE	Usado na leitura do teclado.
N1	23560	LAST K	Guarda a última tecla premida.
1	23561	REPDEL	Tempo (em 1/50 do segundo) que uma tecla tem de ser premida para começar a repetir.
1	23562	REPPER	Intervalo (em 1/50 do segundo) entre duas repetições sucessivas de uma tecla continuamente premida: inicialmente 5.
N2	23563	DEFADD	Endereço dos argumentos de uma função definida pelo utilizador se se estiver a calcular alguma; senão 0.
N1	23565	K DATA	Guarda o segundo «byte» do controlo de cor introduzido a partir do teclado.
N2	23566	TVDATA	Guarda «bytes» de cor e controlos AT e TAB que vão para a televisão.
X38	23568	STRMS	Endereços dos canais ligados ao sistema.
2	23606	CHARS	256 menos do que o endereço do conjunto de caracteres (que começa num espaço e continua até ao símbolo de «copyright»). Normalmente em ROM, mas pode estabelecer os seus conjuntos de caracteres em RAM e fazer o CHARS apontar para eles.
1	23608	RASP	Intervalo do zumbido.
1	23609	PIP	Intervalo do estalido do teclado.
1	23610	ERR NR	1 a menos que o código da mensagem. Começa em 255 (para — 1) de modo que **PEEK** 23610 dá 255.
X1	23611	FLAGS	Diversas «flags» de controlo do sistema BASIC.
X1	23612	TV FLAG	«Flags» associadas à televisão.
X2	23613	ERR SP	Endereço do elemento da pilha da máquina que deve ser usado como retorno de erro.
N2	23615	LIST SP	Endereço do endereço de retorno para listagem automática.

Apêndice D: As Variáveis do Sistema

Nota	Endereço	Nome	Conteúdo
N1	23617	MODE	Especifica cursor tipo K, L, C, E ou G
2	23618	NEWPPC	Linha para que se deve saltar.
1	23620	NSPPC	Número da instrução na linha para a qual se deve saltar. Fazendo um **POKE** primeiro para NEWPPC e depois NSPPC força um salto para uma instrução específica de uma linha.
2	23621	PPC	Número de linha da instrução que está a ser executada.
1	23623	SUBPPC	Número dentro da linha da instrução que está a ser executada.
1	23624	BORDCR	Cor do **BORDER** vezes 8; também contém os atributos para a parte inferior do ecrã.
2	23625	E PPC	Número de linha corrente (com cursor de programa).
X2	23627	VARS	Endereço das variáveis.
N2	23629	DEST	Endereço da variável a ser modificada.
X2	23631	CHANS	Endereço dos dados do canal.
X2	23633	CURCHL	Endereço da informação que está a ser usada par entrada/saída.
X2	23635	PROG	Endereço do programa BASIC.
X2	23637	NXTLIN	Endereço da próxima linha de programa.
X2	23639	DATADD	Endereço do indicador do último elemento DATA.
X2	23641	E LINE	Endereço da instrução a ser introduzida pelo teclado.
2	23643	K CUR	Endereço do cursor.
X2	23645	CH ADD	Endereço do próximo caracter a ser interpretado: o caracter depois do argumento **PEEK**, ou **NEWLINE** no fim de uma instrução **POKE**.
2	23647	X PTR	Endereço do caracter depois do ?.
X2	23649	WORKSP	Endereço do espaço de trabalho temporário.
X2	23651	STKBOT	Endereço do fim do «stack» do calculador.
X2	23653	STKEND	Endereço do princípio do espaço livre.
N1	23655	BREG	Registo b do calculador.
N2	23656	MEM	Endereço da área usada para memória do calculador (geralmente MEMBOT, mas nem sempre).
1	23658	FLAGS2	Mais 'flags'.
X1	23659	DF SZ	O número de linhas (incluindo uma em branco) na parte inferior do ecrã.
2	23660	STOP	O número de linha no início de um programa para listagem automática.
2	23662	OLDPPC	Número de linha para a qual o **CONTINUE** deve saltar.
1	23664	OSPPC	Número da instrução dentro da linha para a qual **CONTINUE** salta.
N1	23665	FLAGX	Várias 'flags'.

Apêndice D: As Variáveis do Sistema

Nota	Endereço	Nome	Conteúdo
N2	23666	STRLEN	Comprimento da «string».
N2	23668	T ADDR	Endereço do próximo elemento da tabela de sintaxe.
2	23670	SEED	O primeiro número para a instrução **RND**. Esta variável é iniciada por **RANDOMIZE**.
3	23672	FRAMES	3 «bytes» (o menos significativo em primeiro lugar), o contador de imagens. Incrementados todos os 16 ms.
2	23675	UDG	Endereço do primeiro caracter gráfico definido pelo utilizador. Pode modificá-lo, por exemplo, para arranjar espaço, tendo menos caracteres definidos.
1	23677	COORDS	Coordenada x do último ponto de um **PLOT**.
1	23678		Coordenada y do último ponto de um **PLOT**.
1	23679	P POSN	Número até 33 para a coluna da posição da impressora.
1	23680	PR CC	«Byte» menos significativo do endereço da próxima posição de **LPRINT** em que se deve escrever (BUFFER da impressora).
1	23681		Não usado.
2	23682	ECHO E	Número de coluna (até 33) e número linha (até 24) (na parte de baixo) do BUFFER de entrada.
2	23684	DF CC	Endereço no ficheiro de imagem da posição de **PRINT**.
2	23686	DFCCL	Como DF CC para a parte inferior do ecrã.
X1	23688	S POSN	Número de coluna (até 33) para a posição de **PRINT**.
X1	23689		Número de linha (até 24) da posição de PRINT.
X2	23690	SPOSNL	Como S POSN para parte de baixo.
1	23692	SCR CT	Contador para »**scroll**«: é sempre um a mais do que o número de «**scrolls**» que serão feitos antes de parar para perguntar «**scroll?**». Se introduzir aqui (por **POKE**) continuamente um número maior do que 1 (por exemplo 255), o ecrã continuará a fazer «**scroll**» sem lhe perguntar.
1	23693	ATTR P	Cores correntes permanentes, etc. (estabelecidas pelas instruções de cores)
1	23694	MASK P	Usado para cores transparentes, etc. Qualquer «bit» que seja 1 mostra que o atributo correspondente é tomado não do **ATTR** P, mas do que já está no ecrã.
N1	23695	ATTR T	Cores correntes temporárias, etc. (estabelecidas pelos elementos de cores).
N1	23696	MASK T	Como MASK P, mas temporário.
1	23697	P FLAG	Mais 'flags'.
N30	23698	MEMBOT	Área de memória do calculador usado para Guardar números que não podem ser colocadas convenientemente no «stack» do calculador.
2	23728		Não usado.
2	23730	RAMTOP	Endereço do último «byte» da área do sistema BASIC.
2	23732	P-RAMT	Endereço do último «byte» da RAM física.

Este programa dá-lhe os 22 primeiros «bytes» da área de variáveis:

```
10 FOR n = 0 TO 21
20 PRINT PEEK (PEEK 23627 + 256*PEEK
23628 + n)
30 NEXT n
```

Tente comparar a variável de controlo n com as descrições dadas acima.
Agora modifique a linha 20 para

```
20 PRINT PEEK (23755 + n)
```

Isto diz-lhe quais os 22 primeiros «bytes» da área do programa. Compare-os com o programa em si.

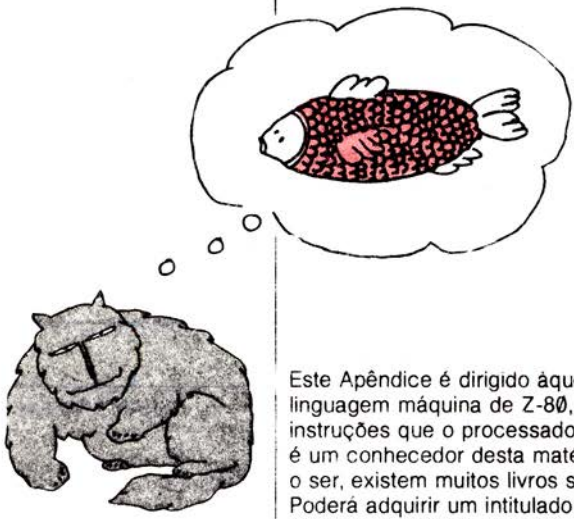
ENDEREÇOS DO «DISPLAY»

Endereço	**Hexadecimal**	**Decimal**
Display File 1	4000-57FF	16384-22527
Attribute File 1	5800-5AFF	22528-23295
Display File 2	6000-77FF	24576-30719
Attribute File 2	7800-7AFF	30720-31487

POKE statements must use decimal addresses.

Instruções POKE devem utilizar endereços decimais.

Apêndice E: Como Usar a Linguagem Máquina



Este Apêndice é dirigido àqueles que entendem a linguagem máquina de Z-8Ø, o conjunto de instruções que o processador Z-8Ø utiliza. Se não é um conhecedor desta matéria, mas gostaria de o ser, existem muitos livros sobre este assunto. Poderá adquirir um intitulado Z-8Ø MACHINE CODE (ou ASSEMBLY LANGUAGE) for the absolute Beginner.

Estes programas são normalmente escritos em assembly que embora não é com a prática tão difícil de compreender.

No entanto, para que o programa «corra» no computador é preciso codificá-lo em sequências de «bytes» — nesta forma é chamado código máquina. Esta tradução é normalmente feita pelo próprio computador, utilizando um programa chamado assemblador. Não há um assemblador já feito no TC 2048, mas será com certeza capaz de adquirir um em cassette.

À falta deste, terá de fazer a tradução por si, desde que o programa não seja demasiado extenso.

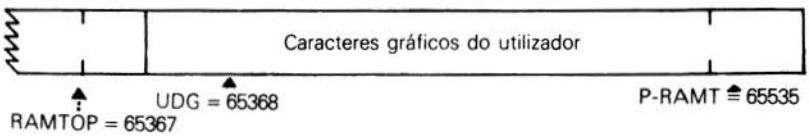
Veja, por exemplo, programa

```
ld bc, 99.
ret
```

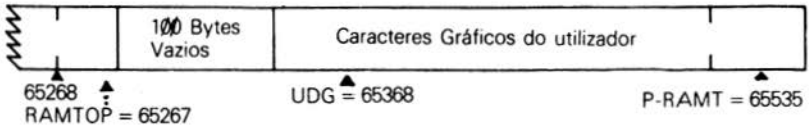
que «carrega» o par de «registos» bc com o valor 99.
Isto traduz-se em código máquina nos quatro «bytes» 1,99,Ø, (para o ld bc, 99) e 2Ø1 (para ret). (Se procurar o 1 e 2Ø1 no Apêndice B, encontrará ld bc, NN — onde NN corresponde a um número de dois «bytes» — e ret).

Quanto tiver o seu programa em linguagem máquina, o passo seguinte é introduzi-lo no computador (Um assemblador provavelmente fá-lo-ia automaticamente). Precisa de se decidir em que posição de memória o vai colocar, e o melhor é arranjar espao entre a zona BASIC e os caracteres gráficos definidos pelo utilizador (UDG).

A sua máquina é de 48 K, então a parte final da RAM tem



Se «teclar»
CLEAR 65267
obterá um espaço de 1ØØ (para uma boa medida) «bytes» começando no endereço 65268.



Para introduzir o programa em linguagem máquina, tem de executar um programa em BASIC, que será algo como

```
10 LET a = 65268
20 READ n: POKE a,n
30 LET a = a + 1: GOTO 20
40 DATA 1,99,0,201
```

(Este programa vai parar com a mensagem **E out of DATA**, quando tiver preenchido os quatro «bytes» específicos).
Para executar o programa em linguagem, máquina, utiliza-se a função **USR** — mas desta vez com argumento numérico, o endereço inicial. O resultado é o valor do «registo» bc quando voltar ao programa em código máquina. Por isso se fizer

PRINT USR 65268

obterá a resposta 99.

O endereço de regresso ao BASIC está num «stack» na forma usual e por isso o regresso é feito pela instrução RET do Z80.

Contudo se o utilizador voltar ao BASIC é necessário restituir os valores anteriores dos registos iy e i.
Pode gravar o seu programa em código máquina com:

SAVE "QUALQUER NOME" **CODE** 65268
Vendo bem, não é possível gravá-lo de forma a que, quando o «carregar» ele «corra» automaticamente, mas pode-se obviar a isso usando um programa em BASIC.

```
10 LOAD " " CODE 65268,4
20 PRINT USR 65268
```

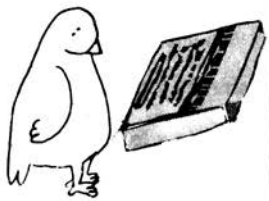
Primeiro «tecle»
SAVE "QUALQUER NOME" **LINE** 10
e então
SAVE "xxxx" **CODE** 65268,4
LOAD "QUALQUER NOME"

Assim «carregará» automaticamente e «correrá» o programa em BASIC que vai, por sua ez, «carregá-lo e corrê-lo« em linguagem máquina.

Notas:

1. Se «INTERRUPTS» estiverem activos, não utilize o registo iy.
2. O registo i não suportará o valor superior a 3Fh.

Apêndice F:
Tabela de «comandos»



Cursor		Definição
K	**Keyword** («comando»)	Se uma tecla for premida quando o cursor K está no ecrã, o comando impresso na tecla pode ser utilizado (por exemplo PRINT).
L	**Letter** («letra»)	Se uma tecla é premida quando o cursor L está no ecrã, a letra ou símbolo impresso na tecla é considerado pelo computador e apresentado no ecrã (isto é, Ø ou Å).
C	**Caps Lock** («maiúsculas»)	Quando uma tecla alfabética é premida produz essa letra em maiúsculas.
G	**Graphics** («gráficos»)	Inicia o «modo» gráfico de tal modo que os caracteres gráficos impressos nas teclas 1 a 8 podem ser utilizados e UDG 1
E	**Extended** («extensivo»)	Se uma tecla for premida quando o cursor E está no ecrã, o comando impresso acima da tecla pode ser utilizado. (Por exemplo, LPRINT); se premirmos simultaneamente a tecla SYMBOL SHIFT, o comando impresso abaixo da tecla será utilizado (por exemplo, BEEP).

Apêndice F: Tabela de «Comandos»

Símbolos e funções	Nome da tecla	Instruções
ABS	G	Cursor E. Premir tecla.
ACS	W	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
AND	Y	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
ASN	Q	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
AT	I	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
ATN	E	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
ATTR	L	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
BEEP	Z	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
BIN	B	Cursor E. Premir a tecla.
BORDR BORDER)	B	Cursor K. Premir a tecla.
BREAK	BREAK	Premir a tecla.
BRIGHT	B	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
CAPS LOOCK	2	Cursor L ou C. Premir simultaneamente CAPS SHIFT e a tecla.
CAT	9	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
CHR$	U	Cursor E. Premir a tecla.
CIRCLE	H	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
CLEAR	X	Cursor K. Premir a tecla.
CLOSE#	5	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
CLS	V	Cursor K. Premir a tecla.
CODE	I	Cursor E. Premir a tecla.
CONT	C	Cursor K. Premir a tecla.
COPY	Z	Cursor K. Premir a tecla.
COS	W	Cursor E. Premir a tecla.
DATA	D	Cursor E. Premir a tecla.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
DEF FN	1	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
DELETE	Ø	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
DIM	D	Cursor K. Premir a tecla.
DRAW	W	Cursor K. Premir a tecla.
EDIT	1	Cursor K. Premir simultaneamente CAPS SHIFT e a tecla.
ERASE	7	Cursor E. Premir simultaneamente SYMBOL SHIFT e a tecla.
EXP	X	Cursor E. Premir a tecla.
FLASH	V	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
FN	2	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
FOR	F	Cursor K. Premir a tecla.
FORMAT	Ø	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
GOSUB	H	Cursor K. Premir a tecla.
GOTO	G	Cursor K. Premir a tecla.
GRAPHICS	9	Cursor K L ou C. Premir simultaneamente a tecla e CAPS SHIFT.
IF	U	Cursor K. Premir a tecla.
IN	I	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
INK	X	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
INKEY$	N	Cursor E. Premir a tecla.
INPUT	I	Cursor K. Premir a tecla.
INT	R	Cursor E. Premir a tecla.
INV. VIDEO	4	Cursor K, L ou C. Premir simultaneamente a tecla e CAPS SHIFT.
INVERSE	M	Cursor E. Premir simultaneamente SYMBOL SHIFT e a tecla.
LEN	K	Cursor E. Premir a tecla.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
LET	L	Cursor K. Premir a tecla.
LINE	3	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
LIST	K	Cursor K. Premir a tecla.
LLIST	V	Cursor E. Premir a tecla.
LN	Z	Cursor E. Premir a tecla.
LOAD	J	Cursor K. Premir a tecla.
LPRINT	C	Cursor E. Premir a tecla.
MERGE	T	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
MOVE	6	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
NEW	A	Cursor K. Premir a tecla.
NEXT	N	Cursor K. Premir a tecla.
NOT	S	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
OPEN #	4	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
OR	U	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
OUT	O	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
OVER	N	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
PAPER	C	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
PAUSE	M	Cursor K. Premir a tecla.
PEEK	O	Cursor E. Premir a tecla.
PI	M	Cursor E. Premir a tecla.
PLOT	Q	Cursor K. Premir a tecla.
POINT	8	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
POKE	O	Cursor K. Premir a tecla.
PRINT	P	Cursor K. Premir a tecla.
RAND	T	Cursor K. Premir a tecla.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
READ	A	Cursor **E**. Premir a tecla.
REM	E	Cursor **K**. Premir a tecla.
RESTORE	S	Cursor **E**. Premir a tecla.
RETRN (RETURN)	Y	Cursor **K**. Premir a tecla.
RND	T	Cursor **E**. Premir a tecla.
RUN	R	Cursor **K**. Premir a tecla.
SAVE	S	Cursor **K**. Premir a tecla.
SCREEN$	K	Cursor **E**. Premir simultaneamente a tecla e SYMBOL SHIFT.
SGN	F	Cursor **E**. Premir a tecla.
SIN	Q	Cursor **E**. Premir a tecla.
SPACE BAR	Barra de espaços	Premir a tecla
SQR	H	Cursor **E**. Premir a tecla.
STEP	D	Cursor **K**, **L** ou **C**. Premir simultaneamente a tecla e SYMBOL SHIFT.
STOP	A	Cursor **K**, **L** ou **C**. Premir simultaneamente a tecla e SYMBOL SHIFT.
STR$	Y	Cursor **E**. Premir a tecla.
SYMBOL	SYMBOL	Premir a tecla.
SHIFT	SHIFT	Premir a tecla.
TAB	P	Cursor **E**. Premir a tecla.
TAN	E	Cursor **E**. Premir a tecla.
THEN	G	Cursor **K**, **L** ou **C**. Premir simultaneamente a tecla e SYMBOL SHIFT.
TO	F	Cursor **K**, **L** ou **C**. Premir simultaneamente a tecla e SYMBOL SHIFT.
TRUE VIDEO	3	Cursor **K**. Premir simultaneamente a tecla e CAPS SHIFT.
USR	L	Cursor **E**. Premir a tecla.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
VAL	J	Cursor E. Premir a tecla.
VAL$	J	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
VERIFY	R	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
!	1	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
"	P	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
#	3	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
$	4	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
%	5	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
>	T	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
/	D	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
↑	H	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
£	X	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
?	C	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
\	V	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
<=	Q	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
>=	E	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
©	2	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
[	Y	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
]	U	Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
&	6	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
'	7	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
(	8	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
)	9	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
*	B	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
+	K	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
,	N	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
-	0	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
_	J	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
.	M	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
:	Z	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
;	O	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
=	L	Cursor K L ou C Premir simultaneamente a tecla e SYMBOL SHIFT.
▙	1	Cursor G Premir a tecla.
▛	2	Cursor G Premir a tecla.
▜	3	Cursor G Premir a tecla.
▟	4	Cursor G Premir a tecla.
←	5	Cursor K L ou C Premir simultaneamente a tecla e CAPS SHIFT.
▌	5	Cursor G Premir a tecla.
↓	6	Cursor K L ou C Premir simultaneamente a tecla e CAPS SHIFT.
▚	6	Cursor G Premir a tecla.
↑	7	Cursor K L ou C Premir simultaneamente a tecla e CAPS SHIFT.
▞	7	Cursor G Premir a tecla.
→	8	Cursor K L ou C Premir simultaneamente a tecla e CAPS SHIFT.
▐	8	Cursor G Premir a tecla.

Apêndice F: Tabela de «comandos»

Símbolos e funções	Nome da tecla	Instruções
<>	W	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
<	R	Cursor K, L ou C. Premir simultaneamente a tecla e SYMBOL SHIFT.
©		Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
~		Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
|		Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
\		Cursor E. Premir simultaneamente a tecla e SYMBOL SHIFT.
]		Cursor E. Premir simultaneamente a tecla C SYMBOL SHIFT.

Apêndice G: Indice



A

ABS	211
ACS	211
AND	128, 131, 211
apóstrofo	82, 88, 208
arranque automático	51
ASN	211
aspas	30, 85 88
AT	84, 88
ATN	211
ATTR	186, 187, 211
atributos	206

B

basic 27
BEEP 66, 69 216
BIN 164, 166, 212
BORDER 55, 58, 212, 216
BREAK 74, 78, 192, 194, 205
BRIGHT 176, 177, 181, 207, 216

C

© copyright 25
cabo audio 6, 9
cabo TV 6, 8
CAPS LOCK 18, 73, 204
CAPS SHIFT 15, 204
CAT 200, 216
chaveta esquerda 25
chaveta direita 25
CHR$ 183, 186, 212
ciclo contendo outro ciclo 118
ciclo **FOR/NEXT** 114, 119
ciclo sem fim 114
CIRCLE 62, 64, 216
CLEAR 91, 93, 216
CLOSE 200, 216
CLS 91, 93, 217
CODE 184, 187, 212
código de números 128, 183
código máquina 252
COS 212
como «cortar» strings 147, 149, 210
comparando strings 127
comparando valores 126
compilador 139
CONT 76, 78, 217
COPY 191, 194, 217
cursor 18, 204
cursor 23, 204
cursor 22
cursor 13, 203
cursor 14, 204
cursor do programa 73, 79, 205
cursores das setas 16, 33, 204

D

DATA 108, 111, 217
declaração de atribuição 90, 124
DEF FN 217
DELETE 17, 32, 204
DIM 143, 149, 217
dois pontos 87, 88
DRAW 60, 64, 218

E

E (expoente) 96, 97
EDIT 80, 105, 205
elementos de uma matriz 142, 149
endereços de display 236, 250
ENTER 29, 36, 205
ERASE 201, 218
erros por arredondamento 98
estrutura de um ciclo **FOR/NEXT** 244
estrutura da linha basic 242
estrutura da memória 242
estrutura duma matriz de caracteres 245
estrutura duma matriz numérica 244
EXP 212

F

file do display 239
FLASH 177, 178, 181, 207
forma do byte de atributo 240
FORMAT 200, 218
fonte de alimentação 7, 8
FOR 114, 119, 218
FN 212
funções 99, 211

G

gerador de números aleatórios 99
GOTO 74, 78, 93, 218
GOSUB 131, 140, 218
gráficos definidos pelo utilizador 162, 204
gravação 48
gravador 7, 9, 201

I

IF 121, 125, 130, 218
impressora 6, 189, 201, 208
IN 199, 212
indicador de «erro de sintaxe» 33, 205
INK 56, 58, 60, 105, 206, 219
INKEY$ 168, 173, 212
INPUT 104, 110, 157, 219
instrução 27
INT 100, 212
interferência TV 9
interpretador 139
interruptor ON/OFF 6, 9
INVERSE 176, 177, 181, 207, 219
INVERSE VIDEO 20

J

joystick 6, 172, 173, 201

L

LEN 212
LET 90, 93, 219
letras maiúsculas 73
LINE 51, 53, 224
linguagem máquina 254
linha de programação 73
LIST 115, 119, 220
LLIST 190, 194, 220
LN 212
LOAD 39, 220
LOAD...CODE 220
LOAD...DATA 148, 149, 220
LOAD...SCREEN$ 220

M

mapa da memória 242
matriz 142, 149
matriz alfanumérica 210
mensagem codificada 30, 42
mensagem de copyright 9, 13
MERGE 52, 54, 220
modem 201
modo extensivo 23, 35
modo extensivo cor 152
modo gráfico 22, 34, 150, 204
modo imediato 71
modos de apresentação do ecrã 231
módulos 133
monitor 6, 201
MOVE 200, 220

N

NEW 32, 36, 78, 93, 220
NEXT 114, 119, 221
nome da variável 90, 93, 209
nome da variável da matriz 144, 149, 209
nome das variáveis de controlo 209
nome das variáveis «string» 90, 93
nome do programa 40, 53
NOT 129, 131, 212
notação científica 97, 102

O

onda sinusoidal 158
OPEN 200, 221
OR 129, 131, 213
OUT 199, 221
OVER 178, 181, 207, 221

P

palavras-chave 14, 27, 203
PAPER 57, 58, 206, 222
parêntesis 97, 102
PAUSE 171, 173, 222
PEEK 198, 213, 238
periféricos 199, 200, 201

PI	213
PIXEL	59, 160, 163
PLAY	41
PLOT	62, 64, 157, 160, 222
POINT	185, 187, 213
POKE	163, 166, 198, 222, 247
ponto e vírgula	80, 88
posição de **PRINT**	80, 151
posições de **PLOT**	151, 160
PRINT	28, 36, 72, 87, 222
prioridades	96, 102, 225
procrusteana	145
programa	3, 4, 71
programa de treino sobre o teclado	25
programa estruturado	133, 140
programa ramificado	121
programa em cassettes	38
pseudo-gerador	100

R

ramificação	121
RAND	101, 223
READ	108, 111, 224
relações	128, 131
relações lógicas	128
REM	47, 77, 224
RESTORE	110, 111, 224
RETURN	134, 140, 224
RND	99, 102, 213
RUN	42, 74, 78, 93, 224

S

SAVE	48, 53, 224
SAVE...CODE	224
SAVE...DATA	148, 149, 224
SAVE...SCREEN$	230
SCREEN$	159, 160, 213
scroll?	74, 78
seta de baixar	81
seta de subir	79, 81
SGN	213
SIN	158, 213
sintonia canal TV	9
SQR	35, 99, 213
«stack» do calculador	245
«stack» máquina	245
STEP	117, 119, 218

STOP	76, 106, 111, 225
«string»	32, 36, 210
«string» de uma matriz	145, 149, 210
STR$	214
subrotinas	134, 140
subscrito	143, 149
substrings	147, 149, 210
SYMBOL SHIFT	18, 204
T	
TAB	83, 88, 194, 208
TAN	214
THEN	125, 130, 218
TO	114, 119, 218
tomada de joystick	6
TRUE VIDEO	21
U	
USR	164, 166, 214
V	
VAL	214
VAL$	214
variável	90
variável de controlo	114, 119
variável «string»	90, 93
variáveis do sistema	247
VERIFY	49, 53, 224
vírgula	80, 87
X	
X/Y eixos	158
Z	
Z8Ø chip processadora	254
+,−,•,/,↑	96, 101, 215
=,<,>,<=,<>	126, 131, 215
$	90

Apêndice H: Lista de Códigos



As mensagens aparecem no fundo do ecrã quando o computador pára a sua execução em BASIC, e explica por que razão parou, quer por um motivo natural, quer pela ocorrência de um erro.

A mensagem tem um número ou letra de código, de modo que pode referenciar-se à tabela dada a seguir. Igualmente, uma mensagem curta explica o que aconteceu, o número da instrução da linha onde a paragem ocorreu.
(Um comando directo é mostrado como linha Ø. Dentro de uma linha, a instrução 1 encontra-se no princípio, a instrução 2 vem a seguir a dois pontos ou **THEN**, e assim por diante).

O comportamento do comando **CONTINUE** depende muito destas mensagens. Normalmente, **CONTINUE** retoma a linha e instrução especificadas na última mensagem, excepto se se tratar de Ø, 9 e D.

Segue-se uma tabela que apresenta todas as mensagens. Também lhe diz em que circunstâncias uma mensagem pode ocorrer. Por exemplo, um erro **A Invalid argument** pode ocorrer com **SQR**, **IN**, **ACS** e **ASN**.

Apêndice H: Lista de Códigos

Cód.	Situação	Significado
0	Qualquer	OK Final bem sucedido, ou salto para uma linha maior do que as existentes. Esta mensagem não modifica a linha e instrução para a qual saltou o CONTINUE.
1	NEXT	NEXT without FOR (NEXT sem FOR). Variável do controlo não existe (não foi iniciado por uma instrução FOR), mas há uma variável vulgar com o mesmo nome.
3	Qualquer	Variable not found (Variável não encontrada). Para uma variável simples isto ocorre se a vriável for usada antes de se lhe ter atribuído um valor por uma instrução LET, READ ou INPUT ou «carregada» de «cassette» ou estabelecida por uma instrução FOR. Para uma variável indexada ocorrerá se ela for usada antes de ter sido dimensionada numa instrução DIM ou lida de uma cassette.
3	Variáveis indexadas Subcadeias	Subscript wrong (Índice errado). Um índice está para além da dimensão da matriz, ou há um número errado de índices. Se o índice for negativo ou superior a 65535 então ocorrerá erro B.
4	LET, INPUT, FOR, DIM, GOSUB, LOAD, MERGE. Por vezes durante o cálculo de uma expressão.	Out of memory (Memória esgotada). Não há espaço suficiente no computador para o que está a tentar fazer. Se o computador parecer mesmo «preso» nesta situação, pode ter de apagar a linha de comando usando DELETE e a seguir apagar uma ou duas linhas de programa (com a intenção de as voltar a introduzir mais tarde) para dar espaço de manobra como por exemplo — CLEAR.
5	INPUT, PRINT AT	Out of screen (Fora do ecrã). Uma instrução (INPUT tentou gerar mais do que 23 linhas na parte inferior do ecrã. Também ocorre com PRINT AT 22,...
6	Aritmética	Number too big (Número demasiado grande). O resultado de cálculos aritméticos foi superior a 10^{38}
7	RETURN	RETURN without GOSUB (RETURN sem GOSUB). Existe um RETURN a mais do que o número de GOSUBs.
8	Operações com periféricos	End of file (Fim do Ficheiro).

Apêndice H: Lista de Códigos

Código	Situação	Significado
9	STOP	**Stop statement (Instrução STOP).** Depois desta situação **CONTINUE** não repete o **STOP**, mas continua na instrução seguinte.
A	SQR, LN, ASN, ACS, USR (com argumento de «string»)	**Invalid argument (Argumento inválido).** Argumento de uma função incorrecto, por qualquer motivo.
B	RUN, RANDOMIZE, POKE, DIM, GOTO, GOSUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR (com argumento numérico) ACESSO A MATRIZES.	**Integer out of range (Inteiro fora da gama permitida).** Quando um número inteiro é exigido, o número em ponto flutuante é arredondado ao inteiro mais próximo. Se este estiver fora da gama permitida, ocorrerá erro B. No acesso a matrizes, veja também erro 3.
C	VAL, VAL$	**Nonsense in basic (Disparate in BASIC).** O texto do argumento da «string» não forma uma expressão válida.
D	LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY. Também quando o computador pergunta scroll? e se «teclar» N, SPACE ou STOP.	**BREAK — CONT repeats.** Foi premiada a tecla **BREAK** durante uma operação periférica. O comportamento de **CONTINUE** depois desta mensagem é normal dado que repete a instrução. Comparar com a mensagem L.
E	READ	**Out of DATA (Fora da lista de dados).** Tentou utilizar a instrução **READ** depois do fim da lista **DATA**.
F	SAVE	**Invalid file name (Nome do ficheiro inválido).** Gravação com nome vazio ou com mais 1Ø caracteres.
G	Ao introduzir uma linha de programa	**No room for line (Sem espaço para a linha).** Não há espaço suficiente na memória para receber uma nova linha de programa.
H	INPUT	**STOP in INPUT (STOP num INPUT).** Um dado para **INPUT** começou por **STOP**, ou foi «teclado» para um **INPUT LINE**. Ao contrário da mensagem 9, depois da mensagem H faz com que **CONTINUE** se comporte normalmente, repetindo a instrução **INPUT**.

Apêndice H: Lista de Códigos

Código	Situação	Significado
I	FOR	**FOR without NEXT (FOR sem NEXT).** Houve um ciclo **FOR** para ser executado vez nenhuma. (Por exemplo n=1 to Ø) e a instrução **NEXT** correspondente não foi encontrada.
J	OPERAÇÃO COM PERIFÉRICOS	**Invalid I/O Device (Dispositivo I/O inválido).**
K	**INK, PAPER BORDER, FLASH, BRIGHT, INVERSE, OVER.** Também depois de um dos caracteres de controlo correspondentes	**Invalid color (Cor inválida).** O número especificado não é válido.
L	QUALQUER	**BREAK into program (BREAK no programa).** **BREAK** premido e detectado entre duas instruções. A linha e o número da instrução na mensagem referem-se à instrução antes de **BREAK** ser premido, mas **CONTINUE** vai para a instrução seguinte (permitindo qualquer salto) e por isso nenhuma instrução é repetida.
M	**CLEAR**; possível com **RUN**	**RAMTOP no good (RAMTOP não serve).** O número especificado para o **RAMTOP** é ou muito grande ou muito pequeno.
N	**RETURN, NEXT, CONTINUE**	**Statment lost (Instrução perdida).** Salto para uma instrução inexistente.
O	OPERAÇÃO COM PERIFÉRICOS	**Invalid stream (Corrente inválida).**
P	**FN**	**FN without DEF (FN sem DEF).** Função definida pelo utilizador.
Q	**FN**	**Parameter error (Erro nos parâmetros).** Número de argumentos errado, ou um deles é de tipo errado («string» em vez de número, ou vice-versa).
R	**VERIFY, LOAD** ou **MERGE**	**Tape loading error (Erro na leitura da fita).** Um ficheiro em fita foi encontrado, mas por qualquer motivo não pode ser lido ou verificado.

Apêndice I: Como Utilizar o Joystick

O seu TC 2048 possibilita-lhe a utilização de um Joystick «standard» com "ficha de 9 pinos tipo D" O seu TC 2048 possui um melhoramento que lhe permite uma fácil utilização do seu Joystick para quase todos os jogos ou programas existentes no mercado.

A sua utilização é bastante simples e deve proceder-se do seguinte modo:

1. Ligue o Joystick à ficha situada na parte lateral esquerda do seu computador TC 2048.
2. Ligue o cabo de antena e o cabo de alimentação ao seu computador. Coloque o botão ON/OFF, situado no lado direito, na posição ON.
3. O computador deverá mostrar o logotipo usual (© 1982 Sinclair Research Ltd.). Se esta mensagem não aparecer reinicie o processo.
4. O utilizador «BASIC» que queira utilizar o Joystick nos seus programas ou jogos deverá ler o Port 223 que é feito através da instrução **PRINT IN** 223 ou **LET** A = **IN** 223.
   A "posição" do Joystick ser-lhe-á indicada.
   Faça então **LOAD**.
5. Quando o computador lhe apresentar o menu de opções, introduza o «comando» para Joystick Kempston ou Quickshot.
6. No caso do utilizador «ASSEMBLER» deverá ler o Port **IN** A, (CF) e o valor do Joystick virá nos bits D Ø a D 4 sendo os bits D 5 a D 7 mantidos a zero.

Bons Jogos.

Notas

Notas

306995700

TIMEX

TMX PORTUGAL LDA.
QTA. DOS MEDRONHEIROS - LAZARIM
2825 MONTE DA CAPARICA - TEL. 295 20 69