OUTROS LIVROS NA ÁREA

Carvalho — Assembler para o TK 90X
Hurley — Programas para Jovens Programadores
TK-82 / TK-83 / TK-85 e CP 200
Hurley — TK 90X - Programas para Jovens Programadores

0-07-450226-3

CARVALHO BASIC AVANÇADO Para o TK 90X

BASIC AVANÇADO

Para o TK 90X

JOSÉ EDUARDO M. DE CARVALHO

COMPATÍVEL COM A LINHA ZX SPECTRUM ®

McGraw-Hill

BASIC Avançado para o TK 90 X

BASIC Avançado para o TK 90 X

José Eduardo Maluf de Carvalho

McGraw-Hill
São Paulo
Rua Tabapuã, 1.105, Itaim-Bibi
CEP 04533
(011) 881-8604 e (011) 881-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá • San Juan • Santiago

Auckland • Hamburg • Johannesburg • Kuala Lumpur • London • Montreal • New Delhi • Paris • Singapore • Sidney • Tokyo • Toronto

Coordenadora de Revisão: Daisy Pereira Daniel
Supervisor de Produção: Edson Sant'Anna
Capa: Lay-out: Cyro Giordano
Arte Final: H. Jaime Marques

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

	Carvalho, José Eduardo Maluf de.	
C324b	Basic para o TK 90X / José Eduardo Maluf de Carvalho. — São Paulo: McGraw-Hill, 1986.	
	1. BASIC (Linguagem de programação para computadores) 2. Microcomputadores — Programação 3. Programas de computador 4. TK 90X (Computador) — Programação I. Título.	
		17. CDD–651.8
		18. –001.6425
		18. –001.6424
85–2092		18. –001.642

Índices para catálogo sistemático:

1. BASIC: Linguagem de programação: Computadores: Processamento de dados 651.8 (17.) 001.6424 (18.)
2. Computadores: Programas: Processamento de dados 651.8 (17.) 001.642 (18.)
3. Microcomputadores: Programação: Processamento de dados 651.8 (17.) 001.642 (18.)
4. Programas: Computadores: Processamento de dados 651.8 (17.) 001.6425 (18.)
5. TK 90X: Computadores: Programação: Processamento de dados 651.8 (17.) 001.641 (18.0)

Dedicatória — Agradeço a todos os que me incentivaram e encorajaram a escrever este livro, em particular a Jorge dos Santos, mais particularmente ainda, a minha adorável esposa Vivian e meus queridos filhos Felipe e Marina, que se contentaram apenas em me olhar, enquanto eu escrevia.

O AUTOR

José Eduardo Maluf de Carvalho, 32 anos, arquiteto, atuando em Planejamento Urbano, na Secretaria Municipal de Planejamento de São Paulo e em projetos de edificações e construção civil em seu escritório próprio, sentiu necessidade de agilizar seu trabalho particular, nas tarefas mais repetitivas do arquiteto, desde uma consulta às leis de zoneamento, que regem o uso e aproveitamento do solo no Município de São Paulo, até as fases finais de detalhamento, quantificação e custo de um projeto de arquitetura.

Para isso, adquiriu, como um curioso, em março de 1982, um TK 82-C, e imediatamente conseguiu fazer deste pequeno micro uma poderosa ferramenta de trabalho.

Sentindo falta de uma melhor resolução gráfica, bem como de uma maior capacidade de armazenamento, partiu para um micro que foi um caso de "amor a primeira vista". Era o ZX Spectrum da Sinclair.

Desde setembro de 1983, o domínio sobre essa máquina foi num crescente cada vez maior, incluindo aí os periféricos ZX Microdrive, que nada mais é que um micro acionador de cartuchos (em vez de discos) e a Interface One (de compatibilização com o Microdrive, saída RS 232 C e rede local).

Trabalhando principalmente em linguagem Basic, com algumas rotinas em código de máquina, conseguiu elaborar um pacote de programas aplicativos para arquitetura, que vão desde a análise do zoneamento, otimizando a ocupação do terreno, definição do programa do projeto de arquitetura, elaboração da planta baixa, vegeta-

ção, detalhes construtivos, perspectivas, culminando com o orçamento e o cronograma da obra. Ou seja, não se sai do micro sem ter o projeto de arquitetura completo.

Hoje, após o lançamento do TK 90X, da Microdigital, totalmente compatível com o Spectrum, e até com algumas inovações e levando em consideração o ineditismo do seu trabalho, o autor fundou a empresa ARQUITRON INFORMÁTICA LTDA., de comércio e prestação de serviços em informática; juntamente com o Clube de Usuários do TK 90, sem no entanto abandonar suas atividades no escritório de arquitetura.

Mas nem por isso abandona os deliciosos momentos de lazer que tem com este grande parceiro de video games.

BEM-VINDOS AO MUNDO DO PRIMEIRO MICRO PESSOAL EM CORES LANÇADO NO MUNDO!!!

Sumário

Introdução *

Bem, leitor, aqui estamos nós juntos, nesse mundo cheio de surpresas agradáveis e, na maioria dos casos, quase que totalmente desconhecidas, deste pequeno notável, de potencial ilimitado, que é o TK 90X.

Provavelmente você é daquele tipo de pessoa que compra um computador como um curioso (foi assim que eu comecei), sem saber exatamente se ele é adequado às suas necessidades (se é que você sabe quais são elas).

Sem dúvida alguma é isso que está acontecendo nos grandes mercados mundiais, com o advento da microinformática, que introduziu na maior parte dos lares americanos, ingleses, italianos, franceses etc., sem saber exatamente como funcionam e para que servem, essas maravilhas que são os microcomputadores.

Você faz agendas telefônicas na sua residência, faz cadastramentos diversos, faz receitas de bolos e uma infinidade de aplicações domésticas. Só que já imaginou uma dona-de-casa, ao fazer uma receita, e de repente, esquecer um trecho dela, ter que ligar toda a parafernália eletrônica, após, evidentemente ter lavado as mãos, para uma simples consulta? Ou, então, se você quiser procurar um telefone de um amigo, se o esqueceu, ter de ligar o computador, ligar a tevê, ligar o cassete, carregar o programa, digitar o nome de quem você quer o telefone? É cansativo, muito mais que abrir uma simples cadernetinha de telefones!

* NOTA: Todos os programas deste livro foram digitados e testados num TK 90X, de 48K de memória RAM, n. 738/CQF/250/11015/6.

O mercado externo está exatamente nessa situação, em relação aos micros domésticos. Muitas firmas fabricantes de microcomputadores estão falindo, devido à guerra agressiva do capitalismo, que baixa até níveis insustentáveis os preços de suas mercadorias.

Nós estamos indo para o mesmo caminho. Computadores, custando os olhos da cara, são anunciados como "máquinas mágicas" que fazem o que nenhuma outra faz. Dizem que o que é bom para os EUA é bom para o Brasil. Será?

Os computadores mais baratos são vistos como máquinas apenas de "jogos eletrônicos", substituindo os videos games com grande vantagem. É o caso típico desta máquina, cuja potencialidade é totalmente desconhecida aqui no Brasil.

Apenas a título de exemplo, somente na Inglaterra, existem cerca de 30 000 escolas (!!!) que utilizam o micro ZX Spectrum da Sinclair, com a Interface One, que permite fazer uma rede local de computadores, interligando até 64 máquinas, e cada uma com até oito microdrives (faça as contas: 64 x 8 x 90K de capacidade de armazenamento de cada cartucho do microdrive resulta num total de cerca de 46 000 Kbytes, ou 46Mb – é memória de massa para ninguém colocar defeito). E, no mundo inteiro, são mais de cinco milhões de ZX Spectrum vendidos, em apenas três anos de existência.

Aqui, no Brasil, essa capacidade de se ligar em rede local é anunciada por quase todos os fabricantes com grande alarde, encarecendo o produto e servindo apenas a "grandes micros".

Eu, pelo que conheço deste micro, posso afirmar, categoricamente, que ele não é absolutamente uma máquina de jogos apenas. É muito, mas muito mais que isso. Ele é um dos microcomputadores de tecnologia de ponta mais desenvolvida do mundo inteiro, em relação ao seu microprocessador Z 80A, sendo o primeiro do mundo a apresentar alta resolução gráfica com cores!

Bem, mas vamos ver o que é e o que pode fazer esta máquina. É isto que eu pretendo neste livro. Com uma abordagem extremamente didática, a máxima que eu pude atingir, você verá diversos programas demonstrativos, jogos e utilitários que exploram a sua capacidade total da linguagem Basic.

E o que é a linguagem Basic? É a linguagem de alto nível mais difundida, mais versátil e mais fácil de ser entendida por nós, que a grande maioria de microcomputadores pessoais e domésticos utiliza por todo o mundo. Ela é relativamente nova, tendo sido criada em 1964, no Dartmouth College, nos EUA.

A teoria dessa linguagem Basic é exatamente aquilo que você leu no manual que acompanha o micro, faltando alguns truques e segredos aos quais se poderá ter acesso, no decorrer deste livro.

Quanto ao manual, como todo e qualquer manual de computador, ele não é suficientemente elaborado para nos ensinar a explorar ao máximo o potencial da máquina. Ele nos dá uma visão ampla do que pode ser feito com o micro, mas não se aprofunda em detalhes.

Eu espero que, ao comprar este livro, você já tenha explorado esse manual (mas que não esteja satisfeito apenas com o que leu nele), e que também tenha entendido como funciona a linguagem Basic e suas aplicações.

Você não vai encontrar informações repetidas neste livro, a partir do manual, ou seja, as informações contidas lá não estão neste livro. O nível de teoria é mais elevado do que no manual.

Portanto, você já deve estar com o micro ligado, deve ter-se exercitado nos exemplos do manual, provavelmente já fez o seu primeiro programa Basic, e está ansioso por saber mais. Vamos lá!

Na minha opinião, a computação, ciência como qualquer outra, é cerca de 30% de teoria e o restante de prática, aplicação. E, no manual, existe cerca de 90% dessa teoria e 10% da prática.

Portanto, para você realmente aprender Basic, não deve ter medo de pressionar qualquer tecla, não importa se o manual diz o contrário, porque o máximo que pode acontecer é um "crash", ou seja, o computador pára de entender Basic. Você deverá então desligá-lo e ligá-lo novamente, que ele continuará inteiro à sua disposição.

Vamos começar com uma pequena revisão da linguagem Basic.

Você já viu, no manual, as palavras-chave que a linguagem Basic utiliza. Quando o cursor, aquele pequeno quadradinho que aparece na tela da sua televisão, está com a letra K, significa que ele está aguardando uma palavra-chave de Basic. A partir disso, ele muda para a letra L, significando que agora está aguardando um valor literal. Basicamente, essa é a estrutura de uma linha de comandos em Basic: em modo imediato, você diz a palavra-chave, em seguida o seu operando, ou seja, aquilo sobre o qual ele vai trabalhar e, finalmente, um indicador de final de comando, para que ele execute a sua ordem. É por isso que nós costumamos dizer que o computador é uma "máquina burra", pois, além de dar-lhe uma ordem, você deve mandá-lo executar aquela ordem!

A diferença entre esse modo imediato, onde nós usamos o micro quase que como uma calculadora comum, e o modo programado é um indicador sobre a ordem dos comandos, ou seja, o número do comando, ou melhor, o número da linha de programa em Basic.

Portanto, um programa em Basic é uma seqüência ordenada de instruções ou comandos nessa linguagem que vai dizer ao computador o que ele deve fazer com os nossos operandos, objetivando o nosso resultado.

Não se preocupe, pois a máquina coloca a numeração das linhas em ordem crescente automaticamente, apagando as coincidentes.

A linha de programa pode ter mais de um comando, separados pelo indicador de final de comando (e não final de linha), que é o símbolo : (dois pontos).

Os números de linhas aceitáveis pelo TK 90X vão de 1 até 9999, inclusive. Convém espaçarmos esses números, para futuras inserções, quando das correções do programa (ninguém nunca ficou satisfeito com a primeira versão do seu programa).

Com o programa feito, então você deve ordenar ao computador para que ele execute aquela seqüência de comandos.

Esses comandos, basicamente, se dividem em:

1 - Comandos de saída de dados;
2 - Comandos de entrada de dados;
3 - Comandos de atribuição;
4 - Comandos de controle e edição;
5 - Comandos de desvios de processamento;
6 - Comandos de armazenamento;
7 - Comandos condicionais e
8 - Comandos gráficos.

Procure, a título de exercício, colocar numa folha à parte os diversos comandos de cada grupo citado, a partir do manual.

Para que o computador possa entender as nossas ordens em uma linguagem de alto nível, existe dentro dele um "dicionário", ou seja, um interpretador que traduz as nossas ordens em instruções que o computador possa entender, em linguagem binária, de dígitos 0 e 1, que é a linguagem do computador, dificílima de ser entendida por nós, mas com duas grandes vantagens sobre a linguagem Basic: economia de memória e maior rapidez de processamento (por não precisar do interpretador).

Portanto, a maneira usual de um computador executar um programa em Basic é interpretá-lo, comando por comando. A outra alternativa é compilar este programa, ou seja, através de um programa escrito em linguagem de máquina, chamado *compilador*, o programa em linguagem de alto nível é totalmente traduzido para a linguagem de máquina, para que depois seja processado de uma vez, em seqüência, ganhando velocidade de execução.

Portanto, arregace as mangas, ligue o seu pequeno notável e mãos à obra!

Vamos, antes de iniciar o conteúdo propriamente dito deste livro, exercitar alguns programas em Basic.

Se você já carregou toda a fita de demonstração que acompanha o micro, já deve estar familiarizado com o teclado. Se ainda não o fez, experimente, pois as lições sobre como manipular o teclado são muito boas.

Vamos então ao treino de programas.

Experimente digitar o programa nº 1, a seguir.

Digite os algarismos 10 e repare que o cursor está em K.

Pressione então a tecla P, para imprimir a palavra-chave daquela tecla, PRINT. O cursor passou para L. Pressione SYMBOL SHIFT e junto a tecla P, outra vez. Agora foi impressa a aspa e o cursor continua em L. Digite qualquer nome, não esquecendo que para se obter letras maiúsculas deve-se pressionar CAPS SHIFT junto com a letra desejada. Não se esqueça de fechar a aspa, mas, caso o tenha esquecido, o micro o avisará que ele entendeu aquela linha de programa (evidentemente, após você ter digitado ENTER, para avisá-lo de que a sua linha terminou ali) até o sinal de interrogação, e que, dali para a frente, algo está estranho para ele.

A linha 20 significa o comando SOUND 0,5 segundos e tonalidade 10.

A linha 30 faz o processamento do programa ser desviado para a linha 10 novamente, através do comando GOTO 10.

```
  10 PRINT "Jose Eduardo "
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

Agora altere esse mesmo programa, para o seguinte:

```
  10 PRINT "Jose Eduardo ";: REM
 repare no ponto e virgula
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

Observe a linha 10, onde foi introduzido outro sinal:

```
  10 PRINT "Jose Eduardo ",: REM
 repare na virgula
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

Mude novamente a linha 10 para:

```
  10 PRINT "Jose Eduardo ",,,: R
EM repare nas virgulas
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

Você sabe "mudar" essas linhas, não sabe?

Se não, vamos lá.

Primeiramente digite, em modo direto, qualquer palavra-chave.

Suponhamos que não era essa palavra que você queria digitar. Você, então, deverá deletá-la. Para isso, primeiramente, deve pressionar CAPS SHIFT e, em seguida, a tecla 0, onde está escrito em branco a palavra DELETE. Pronto, a palavra-chave foi apagada.

Como fazer isso em modo programado?

Você digita uma linha de programa, e em seguida avisa ao micro que sua linha terminou, através do comando ENTER. Se a linha estiver absolutamente com a sintaxe correta, ela é aceita pelo micro, e passa, para a parte superior da tela, o campo de programa. Para corrigi-la, primeiramente coloque o cursor indicador de linha corrente, aquele símbolo ">", que deve estar imediatamente após a última linha digitada, na linha que se quer corrigir, através da tecla 7, que tem sobre ela uma seta apontando para cima. É uma das quatro teclas cursoras do micro. Quando o cursor chegar até a sua linha, você então deve trazê-la para a área de edição, que é a área inferior da tela, a fim de ser editada. Você consegue isso pressionando CAPS SHIFT e a tecla 1, que tem escrito sobre ela a palavra-chave EDIT.

Pronto, a linha de programa está na parte inferior da tela. Agora, com outra tecla cursora, a 8, leve o cursor até uma posição após o que você vai "deletar". Novamente com CAPS SHIFT e a tecla 0 você apaga o erro, e está pronto para introduzir a alteração.

Agora, altere novamente a linha 10 do programa nº 1:

```
  10 PRINT "Jose Eduardo "'': RE
M repare nos apostrofes
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

E outra linha 10:

```
  10 PRINT "Jose Eduardo "': REM
 repare no apostrofe
  20 BEEP .5,10: REM este e o co
mando SOUND do TK
  30 GO TO 10
```

Agora, altere as linhas 20 e 30 para:

```
  10 PRINT "Jose Eduardo "'': RE
M repare nos apostrofes
  20 BEEP .5,RND*69: REM este e
 o comando SOUND do TK
  30 GO TO 10
```

Quais são as diferenças?

Com a função RND na linha 20, o computador passa a gerar diversos tons de notas musicais, na faixa de 0 até 69, aleatoriamente, ou randomicamente.

Na linha 30, a introdução do comando RUN mudou todo o programa. Isto porque este comando faz a inicialização da memória, ou seja, limpa tudo o que lá existe, inclusive alguma posição de impressão, para começar tudo outra vez.

Passemos agora para o programa nº 2, um pouco mais avançado:

```
  10 INPUT "Qual e o seu nome ";
a$
  20 INPUT "E a sua idade ";i
  30 PRINT "Ola ";a$
  40 IF i<=30 THEN PRINT "Voce a
inda e jovem, pois tem so ";i;"a
nos de idade"
  50 IF i>30 THEN PRINT "Cuidado
 - voce ja passou dos 30"
```

```
60 LET c=i+i
70 PRINT "Daqui ha ";i;" anos,
"oce tera ";c;" anos"
80 PRINT
90 GO TO 10
```

Na linha 10, o computador vai imprimir a mensagem entre aspas e esperar que você digite um valor alfanumérico (observe as aspas, no canto inferior esquerdo da tela – é um lembrete do micro para você), para que ele faça a associação dessa *série de caracteres* à variável alfanumérica de nome a$.

Na linha 20, o mesmo raciocínio para uma variável numérica.

Na linha 30, ele imprime a *série* "OLA" e a variável alfanumérica a$.

Na linha 40, existe uma condição para que seja executada a alternativa de impressão: se o valor numérico digitado na linha 20 satisfizer essa condição, ou seja, a condição passa a ser verdadeira, então a alternativa será executada.

Na linha 50, outra condição, de idêntico raciocínio da linha anterior.

Na linha 60, foi feita uma atribuição de valores. Atribuiu-se à variável de nome c o dobro do valor digitado na linha 20, através da operação aritmética da soma.

Na linha 70, uma simples impressão de *séries* junto com variáveis.

Na linha 80, o micro pulará três linhas, antes de executar o comando da linha 90, que desvia o processamento do computador para a linha 10, ou a primeira linha do programa, mas não faz a inicialização da memória, como o comando RUN.

Agora, faça exatamente isso – a inicialização da memória, alterando a linha 90 e introduzindo a linha 100.

```
  10 INPUT "Qual e o seu nome ";
a$
  20 INPUT "E a sua idade ";i
  30 PRINT "Ola ";a$
  40 IF i<=30 THEN PRINT "Voce a
inda e jovem, pois tem so ";i;"a
nos de idade"
  50 IF i>30 THEN PRINT "Cuidado
 - voce ja passou dos 30"
  60 LET c=i+i
  70 PRINT "Daqui ha ";i;" anos,
 voce tera ";c;" anos"
  80 PRINT ""
  90 RUN
```

Experimente rodar esta versão do programa sem a linha 90 para ver o que acontece. Quase não dá tempo de se ver o que está na tela.

Essa é a função da pausa. Aguarde um instante.

Agora, vamos "formatar" os dados de saída, ou seja, vamos alterar a posição de impressão das mensagens na tela, através da tabulação nas linhas 30 e 70, e a mudança de posição das linhas 40 e 50, alterando a posição relativa a uma coluna e a uma linha.

```
  10 INPUT "Qual e o seu nome ";
a$
  20 INPUT "E a sua idade ";i
  30 PRINT TAB 15;"Ola ";a$
  40 IF i<=30 THEN PRINT AT 5,10
;"Voce ainda e jovem, pois tem s
o ";i;"anos de idade"
  50 IF i>30 THEN PRINT AT 5,10;
"Cuidado - voce ja passou dos 30
"
  60 LET c=i+i
  70 PRINT TAB 10;"Daqui ha ";i;
" anos, voce tera ";c;" anos"
  80 PRINT ""
  90 PAUSE 100
 100 RUN
```

Para que você se acostume com as posições de impressão, introduza no computador o programa nº 3, a seguir:

```
  10 INPUT "Cor da tinta ";t
  20 INPUT "Cor do papel ";p
  30 INPUT "Cor da borda ";b
  40 BORDER b: PAPER p: INK t: C
LS
  50 INPUT "Qual coluna para esc
rever ";c
  60 INPUT "E qual linha ";l
  70 INPUT "Qual o seu nome ";n$
  80 PRINT AT l,c;n$
  90 GO TO 80
```

Observou o que acontece com esse programa?

A linha 90 faz o desvio do processamento do programa para a linha 80, que possui uma mensagem de impressão sempre na mesma posição, o que dá a impressão de que o micro parou de trabalhar.

Procure alterar, como exercício, os valores dessa linha, incrementando-os ou decrementando-os.

Mas, e se você digitar um número de linha ou de coluna maior que o permitido? Certamente o micro lhe devolverá uma mensagem de erro, não aceitando esses valores.

Digite essa nova versão do mesmo programa:

```
10 INPUT "Cor da tinta ";t
20 INPUT "Cor do papel ";p
30 INPUT "Cor da borda ";b
40 BORDER b: PAPER p: INK t: C
LS
50 INPUT "Qual coluna para esc
rever ";c
55 IF c>31 THEN GO TO 50
60 INPUT "E qual linha ";l
65 IF l>21 THEN GO TO 60
70 INPUT "Qual o seu nome ";n$
80 PRINT AT l,c;n$
90 GO TO 10
```

Agora, passe para o programa nº 4. Digite-o e rode-o.

```
10 BORDER 0: PAPER 0: INK 5: B
RIGHT 1: CLS
20 INPUT "Digite um numero de
1 a 10 ";a
30 INPUT "Digite outro ";b
40 IF INT (a/2)=a/2 THEN PRINT
"O numero ";a;" e par"
50 IF b>=a THEN PRINT "O segun
do e maior que o primeiro"
60 PRINT a,b
70 PAUSE 100
80 RUN
```

Note, na linha 40, a condição que examina se o número é par ou não.

Experimente alterar a linha 80, de modo que ela não apague o que já existe na tela.

Agora veja, no programa nº 5, a função RND sendo executada infinitas vezes. Maiores detalhes sobre essa função e esses números "aleatórios" serão dados mais tarde, em outro capítulo. Por enquanto, apenas treine a linguagem Basic.

```
10 LET l=RND
20 PRINT l
30 GO TO 10
```

Experimente esta nova versão do mesmo programa, só que mais rápida.

```
10 PRINT RND
20 GO TO 10
```


E altere a linha 20, de modo que haja inicialização da memória:

```
10 PRINT RND
20 RUN
```

No próximo programa, de nº 6, está uma sugestão para se fazer cartões de cinco dezenas na LOTO, com os números que o micro fornecerá. Experimente, quem sabe ele tem sorte!

Note a presença de outro comando, que denominamos LOOP através dos comandos FOR para abri-lo e NEXT para fechá-lo:

```
10 FOR f=1 TO 5
20 LET l=INT (RND*99)+1
30 PRINT l,
40 NEXT f
```

Experimente o programa nº 7, que varia tons de notas musicais:

```
10 FOR f=1 TO 25: BEEP .05,f:
NEXT f
```

Altere-o, introduzindo a linha 20, para que ele seja processado infinitas vezes.

```
10 FOR f=1 TO 25: BEEP .05,f:
NEXT f
20 RUN
```

Altere-o novamente, com a introdução de outras tonalidades:

```
10 FOR f=1 TO 25: BEEP .05,50-
BEEP .05,f: NEXT f
20 RUN
```

E outras:

```
10 FOR f=1 TO 25: BEEP .05,-f:
BEEP .05,50-f: BEEP .05,f: NEXT
20 RUN
```

Passemos então para o programa nº 8, que mostra o efeito dos comandos FLASH e INVERSE separadamente e depois, na outra versão, esses mesmos comandos juntos e finalmente o comando BRIGHT.

Note que nas três versões você deverá pressionar qualquer tecla, para que ele passe para a linha seguinte. O modo mais fácil é pressionando-se somente ENTER, ou seja, uma *série* vazia ou nula.

Pressionando-se "s" o programa será interrompido.

Se não houvesse essa condição das linhas 20 e 40, como interromper um programa que aceita até a palavra STOP como *série,* ou considera BREAK como um espaço? Simples: basta apagar uma aspa da parte inferior da tela e, então, através de SYMBOL SHIFT e tecla A, acionar o comando STOP, que agora funciona.

```
  10 PRINT INVERSE 1;AT 0,0;"TK
90X"
  20 INPUT a$: IF a$="s" THEN ST
OP
  30 PRINT INVERSE 0;AT 0,0;"TK
90X"
  40 INPUT a$: IF a$="s" THEN ST
OP
  50 RUN
```

```
  10 PRINT FLASH 1;AT 0,0;"TK 90
X"
  20 INPUT a$: IF a$="s" THEN ST
OP
  30 PRINT FLASH 0;AT 0,0;"TK 90
X"
  40 INPUT a$: IF a$="s" THEN ST
OP
  50 RUN
```

```
  10 PRINT FLASH 1; INVERSE 1;AT
 0,0;"TK 90X"
  20 INPUT a$: IF a$="s" THEN ST
OP
  30 PRINT FLASH 0; INVERSE 0;AT
 0,0;"TK 90X"
  40 INPUT a$: IF a$="s" THEN ST
OP
  50 RUN
```



```
  10 PRINT BRIGHT 1; FLASH 1; IN
VERSE 1;AT 0,0;"TK 90X"
  20 INPUT a$: IF a$="s" THEN ST
OP
  30 PRINT BRIGHT 0; FLASH 0; IN
VERSE 0;AT 0,0;"TK 90X"
  40 INPUT a$: IF a$="s" THEN ST
OP
  50 RUN
```

Agora experimente o programa nº 9, que utiliza dois loops que costumamos chamar de "aninhados", ou seja, um tem obrigatoriamente de estar totalmente contido dentro do outro.

```
  10 INPUT "Quantas colunas";c
  20 INPUT "Quantas linhas";l
┌─30 FOR f=0 TO l
│ ┌40 FOR g=0 TO c
│ │50 PRINT AT f,g;"TK 90X"
│ └60 NEXT g
└─70 NEXT f
```

Digite esta nova versão do programa nº 9, onde o micro vai perguntar com quantas linhas e colunas você quer que ele trabalhe e, além disso, mostra uma nova utilização do comando OVER 1, ou seja, imprimindo na mesma posição, a mesma *série,* e resultando então em nada, ou seja, apagando aquela mensagem impressa. Verifique a velocidade da execução deste comando, pois, comparativamente com a versão anterior, a mudança de velocidade é quase que imperceptível.

```
  10 INPUT "Quantas colunas";c
  20 INPUT "Quantas linhas";l
  30 FOR f=0 TO l
  40 FOR g=0 TO c
  50 PRINT AT f,g;"TK 90X"; OVER
1;AT f,g;"TK 90X"
  60 NEXT g
  70 NEXT f
```

Neste programa, de nº 10, utilizamos uma matriz numérica de cinco valores. Digite-o para perceber, através das linhas 50 em diante, que aqueles valores digitados no início agora estão guardados na memória do micro, para que possamos acessá-los quando quisermos, a menos que se digite um comando RUN (lembra-se da inicialização da memória?), ou se desligue o micro, ou mesmo que o processamento passe outra vez pelo comando DIM, apagando então os conteúdos existentes e reservando espaço para novos conteúdos.

```
10 DIM d(5)
20 FOR f=1 TO 5
30 INPUT "Digite um numero ";d
f)
40 NEXT f: PRINT "Os numeros s
ao "
50 FOR m=1 TO 5
60 PRINT d(m)
70 NEXT m
```

No programa seguinte, também sobre matrizes, de nº 11, criamos uma matriz alfanumérica de três elementos de comprimento 5. Repare, ao rodar o programa, que qualquer comprimento de nome digitado superior a 5 é desprezado pelo micro.

```
10 DIM d(3)
20 DIM d$(3,100)
30 FOR b=1 TO 3
40 INPUT "Digite um nome ";d$(
b)
50 NEXT b
60 PRINT #1;AT 1,0;"Pressione
qualquer tecla ": PAUSE 0
70 PRINT "Os nomes sao"
80 FOR k=1 TO 3
90 PRINT d$(k)
100 NEXT k
```

Altere, então, o comprimento dos elementos dessa matriz, conforme linha 20, da nova versão do mesmo programa.

Repare uma novidade na linha 60: ela simplesmente imprime, na parte inferior da tela, qualquer mensagem que queiramos. (Maiores detalhes sobre fluxos e canais, em outro capítulo do livro.)

Veja também que utilizamos PAUSE 0, que significa pausa por tempo infinito (nem tanto), até que digitemos alguma tecla.

```
10 DIM d(3)
20 DIM d$(3,100)
30 FOR b=1 TO 3
40 INPUT "Digite um nome ";d$(
b)
50 NEXT b
60 PRINT #1;AT 1,0;"Pressione
qualquer tecla ": PAUSE 0
70 PRINT "Os nomes sao"
80 FOR f=1 TO 3
90 PRINT d$(f)
100 NEXT f
```


E, para terminar esta introdução, o programa de nº 12, que mostra o funcionamento do comando INKEY$, que lê o teclado.

```
10 PRINT #1;AT 1,0;"Pressione
qualquer tecla"
20 PRINT INKEY$;
30 GO TO 20
```

Bem, espero que, com esta breve introdução e revisão da linguagem Basic, você tenha adquirido uma maior intimidade com o seu micro, e percebido que ele obedece sem reclamar a qualquer ordem que lhe é dada (desde que ele a entenda!).

É assim que se aprende computação: praticando, exercitando, sem medo de errar. No decorrer deste livro, procure entender mentalmente a execução dos problemas que surgem, antes de digitá-los. Facilita demais a tarefa de aprender.

E, se você quiser algo mais que a linguagem Basic, esta editora lançará um livro sobre Linguagem de Máquina para este micro, desde mesmo autor, com exemplos incríveis sobre a total capacidade desta pequena máquina maravilhosa.

Capítulo I

O que é o TK 90X

É gratificante, ao se comprar um livro, abri-lo em uma página qualquer, ler um trecho, retornar, ler o capítulo anterior, ler o último capítulo. . . e descobrir logo quem é o assassino. Com os computadores é um pouco diferente – como são as máquinas mais lógicas que existem, qualquer pessoa que queira entendê-las deve começar, logicamente, pelo princípio.

Podemos descrever qualquer sistema de computação, sob três aspectos diferentes.

O primeiro aspecto é relativo à descrição sucinta do sistema de um modo geral, a partir de sua CPU e seus periféricos.

O segundo aspecto refere-se ao hardware (partes físicas) do sistema propriamente dito.

O terceiro aspecto é sobre o desempenho lógico do micro.

O SISTEMA

O microcomputador TK 90X é uma pequena caixa preta, de plástico, medindo 236mm de largura, por 146mm de profundidade e 44mm de altura. Na superfície horizontal superior estão dispostas as quarenta teclas de um composto de silicone, com as legendas impressas pelo processo chamado *hot-stamping,* ou seja, *silk screen* a quente, para maior durabilidade, que compõe o seu teclado.

Na traseira da caixa estão dispostos, da esquerda para a direita, o plug que conecta com a entrada da tevê (plug da antena externa); o soquete de entrada que conecta com a saída do provador cassete; o soquete de saída que conecta com a entrada do gravador cassete; o plug de conexão para joysticks tipo Atari; e o slot de expansão da máquina, onde se conectam diversos periféricos de entrada/saída, tais como uma impressora (ZX Printer, Alphacom 32 ou TK 5005), uma interface RS 232, a interface de compatibilização com o microdrive, e finalmente o plug de voltagem.

A placa de circuitos, incluindo o microprocessador Z80 e outros componentes eletrônicos, encontra-se dentro dessa caixa preta, tomando toda a sua dimensão, e é separada do teclado por dois cabos de ligação.

Esse sistema pode ser visto esquematicamente na Figura 1.

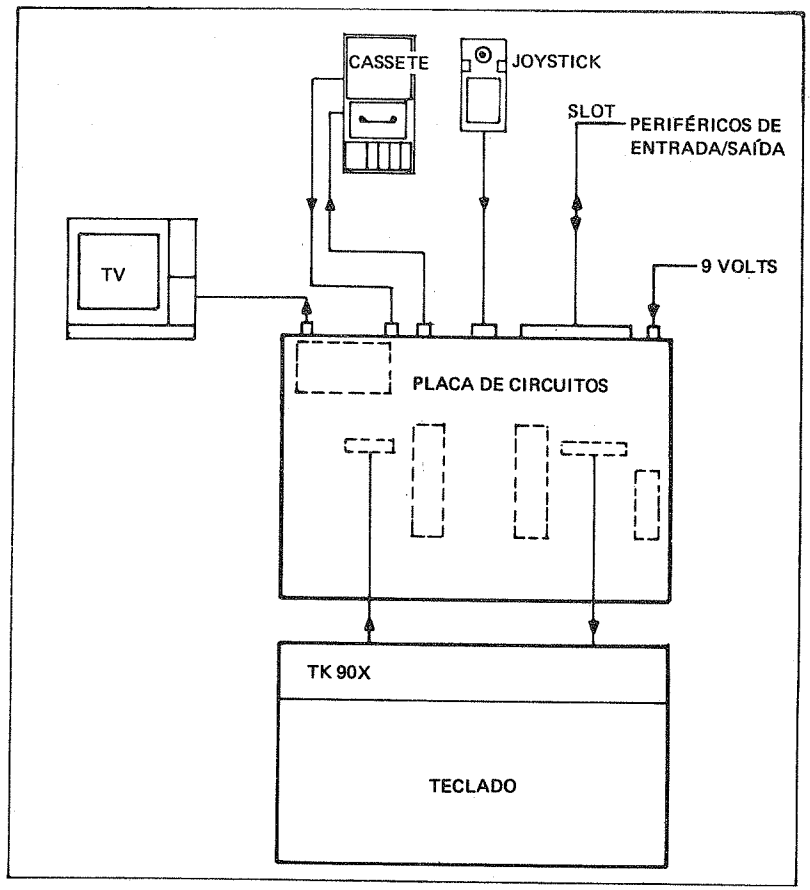


Figura 1 - O sistema do micro TK 90X (sem escala de desenho).

O HARDWARE (os componentes físicos)

A placa de circuitos do TK 90X pode facilmente ser acessada, retirando-se os quatro parafusos inferiores que prendem a tampa, onde está instalado o teclado, à parte inferior da caixa, onde se encontra a placa de circuitos. Se o seu micro estiver na garantia não convém abri-lo, pois destruirá o lacre da garantia, anulando-a.

Se a curiosidade for muito grande, retire os parafusos, segurando as duas partes da caixa, para que não se separem e, com o micro em posição de funcionamento, levante o teclado muito vagarosamente, procurando não desconectar os seus dois cabos de ligação com a placa, pois são muito frágeis e suas pontas se desgastam muito facilmente. Estes cabos podem ser desconectados, puxando-os suavemente dos seus soquetes, mas procure não fazê-lo: use duas hastes (canetas) de igual comprimento para sustentar a tampa, tal qual o capô de um carro.

Os componentes principais da placa de circuito estão desenhados na Figura 2.

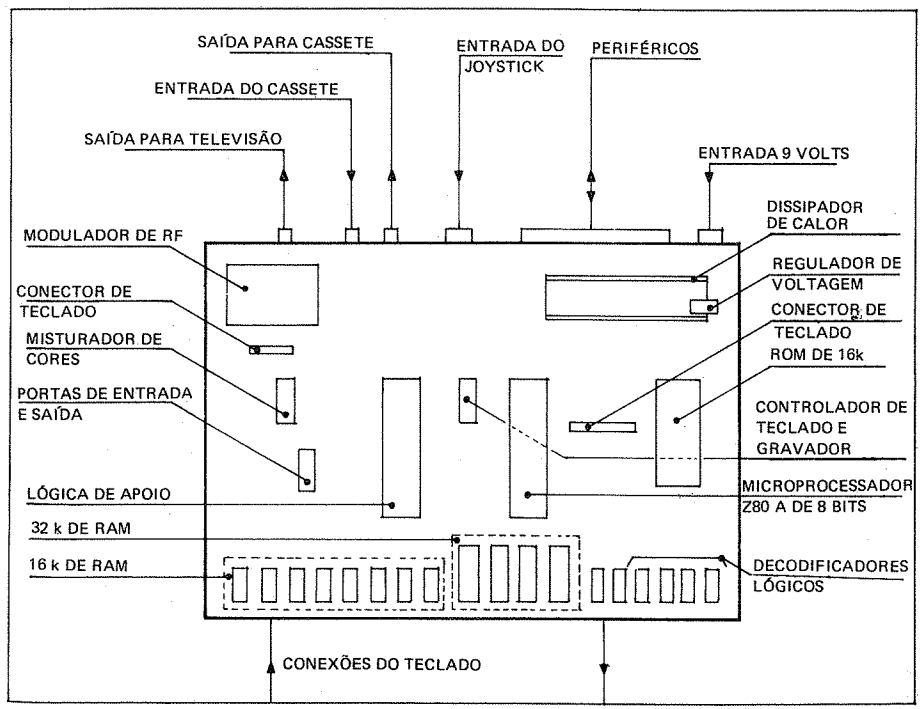


Figura 2 - Placa de circuitos eletrônicos e componentes principais do TK 90X.

Cada um desses componentes principais será descrito a seguir:

– O MICROPROCESSADOR Z 80

Este chip, muito conhecido entre os aficionados da informática, visto que é o coração de muitos sistemas que existem por aí, é o mais importante do TK 90X e o mais sofisticado (também o mais complicado, em termos de código de máquina) dos microprocessadores de 8 bits.

Como microprocessador, tem capacidade para trabalhar como um "computador", de uma maneira genérica, capaz de executar instruções armazenadas (programa). Os programas para o TK 90X, quaisquer que sejam eles, serão sempre convertidos internamente em instruções em código de máquina do Z 80.

No TK 90X, o "clock" (relógio medidor de tempo interno de execução de instruções) é de 3,75 MHZ e a esta velocidade ele é capaz de processar cerca de 900 000 das mais simples instruções em código de máquina. É interessante notar que, a partir do momento em que se liga a fonta de energia do micro, o seu microprocessador imediatamente começa a funcionar, apesar de não mostrar qualquer resultado. Ele simplesmente fica aguardando uma ordem.

– A ROM DE 16 KBYTES (ROM = READ ONLY MEMORY – OU MEMÓRIA APENAS DE LEITURA)

O programa em código de máquina que é normalmente executado pelo microprocessador Z 80 está contido num único chip de ROM, que manipula 128 kbits, ou 16 kbytes de informação.

Nestes 16 k de "programa monitor" do TK 90X, os aproximadamente primeiros 7 k estão dedicados ao sistema operacional, os 8 k seguintes ao interpretador Basic e o k restante, ao gerador de caracteres.

– A RAM (RANDOM ACCESS MEMORY – MEMÓRIA DE ACESSO RANDÔMICO)

Na versão standard de 16 k do TK 90X, existem oito chips de memória RAM de 2 k cada, ou 16 kbits.

Três destes 8 chips formam o "arquivo da imagem" ou mapeamento da memória da tela, e podem normalmente ser usados somente com este propósito. O quarto chip é dedicado à manipulação dos bytes dos atributos (cores de papel, tinta,

flash, bright etc.) das 768 posições de caracteres da tela, e as variáveis do sistema. Um pouco mais de 8 k está disponível ao usuário na versão de 16 k.

Para a RAM total de 48 k, são adicionados mais quatro chips de 8 k cada, ou 64 kbits. Nesta memória, estão disponíveis para o usuário um pouco mais de 40 k.

– A LA (LÓGICA DE APOIO)

Este pode ser considerado como sendo um "grande chip feito de diversos pequenos chips". É um dos chips conhecidos como "customizado", ou seja, só serve para esse computador. Ele atua como um "centro de comunicação", verificando se tudo o que o microprocessador requer ou ordena está sendo efetuado; ele também "lê" a memória para ver em que consiste a imagem da tevê, e envia os sinais apropriados para a interface da tevê.

– MISTURADOR DE CORES

Este chip recebe as informações sobre cores da LA e as converte para os sinais apropriados a serem enviados ao modulador VHF. O sinal produzido pelo modulador é ajustado para o canal 3.

Em adição a estes componentes principais existem ainda: a interface que produz som no alto-falante da tevê; a interface para o gravador cassete; o dissipador de calor; o regulador de voltagem, que converte a tensão de entrada em 5 volts absolutos, sem oscilação; alguns decodificadores de endereços; e outros componentes menores.

A LÓGICA DO SISTEMA

Neste aspecto, todas as ligações entre os vários componentes internos do microcomputador são consideradas. Estas ligações possuem uma existência real – são "caminhos" impressos na placa de circuitos ou, eventualmente, fios de ligação.

O microprocessador Z 80 pode gerar um endereçamento individual de até 65536 endereços diferentes de memória (65536/1024 = 64 kb), de cada vez. No TK 90X, versão 16 k, apenas os endereços 0 a 32767 (32 kb, sendo 16 k de ROM e 16 k de RAM) estão disponíveis. Na versão de 48 K, todos os endereços são utilizados.

No TK 90X, os endereços são produzidos na forma de 16 sinais binários (bits - algarismos 0 e 1). O endereço 0 é, portanto, 00000000 00000000, e o endereço 65535 é 11111111 11111111. Isto porque o computador trabalha com o sistema numérico binário, com os algarismos 0 e 1.

Não entendeu? Vamos ver:

No nosso sistema decimal, temos que:

1985

$5x10^0 = 5 \times 1 = 5$
$8x10^1 = 8 \times 10 = 80$ +
$9x10^2 = 9 \times 100 = 900$
$1x10^3 = 1 \times 1000 = 1000$
1985

ou

4096

$6x10^0 = 6 \times 1 = 6$
$9x10^1 = 9 \times 10 = 90$ +
$0x10^2 = 0 \times 100 = 0$
$4x10^3 = 4 \times 1000 = 4000$
4096

Analogamente, no sistema binário, que utiliza somente os algarismos 0 e 1, ou bits, temos para 1 byte (conjunto de 8 bits):

1111 1111

$1x2^0 = 1 \times 1 = 1$
$1x2^1 = 1 \times 2 = 2$
$1x2^2 = 1 \times 4 = 4$
$1x2^3 = 1 \times 8 = 8$ +
$1x2^4 = 1 \times 16 = 16$
$1x2^5 = 1 \times 32 = 32$
$1x2^6 = 1 \times 64 = 64$
$1x2^7 = 1 \times 128 = 128$
255

Ou seja, 256 valores diferentes para um byte, que começam em 0 e terminam em 255. Portanto, $2^8 = 256$

$256 * 256 = 2^8 * 2^8 = 2^{16} = 65536$ ou 11111111 11111111

Da mesma forma, no sistema hexadecimal, que utiliza os algarismos 0 a 9 para valores decimais de 0 a 9, e letras de A até F para valores de 10 a 15, temos:

HEXADECIMAL	DECIMAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Portanto, para o valor decimal 255, temos em hexa FF e em binário 11111111

FF

$15x16^0 = 15$ +
$15x16^1 = 240$
255

O programa a seguir faz as conversões automaticamente entre as bases numéricas decimal, binária e hexadecimal.

```
  0 REM
  1 REM ************************
  2 REM *Programa de conversao*
  3 REM *  de bases numericas  *
  4 REM ************************
 10 CLS : PRINT "Este programa
faz a conversao dequalquer base
<=16 para qualqueroutra base <=1
6"
 15 INPUT "Digite a base anteri
or ";x$
 20 LET e=0
 25 IF VAL x$>16 OR x$="" THEN
GO TO 10
 30 GO SUB 185
 35 LET b=n
 40 IF n<2 OR n>16 THEN GO TO 1
```

```
0
   45 INPUT "Qual e o numero ";x$
   50 IF x$="" THEN GO TO 45
   55 GO SUB 205
   60 IF e=1 THEN PRINT "Erro ":
LET e=0: GO TO 45
   65 LET n1=n
   70 PRINT x$;" na base 10 e ";n
1
   75 IF n1<1000000 THEN GO TO 85
   80 PRINT "O numero na base 10
e >= 1000000podendo por isso oco
rrer erros"
   85 INPUT "Qual e a nova base "
;x$
   90 IF x$="" THEN GO TO 85
   95 GO SUB 185
  100 LET b1=n: IF n<2 OR n>16 TH
EN GO TO 85
  105 LET b$=""
  110 LET v=INT (n1/b1)
  115 LET r=n1-v*b1
  120 IF r>9 THEN GO TO 140
  125 LET b$=b$+CHR$ (r+48)
  130 LET n1=v: IF v=0 THEN GO TO
 145
  135 GO TO 110
  140 LET r=r+55: LET b$=b$+CHR$
(r): LET n1=v: IF v<>0 THEN GO T
O 110
  145 PRINT "O numero na base ";b
1;" e ";
  150 FOR j=LEN b$ TO 1 STEP -1
  155 PRINT b$(j);: NEXT j
  160 PRINT
  165 INPUT "Mais numeros (s/n) "
;x$
  170 IF x$="s" THEN GO TO 10
  175 IF x$="n" THEN NEW
  180 GO TO 165
  185 LET n=0
  190 FOR j=1 TO LEN (x$): LET d=
CODE x$(j)
  195 LET n=n*10+d-48: NEXT j
  200 RETURN
  205 LET n=0
  210 FOR j=1 TO LEN x$: LET d=CO
DE x$(j)
  215 IF d>47 AND d<58 THEN LET d
=d-48: GO TO 230
  220 IF d>64 AND d<71 THEN LET d
=d-55: GO TO 230
  225 LET e=1: RETURN
  230 IF d>=b THEN LET e=1: RETUR
N
  235 LET n=n*b+d
  240 NEXT j
  245 RETURN
```


Os endereços são gerados pelo microprocessador Z 80 e transportados pelo computador através das Vias de Endereçamento (Address Bus). São no total 16 vias, nas quais um endereço pode ser especificado considerando-se uma via possuindo uma "alta" voltagem (5 volts), ou "baixa" voltagem (0 volts), ou ainda, respectivamente, 1, para alta voltagem e 0 para baixa voltagem.

Enquanto as vias de endereço possuem 16 contatos, as de dados (Data Bus) do TK 90X possuem somente oito contatos, pelo fato de o seu microprocessador ser de 8 bits. Portanto, qualquer dado que estiver sendo manipulado pelo sistema, seja ele uma instrução em código de máquina, ou um valor qualquer, ou uma instrução em Basic, deve ser considerado como sendo um número decimal na faixa de 0 a 255 (valor máximo de 1 byte), ou na faixa binária de 00000000 até 11111111.

A figura abaixo mostra esquematicamente como as vias de endereçamento e as vias de dados são interligadas aos outros componentes do TK 90X.

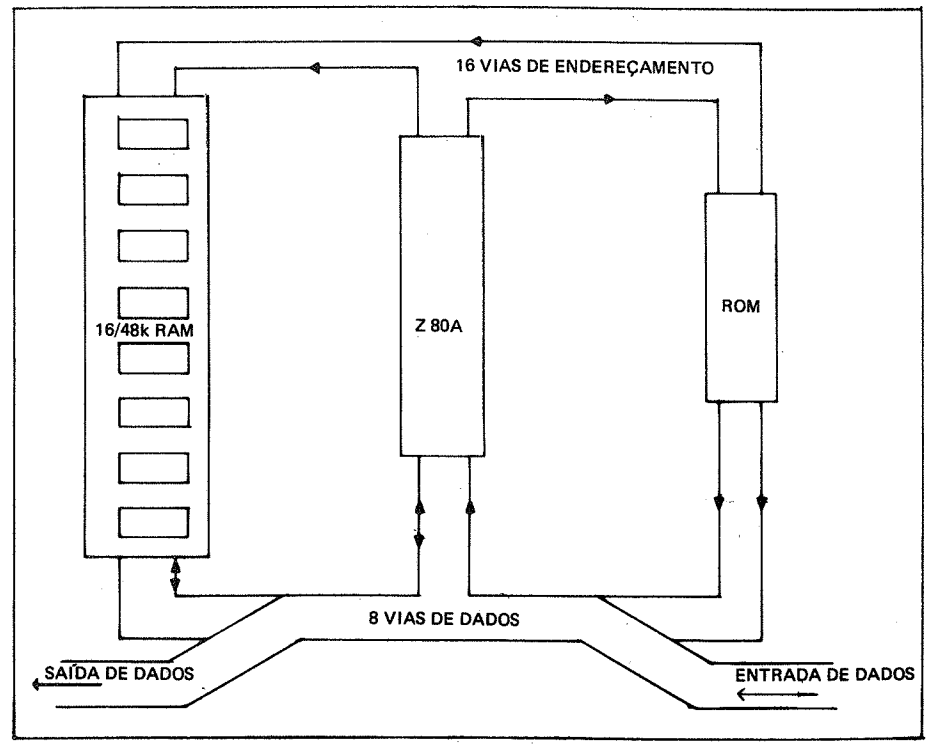


Figura 3 - Vias de endereçamento e de dados do TK 90X.

O TK 90X é fornecido pela Microdigital com um programa monitor de 16K, iniciando no endereço 0 e terminando no endereço 16383, que proporciona ao usuário o emprego de um sistema operacional bem como de um interpretador Basic. Também é possível o "abandono" deste programa monitor permitindo-se que o microprocessador Z 80 execute um programa em código de máquina escrito pelo usuário. Em modo normal de operação, o sistema operacional do TK 90X não requer qualquer ação por parte do usuário, e todas as ações desse programa monitor são "transparentes" para ele.

Entretanto, isso acontece sempre que o TK 90X é ligado, mas não em operação; o seu interpretador Basic está constantemente "lendo" o teclado, através de uma rotina do programa monitor, aguardando uma declaração em Basic do usuário, seja em modo imediato ou em modo programado.

Note que de nenhuma maneira o microprocessador Z 80 executará um programa em Basic – mas, sim o programa monitor, que é uma rotina em código de máquina do Z 80. A única exceção a este procedimento é quando o próprio usuário insere no micro uma rotina em código de máquina.

O mapeamento da memória do TK 90X, versões 16K e 48K, é mostrado esquematicamente na Figura 4 e cada uma dessas áreas será discutida a seguir.

Nota: A partir daqui, todos os endereços que contiverem apenas algarismos serão da base decimal; se contiverem após os algarismos a letra "h" serão da base hexadecimal, e se contiverem a letra "b" serão da base binária.

– ÁREA DE ROM

Os 16K de ROM contêm o sistema operacional, o interpretador Basic e o gerador de caracteres, ocupando os endereços de 0 a 16383, ou 0000 a 3FFFh. Como em qualquer microcomputador baseado no Z 80, o início do programa em código de máquina está no endereço 0.

– ÁREA DE MAPEAMENTO DA MEMÓRIA DA TELA – ARQUIVO DA IMAGEM

Os 6K de memória, do endereço 16384 até 22527, ou 4000 a 57FFh, formam a área de alta resolução gráfica da tela. É importante notar que esta área está fixada nessa dimensão pelo hardware do TK 90X, e não pode ser alterada por intermédio do software.

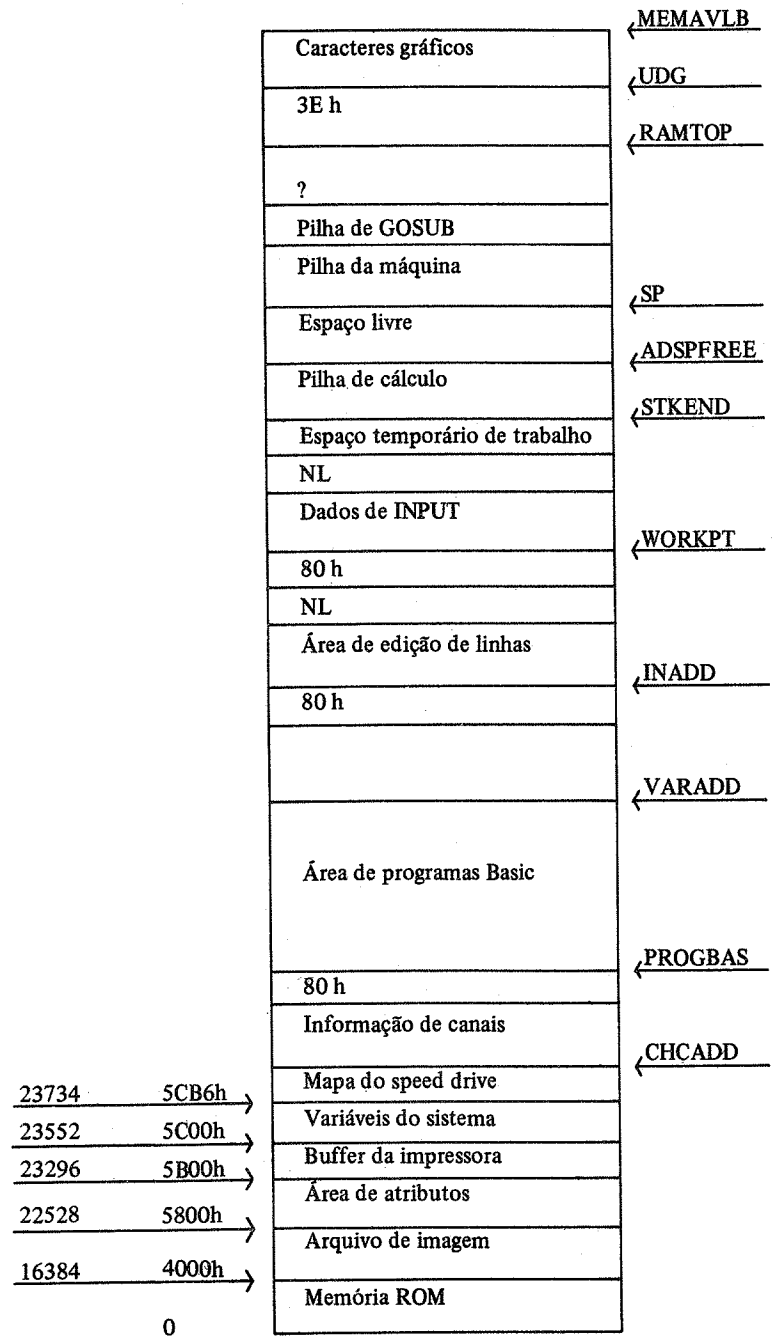


Figura 4 - Mapeamento da memória do TK 90X.

Existe uma relação de um para um entre todos os bits desta área de memória e os pixéis (*picture elements* – menor ponto de impressão) que aparecem na tela, e o cálculo seguinte mostra que o número de bits em 6K de memória é igual ao número de pixéis que podem ser mostrados na tela.

Número de bytes em 6K de memória	= 1024 x 6 = 6144
Número de bits em 6K de memória	= 6144 x 8 = 49152
Número de pixéis em 32 colunas por 24 linhas da tela, com 64 pixéis por caractere (matriz de 8 x 8)	= 32 x 24 x 64 = 49152

Agora vamos ver isso em termos práticos, ou seja, como são dispostos todos esses elementos na tela. (Aguarde mais um pouco – logo ligaremos o computador. . .)

Inicialmente, vamos considerar a tela da tevê dividida em três blocos horizontais iguais. O bloco mais alto, começando na linha 0 e terminando na linha 7 é produzido através dos conteúdos dos endereços 16384 a 18431, ou 4000 a 47FFh. O bloco central, das linhas 8 a 15, utiliza os conteúdos dos endereços 18432 a 20479, ou 4800 a 4FFFh; o bloco inferior, ou seja, o restante da tela do TK 90X, das linhas 16 a 23, utiliza os endereços 20480 a 22527, ou 5000 a 57FFh.

Preste atenção nos números em hexadecimal, que sugerem uma fácil manipulação do arquivo de imagem. Vamos analisar o bloco superior (e, por analogia, os outros dois blocos).

Cada posição PRINT necessita de 8 x 8 bits (64); e há 32 dessas posições em cada linha da tela (32d = 20h). Por outro lado, cada bloco da tela tem 32 x 8 x 8 = 2048 bytes, que em hexadecimal equivale a 0800h. Vimos que os bytes correspondem à ordem: todos os primeiros bytes do primeiro bloco, os segundos, os terceiros etc., o que nos dá um salto de 7! Isso significa, por exemplo, que o primeiro byte da primeira posição PRINT da tela dista 32 x 8 = 256, ou 0100h, do segundo.

Conclusão:

– Para cada posição PRINT, o primeiro byte dista dos seguintes, um múltiplo de 0100h.

– Um byte de uma posição PRINT dista do seu byte análogo na posição PRINT imediatamente abaixo ou acima, num mesmo bloco, de 0100h.

– Um byte de um bloco da tela dista do byte com a mesma posição relativa noutro bloco, de 0800h, ou 1600h, dependendo do bloco de referência e o bloco a que se refere.

Se você já fez algumas experiências em Basic com seu TK 90X e gostou, não imagina o quanto isto pode ser útil em programas de manipulação de imagens, tanto em Basic como principalmente em Assembler.

– ÁREA DE ATRIBUTOS

O arquivo de imagens possui ainda 768 áreas de caracteres, que vão do endereço 22528 a 23295, ou 5800 a 5AFFh, onde cada uma delas contém os códigos para cor de papel, cor da tinta, flash e bright, e são usados para armazenar valores que determinam os atributos dos 768 caracteres que podem ser impressos na tela.

Os valores (menores que 255) dos atributos podem ser considerados a partir da expressão:

ATTR (lin., col.) = INK + PAPER * 8 + BRIGHT * 64 + FLASH * 128

Essa função ATTR (lin., col.) é equivalente a PEEK (22/528 + 32 * lin. + col.)

Ou seja, nos bytes dos atributos, os bits 0, 1 e 2 determinam a cor da tinta; os bits 3, 4 e 5 determinam a cor do papel; o bit 6, se a posição está em modo bright ou não, e o bit 7, se está em modo flash ou não.

Por exemplo:		
	INK azul	= 1 1
	PAPER branco	= 7 (7 x 8 = 56) 56
	BRIGHT 0	= 0 0
	FLASH 1	= 1 (1 x 128) 128
		185

A seguir, uma tabela de todos valores possíveis para os atributos de qualquer posição.

TABELA DE ATRIBUTOS

PAPER	INK								MODO
	preto	blue	red	mag.	green	cyan	yell.	white	
BLACK	0	1	2	3	4	5	6	7	normal
	64	65	66	67	68	69	70	71	bright
	128	129	130	131	132	133	134	135	flash
	192	193	194	195	196	197	198	199	bright + flash
BLUE	8	9	10	11	12	13	14	15	normal
	72	73	74	75	76	77	78	79	bright
	136	137	138	139	140	141	142	143	flash
	200	201	202	203	204	205	206	207	bright + flash

RED	16	17	18	19	20	21	22	23	normal
	80	81	82	83	84	85	86	87	bright
	144	145	146	147	148	149	150	151	flash
	208	209	210	211	212	213	214	215	bright + flash
MAGENTA	24	25	26	27	28	29	30	31	normal
	88	89	90	91	92	93	94	95	bright
	152	153	154	155	156	157	158	159	flash
	216	217	218	219	220	221	222	223	bright + flash
GREEN	32	33	34	35	36	37	38	39	normal
	96	97	98	99	100	101	102	103	bright
	160	161	162	163	164	165	166	167	flash
	224	225	226	227	228	229	230	231	bright + flash
CYAN	40	41	42	43	44	45	46	47	normal
	104	105	106	107	108	109	110	111	bright
	168	169	170	171	172	173	174	175	flash
	232	233	234	235	236	237	238	239	bright + flash
YELLOW	48	49	50	51	52	53	54	55	normal
	112	113	114	115	116	117	118	119	bright
	176	177	178	179	180	181	182	183	flash
	240	241	242	243	244	245	246	247	bright + flash
WHITE	56	57	58	59	60	61	62	63	normal
	120	121	122	123	124	125	126	127	bright
	184	185	186	187	188	189	190	191	flash
	248	249	250	251	252	253	254	255	bright + flash

Podemos imaginar cada pixel da tela da televisão (colorida) como sendo um pequeno triângulo eqüilátero, cujos vértices são iluminados. Para cada vértice há um pixel vermelho, um azul e um verde, e o arquivo de atributos é usado para controlar a iluminação (acesa ou não) destas três cores diferentes. O arquivo da imagem vai indicar se um determinado pixel deve ser plotado com uma cor de tinta em particular. Os três bits mais baixos (0, 1 e 2) do valor do atributo daquela área são usados para decidir quais vértices do nosso triângulo imaginário, o verde, o azul ou o vermelho, devem ser acesos. Nossos olhos contêm apenas três tipos de sensores de cores (verde, vermelho e azul). Nosso cérebro recebe sinal dos três vértices coloridos e combina-os em um simples pixel de cor composta. Por exemplo, se os três últimos bits de atributo são 111, equivalente à cor 7, nós temos verde + azul + vermelho, o que corresponde à luz branca. Os outros códigos de cores, quando escritos na forma binária, podem ser interpretados da seguinte forma:

COR	CÓDIGO	BINÁRIO	VÉRTICES ACESOS		
Black	0	000			
Blue	1	001			Blue
Red	2	010		Red	
Magenta	3	011		Red	Blue
Green	4	100	Green		
Cyan	5	101	Green		Blue
Yellow	6	110	Green	Red	
White	7	111	Green	Red	Blue

Para os interessados, aí está uma boa dica para se obter mais de 8 cores do nosso TK 90X.

– BUFFER DA IMPRESSORA

Os endereços entre 23296 e 23551, ou 5B00 a 5BFFh são usados como o buffer da impressora (ZX Printer ou Alphacom 32 ou Timex 2040). Estes 256 bytes são suficientes para armazenar os 32 caracteres em sua forma de alta resolução, os primeiros 32 bytes armazenarão os bits da primeira linha dos caracteres, os próximos 32 bytes armazenarão os bits da segunda linha, e assim por diante.

Se não houver impressora ligada ao seu micro, esta área pode ser usada como área de armazenamento de rotinas em código de máquina, elaboradas pelo usuário... (boa dica essa!). Não há interferência com qualquer programa que esteja no micro, mesmo que ele ocupe toda a memória.

– VARIÁVEIS DE SISTEMA

As 182 posições de memória entre 23552 e 23733, ou 5C00 a 5CB5h, armazenam as diferentes variáveis do sistema do TK 90X. Se estiver conectado ao seu micro uma Interface One da Sinclair, ou um microdrive, haverá uma ampliação desta área, com o surgimento de novas variáveis de sistema que cuidarão exclusivamente desses periféricos. No Capítulo 3, estas variáveis serão melhor estudadas.

– MAPEAMENTO DO MICRODRIVE

Esta área da memória começa no endereço 23734, ou 5CB6h, e tem apenas uma existência teórica na versão standard do TK 90X, ou seja, é uma área que não é usada com esse propósito, a menos que a ele seja ligado um microdrive.

Sem o periférico, esta área também pode ser usada para armazenar rotinas em código de máquina elaboradas pelo usuário.

– ÁREA DE INFORMAÇÃO DE CANAIS

Esta área especial de memória tem início no endereço apontado pela variável de sistema CHCADD, armazenada nos endereços 23631 e 23632, ou 5C4F e 5C50h. Esta área tem tamanho variável, mas termina num endereço que armazena um marcador final de valor 128 ou 80h.

Na versão-padrão do TK 90X, ou seja, o micro sem periféricos conectados, existem detalhes de entrada/saída para quatro canais. Estes canais são:

– Canal "K" (palavra-chave) – Permite entradas via teclado e saídas para a parte inferior da tela.

– Canal "S" (screen) – Não permite entradas, mas somente saídas normais para a tela.

– Canal "R" (RS 232) – Também não permite entradas, mas permite saídas para a parte inferior da tela. O tamanho desta área de vídeo pode ser expandido, se necessário.

– Canal "P" (Printer) – Também não permite entrada, mas somente saída para a impressora, via slot traseiro.

As informações de canais consistem, por canal, em 5 bytes de dados. Estes bytes fornecem o endereço da rotina de saída, que toma dois bytes, o endereço da rotina de entrada, que toma outros dois bytes, e o nome do arquivo, que é um simples código da letra correspondente.

Como existem quatro canais-padrão e um indicador de final, área de informação de canais no TK 90X standard ocupa os 21 endereços entre 23734 e 23754, ou 5CB6 e 5CCAh.

Os percursos ou caminhos que os dados percorrem para chegar ou vir destes canais são chamados de fluxo. Associados a estes quatro canais-padrão existem quatro fluxos:

Fluxo # 0 – Imprime dados na parte inferior da tela que

Fluxo # 1 – são lidos do teclado.

Fluxo # 2 – Escreve dados na parte superior da tela, mas não pode ler dados.

Fluxo # 3 – Envia dados de saída para a impressora, mas também não lê dados.

Cada instrução de entrada/saída de dados usa automaticamente um destes fluxos. O comando PRINT usa o fluxo # 2 e o comando LPRINT usa o fluxo # 3.

Portanto, PRINT "TK 90X" é uma abreviação de PRINT # 2; "TK 90X".

Se você escrever LPRINT "TK 90X", o micro manda a mensagem para a impressora; mas, se você digitar LPRINT # 2; "TK 90X", esta mensagem é desviada para a tela (isso funciona com os diversos comandos de saída, tais como LIST, LLIST etc.).

Analogamente, se você escrever PRINT #1; AT 0,0; "TK 90X" o micro imprimirá a mensagem na linha 0, coluna 0, da parte inferior da tela (ou seja, a partir da linha 23). Experimente, pois isto é muito interessante na impressão de mensagens, sem interferir no arquivo de imagens! (O que deveria ter sido feito com o comando TRACE.)

– ÁREA DE PROGRAMAS BASIC

Esta área da memória armazena as linhas de programas em Basic, se existirem. O seu tamanho depende justamente de quantas linhas existirem.

O início da área de programa é sempre dado pelo valor armazenado na variável de sistema PROGBAS, que ocupa os endereços 23635 e 23636, ou 5C53 e 5C54h.

Note que, na versão standard do TK 90X, a variável de sistema PROGBAS vai indicar que esta área de programas Basic começa no endereço 23755, ou 5CCBh, e isto sempre vai acontecer, a menos que haja um microdrive conectado ao micro, ou seja, a área de mapeamento do microdrive esteja sendo usada ou endereços extras tenham sido reservados para informações adicionais de canais.

Na área de programas Basic, as linhas são armazenadas no seguinte formato:

Os primeiros dois bytes de qualquer linha armazenam o seu número, sendo o primeiro byte o mais significativo, e o outro, o menos significativo.

O terceiro e o quarto bytes armazenam o comprimento da linha. Aqui, o byte menos significativo vem antes do byte mais significativo. O comprimento da linha significa o número de bytes, a partir do quinto byte até o byte final, o ENTER (indicador de final de linha), inclusive.

Agora começa a linha de Basic propriamente dita. Os códigos do TK 90X são usados para símbolos, palavras-chave, outros caracteres, e os códigos ASCII para caracteres alfanuméricos standard.

O último byte de qualquer linha é sempre o caractere ENTER.

Dentro de uma linha em Basic, múltiplas declarações são separadas entre si por dois pontos, caractere 58, ou 3AH. Note que se numa linha Basic existir um número, em decimal, ele será armazenado em caracteres ASCII e seguido pelo caractere NÚMERO – código 14, ou OEh, e em seguida forma de ponto flutuante para números quaisquer ou a forma inteira para números inteiros da faixa -65535 a +65535, onde, em ambos os casos, são consumidos 5 bytes. Isto significa que são reservados 6 bytes de RAM para cada número decimal que estiver incluído num programa Basic!!!

O programa-demonstração a seguir mostra o seu próprio conteúdo armazenado na área de programas.

```
 10 FOR a=23755 TO 23817: PRINT
a;TAB 9;PEEK a;TAB 15;CHR$ PEEK
a: NEXT a
```

```
23755    0        ?
23756    10       ?
23757    59       ;
23758    0        ?
23759    245      FOR
23760    97       a
23761    61       =
23762    0        ?
23763    0        ?
23764    51       3
23765    0        ?
23766    0        ?
23767    14       ?   ──► INDICA NÚMERO
23768    0        ?
23769    0        ?
23770    203      THEN ──► 203        203
23771    2        ?    ──► 92x256 = 23552 +
23772    0        ?                  23755
23773    204      TO
23774    0        ?
23775    61       =
23776    0        ?
23777    7        7
23778    0        ?
23779    14       ?   ──► INDICA NÚMERO
23780    0        ?
23781    0        ?
23782    0        ?   ──►  9          9
23783    9        :   ──► 93 x 256 = 23808 +
23784    58       ?                  23817
23785    58       :
```



```
23785    245      PRINT
23787    97       a
23788    59       ;
23789    173      TAB
23790    57       9
23791    1        ?
23792    244      POKE
23793    9        :
23794    168      FN
23795    16
```

```
23796    0        ?
23797    59       ;
23798    190      PEEK
23799    97       a
23800    59       ;
23801    173      TAB
23802    49       1
23803    53       5
23804    14       ?   ──► INDICA NÚMERO
23805    0        ?
23806    0        ?
23807    15       ?   ──► 15
23808    0        ?
23809    0        ?
23810    59       ;
23811    194      CHR$
23812    190      PEEK
23813    97       a
23814    58       :
23815    243      NEXT
23816    97       a
23817    13
```

– ÁREA DE VARIÁVEIS

O endereço inicial desta área que armazena as variáveis correntes de um programa Basic é sempre dado pelo valor contido na variável de sistema VARADD, que ocupa os endereços 23627 e 23628, ou 5C4B e 5C4Ch.

No TK 90X, o início dessa área não vai mudar durante a execução de um programa Basic, mas sim quando nela for introduzida uma nova variável.

O último endereço desta área sempre contém o marcador de final de área, caractere 128 ou 80h.

O programa a seguir mostra o conteúdo da área de variáveis, que contém apenas a variável de controle do loop.

Programa-demonstração da área de variáveis:

```
    0 REM NAO DIGITE AS LINHAS RE
M - SOMENTE A LINHA 10
    1 REM PRIMEIRO DIGITE EM MODO
 DIRETO:
    2 REM PRINT PEEK 23627+256*PE
EK 23628
    3 REM O RESULTADO SERA 23818
    4 REM QUE, POR SUA VEZ E O VA
LOR INICIAL DA VARIAVEL DE CONTR
OLE <A> DO LOOP
    5 REM
   10 FOR A=23804 TO 23823: PRINT
 A;TAB 9;PEEK A;TAB 15;CHR$ PEEK
 A: NEXT A
```

– ÁREA DE EDIÇÃO DE LINHAS

O início desta área que armazena a linha Basic que está sendo digitada, ou editada, é sempre dado pelo valor armazenado na variável de sistema INADD, que ocupa os endereços 23641 e 23642, ou 5C59 e 5C5ah.

Quando a parte inferior da tela mostra apenas o cursor intermitente, a área de edição de linhas tem três bytes reservados para ela. O primeiro, cujo endereço é dado pelo conteúdo da variável de sistema CURADD, armazena um caractere ENTER, e o segundo armazena um marcador de final, caractere 128, ou 80h.

Os caracteres digitados a partir do teclado são, então, transferidos para esta área, que é expandida, para armazená-los.

Um procedimento similar é obtido quando se digita a tecla EDIT (CAPS SHIFT e tecla 1), para corrigir uma linha de programa. Em primeiro lugar, a área de edição é expandida para o comprimento igual ao da linha a ser editada. Então esta linha é copiada sobre o conteúdo anterior da área.

– INPUT DADOS E ÁREA DE TRABALHO TEMPORÁRIO

Esta área da memória é usada para muitos propósitos, como, por exemplo, dados de entrada, concatenação de cadeia de caracteres etc. O endereço inicial da área é dado pelo valor armazenado na variável de sistema WORKPT, de endereços 23649 e 23650, ou 5C61 e 5C62h. Quando se faz necessário uma maior área de trabalho, esta área é expandida. Após o uso, ela é então esvaziada.

– ÁREA DA PILHA DE CÁLCULO

Esta é uma área muito importante da memória e começa no endereço apontado pela variável de sistema STKCEND, de endereços 23651 e 23652, ou 5C63 e 5C64h, e vai até o endereço apontado pela variável de sistema ADSPFEE, de endereços 23653 e 23654, ou 5C65 e 5C66h.

Esta área é usada para armazenar os números em ponto flutuante em cinco bytes e quando manipula cadeias de caracteres utiliza grupos de cinco bytes como parâmetros delas.

A manipulação desta pilha de cálculo é feita sob a forma conhecida como "LIFO", do inglês "Last-in First-out", ou seja, o último valor depositado na pilha é o primeiro a ser retirado.

– ESPAÇO LIVRE

A área de memória entre a pilha de cálculo e a pilha da máquina representa a quantidade de memória que é disponível ao usuário. Na versão standard do TK 90X existem nominalmente 8839 endereços nesta área quando se liga o computador. Para a versão de 48K de RAM existem esses 8839 mais os adicionais 32K. Entretanto, é interessante notar que o valor mais baixo aceitável para CLEAR é 23821, que traz a RAMTOP 8878 bytes para baixo, ou na RAM de 48K, 8878 mais 32K (41646K).

– ÁREA DA PILHA DA MÁQUINA

O microprocessador Z 80 deve ter uma área de trabalho disponível apenas para ele, chamada de pilha da máquina, onde ele vai manipular valores dos programas armazenados na memória.

– ÁREA DA PILHA DE GO SUB

Sempre que existir um loop de GO SUB, o número da linha desse loop será armazenado nesta área.

Cada número de linha de GO SUB requer três endereços. O mais alto armazena o número da declaração dentro de uma linha de programa Basic, para onde o retorno deverá ser feito. O próximo endereço armazena a parte menos significativa do número da linha de programa, e o último armazena a parte mais significativa desse número.

A variável de sistema RAMTOP, que ocupa os endereços 23730 e 23731 ou 5CB2 e 5CB3h, armazena o endereço que contém o valor 62, considerado como sendo o último endereço da área de Basic.

– ÁREA DE DEFINIÇÃO DE GRÁFICOS DO USUÁRIO

Os últimos 168 endereços da memória RAM do TK 90X são reservados para armazenar as representações dos bits dos caracteres gráficos definidos pelo usuário, a menos que tenha sido invadida pelo sistema Basic, quer pela modificação de endereços, através do uso de CLEAR, ou por um programa Basic muito extenso.

Como parte do procedimento de inicialização do TK 90X, a representação bit a bit das letras de "A" até "U" é copiada nesta área. O usuário, ao definir alguns ou todos os 21 caracteres gráficos, muda esta representação.

O último endereço da memória é sempre fornecido pela variável de sistema MEMAVLB, que ocupa os endereços 23732 e 23733, ou 5CB4 e 5CB5h.

Na versão de 16K do TK 90X, este endereço deve ser 32767, e na de 48K, deve ser 65535, valor obtido pela expressão Basic, em modo direto:

```
PRINT PEEK 23732 + 256 * PEEK 23733
```

Capítulo II

A LINGUAGEM BASIC – COMANDOS E FUNÇÕES

Como citei na Introdução, espero que você já tenha adquirido noções de programação em Basic, e possua uma certa familiaridade com seus comandos e funções.

Neste capítulo, serão mencionadas todas as declarações e funções do TK 90X, com assuntos novos, que não foram citados no manual da máquina.

O Interpretador Basic do TK 90X standard reconhece cerca de cinqüenta comandos diferentes e trinta e três funções. Cada um deles será discutido neste capítulo, com alguns exemplos práticos. Eles estão organizados em ordem alfabética, para facilitar a sua consulta.

OS COMANDOS BASIC

– BORDER M

Existem no total oito cores disponíveis para a área da borda da tela que o TK 90X utiliza. A faixa de números (m) inteiros vai de 0 a 7. Entretanto (m) é arredondado e a verdadeira faixa de números aceitáveis começa em -0,5 e vai até 7,49.

(m) pode também ser uma expressão cujo resultado se situe dentro da faixa especificada.

O efeito do comando BORDER m é o de enviar um sinal OUT para a porta 254, e isto pode ser mostrado através da linha:

OUT 254,m (onde m=2 resulta borda vermelha)

Mas, OUT e BORDER são muito diferentes. A cor assumida pela área da borda da imagem, através do comando OUT, é uma cor "provisória", enquanto um comando BORDER muda para uma cor "permanente", com o código dessa cor sendo armazenado na variável de sistema BORCLR, de endereço 23624, ou 5C48h.

Esse efeito pode ser mostrado da seguinte maneira:

Digite BORDER 2 e ENTER.

A cor da borda passará para vermelho.

Então digite OUT 254,6 e ENTER.

A cor da borda passará, temporariamente, para amarelo, mas se você pressionar ENTER novamente a cor da borda voltará para vermelho, a cor permanente. Note, também, como BORCLR manipula a cor do papel na parte inferior da tela.

Experimente digitar este programa:

```
10 POKE 23624, RND * 255
20 OUT 254, RND * 7
30 RUN
```

E veja como é interessante o resultado.

Agora, experimente a linha de programa:

```
10 PAUSE 1: BORDER 1: BORDER 2: BORDER 3: BORDER 4: BORDER 5: BORDER 6: BORDER 7: BORDER 0: GOTO 10
```

– BRIGHT M

Este é o primeiro comando de cores. Todos estes comandos de cores podem ser utilizados como comandos únicos em modo imediato, ou em uma declaração Basic, sendo, neste caso, comandos com resultados permanentes; se forem utilizados em declarações tipo PRINT, os resultados serão "provisórios", pois só atingem parte da tela.

m pode ser uma expressão numérica, mas somente com resultado igual a 0, 1 ou 8.

Com m = 0 a tela voltará ou permanecerá com o brilho normal das cores, ao passo que com m = 1 qualquer impressão futura será feita de modo mais brilhante, alterando a tonalidade-padrão das oito cores do TK 90X. O emprego de BRIGHT 1 e CLS é o modo mais fácil para que toda a imagem passe a modo brilhante.

Com m = 8 qualquer impressão a ser feita na tela considerará o estado em que já se encontrava a tela, ou seja, m = 8 quer dizer "transparente", sem alterar nada.

As linhas abaixo mostram os quatro diferentes modos de como um comando de cor, com o BRIGHT, pode ser usado.

```
10 BRIGHT 1: PRINT "BRIGHT –";:
   BRIGHT 0: PRINT "NORMAL"
```

O brilho da cor é alterado de duas maneiras, sempre permanentemente.

```
20 PRINT BRIGHT 1; "BRIGHT –"; BRIGHT 0; "NORMAL"
```

que muda o brilho temporariamente, durante a declaração.

```
30 PRINT CHR$ 19 + CHR$ 1; "BRIGHT –"; CHR$ 19 + CHR$ 0; "NORMAL"
```

que substitui o comando pelo seu CHR$ equivalente.

```
40 LET A$ = CHR$ 19 + CHR$ 1: LET B$ = CHR$ 19 + CHR$ 0: PRINT A$; "BRIGHT –"; B$; "NORMAL"
```

que coloca os comandos de cor em uma variável tipo *série,* ou tipo *série* de caracteres.

Existe outro modo de se "ligar" ou "desligar" o BRIGHT, que não foi citado no manual do TK 90X. Com o cursor em modo E, ou seja "extended mode", se você pressionar a tecla 9, automaticamente obterá o modo BRIGHT 1; se pressionar a tecla 8, desativará, ou seja, retornará ao modo BRIGHT 0. Isso pode ser utilizado dentro de aspas, na impressão de mensagens, ou mesmo com comandos de atribuição, como por exemplo LET A$ = " " (com o modo extendido dentro das aspas). Digitando PRINT A$ lhe dará o mesmo efeito que BRIGHT 1 ou BRIGHT 0, dependendo do que você colocou entre aspas.

– CAT M

Para uso exclusivo com microdrives.

Fornece a listagem de todos os arquivos gravados no cartucho inserido no Microdrive m (com m indo de 1 a 8). A listagem é apresentada em ordem alfabética,

precedida pelo nome do cartucho e seguida pela quantidade de memória disponível naquele cartucho, expressa em Kb.

– CIRCLE x, y, z

Este comando desenha um círculo de raio z, e coordenadas x e y do centro.

O valor z deve ser um número absoluto inteiro, enquanto x e y são manipulados como números de ponto flutuante.

O maior círculo que pode ser desenhado possui raio 88, como na linha CIRCLE 127,5, 87,7, 88. Se o raio é zero, o círculo é um simples ponto.

Qualquer comando de cores pode ser inserido num comando CIRCLE, mas seu efeito será sempre provisório.

Experimente a linha:

10 FOR f = 168 TO 4 STEP -4: CIRCLE INK RND * 6; f, f/2, f/2: NEXT f

E perceba a diferença entre um arquivo de imagens de alta resolução gráfica e um arquivo de atributos de baixa resolução gráfica.

– CLEAR E CLEAR N

Como o TK 90X possui uma grande quantidade de memória RAM disponível para o usuário, o emprego do comando CLEAR é muito vantajoso. Ele tem a facilidade de mover a RAMTOP, fazendo a área de programas Basic aumentar ou diminuir. Como a memória RAM faz o papel de um bloco de rascunho, tudo o que estiver dentro dela será apagado pelo comando NEW. O que estiver acima da RAMTOP, porém, não será atingido por esse comando.

O limite inferior para n é 23821, e após um comando CLEAR 23821, o TK 90X emitirá um sinal sonoro avisando que não cabe qualquer declaração ou linha de programa Basic na memória.

O limite superior para n é 32767 em máquinas com 16K de RAM, e 65535 em máquinas de 48K. O uso de CLEAR n com um número apropriado tem o efeito de colocar as pilhas da máquina e do GOSUB na área usada para caracteres gráficos do usuário. É interessante tentar os passos a seguir que ilustram este efeito.

Digite CLEAR 32767 (ou 65535)

Mude o cursor para modo gráfico (G) e pressione as letras de A até U. Então mantenha pressionada a tecla de SPACE por alguns segundos e observe as alterações de valores de cada tecla pressionada, indicando o uso da pilha da máquina.

– CLOSE = FLUXO

Para uso com microdrives.

Desconecta um canal do fluxo especificado.

– CLS

Trata-se de um comando aparentemente simples de ser utilizado, que toma menos de um décimo de segundo na sua execução, mas que envolve muito trabalho por parte do microprocessador.

O comando CLS limpa o arquivo de imagem. Ele "zera", ou seja, inicializa todos os endereços de 16384 a 22527, ou 4000 a 57FFh, bem como inicializa todos os endereços do arquivo de atributos, que vão de 22528 a 23295, ou 5800 a 5AFFh.

Na realidade, essa inicialização é feita através da cópia do valor armazenado na variável de sistema ATCLR P em cada posição do arquivo.

Fazendo-se diversos POKE na variável ATCLR P, vemos claramente a atuação de CLS. Digite:

10 POKE 23693, RND * 255

20 PAUSE 0: CLS: GOTO 10

Em ATCLR P, o bit 7 controla o FLASH, o bit 6, o BRIGHT, os bits 5-3 controlam a cor de PAPER e os bits 2-0 controlam INK.

– CONTINUE

Muitas vezes este comando trabalha muito bem. Mas, em modo direto, o usuário perceberá que o computador entra num "loop" contínuo, até que se pressione BREAK.

Existem duas facetas distintas do comando CONTINUE. A primeira é a que permite ao usuário, em tendo declarações STOP dentro de um programa Basic, fazer o seu processamento parar e, em seguida, continuar, através da tecla CONT. Des-

ta forma, pode-se examinar o programa, suas declarações, suas variáveis, alterar valores em suas linhas etc. Facilita o "debugging" (correção) de programas. Também pode ser utilizada a tecla BREAK.

A segunda faceta permite ao usuário repetir a interpretação de uma declaração, após corrigir um erro. Por exemplo, se um programa pára por falta de definição de variáveis, ou "variável não encontrada", esta pode ser definida pelo usuário através de um comando direto, e o programa pode ser reiniciado pelo comando CONT.

Seis variáveis de sistema estão envolvidas de alguma forma na execução do comando CONTINUE. São elas: INJMP, INSTRNR, EXCLINE, SUBLEXC, CONTJMP e CONTNR.

– COPY

Este é um comando muito particular.

Se houver uma impressora conectada ao slot traseiro do TK 90X, as 22 linhas superiores da tela serão enviadas a essa impressora. As 176 linhas de alta resolução do conteúdo do arquivo de imagem serão enviadas em partes. Este comando COPY é um dos muitos que desligam o "relógio de tempo real" (o clock do Z 80), porque o comando é mais lento do que esse relógio. Pode-se verificar isto, examinando-se a variável de sistema TVCOUNT antes e depois do uso de COPY.

– DATA a, b, c,. . .

Este comando, que só pode ser utilizado em modo programado, armazena uma listagem de valores que são associados aos nomes de variáveis contidos em outra linha do programa que obrigatoriamente começa com o comando *READ*. É uma dupla de comandos muito útil, pois economiza memória, já que pode agrupar diversos comandos LET (comparativamente) em uma mesma linha.

– DEF FN c (d,. . .w)=g & DEF FN b$ (d,. . .f)=r

Este é um comando muito poderoso no TK 90X.

O usuário pode definir um total de 52 funções – 26 numéricas e 26 alfanuméricas. Os nomes usados para essas funções devem portanto ser de um simples caractere (com $ para alfanuméricas).

– DIM x (al,. . .ak) & DIM x$ (al,. . .ak)

O comando DIM reserva espaço na memória RAM do computador para uma matriz numérica ou alfanumérica, de quantos elementos forem definidos em sua declaração, através da variável subscrita, que aparece em primeiro lugar entre parênteses, iniciando obrigatoriamente em 1 (outros sistemas permitem iniciar em 0). O último elemento dentro do parênteses é o comprimento de matrizes alfanuméricas. Siga o exemplo:

```
10 DIM a$ (1, 704)
20 FOR f = 1 TO 168 STEP 4: CIRCLE f, 87, f/2: NEXT f
30 PRINT FLASH 1; AT 11, 13; "TK 90X"
40 INPUT "BRILHANTE OU NÃO (1 ou 0)"; b
50 INPUT "Que cor de papel?"; p
60 PRINT OVER 1; BRIGHT b; PAPER p; INK 9; AT 0,0; a$ (1)
70 GOTO 50
```

– DRAW x, y & DRAW w, y, z

Este comando desenha uma linha a partir da última posição PLOT, exclusive, até um ponto que está a distância x horizontalmente e a distância y verticalmente. Se o argumento z é especificado, então é desenhado um arco de círculo, em vez de um segmento de reta. Esse argumento z deve ser especificado em radianos, onde se z= PI, o arco de círculo é um semicírculo. Se z é positivo, o arco será desenhado no sentido anti-horário, e se z é negativo, o arco será desenhado no sentido horário.

Qualquer comando de cor pode ser utilizado juntamente com o comando DRAW; não se esqueça de que o efeito desse comando de cor, se utilizado desta forma, é temporário.

Agora, coloque a ROM do seu micro em transe:

```
10 FOR f= 201 TO 1000 STEP 20: BORDER RND * 6: PAPER RND * 6:
INK 9: CLS: PLOT 100,60: DRAW 70, 70, f * PI: NEXT f.
```

– ERASE "m";y; "nome"

Para ser usado com microdrive.

Apaga o arquivo especificado pelo "nome" no cartucho inserido no microdrive de número y (de 1 a 8).

– FLASH m

Este é o segundo comando de cor. Quando usado sozinho em uma declaração Basic, seu efeito é de cor permanente, mas quando utilizado juntamente com uma declaração PRINT, seu efeito passa a ser "provisório".

Como BRIGHT, m pode assumir os valores 0, 1 e 8.

Com m=0 → área sem intermitência

m=1 → área com intermitência

m=8 → área não sofre alteração

FOR x= a TO b STEP c

O comando FOR é um dos comandos mais interessantes da Basic, e geralmente muito pouco compreendido, dada a sua grande faixa de atuação.

As fases executadas pelo interpretador, quando encontra um comando FOR, são:

– Cancela qualquer variável existente que possua o mesmo nome da variável de controle.

– Adiciona às variáveis existentes no programa a nova variável de controle (x). Esta variável ocupa dezenove endereços de memória. O primeiro endereço armazena o código da letra que é o nome da variável. Os próximos cinco endereços armazenam o valor inicial (a) da variável de controle, como um número decimal de ponto flutuante. Outros cinco endereços são consumidos para armazenar o valor-limite (b) da variável de controle, sob a mesma forma do valor inicial. Os outros cinco endereços seguintes são usados para armazenar o valor do passo (c); caso não seja especificado, é assumido automaticamente o valor 1. Os três endereços finais da linha de abertura do loop armazenam detalhes sobre ele. Os dois primeiros armazenam o número da linha onde se encontra o comando FOR, e o terceiro armazena o número da declaração dentro de uma linha de programa, incrementado em um.

Se os valores limite e inicial são inteiros entre -65535 e 65535 então são armazenados como números inteiros, em vez de números de ponto flutuante, e são manipulados cerca de 20% mais rápido.

– A terceira ação executada pelo interpretador é muito especial. No TK 90X, não existe indicação de erro quando especificamos um passo com valor errado (negativo em vez de positivo, por exemplo). Ele simplesmente ignora o próximo valor a ser assumido pela variável de controle.

Por exemplo, no programa:

```
10 FOR a= 1.6 TO 2.1 STEP -.1
20 PRINT a
30 NEXT a
```

E o micro executará o programa, só que não haverá impressão alguma na tela, mostrando o resultado.

– FORMAT "m";y; "nome"

Para ser usado com microdrive.

Prepara para uso um cartucho novo, inserido no microdrive de número y, associando o nome especificado a esse cartucho.

– FORMAT "n"; x

Atribui à estação (microcomputador trabalhando em rede local, até 64) o número x especificado.

– FORMAT "t"; x

– FORMAT "b"; x

Seleciona a taxa de baud para a interface RS 232 especificada pelo valor x, que deve estar contido na faixa-padrão da taxa de baud (50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200). Outras taxas de baud, que não estejam na faixa-padrão, podem ser obtidas através da alteração de valores da variável do sistema BAUD (na versão standard do TK 90X, esta variável inexiste).

– GO SUB n

Este comando transfere a execução de um programa para a linha especificada pelo número n, ou para a linha imediatamente posterior.

Após a execução desta sub-rotina, o controle do programa retorna para a primeira declaração posterior ao GO SUB que o desviou.

Uma poderosa capacidade do Basic do TK 90X é a que permite aos GO SUB e GO TO executarem saltos, com valores provenientes de expressões aritméticas, o que não está disponível em muitas máquinas que possuam o comando ON x GOSUB, e ON x GOTO.

– GO TO n

No programa, significa que a próxima linha a ser interpretada é a linha de número n.

O comando GO TO armazena sua linha de destino, nas variáveis de sistema INJMP e INSTNR, nos endereços 23618 e 23620, ou 5C42 e 5C44h.

– IF x THEN y

No sistema do TK 90X, a quantidade zero é considerada logicamente "falsa", e qualquer outra quantidade é considerada "verdadeira". Esta declaração IF x THEN y somente executará alternativa y quando a condição x for verdadeira.

Estes comandos são muito poderosos, permitindo diversas combinações de operações lógicas e aritméticas, mas o programa abaixo mostra o uso de divisões repetitivas que levam quantidades numéricas a nunca assumirem um valor "falso". O problema surge em decorrência do modo como o TK 90X trata números, ou quantidades 2 ↑ - 128 (2 elevado à potência - 128).

Programa-demonstração de um caso especial de IF. . .THEN. . .

```
10 LET a=1
20 IF NOT a THEN PRINT a
30 PRINT a
40 LET a=a/2
50 GO TO 20
RUN
```


E a será impresso na tela, diversas vezes, até atingir o valor 2.9387359E-39, quando se repete.

– INK n

Terceiro item de cores. Quando usado sozinho numa declaração Basic, tem o efeito permanente, mas quando utilizado dentro de declaração PRINT, por exemplo, tem efeito provisório.

O valor de n deve estar na faixa de 0 a 9. As oito principais cores do TK 90X possuem os códigos entre 0 e 7, inclusive, e são mostradas claramente no teclado. O uso de INK 8 demonstra que qualquer impressão a seguir será feita de modo a não alterar a tinta existente na posição da tela. O uso de INK 9, entretanto, é um pouco mais complicado, pois resulta numa tinta contrastante com o papel de fundo. Ou seja, se o papel é escuro, a tinta será branca, e se o papel for claro, a tinta será preta. Os papéis considerados escuros são o preto, o azul, o vermelho, e o magenta; os papéis claros são o verde, o cyan, o amarelo e o branco. A ação do comando INK permanente é: para n=0 a 7, seta os bits 0, 1 e 2 de ATCLR P conforme a cor; para n= 8, seta os bits 0, 1 e 2 de MASKCLRP; e para n= 9, seta o bit 5 de SFLAG4. Se o comando INK é provisório, os bits das variáveis de sistema que cuidam das cores provisórias são setados conforme a cor desejada.

– INPUT n

Este é um comando muito poderoso do TK 90X, pois permite ao usuário a inserção de mensagens entre aspas a serem impressas, simulando o comando PRINT. Estas mensagens, quando usadas com o comando INPUT LINE, devem vir obrigatoriamente antes da palavra LINE.

O programa abaixo mostra o uso de diversos itens num mesmo INPUT:

```
10 INPUT "Qual é o seu nome?"; a$; CHR$ 13; "E qual é a sua idade?"; LINE b$; "E onde você mora?"; LINE c$
20 PRINT AT 2,5; "Nome:"; a$; CHR$ 13; "Idade:"; b$
30 PRINT AT 4,5; "Endereço:"; c$
```

Agora, experimente o seguinte programa, substituindo a linha 10 por:

```
10 INPUT AT 5,0; "Qual é o seu nome?"; LINE a$; AT 3,8; "E a sua idade?"; LINE b$; AT 7,15; "E onde você mora?"; LINE c$
```

Veja que efeito interessante!

Experimente substituir os valores das linhas e colunas do comando INPUT AT, respeitando os números de linhas (de 0 a 21) e colunas (de 0 a 31) da tela, e verificará que se pode, em diversos programas, colocar as entradas diretamente na posição que serão impressas.

– INVERSE n

Outro comando de cores, agora o quarto. Mais uma vez, seus efeitos podem ser provisórios ou permanentes.

Se é utilizado INVERSE 1, então a impressão se fará, conforme o nome diz, inversamente, ou seja, com a tinta da cor do papel, e o papel com a cor da tinta (inverte as cores). Para anular esse efeito, usa-se INVERSE 0.

Quando se usa INVERSE 1, para efeito permanente, o que o TK 90X faz é setar o bit 5 de SFLAG4, e seta o bit 4 daquela variável, quando se quer efeito provisório.

– LET x=y

Este é, sem dúvida alguma, o mais importante dos comandos Basic do TK 90X. Uma variável é "inicializada" pelo usuário e o interpretador tem de determinar se a variável é simples, *é série,* é controle de loop, ou parte de uma matriz.

Todos os valores numéricos empregam cinco endereços para armazenar o seu conteúdo. Este valor é então manipulado como um número de ponto flutuante ou um inteiro. *Séries* de caracteres que tenham comprimento dinâmico, ou seja, o número de caracteres reservados para seu comprimento depende do instante do programa, também tem endereços reservados de valores variáveis. Todas as matrizes de comprimento fixo e as cadeias de caracteres que sejam elementos de matrizes serão fatiadas, se o seu conteúdo for maior do que o comprimento especificado, e serão completadas com espaços em branco, se o seu comprimento for menor que o especificado.

O programa-demonstração a seguir mostra a área de variáveis manipulando diversos valores digitados a partir do teclado.

```
10 REM Entre com a sua variável
20 LET m= PEEK 23627 + 256 * PEEK 23628
30 PRINT m, PEEK m: LET m=m + 1
40 GO TO 30
```


Algumas sugestões para a linha 10 acima:

```
10 LET c=0
10 LET c=15E3
10 LET c$="CCC"
10 DIM c(3): LET c(1)= 7E7
10 DIM c$(3): LET c$="ABCDEFGHIJKLM"
10 DIM c$(2,10): LET c$(1)="TK 90X"
```

Veja, no seu manual, a correspondência de caracteres e seus respectivos códigos para entender todos esses números que apareceram na tela.

– LIST n

Neste comando, se n=0, é assumido o valor da primeira linha existente do programa em Basic, e não é necessário especificá-lo. Se a linha n existe, então o cursor é colocado após seu número, indicando que ela passou a ser a linha corrente, a fim de ser editada.

Digite qualquer programa no micro, a partir da linha 1, com linhas de 1 em 1. Inclua uma linha qualquer, de número 20, e digite mais algumas linhas. Em seguida dê LIST 49172, para ver o que acontece (é o mesmo que LIST 20).

– LLIST n

Este comando tem atividade semelhante a LIST n, só que desvia a listagem do programa para a impressora. Experimente colocar em seus programas LIST # 3 e LLIST # 2, para ver o que acontece.

– LOAD

No TK 90X, o comando LOAD possui diversos modos de utilização, a fim de permitir ao usuário carregar programas Basic, matrizes ou blocos de bytes.

A informação, código ou programa armazenado numa fita cassete, é manipulado em duas partes. A primeira parte é um "cabeçalho" de dezessete bytes e a segunda parte é um bloco de bytes.

O "cabeçalho" pode ser dividido em quatro partes, a saber:

1- Um simples byte de informação, que vale 0 para programas Basic, 1 para matrizes numéricas, 2 para matrizes alfanuméricas e 3 para blocos de bytes.

2- Os próximos dez bytes armazenam o nome do arquivo. Se o nome tiver mais do que dez caracteres, ele é rejeitado.

3- Dois bytes seguintes são usados para informar o comprimento total do bloco de bytes. No caso de programa Basic, apenas a área de programas e a área de variáveis são armazenadas na fita cassete.

4- Dois outros bytes que armazenam, no caso de programa Basic, o número da linha de programa que será o seu início, e no caso de bloco de bytes, o endereço de início daqueles bytes.

O bloco de bytes é simplesmente carregado na área de RAM requerida, de acordo com a informação do cabeçalho.

– LPRINT

Este comando desvia os dados de saída para a impressora. Para maiores detalhes, veja PRINT.

– MERGE

Este comando permite que um programa Basic e suas respectivas variáveis e informações sejam carregados na memória do micro, a partir de um gravador cassete ou um microdrive, cancelando apenas o que for coincidente em relação ao programa que já estava na memória, como, por exemplo, números de linhas de programa, nomes de variáveis etc., fazendo então com que os dois programas "se misturem".

O modo de trabalhar da rotina da ROM do comando MERGE faz o programa, a ser carregado da fita cassete, ser tratado como um bloco de bytes e ser inicialmente armazenado na área de trabalho da memória, para, a seguir, ser comparado, linha por linha, com o programa existente, e então serem copiadas na área de programas Basic as novas linhas e serem sobrepostas as coincidentes e as variáveis, de mesmo nome.

– MOVE FONTE TO DESTINO

Para uso com microdrive.

Transfere dados de uma fonte qualquer de dados, fluxo ou canal, para a fonte destino.

Em termos de microdrive, é um comando muito poderoso.

– NEW

Este comando faz uma reinicialização total do sistema, ou seja, apaga todo o conteúdo da memória e "zera" qualquer variável existente, exceto as variáveis de sistema RAMTOP, MEMAVLB, BUZCCLE, KCLICK e UDGRAFH, não afetando caracteres gráficos definidos pelo usuário, visto que se encontram em área fora da área Basic. É o mesmo que RAND USR ϕ.

– NEXT x

Este comando pode ser considerado como o comando de fechamento de um loop aberto através de FOR x=. . . TO y. . . STEP. . . , e tudo que se encontra "dentro" dessas duas linhas tenha o seu processamento repetido y vezes. Pode-se dizer que o comando FOR não trabalha sem NEXT, e o comando NEXT não trabalha sem FOR. Para cada FOR de um programa, deve existir um NEXT.

As fases envolvidas na execução do comando NEXT são as seguintes:

1- A variável de controle é armazenada na área de variáveis da memória, e o valor do STEP é adicionado ao valor da variável de controle, não importando se é negativo ou positivo.

2- Então o novo valor da variável de controle é comparado com o limite, e se este novo valor o ultrapassar, então o comando NEXT estará terminado.

– OPEN

Para uso com microdrive, entretanto, é possível "abrir" ou "fechar" (OPEN-CLOSE) fluxos num micro standard, sem periféricos, da seguinte maneira:

– Experimente a linha PRINT # 5 "Será que funciona?", e certamente não funcionará, a menos que o fluxo seja aberto com a linha OPEN # 5, "S", direcionando o fluxo 5 ao canal S (de screen). A linha CLOSE # 5 fechará o fluxo.

Experimente digitar o programa a seguir, que mostra entradas sendo aceitas a partir do teclado e sendo passadas para a tela.

```
10 OPEN # 5, "k"
20 INPUT # 5; a$
30 OPEN # 5, "s"
40 PRINT # 5; a$
```

– OUT f,g

Este comando permite que o usuário envie sinais às portas de saída do TK 90X, a partir do Basic.

No comando BORDER, já visto, foi mostrado que a porta 254 controla a cor da borda, para valores entre 0 e 7.

Entretanto, essa mesma porta 254 pode ser empregada para controlar a saída de som através do alto-falante da televisão. O programa mostra também uma estimativa do tempo que o TK 90X leva para executar uma instrução. Se esta for rápida, o som equivalerá a uma nota alta, e, se ela for mais lenta, o som equivalerá a uma nota mais baixa.

```
– Programa de controle do alto-falante da televisão
10 OUT 254, 23
20 REM Nesta linha pode-se colocar instruções das mais variadas, como, por exemplo PRINT;: ou RAND ou LET t=0 ou PAUSE 1 ou PAUSE 5 etc.
30 OUT 254,7
40 GO TO 10 ou RUN
```

Nesse programa, o comando OUT 254,23 desliga o som, enquanto o comando OUT 254,7 liga o som. A operação é repetida muito rapidamente para produzir o som do alto-falante da tevê.

Quando houverem diversos periféricos no mercado, este será um comando muito utilizado.

– OVER x

Mais um comando de impressão de cores (o quinto). Mais uma vez, seus efeitos podem ser provisórios ou permanentes. Se foi especificado OVER 1, então a impressão que segue será resultado da operação binária XOR com a impressão existente na área do caractere que está sendo processado.

O resultado da operação binária XOR será 1 (um) quando somente um dos bits envolvidos for um, caso contrário, será zero.

Isto quer dizer que se um caractere qualquer for impresso duas vezes com OVER 1, ele desaparecerá.

Com OVER 0, todos os bits de uma dada área de caractere envolvida serão plotados.

– PAPER x

Último item de impressão de cores. Mais uma vez, seu efeito pode ser temporário ou permanente.

Como INK, o comando PAPER pode ser usado com x variando de 0 a 9. Para se utilizar as cores normais do TK 90X, utiliza-se x na faixa de 0 a 7. Com x=8, emprega-se o modo transparente, ou seja, não se altera a cor do papel já existente na tela.

O uso de PAPER 9 é muito útil, pois nos permite utilizar o papel sob forma contrastante, ou seja, se a tinta com a qual se está escrevendo for escura, o papel a ser utilizado será branco, e vice-versa.

A ação do comando PAPER, usado no modo permanente, seta os bits 3, 4 e 5 da variável de sistema ATCLR P, conforme a cor especificada (com x de 0 a 7); com x=8, seta os bits 3, 4 e 5 de MASKCLRP; e com x=9 seta o bit 7 de SFLAG4.

– PAUSE s

Este comando aplica ao Z 80 a instrução em linguagem de máquina HALT, por s interrupções. O único trabalho que é realizado durante PAUSE é a manipulação da rotina da ROM que lê o teclado para saber se alguma tecla foi pressionada, interrompendo a pausa.

O resultado da divisão de s/60 fornece o tempo da pausa em segundos.

– PLOT x,y

Na tela do TK 90X existem 256 x 192 pixéis, cada um deles controlado individualmente pelo comando PLOT. No modo normal de operação, o comando PLOT vai setar o bit correspondente àquele pixel, no arquivo de imagem. O comando também faz o armazenamento das coordenadas do último pixel plotado nas variáveis de sistema LSTPLOT e COORDS, nos endereços 23677 e 23678.

Qualquer comando de cor do TK 90X pode ser utilizado dentro de uma declaração PLOT, e o emprego de PLOT OVER 1 significa, em equivalência ao TK 85, o comando UNPLOT, que apaga o ponto setado. Com o uso de PLOT INVERSE 1, simulamos um comando que "não plota", porque inverte as cores da tinta e do papel. O comando PLOT transfere a cor permanente para toda a área do caractere envolvido, mas não afeta nenhum outro comando de cor, a menos que ele esteja especificado, isto é, PAPER 8; FLASH 8; BRIGHT 8.

– POKE end, b

Este comando permite que o usuário armazene, diretamente nos endereços da memória RAM do TK 90X, seus valores especificados. A faixa aceitável de endereçamento começa com 0 e termina em 65535 (p/48K de RAM) valendo a ressalva que os endereços entre 0 e 16383, pertencentes à memória ROM (memória apenas de leitura), aceitem os valores a ser introduzidos nos endereços (não transmitem mensagem de erro), mas não os escrevem nesses endereços – simplesmente despreza-os. A faixa aceitável para os valores de b, conforme no capítulo anterior, é de valores de bytes, ou seja, de 0 a 255. Quando são valores negativos, na faixa de -255 a -1, a eles é adicionado o valor 256, para convertê-los em positivos.

– PRINT...

Este comando permite ao usuário diversas "formatações de dados de saída", na tela do TK 90X.

As mensagens ou os símbolos a serem impressos são separados do anterior por "delimitadores", que determinam onde e se devem ou não ser impressos.

O delimitador ";" (ponto e vírgula) significa que o próximo item a ser impresso deverá vir imediatamente após o existente, sem espaços intermediários.

O delimitador "," (vírgula) significa tabular até a próxima metade da linha que está sendo impressa na tevê, ou seja, divide a tela horizontalmente em duas metades iguais.

O delimitador "i" (apóstrofo) significa pular uma linha, começando uma impressão no início da próxima linha.

A posição de impressão também pode ser "formatada" através do uso de TAB y e AT x, y.

Os itens a serem impressos também podem ser coloridos, tanto provisória quanto permanentemente, através dos comandos de cor, ou através de acesso direto às cores desejadas.

Note que todos os comandos de cores, bem como os delimitadores e controladores de posição de impressão, também podem ser utilizados através dos seus códigos apropriados (CHR$...), o que pode ser muito útil.

– RANDOMIZE x

Este comando inicializa o valor armazenado na variável de sistema INITRND, nos endereços 23670 e 23671, ou 5C76 e 5C77.

Se x não é especificado, o valor de INITRND é tomado dos dois bytes menos significativos da variável de sistema TVCOUNT, nos endereços 23672, 23673 e 23674. Se foi declarado um valor para x, este número é copiado em INITRND.

– READ x1, x2, x3,...

Este comando, usado conjuntamente com uma listagem de dados em uma declaração DATA, pode ser considerado como sendo uma série de comandos LET, já que faz a atribuição aos nomes das variáveis contidas na linha READ, com os seus respectivos valores contidos na linha DATA.

O número de valores de uma linha DATA deve obrigatoriamente ser igual ao número de nomes de variáveis contidos em uma linha READ, pois se assim não for ocorrerá o surgimento de uma mensagem de erro.

– REM... (DO INGLÊS *REMARKS*, OU OBSERVAÇÕES)

Este é um comando muito útil para o usuário, e não tem efeito algum sobre o processamento do programa em Basic. Note que, em uma linha que inicie com o comando REM, se houver qualquer outro comando após o REM, separado por dois pontos, ele não será processado, sendo considerado, pelo interpretador Basic, como declaração REM.

– RESTORE x

Este comando é usado em conjunto com uma listagem de um comando DATA, ou mais. Se X = 0, então o apontador da variável de sistema ENDDATA, de endereços 23639 e 23640, ou 5C57 e 5C58h, é direcionado para apontar em direção ao endereço após a área de programa. Se foi especificado um valor para x, então ENDDATA é ajustado para apontar para o endereço depois do início daquela linha, se existir, ou a primeira linha posterior àquela. Se x exceder o valor 9999, que é o último valor válido para números de linhas de programa, então a variável de sistema ENDDATA apontará para o último endereço da área de programas da memória RAM.

– RETURN

Este comando procura a última entrada na pilha do GOSUB. Se essa entrada for um número válido de declaração, então o interpretador Basic executará a próxima declaração. Se a entrada não for válida, então surgirá a mensagem de erro adequada.

– RUN x

Este importante comando em Basic permite que o usuário execute seus programas. Se não foi especificado um valor para x, então significa que o usuário quer que o interpretador procure, na área de programas Basic da memória, uma linha com aquele número, ou a primeira linha após aquele número, caso não exista a linha x, para então interpretá-la.

O comando RUN faz a inicialização de todas as variáveis existentes na memória, ou seja, ele é equivalente ao comando CLEAR acrescido do comando GOTO primeira linha.

– SAVE

Este comando está muito bem explicado no capítulo 23 do manual do TK 90X. Para outros detalhes, veja neste capítulo o comando LOAD.

– SOUND x, y

Este comando emite uma nota musical através do alto-falante da televisão. O valor x é a duração em segundos, e o valor y é o tom da nota, em função da nota

DO central, na faixa de valores entre - 60 e + 69,8. Tanto x quanto y podem ser expressões numéricas.

Note que enquanto o comando SOUND estiver ativado, ele não permite interrupções (BREAK) através de Basic, porque sua rotina de comandos na ROM não checa se a tecla BREAK foi pressionada. Após a sua execução, pode-se dar um BREAK.

Experimente o seguinte programa, para testar tempo e tom das notas musicais emitidas pelo TK 90X:

```
10 LET a=- 60

20 SOUD RND * .05, a

30 LET a = a + 1

40 PRINT a,

50 GOTO 20
```

Veja que, quando a atingiu o valor 70, o computador emitiu a mensagem de erro.

Acrescente as linhas

```
15 POKE 23692,255 e
45 IF a = 69 THEN RUN
```

– STOP

Sempre que este comando for interpretado, surgirá a mensagem de erro "9 STOP executado". O uso do comando CONT, em modo direto, permite a execução do programa a partir do comando imediatamente após o comando STOP.

Assim como uma situação de erro, o efeito do erro "STOP" é o de transferir o número do erro para a variável de sistema ERRCD de endereço 23610, ou 5C3Ah. O número do erro é sempre um a menos que o código do erro que aparece antes da mensagem de erro.

Uma vez ocorrido um erro, o interpretador interrompe a execução da rotina do programa e salta para a rotina de manipulação de erros. Então, o número do erro é transferido para a variável de sistema ERRCD e o endereço de retorno ao modo imediato, localizado em 4867, ou 1303h, é coletado. Este endereço de retorno está sempre presente durante a interpretação de um programa, nos dois endereços abaixo da pilha do GOSUB. Este endereço está sempre apontado pela variável de sistema P ERR, de endereços 23613 e 23614, ou 5C3D e 5C3Eh.

– TRACE (1 OU 0)

Comando de saída, que, quando acionado (TRACE 1), mostra o número da linha de programa Basic que está sendo executada, após o sinal de #.

Para cessar o comando, usa-se ou TRACE 0 ou TRACE.

Infelizmente, na minha opinião, o campo de saída do comando é a própria tela, não havendo preocupação dos projetistas em não atrapalhar a saída de dados do programa, abrindo, por exemplo, quando do acionamento do comando, uma janela na parte inferior do vídeo. (Existe uma variável de sistema que armazena o número da linha de programa que está sendo processada).

– VERIFY

A presença deste comando em Basic do TK 90X é muito tranqüilizadora para o usuário, pois permite que qualquer programa gravado numa fita cassete, ou num microdrive, seja comparado com o conteúdo da memória, acusando, através de uma mensagem de erro, se o conteúdo da fita está diferente do conteúdo da memória.

AS FUNÇÕES BASIC

– ABS

Esta função converte todos os números negativos em positivos e não altera os números positivos.

– ACS

Retorna o arco do co-seno do ângulo em questão, em radianos.

– AND

Esta função é, na realidade, uma operação binária que requer dois operandos. Se os dois operandos são logicamente verdadeiros, então a operação é executada. Entretanto, se um dos operandos for logicamente falso, a operação deixa de ser executada. Uma expressão numérica "falsa", equivalerá, para a operação binária, ao valor 0, e uma cadeia de caracteres também "falsa", terá comprimento zero e conteúdo nulo. Quando o resultado é verdadeiro, o primeiro operando é o que é mostrado ao usuário.

– ASN

Retorna o arco do seno do ângulo em questão, em radianos.

– ATN

Retorna o arco da tangente do ângulo em questão, em radianos.

– ATTR

Esta função tem a forma ATTR (linha, coluna). Retorna ao usuário o valor armazenado no byte do arquivo de atributos daquela respectiva posição. A função é equivalente a:

PEEK (22528 + linha * 32 + coluna)

O valor do atributo pode ser considerado como sendo:

INK + PAPER * 8 + BRIGHT + FLASH * 128

– BIN

Esta é uma função muito interessante, pois permite que qualquer número inteiro decimal entre 0 e 65535 seja digitado em sua forma binária. Qualquer número que possua até 16 dígitos binários é permitido.

O programa a seguir mostrará, a partir de números binários, decimais de 0 a 10.

```
10 FOR a =BIN 1 TO BIN 1010 STEP 00000001
20 PRINT a
30 NEXT a
```

Note apenas que os números binários não são impressos em comandos de saída, a menos que se use uma rotina em Basic que o faça.

– CHR$

Esta função retorna ao usuário o caractere correspondente a um determinado código.

O programa a seguir mostra a impressão desses caracteres. Os comandos CONT ou NEXT a, deverão ser utilizados diversas vezes, para não interromperem o loop de saída, quando for impressa a mensagem de erro referente aos códigos de cor.

```
10 FOR a=0 TO 255
20 PRINT a; TAB 8; CHR$ a
30 NEXT a
```

– CODE

Esta função retorna ao usuário o código relativo ao primeiro caractere da respectiva série de caracteres, correspondendo ao inverso da função anterior. Se a série de caracteres for nula, o código será zero.

– COS

Retorna o co-seno do ângulo em questão, em radianos.

– EXP

Para um dado argumento x, a função retorna o valor "e elevado à potência x", onde e vale 2,7182818. . .

– FN

No TK 90X podem existir até 26 funções numéricas e até 26 funções alfanuméricas definidas pelo usuário. A função Basic "FN" deve vir seguida de uma simples letra, para funções numéricas, ou de simples letras seguidas do símbolo "$", para funções alfanuméricas, e o argumento requerido pela função deve vir entre parênteses.

– IN endereço, byte

O argumento desta função é usado como código de endereçamento de uma porta de entrada, e o valor resultante deverá estar na faixa de 0 a 255. Se uma porta não estiver sendo usada, o valor nela será 255.

– INKEY$

Esta função permite que o teclado do TK 90X seja lido sem que se faça uso do comando INPUT. Se uma simples tecla é pressionada, então é criada uma *série* de caracteres de comprimento um, cujo conteúdo é aquela tecla pressionada.

– INT

Retorna ao usuário a parte inteira de um número decimal, ou seja, arredonda aquele número para baixo. Números negativos são primeiramente truncados, para então se subtrair 1 dele e em seguida o resultado ser arredondado.

Portanto, + 4.66 resultará em +4, enquanto -5.66 será truncado para -5, ou arredondado para -6.

– LEN

Esta função calcula o comprimento, em número de caracteres de uma dada *série* de caracteres. Uma *série* nula terá comprimento zero.

Entretanto é bom salientar que as cores, tanto de papel, quanto de tinta, bem como os atributos obtidos diretamente, via teclado, serão considerados para efeito de comprimento de *séries,* e cada inclusão dessa consumirá dois caracteres de comprimento.

– LN

Retorna o logaritmo da base e, do argumento dado.

– NOT

Esta é uma função muito interessante por ser a única que fornece o "estado lógico" de um valor no sistema do TK 90X. Entretanto é importante notar que o resultado é invertido e que o verdadeiro resultado lógico é dado pela operação lógica "NOT NOT x".

Isto é, para x=2, um valor "verdadeiro"

NOT x será 0, que é "falso"

NOT NOT x será 1, que é a resposta correta e "verdadeira".

– OR

Esta é uma operação binária que requer dois operandos. Se um dos dois operandos for logicamente verdadeiro, então a operação será efetuada. Entretanto, se nenhum dos dois operandos for verdadeiro, então a operação será logicamente falsa. Um resultado verdadeiro terá o valor 1 se o segundo operando for zero e terá o valor do primeiro operando se o segundo operando não for zero.

– PEEK

Esta função retorna o valor encontrado no endereço especificado da memória. O resultado será sempre um inteiro entre 0 e 255, inclusive.

– PI

Uma função muito particular do TK 90X. A representação em ponto flutuante de PI/2 é armazenada como uma constante na "tabela de constantes da ROM". Este valor é de lá coletado, duplicado e colocado à disposição do usuário.

O valor de PI é aproximadamente 3.141592653. . .

– POINT

Esta função tem a forma POINT (x plot, y plot), e retorna o valor 1 se o pixel daquela posição estiver impresso na tela, e o valor 0, se não estiver impresso.

– RND

No TK 90X, os números "aleatórios" são gerados por um pseudogerador de números aleatórios. A variável de sistema INITRND, de endereços 23670 e 23671, ou 5C76 e 5C77h, é coletada, modificada e reinicializada a cada chamada da função RND. O número aleatório gerado para o usuário é o novo valor de INITRND dividido por 65536.

O valor de INITRND é zerado após cada inicialização do TK 90X. Entretanto, existe uma seqüência de números entre os valores inteiros de 0 a 65535. Nenhum valor será repetido até que tenham sido usados 65535 números.

Os dois exemplos Basic a seguir mostram estas ações.

1- A natureza de RND

```
10 PRINT RND: FOR a=1 TO 65535: POKE 0, RND: NEXT a: PRINT RND
```

Este pequeno programa leva cerca de vinte minutos para ser executado, mas mostra que o mesmo número RND é gerado após 65535 chamadas. O comando intermediário POKE é apenas uma sugestão.

2- A modificação de INITRND

```
10 POKE 23670,0: POKE 23671,0
20 LET INIT=0
30 LET INIT=INIT + 1
40 LET INIT=INITx75
50 LET INIT=INIT-INT (INIT/65537) x 65537
60 LET INIT=INIT-1
70 PRINT INIT/65536, RND
```

A linha 10 zera a variável de sistema INITRND. O programa mostra que o mesmo resultado é produzido através das modificações das linhas 30 a 60, ou pela chamada de RND.

– SCREEN$

Função muito interessante do TK 90X, que foi pouco explorada no manual.

A forma desta função é SCREEN$ (linha, coluna), que retorna ao usuário uma *série* contendo o caractere da posição especificada. A função reconhece os códigos de caracteres da faixa 32 a 127, tanto normais ou invertidos.

É possível, no entanto, fazer SCREEN$ reconhecer caracteres gráficos definidos pelo usuário. Basta alterar a variável de sistema PTBLCHR, nos endereços 23606 e 23607, ou 5C36 e 5C37h, tornando o endereço apontador dos caracteres, para efeito de reconhecimento de SCREEN$, o primeiro da área de caracteres gráficos.

Em outros termos, você faz POKE 23606, 88 e POKE 23607, 254. Com estes dois comandos a tabela de caracteres foi desviada para a área de caracteres gráficos, e SCREEN$ vai reconhecer o seu caractere gráfico, através, por exemplo, do co-

mando IF SCREEN$ (linha, coluna) = CHR$ 32 THEN. . . (obrigatoriamente usa-se CHR$ 32, que é, na tabela de código do Apêndice D do manual, o espaço em branco). Para retornar aos caracteres normais de impressão, usa-se POKE 23606,0 e POKE 23607,60.

Esta função trabalha comparando o conteúdo dos oito bytes do caractere especificado com os oito bytes que representam um caractere, armazenados no gerador de caracteres, da memória ROM, que começam no endereço 15360.

Apesar da quantidade de trabalho envolvida pelo hardware da máquina, esta função é surpreendentemente rápida. Isto pode ser demonstrado pelo programa:

```
10 PRINT "&"; SCREEN$ (0,0)
```

que aparece quase instantaneamente.

– SGN

Esta função retorna o sinal do número em questão, ou seja, retorna "+1" para todos os números positivos e "-1" para todos os números negativos.

– SIN

Retorna o seno do ângulo em questão, em radianos.

– SQR

Esta função mostra ao usuário a raiz quadrada do número especificado, desde que esse número seja positivo.

O TK 90X calcula a raiz quadrada de um número elevando-o a 0.5.

– STR$

Esta função converte uma variável numérica em variável alfanumérica.

– TAN

Retorna a tangente do ângulo em questão, em radianos.

Esta tangente é encontrada, a partir do seno e do co-seno daquele ângulo, que são calculados separadamente, e então divididos. O tempo de execução desta função é, portanto, o dobro do gasto pela função seno ou pela função co-seno.

– USR

No TK 90X a palavra-chave USR seguida de uma expressão numérica é diferente da forma USR seguida de uma *série*.

– USR número

Esta função, desta forma, é muito importante para aqueles que desejam utilizar linguagem de máquina no TK 90X. Ela serve para "chamar" rotinas em linguagem de máquina criadas pelo usuário, ou mesmo rotinas da ROM, a partir do número especificado, que deve ser o primeiro endereço da rotina que será executada.

No caso de rotina em linguagem de máquina criada pelo usuário, o Z 80 interrompe a execução do programa monitor armazenado na ROM do TK 90X, e imediatamente inicia a execução da rotina chamada a partir do endereço especificado.

– USR string

A função USR retorna o endereço da memória RAM correspondente ao caractere gráfico definido pelo usuário.

O argumento de USR string deve ser uma *série* de um caractere, que deve estar entre A e U, ou a e u.

No TK 90X standard, com 16K de memória, a área de caracteres gráficos definidos pelo usuário começa em 32600, ou 7F58h.

Na versão de 48K de RAM, esta área começa em 65368, mas estas áreas podem perfeitamente começar em outros endereços, alterando-se o endereço apontado pela variável de sistema UDGRAPH, armazenada nos endereços 23675 e 23676, ou 5C7B e 5C7Ch.

Digite em modo direto:

PRINT USR "a", e terá como resposta 65368

Digite em modo direto:

PRINT PEEK 23675 + 256 * PEEK 23676, e a resposta também será 65368, que é então o primeiro endereço do caractere gráfico correspondente à letra A.

Agora digite:

PRINT PEEK 23675 e PRINT PEEK 23676 separadamente

E dê POKE nesses endereços, para mudar o endereço inicial da área de caracteres gráficos.

Digite novamente a frase PRINT PEEK 23675 + 256x PEEK 23676 para saber o novo endereço inicial da área de gráficos, o que significa digitar PRINT USR "a".

Agora experimente rodar o seguinte programa, em modo direto, considerando-se o endereço inicial da área de gráficos como sendo END.

FOR f= END TO END + 8:POKE END, RND * 255:NEXT f e entrando em modo gráfico (cursor G), veja o que aconteceu com a tecla A.

Percebeu? Você ainda acha que o TK 90X pode criar apenas 21 caracteres gráficos? O tamanho da área de caracteres gráficos definidos pelo usuário é variável, dependendo apenas de quantos caracteres se queira criar para aquele programa. Não esqueça de dar os POKEs respectivos para mudar o endereço inicial da área de caracteres, bem como os POKEs necessários para se retornar ao tamanho padrão. Se quiser, por exemplo, armazenar 42, em vez dos 21 caracteres iniciais, você terá de aumentar a área de caracteres gráficos em 21x8, ou seja 168 endereços. Em 48K de RAM, o endereço inicial é 65368. Portanto, se precisamos de mais 168 endereços, devemos mudar o início para 65368-168, ou seja, para 65200. Se dividirmos 65200 por 256, obteremos 254,6875, ou seja, 254 e alguns quebrados. Multiplique 254 por 256, e o resultado será 65024. Mas 65200 - 65024 = 176. Então, para uma área de 48 caracteres gráficos definidos pelo usuário, precisamos fazer dois POKEs: um, POKE 23675, 176 e outro POKE 23676, 254. Digite agora, em modo imediato, PRINT USR "a", e repita então o programinha em modo imediato, mostrado acima. Entre em modo gráfico e pressione "a". Retorne ao tamanho inicial da área de gráficos, e repita o programa, seguindo o procedimento normal.

Nota: o comando NEW não afeta essa área de caracteres gráficos.

Entendeu o procedimento? Treine bastante, pois isto é muito interessante. Lembre-se que a área de gráficos só retorna ao seu comprimento padrão, através dos respectivos POKEs, ou então se desligar o TK 90X.

Acostume-se a utilizar um papel quadriculado à parte para desenhar os seus caracteres gráficos, se não quiser utilizar UDG 2, para facilitar a conversão em valores decimais ou em valores binários, através de BIN.

– VAL

Esta função requer uma *série* como argumento, cujo conteúdo deve ser numérico. Ela então converte em valor numérico a *série* dada.

É uma função muito útil no sentido de economizar memória, visto que um número qualquer para ser armazenado na memória RAM do TK 90X requer 5 bytes. Utiliza-se esta função em *séries*, e não os números diretos, como por exemplo, em vez de se usar GOTO 900 emprega-se GOTO VAL "900".

– VAL$

Esta função requer uma *série* entre aspas como argumento e retorna ao usuário a *série* original.

Experimente digitar o programa:

```
10 LET a$= "123"
20 PRINT VAL a$
30 PRINT VAL$ "a$"
40 PRINT VAL$ "a$" + STR$ (VAL a$)
```

E assim nós encerramos as explicações de todos os comandos e todas as funções disponíveis na linguagem Basic do TK 90X. Agora, veremos um pouco os caracteres de controle.

CARACTERES DE CONTROLE

No TK 90X existem 12 caracteres de controle que podem ser utilizados pelo usuário.

– CHR$ 6 - vírgula

Equivalente a PRINT, que faz a próxima posição de impressão saltar meia tela, verticalmente.

– CHR$ 8 - caractere à esquerda

A posição de impressão é modificada, no sentido de "retornar" um caractere à esquerda (tecla de backspace em máquinas de escrever).

Um uso útil e particular que se pode obter com este caractere é o de sublinhar certas letras ou palavras, por exemplo:

10 PRINT "TK 90X"; CHR$ 8; CHR$ 8; CHR$ 8; CHR$ 8; CHR$ 8; CHR$ 8; OVER 1; "_____"

Para cada caractere que se quer sublinhar deve haver um CHR$ 8.

Outra utilização é a acentuação de palavras em português, que pode até ser mais fácil do que se usássemos UDG 0...

– CHR$ 9 - caractere à direita

Inversamente ao anterior, este caractere de controle pula uma posição para a direita.

– CHR$ 13 - ENTER

A posição corrente de impressão é alterada para ser impressa no início da próxima linha. Equivale a se usar " ' " após o comando PRINT, ou mesmo a outro PRINT, numa mesma linha de programa.

– CHR$ 16 até 21 - comando de cores

Estes caracteres de controle produzem cores provisórias. Eles são, respectivamente, INK, PAPER, FLASH, BRIGHT, INVERSE e OVER.

Por exemplo:

10 PRINT CHR$ 16; CHR$ 3; CHR$ 18; CHR$ 1; CHR$ 21; CHR$ 1; "TK 90"; CHR$ 19; CHR$ 1; CHR$ 17; CHR$ 1; "TK 90"

– CHR$ 22 - AT

Tabulação da posição de impressão, a partir dos dois próximos números após CHR$ 22. O primeiro número refere-se às linhas e o segundo, às colunas.

Por exemplo:

PRINT AT 12,6; "TK 90X" é idêntico a

PRINT CHR$ 22; CHR$ 12; CHR$ 6; "TK 90X"

– CHR$ 23: TAB

A posição de impressão é modificada, no sentido de ser impressa na coluna especificada pelos dois caracteres que seguem o CHR$ 23. Experimente o programa:

10 LET a=1: POKE 23692,255

20 PRINT PAPER 7; CHR$ 23; CHR$ (RND * 5); CHR$ 0; PAPER RND * 6.5; CHR$ 23; CHR$ (RND * 25 + 5); CHR$ 0; a

30 LET a = a + 1: GOTO 20

Capítulo III

VARIÁVEIS QUE CONTROLAM O SISTEMA

Os bytes na memória, do endereço 23552 até o endereço 23733, são reservados para uso específico do sistema. Pode-se ver o conteúdo de qualquer desses endereços, e, convém até anotar esses valores, pois podem ser úteis mais tarde. Para ver esses conteúdos, digite:

PRINT PEEK endereço

E você também pode mudar esses valores iniciais, sem perigo algum de alterar o hardware standard da máquina. É somente praticando, e "fuçando", que se descobre como ocorrem as coisas dentro dessa pequena maravilha. O máximo que pode acontecer, quando se altera uma variável de sistema, é um "crash", e o passo seguinte nesse caso é desligar a máquina e tornar a ligá-la. Os nomes dessas variáveis não têm nada a ver com a linguagem Basic, e o computador não as reconhece pelo nome (felizmente. . .). Você pode usar em seus programas nomes semelhantes de variáveis — o computador não vai confundir a sua variável com uma variável de sistema.

As abreviações utilizadas na coluna 1 da tabela a seguir têm o seguinte significado:

x — alterando o valor inicial da variável, provavelmente ocorrerá um "crash" no sistema.

n — não haverá efeito algum, alterando o valor inicial.

O número da coluna 1 é o número de bytes daquela variável. Para dois bytes, o primeiro é o menos significativo — o contrário do que normalmente se espera. Portanto, para se dar um POKE de um valor v numa variável de sistema de dois bytes, de endereço n, use

POKE n, v-256 * INT (v/256)

POKE n + 1, INT (v/256)

E, para ver o valor, use a expressão:

PRINT PEEK n + 256 * PEEK (n + 1)

Se quiser alterar algum endereço n, como no caso de aumentar a área de caracteres gráficos definidos pelo usuário, visto no capítulo anterior, você deve efetuar alguns cálculos:

Chamemos de bms, o byte menos significativo e

BMS, o byte mais significativo

VARSIST — a variável de sistema que se queira alterar. Os cálculos são:

BMS = INT (n/256)

bms = n - BMS * 256

E, a seguir:

POKE VARSIST, bms

POKE VARSIST + 1, BMS

VARIÁVEIS DO SISTEMA — TABELA DE DADOS

Nota	Endereço	Nome	Conteúdo
n8	23552/5C00	KBDWORK	Esta variável consiste em 8 bytes, cada um deles contendo informação sobre a tecla pressionada, ou o período de repetição o código em modo "extended" etc.
n1	23560/5C08	KEYPRS	Esta variável contém sempre o código da última tecla pressionada, tendo em conta o modo. Só é alterada quando se aperta outra tecla. A repetição automática afeta a variável. Passando-se a zero, pode-se verificar se foi pressionada alguma tecla.

Nota	Endereço	Nome	Conteúdo
1	23561/5C09	RPTDLAY	Tempo, em 60 ciclos por segundo, durante o qual é necessário pressionar uma tecla para acionar o auto-repeat (repetição automática). Possui o valor inicial 35, podendo ser alterado para mais ou para menos. (Cuidado com valores muitos baixos – a repetição é muito rápida. . .)
1	23562/5C0A	RPTCCLE	Atraso, em 60 ciclos por segundo, entre as sucessivas repetições de uma tecla. Inicialmente contém o valor 5. Pode-se diminuir para o mínimo de 1, a fim de tornar mais rápida a repetição.
n2	23563/5C0B	PT DEF	Endereço dos argumentos das funções definidas pelo usuário. Valor inicial 0.
n1	23565/5C0D	K CLR	Quando é escrito diretamente no teclado um código de comando, o segundo byte, ou seja, a cor e o uso ou não de FLASH, é guardado aqui, enquanto o código de INK, PAPER, FLASH ou INVERSE é impresso. Depois disso, a ROM consulta este byte para impressão a seguir do código de comando.
n2	23566/5C0E	TVCLR	Esta variável é usada pela rotina de impressão, para guardar AT, TAB e os comandos de cor dirigidos para a tela.
x38	23568/5C10	PSTRM	Esta variável é usada para guardar os deslocamentos de CHCADD. Para um total de 19 arquivos, sendo 16 do usuário e 3 da máquina, existe um deslocamento. Quando este é somado a CHCADD, aponta para um endereço que constitui o início da rotina de tratamento desse arquivo.
2	23606/5C36	PTBLCHR	Indica o endereço do conjunto de caracteres, menos 256 (incluindo no conjunto todos os caracteres de código entre 32 e 127, padrão ASCII). A variável contém normalmente o endereço 15360 ou 4C00 (conjunto de caracteres da ROM), mas este valor pode ser alterado, levando-a a apontar para um novo conjunto de caracteres definido pelo usuário.
1	23608/5C38	BUZCCLE	Som produzido pelo computador quando uma linha de programa em edição ultrapassa o comprimento de 23 linhas da tela (tela cheia). Para alterar este comprimento, modifique o conteúdo desta variável, inicialmente em 64.
1	23609/5C39	KCLICK	Duração do som produzido ao se pressionar uma tecla. Pode-se aumentar seu valor inicial, a fim de se tornar mais "simpático" o som produzido.
1	23610/5C3A	ERRCD	Indica o código de mensagens menos 1. Inicialmente em 255. Se for alterada por uma listagem de programa Basic, quando este programa terminar, a máquina imprime a mensagem de erro desejada.
x1	23611/5C3B	SFLAG 0	Esta variável contém um byte cujos valores contêm flags (indicadores) que controlam o sistema Basic.
x1	23612/5C3C	SFLAG 1	Flags associados à impressão na tela e na impressora.
x2	23613/5C3D	P ERR	Esta variável aponta para um elemento da pilha da máquina. Quando ocorre um erro, é para este endereço que a execução salta depois de a pilha ser limpa por RST 08 (instrução em código de máquina). Alterando este elemento, é possível escrever novas rotinas de tratamento de erro.
n2	23615/5C3F	PLIST	Esta variável aponta para o endereço de retorno da pilha da máquina, para o qual salta a execução após uma listagem automática.

Nota	Endereço	Nome	Conteúdo
n1	23617/5C41	CURSOR	Esta variável especifica o cursor em uso (K, L, C, E ou G).
2	23618/5C42	LNJMP	Linha para onde deve saltar a execução do programa. Usada nas instruções GOTO e GOSUB.
1	23620/5C44	INSTRNR	Número da declaração da linha para onde deve saltar a execução. Alterando primeiro a variável LNJMP e depois esta, força-se um salto para uma determinada declaração da linha.
2	23621/5C45	EXCLINE	Número da linha onde se encontra a instrução em execução.
1	23623/5C47	SUBLEXC	Número da instrução em execução dentro de uma linha.
1	23624/5C48	BORCLR	Esta variável contém os atributos da parte inferior da tela. É uma experiência muito interessante alterar seu valor e imprimir mensagens com diversos valores.
2	23625/5C49	CURLINE	Número da linha onde se encontra o cursor.
x2	23627/5C4B	VARADD	Indica o endereço inicial da área de variáveis Basic, quando da execução de um programa.
n2	23629/5C4D	XVARADD	Endereço da variável em atribuição.
x2	23631/5C4F	CHCADD	Indica tabela de endereços usada por PSTRM.
x2	23633/5C51	IOADD	Indica o endereço da tabela anterior (de endereços de tratamentos de arquivos) que está sendo usado pela rotina de tratamento.
x2	23635/5C53	PROGBAS	Indica o endereço de início da área de programas Basic.

Nota	Endereço	Nome	Conteúdo
x2	23637/5C55	NEXEXC	Indica o endereço da linha de programa Basic seguinte à que está sendo executada.
x2	23639/5C57	ENDDATA	Aponta para o separador final do último elemento de uma listagem DATA. Se não existir DATA no programa, aponta para 80h, no fim das informações do canal.
x2	23641/5C59	INADD	Endereço da instrução a ser digitada no teclado.
2	23643/5C5B	CURADD	Endereço do cursor no interior da linha avaliada.
x2	23645/5C5D	CHNXADD	Endereço do próximo caractere a ser interpretado.
2	23647/5C5F	SYCHADD	Endereço do caractere que se encontra depois do sinal "?"
x2	23649/5C61	WORK PT	Endereço da área de trabalho temporário.
x2	23651/5C63	STKEND	Endereço inferior da pilha de cálculo.
x2	23653/5C65	ADSPFREE	Endereço do início da memória livre.
n1	23655/5C67	BREGCAL	Registro de cálculo usado para fins diversos (contagem).
n2	23656/5C68	MEMCADD	Endereço usado para as seis memórias de cálculo (normalmente MENSPCAL, mas nem sempre).
1	23658/5C6A	SFLAG 2	Mais flags controladoras do sistema.
x1	23659/5C6B	SIZE	Número de linhas (incluindo uma em branco), da parte inferior da tela. Ela pode ser alterada fazendo o programa ocupar toda a tela, mas cuidado: antes de cada instrução que ocupe esta área (por exemplo, INPUT), a variável deverá conter o valor original (2), ou seja, reabrir espaço na parte inferior.

Nota	Endereço	Nome	Conteúdo
2	23660/5C6C	LIST NR	Número da linha superior em listagem automática.
2	23662/5C6E	CONTJMP	Número da linha para onde salta o comando CONT.
1	23664/5C70	CONTNR	Número da instrução da linha destino após o comando CONT.
n1	23665/5C71	SFLAG 3	Mais flags controladoras do sistema.
n2	23666/5C72	STRVLEN	Comprimento da variável *série* em atribuição.
n2	23668/5C74	SYTADD	Endereço do elemento seguinte na tabela de sintaxe da ROM. Esta tabela define o local onde se encontra a rotina correspondente a cada comando e o modo de obter a informação necessária.
2	23670/5C76	INITRND	O primeiro número para a instrução RND. Esta variável é inicializada por RAND.
3	23672/5C78	TVCOUNT	Contador de imagens de 3 bytes, incrementado a cada ciclo de alimentação.
2	23675/5C7B	UDGRAPH	Endereço do primeiro caractere gráfico definido pelo usuário. RAMTOP não afeta esta área.
1	23677/5C7D	LSTPLOT	Usada para armazenamento provisório da coordenada x quando são realizados cálculos para definição da posição de PLOT.
1	23678/5C7E	COORDS	Idêntica à anterior, mas para a coordenada y.
1	23679/5C7F	POSIMPR	Número da posição de impressão, em 32 colunas.
1	23680/5C80	PRTADD	Byte menos significativo do endereço da posição que se segue para LPRINT (no buffer da impressora).

Nota	Endereço	Nome	Conteúdo
1	23681/5C81		Não usada pelo sistema. Alguns programas em linguagem de máquina têm a instrução RET (201) armazenada nesta posição, para evitar que, ao se dar o comando LOAD "" CODE, se tenha acesso aos seus códigos. . .
2	23682/5C82	HVBFFIN	Número de colunas (32) e linhas (24) do final da área de entrada.
2	23684/5C84	DFPSPRT	Endereço da posição PRINT no arquivo de imagens. Pode ser dirigida para outra posição.
2	23686/5C86	DFPSPRTL	Idêntica à anterior, mas para a parte inferior da tela.
x1	23688/5C88	HVPOS	Número de colunas para a posição PRINT.
x1	23689/5C89		Número de linhas para a posição PRINT.
x2	23690/5C8A	HVPOSL	Idêntica à anterior, mas para a parte inferior da tela.
1	23692/5C8C	SCRINC	Contador de "scroll" da imagem: contém uma a menos que o número de scrolls realizados antes de a imagem parar imprimindo a mensagem "scroll?". Se for colocado um valor maior aqui, a imagem rolará ininterruptamente.
1	23693/5C8D	ATCLRP	Atributos permanentes definidos por declarações INK, PAPER etc.
1	23694/5C8E	MASKCLRP	Usado para atributos transparentes. Qualquer bit igual a 1 indica que o atributo correspondente não é retirado de ATCLRP, mas do valor que já está na tela.
n1	23695/5C8F	ATCLRT	Atributos provisórios em uso, em instruções tipo PLOT, DRAW etc.
n1	23696/5C90	MASKCLRT	Como MASKCLRP, mas provisório.
1	23697/5C91	SFLAG 4	Outras flags controladoras do sistema.

Nota	Endereço	Nome	Conteúdo
n30	23698/5C92	MEMSPCAL	É nesta área que a unidade de cálculo aritmético pode guardar até seis números diferentes, em notação de ponto flutuante, com 5 bytes cada, utilizando memórias especiais.
2	23728/5CB0	NMIVCT	Ex-vetor de interrupção, não usado devido às características de programação da ROM. Pode ser usada sem restrições pelo usuário.
2	23730/5CB2	RAMTOP	Esta variável contém o endereço do último byte da área Basic.
2	23732/5CB4	MEMAVLB	Endereço do último byte físico da RAM.

Alguns comentários sobre as variáveis de sistema:

1 – KBDWORK

Esta variável armazena:

a- 255 se nenhuma tecla foi pressionada ou

b- o código do maior caractere em branco, à esquerda, impresso em uma tecla.

No último caso, o código pode ser considerado como sendo aquele caractere que INKEY$ produziria se CAPS LOCK (tecla 2) estivesse acionado.

Esta propriedade pode ser usada com vantagens, quando se usa INKEY$ em um programa. Digite o programa a seguir e veja o efeito que CAPS SHIFT ou SYMBOL SHIFT produzem quando pressionados com outras teclas:

```
10 PRINT AT 0,0; INKEY$; "      ": REM 4 espaços dentro das aspas
20 GOTO 10
```

Como se pode notar, o caractere produzido não depende apenas de qual tecla foi pressionada, mas também em que modo se encontrava o computador. Isto conduz a linhas meio "complicadas" quando se usa INKEY$, por exemplo, ao final de um programa:

```
9000 PRINT "Quer rodar outra vez? (s/n)": PAUSE 0:
IF INKEY$ = "s" OR INKEY$ = "S" OR INKEY$ = "NOT" THEN RUN
9010 NEW
```


Podemos utilizar também o byte do endereço 23552:

```
10 PRINT AT 0,0; CHR$ PEEK 23552; "      ";
20 GOTO 10
```

Você vai perceber que para qualquer tecla pressionada, o caractere produzido será aquele que aparece em branco no lado esquerdo da tecla que está sendo pressionada. A solução para o problema:

```
9000 PRINT "Quer rodar outra vez? (s/n)": PAUSE 0:
IF CHR$ PEEK 23552 = "S" THEN RUN
9010 NEW
```

Se pretende usar esta técnica em seus programas, proceda da seguinte maneira:

```
10 LET kbd = 23552
```

E mais adiante,

```
. . . IF CHR$ PEEK kbd = . . . THEN. . .
```

Agora experimente o programa abaixo, e saiba qual endereço desta variável utilizar em seus programas:

```
10 FOR f = 23552 TO 23559
20 PRINT INKEY$; "      "; CHR$ PEEK f;
30 GOTO 20
```

Quando quiser mudar para o próximo endereço, ou o próximo f, basta digitar NEXT f.

Notou que a penúltima tecla pressionada é impressa com a última, através de CHR$ PEEK f/ ? Retire este comando da linha 20 e preste atenção nas diferenças.

2 – KEYPRS

O conteúdo deste endereço vai produzir o código da última tecla pressionada, esteja ou não sendo pressionada outra. Como no caso da variável anterior, se mais

de uma tecla é pressionada, então o código da primeira a fazer contato tem prioridade sobre a outra. É importante notar que, embora CAPS SHIFT ou SYMBOL SHIFT sozinhas não alterem o conteúdo de KBDWORK, ou mesmo de INKEY$, juntas produzem o código 14, que normalmente é usado para representação de números em listagens Basic do TK 90X.

Existem outras quatro combinações de teclas que produzem valores para bytes de 23560 e códigos de INKEY$, que não aparecem no manual do TK 90X. São elas:

Teclas normalmente usadas para produzir:	Valor do código INKEY$
GRÁFICOS (CAPS SHIFT & 9)	15
TRUE VÍDEO (CAPS SHIFT & 3)	4
VÍDEO INVERSO (CAPS SHIFT & 4)	5
CAPS LOCK (CAPS SHIFT & 2)	6

3 – RPTDLAY e RPTCCLE

Experimente rodar o programa abaixo, que é auto-explicativo, e observe o que acontece com o dispositivo de repetição automática de teclado, e o tempo de leitura do teclado:

```
10 PRINT "RPTDLAY - 23561 = "; PEEK 23561 ' "RPTCCLE - 23562 ="; PEEK 23562

20 LET a=PEEK 23561: LET b=PEEK 23562

30 INPUT "Digite valor para mudar RPTDLAY"; m

40 INPUT "Digite valor para mudar RPTCCLE"; n

50 POKE 23561, m: POKE 23562, n

60 INPUT "Experimentar digitar qualquer coisa"; c$

70 PRINT c$

80 POKE 23561, a: POKE 23562, b

90 PRINT "Valor usado para RPTDLAY="; m; "Valor usado para RPTCCLE="; n

100 INPUT "Outra tentativa (s/n) ?"; a$

110 IF a$ = "s" THEN RUN

120 STOP
```


4 – MUDANDO MODOS DO CURSOR

A variável de sistema de endereço 23617 especifica o modo do cursor, e, através dela, o programador pode forçar apenas o modo gráfico e o modo extendido, no próximo comando INPUT.

Para modo gráfico: POKE 23617,2: INPUT a$: PRINT a$

Para modo extendido: POKE 23617,1 : INPUT a$: PRINT a$

Experimente o programa, alternando esses dois POKEs:

```
10 FOR f = 0 TO 255

20 PRINT f; TAB 8;

30 INPUT a$ : PRINT a$ ;: INPUT a$ : PRINT a$

40 NEXT f
```

Portanto, através de cursor, podemos apenas entrar em modo gráfico e modo extendido.

Através de uma variável "misteriosa", localizada no endereço 23658, podemos também entrar em modo maiúsculo (CAPS LOCK). Digitando-se POKE 23658,8, o endereço de SFLAG 2, automaticamente estamos com letras maiúsculas.

Isso pode facilitar em certos programas que possuam uma linha do tipo

```
100 INPUT "Quer rodar outra vez (s/n)"; s$: IF s$ = "S" OR s$ = "s" THEN RUN
```

Podemos passar esta linha para:

```
100 POKE 23658,8: INPUT "Quer rodar outra vez (s/n); s$: IF s$ = "S" THEN RUN: POKE 23658,0: REM Entra em modo maiúsculo e sai desse modo através de POKE 23658,0
```

Pode-se, também, dependendo do tipo de programa, perguntar logo de início se deseja trabalhar somente em letras maiúsculas ou não.

Esta técnica é muito interessante quando existem nos programas diversos comandos tipo INPUT e INKEY$.

5 – CORES DA TELA

BORCLR

Embora o TK 90X proteja o usuário contra acidentes que fazem INK e PAPER em comandos INPUT serem da mesma cor, você, se quiser, pode alterar somente as duas linhas da parte inferior do vídeo, alterando os valores da variável de sistema BORCLR, de endereço 23624. Isto pode ser particularmente interessante, quando não se quer de modo algum que alguém veja a listagem de seu programa, e, desta forma, não é permitido imprimir qualquer coisa na parte inferior da tela.

É interessante notar que, através do uso dessa variável, podemos também fazer FLASH e BRIGHT nas linhas de INPUT, adicionando:

128 para FLASH 1 e

64 para BRIGHT 1

O novo valor de BORCLR permanece inalterado até ser executado um novo comando BORDER ou um NEW.

Por exemplo, para produzir uma borda magenta com tinta amarela e papel magenta, nas linhas inferiores, com FLASH 1 e BRIGHT 1, digite:

```
BORDER 3:POKE 23624, 128 + 64 + 3 * 8 + 6 :CLS
```

ATCLRP – MASKCLRP – ATCLRT – MASKCLRT

Estas variáveis do sistema simplesmente armazenam os valores que estão sendo usados para INK, PAPER, BRIGHT e FLASH. A diferença entre os nomes, ou seja, as letras P e T significam Permanente ou Temporário (estas, quando incluídas em algum comando de saída de dados).

ATCLRP e ATCLRT são pouco utilizadas, mas para referência os valores a serem adicionados nelas, para tratamento de cores, são:

ATCLR = 8 * (cor de PAPER) + cor da INK + (128 para FLASH 1) + (64 para BRIGHT 1)

MASKCLRP e MASKCLRT são mais úteis. Ocupam respectivamente os endereços 23694 e 23696.

Qualquer bit de um byte de uma destas variáveis, que tiver o valor 1, mostra que o bit correspondente de atributos para impressão deve ser tomado da posição corrente de impressão da tela, como no caso de INK, PAPER, BRIGHT e FLASH 8.

O mais interessante destas variáveis é que nós podemos limitar seu efeito a apenas cores primárias (azul, vermelho e verde), de códigos do TK, respectivamente 1, 2 e 4. Os valores para POKE nas variáveis são:

BIT	*EFEITO*	*VALOR*
0	azul INK 8	1
1	verm INK 8	2
2	verde INK 8	4
3	azul PAPER 8	8
4	verm PAPER 8	16
5	verde PAPER 8	32
6	BRIGHT 8	64
7	FLASH 8	128
		255

6 – O RELÓGIO DO TK 90X

Existe, no conjunto das variáveis do sistema do TK 90X, uma variável, que constantemente altera seus valores, como que contando "tempo", chamada TVCOUNT, que possui seu valor inicial incrementado 60 vezes por segundo. Esta freqüência é igual à freqüência ou ciclagem da área, e também o número de vezes por segundo que uma nova imagem é enviada à televisão para formar o que vemos.

TVCOUNT é inicializada em zero, quando o computador é ligado, e incrementada a cada 20 milissegundos aproximadamente, a menos que um comando SOUND, uma operação com fita cassete ou um outro periférico estejam ligados ao TK 90X.

Esta é a justificativa para o comando PAUSE n, que significa simplesmente "aguarde até que TVCOUNT seja incrementado de n".

TVCOUNT está armazenada em 3 bytes: 23672, 23673 e 23674. Cada byte tem 8 bits, e o valor máximo de TVCOUNT é 2 elevado a 24 menos 1, ou seja, 16777215, o que corresponde a aproximadamente 3 dias, 21 horas e alguns minutos, a partir do momento em que o computador for ligado (se fizer esta experiência, faça-a antes que acabe a garantia). Atingindo esse valor, TVCOUNT é zerada e inicia nova contagem.

O valor de TVCOUNT pode ser encontrado através da linha:

```
PRINT PEEK 23672 + 256 * PEEK 23673 + 65536 * PEEK 23674
```

Experimente digitar o programa a seguir, que simula um relógio muito simples:

```
10 LET tvcount = 23672

20 LET min = 0

30 POKE tvcount, 0: POKE tvcount + 1,0: POKE tvcount + 2, 0

40 PRINT AT 1,7; "Segundos = ";

50 LET t = PEEK tvcount + 256 * PEEK (tvcount + 1) + 65536 * PEEK
(tvcount + 2)

60 LET seg =INT (t/50): REM desconte um tempo de processamento

70 PRINT AT 1,1; seg

80 LET min = INT (seg/60)

90 PRINT AT 3,7; "Minutos = "; AT 3, 18; min

100 GOTO 50
```

O potencial desta variável é muito grande: serve de contador de tempo em diversos tipos de programas, até mesmo para aqueles que pretendem dar assessoria ao computador por hora! Serve de alarme, ou mesmo um sofisticado relógio, como se pode ver no capítulo VIII do livro.

7 – ROLANDO A IMAGEM

Um pequeno problema que surge para programadores em Basic de TK 90X é como não deixar o computador perguntar "Scroll?" a cada vez que a tela é preenchida com dados.

Existe uma variável de sistema, chamada SCRINC (de INCrementador de SCRoll), de endereço 23692, que possui o valor de 1 a menos que o número de linhas de tela que serão roladas para cima até a próxima pergunta "Scroll?" (normalmente, esse valor é menor do que 23). Portanto, devemos colocar nessa variável um valor maior do que o número de linhas que rolam para cima na tela, ou seja, de imediato, 255, que é o valor máximo.

Experimente:

```
10 PRINT AT 21,31'

20 PRINT PEEK 23692

30 GOTO 20
```


Rode o programa e introduza a seguir a linha

15 POKE 23692,255 e rode novamente.

8 – VARIÁVEIS DE CONTROLE DE MEMÓRIA

Existem algumas variáveis do sistema que servem apenas para o computador saber o estado em que se encontra a sua memória, como, por exemplo, saber onde começa o programa em Basic, onde estão as variáveis desse programa etc. Muitas destas variáveis do sistema têm pouco interesse para o programador. Veremos algumas delas.

PROGBAS

Os endereços 23635 e 23636 dizem ao computador onde começa o programa Basic

PRINT PEEK 23635 + 256 * PEEK 23636 dá o valor de PROGBAS, ou seja, o início da área de programas Basic. Some-se cinco a este valor e teremos o endereço do primeiro caractere após a declaração REM da primeira linha de um programa qualquer (sem microdrive ou interface), cheia de códigos esquisitos, que são, na verdade, uma rotina em código de máquina.

Se você pretende criar uma rotina em linguagem de máquina e armazená-la dessa forma, não esqueça de deixar após a declaração REM tantos espaços quantos forem os bytes da sua rotina.

Para introduzir uma mensagem de autor de programa e deixá-la protegida na primeira linha, digite:

```
POKE (PROGBAS),0: POKE (PROGBAS + 1),0
```

Se o número anterior da primeira linha do seu programa for menor do que 256, então o primeiro dos dois comandos acima pode ser omitido. Este procedimento "renumera" a primeira linha do programa, atribuindo a ela o número 0, tornando impossível uma simples edição dessa linha.

VARADD

Esta variável armazena o endereço das variáveis Basic, através da expressão:

```
PRINT PEEK 23627 + 256 * PEEK 23628
```

As variáveis Basic estão situadas imediatamente após o programa Basic na memória do TK 90X; portanto, nós podemos encontrar o comprimento de um programa através da simples subtração de PROGBAS de VARADD:

PRINT "O programa tem"; 256 * (PEEK 23628 - PEEK 23636) + (PEEK 23627 - PEEK 23635); "bytes de comprimento"

RAMTOP e ADSPFREE

Estas duas variáveis podem ser utilizadas para medir a quantidade de memória disponível ao usuário. No mapeamento da memória do TK 90X, entre ADSPFREE e RAMTOP, existem apenas o espaço livre, a pilha da máquina e a pilha de GOSUB.

RAMTOP é dada por:

PRINT PEEK 23730 + 256 * PEEK 23731

E ADSPFREE por:

PRINT PEEK 23653 + 256 * PEEK 23654

Como alternativa da segunda expressão, existe na ROM do TK 90X uma rotina que faz isso, e que começa no endereço 7962. Em vez de toda essa expressão pode-se simplesmente digitar PRINT USR 7962, o que dá no mesmo.

Para estimar a quantidade de memória disponível, subtraia ADSPFREE de RAMTOP.

Ainda sobre ADSPFREE, você provavelmente deve saber como parar aquele programa maravilhoso que quando carrega da fita, automaticamente já entra rodando. Sabe-se que ele foi gravado com SAVE. . . LINE. . . Para interromper esse programa, deve-se usar MERGE " ", em vez de LOAD " ", o que cria um grande problema para as software-houses. . . Pois bem, uma maneira de resolver esse problema é gravar o programa como sendo um bloco de bytes, através da inserção das seguintes linhas no final do programa:

9997 LET adspfree = PEEK 23653 + 256 * PEEK 23654

9998 SAVE "nome" CODE 23552, adspfree + 23500

9999 RUN

Desta forma, toda a memória é enviada para a fita cassete, incluindo o programa, as variáveis Basic, as variáveis do sistema, a pilha do calculador, como um bloco de códigos, de tal forma que, quando for carregado de volta ao computador, este vai iniciar o seu processamento a partir da linha em que parou, ou seja, a partir da última linha do programa (não se esqueça de que, para carregar o programa, você deve usar LOAD " " CODE).

9 – DIVERSAS VARIÁVEIS

Para saber qual o número da próxima linha a ser interpretada, digite:

PRINT PEEK 23662 + 256 * PEEK 23663 (será o comando TRACE?)

E, para saber qual o número da próxima declaração, naquela linha, digite:

PRINT PEEK 23664

Os endereços 23677 e 23678 armazenam respectivamente as coordenadas X e Y do último ponto plotado (variáveis de sistema LSTPLOT e COORDS). Você pode criar duas variáveis em Basic, usando linhas com PLOT ou DRAW; se começar o programa com

LET xo = 23677: LE yo = 23678

Toda vez que precisar saber qual foi o último ponto plotado, PEEK xo e PEEK yo lhe darão as coordenadas desse ponto.

Para desenhar uma linha a partir do último ponto plotado

DRAW (x - PEEK xo), (y - PEEK yo)

Ou então, simplesmente, dar POKEs nessas variáveis, para alterar a posição do último ponto plotado, ou então para desenhar pontos e retas, sem utilizar os comandos PLOT e DRAW.

HVPOS

Armazena a posição corrente de impressão, mas não da maneira que você deve estar aguardando. Se lhe foi dado algum comando tipo PRINT AT 1,c, então

endereço 23688 armazena 33 - c

endereço 23689 armazena 24 - 1

Portanto, para encontrar a posição corrente de impressão

LET 1 = 24 - PEEK 23689

```
LET c = 33 - PEEK 23688
```

Se você pretende usar essa técnica em seus programas, até mesmo em conjunto com SCREEN$, então proceda da seguinte forma:

```
DEF FN y( ) = 24 - PEEK 23689
DEF FN x( ) = 33 - PEEK 23688
```

INITRND

Esta é a variável utilizada para gerar o último número randômico (aleatório), e está armazenada nos endereços 23670 e 23671. Experimente digitar:

```
PRINT RND, (PEEK 23670 + 256 * PEEK 23671)/65536
```

E serão impressos dois valores iguais. Cada vez que RND é usada, INITRND é alterada pelo computador da seguinte maneira:

```
Novo INITRND =(75 (INITRND + 1)) mod. 65537 - 1
```

O que corresponde em Basic a:

```
LET INITRND = 75 * (INITRND + 1): LET INITRND = INITRND - 65537
* INT (INITRND/65537) - 1
```

O novo valor de INITRND é então armazenado e dividido por 65536, para produzir um valor para RND entre 0 e 1.

DFPSPRT e DFPSPRTL

Estas duas variáveis de sistema, armazenadas nas locações 23684 e 23686, armazenam o endereço no arquivo de imagens das posições de impressão, tanto para a parte maior da tela quanto para a parte inferior.

Normalmente não usamos PEEK e POKE no arquivo de imagens, pois temos três comandos que suprem qualquer necessidade, que são POINT (x,y), PRINT e SCREENS$ (x,y).

Para qualquer posição de impressão na tela, nas posições X e Y, temos que:

```
DFPSPRT = 2048 * INT (Y/8 + 8) + (Y - 8 * INT (Y/8)) * 32 + X
```

Não se esqueça de que cada caractere na tela é armazenado em 8 bytes na memória (um para cada linha de caractere). Os endereços desse grupo de 8 bytes de cada caractere da tela são separados de outro na mesma posição, em outra parte do arquivo de imagem, por 256. Portanto, se DFPSPRT é a primeira linha, na segunda é DFPSPRT + 256, na terceira é DFPSPRT + 512, e assim por diante.

Experimente digitar o programa a seguir, que ilustra bem essas experiências:

```
10 BORDER 0: PAPER 0: INK 6: CLS
20 PRINT AT INT (RND * 22), INT (RND * 32);
30 LET DF =PEEK 23684 + 256 * PEEK 23685
40 FOR c = 0 TO 1
50 FOR a = DF TO DF + 7 * 256 STEP 256
60 READ b: POKE a, b: NEXT a
70 NEXT c
80 RESTORE: GOTO 40
90 DATA 24, 60, 126, 25, 31, 254, 60, 24, 248, 60, 23, 15, 15, 23, 60, 248
```

Se você realmente quiser proteger seus programas contra aqueles "piratas profissionais", dê um POKE na variável de sistema P ERR, de endereço 23613, com valor 0.

```
POKE 23613, 0
```

E coloque esse comando como sendo o primeiro da primeira linha do programa, para então experimentar dar um BREAK no programa e ver o que acontece...

Capítulo IV

ALGO SOBRE OS COMANDOS IN E OUT

Uma falha gritante do manual do TK 90X!

Estes dois comandos são importantíssimos, tanto a nível de Basic quanto a nível de linguagem de máquina.

Os microcomputadores, ou melhor, qualquer computador, devem poder comunicar-se com o exterior. Este mundo exterior inclui normalmente o leitor, a sua televisão, o seu gravador, uma impressora, microdrives, enfim, tudo que possa ser conectado à CPU do micro.

Dividimos os modos de comunicação de um computador em duas categorias principais: as entradas e as saídas. As entradas são as informações recebidas pelo computador, através de qualquer periférico de entrada. As saídas, ao contrário, são as informações enviadas pelo computador para um periférico de saída. Para poder comunicar-se com o exterior, o computador, qualquer que seja ele, utiliza o que chamamos de portas (*ports*, em inglês). O nome em si tem lógica.

O microprocessador pode ler e escrever dados na memória através das instruções PEEK e POKE. Mas, para ele, não interessa se está lendo ou escrevendo na ROM ou na RAM; ele sabe somente que existem no máximo 65536 endereços disponíveis de uma vez, para sua pesquisa.

Numa total analogia, podemos dizer que existem 65536 portas de entrada ou saída (em inglês, I/O ports – de Input e Output). Estas, como vimos, são usadas pelo sistema para comunicação com o meio exterior, e podem ser controladas pela linguagem Basic, através das instruções IN e OUT.

A instrução IN é similar a PEEK.

IN endereço

Possui um argumento, o endereço da porta, e seu resultado é o byte lido daquela porta.

OUT é uma instrução similar a POKE.

OUT endereço, valor

Escreve naquela porta especificada o valor determinado.

A maneira como esse endereço é interpretado depende muito do estado do computador, ou seja, se existe um periférico conectado a ele. No TK 90X, é preferível imaginarmos esses endereços escritos na sua forma binária, porque cada bit do endereço trabalha de maneira independente, ou seja, depende do estado da máquina, ou da finalidade do processo.

São no total 16 bits, assim denominados:

A15, A14,. . . . A10,. . . . . A2, A1, A0.

Onde A0 é o primeiro bit, A1 é o segundo, e assim por diante, até A15, que é o último. Normalmente esses bits têm o valor 1, mas se um deles tem valor zero, significa que o computador deve realizar uma tarefa específica. Os bits A0, A1, A2, A3 e A4 são os mais importantes. O computador não pode fazer mais que uma tarefa por vez, portanto apenas um destes cinco bits deve ser 0, de cada vez. Os bits A6 e A7 são ignorados. Os melhores endereços para serem utilizados são aqueles múltiplos de 32 menos 1.

Os bits de A8 em diante, algumas vezes, transmitem alguma informação extra ao computador.

O byte lido ou escrito tem 8 bits, que são chamados de D0 a D7.

Existe um grupo de endereços de entrada, que tanto lêem o teclado como o soquete EAR do micro.

O teclado é dividido em oito meias linhas de cinco teclas cada:

– IN 65278 le de CAPS SHIFT até V

– IN 65022 le de A até G

– IN 64510 le de Q até T

– IN 63486 le de 1 até 5

– IN 61438 le de 0 até 6

– IN 57342 le de P até Y

– IN 49150 le de ENTER até H

– IN 32766 le de SPACE até B

(Estes endereços são $254 + 256 * (255 - 2^n)$, com n variando de 0 a 7).

No byte lido, os bits D0 a D4 se associam às cinco teclas daquela meia linha – sendo D0 associado à tecla mais externa e D4 associado à tecla mais próxima ao meio do teclado. O bit será 0 se a tecla foi pressionada, e 1 em caso contrário. D6 é o valor no soquete EAR traseiro.

A porta de endereço 254, com relação a saídas, dirige o som de saída pela tevê (D4), e o soquete MIC traseiro (D3), e também muda as cores da borda da tela (D2, D1, D0).

A porta de endereço 251 dirige a impressora conectada no slot traseiro, tanto em leitura quanto em escrita, ou seja, sabendo se a impressora terminou ou não a mensagem enviada.

As portas de endereço 254, 247 e 239 são usadas para periféricos, como microdrives, interfaces seriais etc.

A porta de endereço 63 é usada para sintetizadores de voz (nem todos – são características de cada um).

Experimente o programa:

```
10 FOR x = 0 TO 7
20 LET m = 254 + 256 * (255 - 2^X)
30 PRINT AT 0,0; IN m: GOTO 30
```

E vá pressionando as teclas para saber os valores respectivos de cada uma. Não é preciso anotar, porém confira com o quadro a seguir os valores correspondentes a cada uma delas, e se houver diferença (é normal), anote-a. Quando completar a digitação de cada meia linha, pressione BREAK e dê, em modo direto, NEXT x, para passar para outra meia linha.

O programa a seguir, de desenho livre, onde se "pega a caneta" do computador, é um grande exemplo do emprego do comando IN, permitindo que você pressione mais de uma tecla ao mesmo tempo. Experimente e divirta-se. (Repare na velocidade.)

TABELA DE ENDEREÇAMENTO DO TECLADO

IN 63496					IN 61438				
1	2	3	4	5	6	7	8	9	0
62	61	59	55	47	47	55	59	61	62
IN 64510					**IN 57342**				
Q	W	E	R	T	Y	U	I	O	P
62	61	59	55	47	47	55	59	61	62
IN 65022					**IN 49150**				
A	S	D	F	G	H	J	K	L	ENTER
62	61	59	55	47	47	55	59	61	62
IN 65278					**IN 32766**				
CAPS SHIFT	Z	X	C	V	B	N	M	SYMBOL SHIFT	SPACE
62	61	59	55	47	47	55	59	61	62

Esta tabela funciona da seguinte maneira:

Suponha que você tenha uma linha de programa com a seguinte configuração:

IF a$ = "s" THEN...

ou então

PAUSE 0: IF INKEY$ = "s" THEN...

Substituindo por endereçamento de teclado:

IF IN 65022 = 61 THEN...

O que agiliza o seu programa, torna a resposta mais rápida que os comandos anteriores, e permite que dois ou mais comandos sejam acionados ao mesmo tempo.

O programa seguinte exemplifica muito bem isso.

Mas, devido às pequenas diferenças existentes entre o ZX Spectrum da Sinclair e o TK 90X, o programa tanto serve para um quanto para outro.

E, nas listagens dos programas seguintes, se por acaso encontrar um comando BEEP, substitua-o pelo comando SOUND; ambos são idênticos. O problema é que este TK 90X não aceitou a impressora ALPHACOM 32 conectada junto com a In-

terface One e o Microdrive. Não se preocupe: o micro é nacional, e os periféricos são importados; num conjunto totalmente nacional isso não irá ocorrer. (Esse é provavelmente um problema da fonte do micro, que não está dimensionada para suportar todos esses periféricos juntos.)

Programa "Desenhando"

```
   1 CLEAR 63999: GO SUB 9000
  10 BORDER 0: PAPER 0: INK 9: B
RIGHT 1: CLS
  20 LET x=128: LET y=87: LET x1
=x: LET y1=y
  30 PLOT x,y
  40 LET x=x+(IN 64510=47 OR IN
64510=175)+(IN 65022=47 OR IN 65
022=175)+(IN 32766=47 OR IN 3276
6=175)-(IN 64510=59 OR IN 64510=
187)-(IN 65022=59 OR IN 65022=18
7)-(IN 65278=55 OR IN 65278=183)
  50 LET y=y+(IN 64510=59 OR IN
64510=187)+(IN 64510=55 OR IN 64
510=183)+(IN 64510=47 OR IN 6451
0=175)-(IN 65278=55 OR IN 65278=
183)-(IN 65278=47 OR IN 65278=17
5)-(IN 32766=47 OR IN 32766=175)
  60 LET x=x+(x<0)-(x>255): LET
y=y+(y<0)-(y>175)
  70 PRINT #1;AT 1,0;x,y
  80 IF IN 32766=55 OR IN 32766=
183 THEN OVER 1: PRINT #1;AT 1,2
6;"OVER 1"
  90 IF IN 65278=59 OR IN 65278=
187 THEN GO TO 10
 100 IF IN 32766=59 OR IN 32766=
187 THEN INVERSE 1: PRINT #1;AT
1,23;"INVERSE 1"
 110 IF IN 57342=59 OR IN 57342=
187 THEN INVERSE 0: OVER 0: PRIN
T #1;AT 1,23;"         "
 120 IF IN 57342=62 OR IN 57342=
190 THEN INPUT "Cor do papel ";P
: DIM a$(1,704): PRINT OVER 1; P
APER p; INK 9;AT 0,0;a$(1)
 130 IF IN 49150=47 OR IN 49150=
175 THEN INPUT "Raio do circulo
";r: CIRCLE OVER 0;x,y,r
 140 IF IN 49150=59 OR IN 49150=
187 THEN CIRCLE OVER 1;x,y,r
 150 IF IN 61438=62 OR IN 61438=
190 THEN INVERSE 0: OVER 0: PRIN
T #1;AT 1,0;"
                "
 160 IF IN 63486=62 OR IN 63486=
190 THEN INK 1
 170 IF IN 63486=61 OR IN 63486=
189 THEN INK 2
 180 IF IN 63486=59 OR IN 63486=
187 THEN INK 3
 190 IF IN 63486=55 OR IN 63486=
183 THEN INK 4
 200 IF IN 63486=47 OR IN 63486=
175 THEN INK 5
 210 IF IN 61438=47 OR IN 61438=
175 THEN INK 6
 220 IF IN 61438=55 OR IN 61438=
183 THEN INK 7
 230 IF IN 61438=59 OR IN 61438=
187 THEN BRIGHT 1
 240 IF IN 61438=61 OR IN 61438=
189 THEN INK 9
 250 IF IN 65022=55 OR IN 65022=
183 THEN PLOT x1,y1: DRAW x-x1,y
-y1
 260 IF IN 65278=61 OR IN 65278=
189 THEN COPY
 270 IF IN 65022=61 OR IN 65022=
189 THEN INPUT "Nome do arquivo
";y$: SAVE y$SCREEN$
 280 IF IN 49150=55 OR IN 49150=
183 THEN INPUT "Nome do arquivo
";y$: LOAD y$SCREEN$
 290 IF IN 49150=61 OR IN 49150=
189 THEN GO SUB 500
 297 REM
 298 REM
 299 REM
 300 IF IN 65278=62 OR IN 65278=
190 THEN RANDOMIZE USR 64000: PR
INT #1;AT 1,0;"
                "
 310 IF IN 32766=62 OR IN 32766=
190 THEN RANDOMIZE USR 65000: PR
INT #1;AT 1,0;"
                "
 320 GO TO 30
 500 LET x2=x: LET y2=y
 510 PLOT x,y: PLOT OVER 1;x,y
 520 LET x=x+(IN 64510=47 OR IN
64510=175)+(IN 65022=47 OR IN 65
022=175)+(IN 32766=47 OR IN 3276
6=175)-(IN 64510=59 OR IN 64510=
187)-(IN 65022=59 OR IN 65022=18
7)-(IN 65278=55 OR IN 65278=183)
 530 LET y=y+(IN 64510=59 OR IN
64510=187)+(IN 64510=55 OR IN 64
510=183)+(IN 64510=47 OR IN 6451
0=175)-(IN 65278=55 OR IN 65278=
183)-(IN 65278=47 OR IN 65278=17
5)-(IN 32766=47 OR IN 32766=175)
 540 LET x=x+(x<0)-(x>255): LET
y=y+(y<0)-(y>175)
 550 IF IN 57342=47 OR IN 57342=
175 THEN RETURN
```

```
 560 IF IN 65022=62 OR IN 65022=
190 THEN INPUT "Arco ";a: DRAW x
2-x,y2-y,PI/a: RETURN
 570 IF IN 64510=61 OR IN 64510=
189 THEN PLOT x2,y2: DRAW x-x2,0
: DRAW 0,y-y2: DRAW -x+x2,0: DRA
W 0,-y+y2: RETURN
 580 IF IN 64510=62 OR IN 64510=
190 THEN DRAW x2-x,y2-y: RETURN
 590 GO TO 510
9000 FOR i=0 TO 24
9010 READ a: POKE 64000+i,a
9020 NEXT i
9030 DATA 6,192,17,0,64,213,225,
35,197,1,31,0,26,237,176,43,119,
0,35,35,19,193,16,240,201
9040 FOR i=0 TO 24
9050 READ b: POKE 65000+i,b
9060 NEXT i
9070 DATA 6,192,17,255,87,213,22
5,43,197,1,31,0,26,237,184,35,11
9,0,43,43,27,193,16,240,201
9080 RETURN
```

Descrição do programa:

Na linha 1, a RAMTOP foi puxada para 63999, liberando a área posterior para entrada e armazenamento de códigos.

Na linha 30 é plotado um ponto, na posição 128,87.

Nas linhas 40 e 50 são as condicionantes que incrementam ou decrementam as coordenadas do ponto, e na linha 60 esses valores são ajustados em função dos limites da resolução gráfica do micro.

Na linha 70 são impressos, na parte inferior do vídeo, os valores correntes de X e Y.

Da linha 80 até a 310 são as teclas condicionantes da posição do ponto plotado.

As teclas que movem o ponto são as que se situam ao redor da tecla F, da seguinte maneira:

```
E   R   T     (movimento em diagonal ascendente à direita)
  ↖ ↑ ↗
D ← F → G     (movimento à direita)
  ↙ ↓ ↘
C   V   B
```


Os outros comandos do programa são:

Linha 80 – tecla N – aciona modo OVER 1, para apagar ponto;

Linha 90 – tecla X – roda o programa outra vez – apaga tudo;

Linha 100 – tecla M – aciona modo INVERSE 1;

Linha 110 – tecla I – aciona modo OVER 0 e INVERSE 0;

Linha 120 – tecla P – muda cor do papel;

Linha 130 – tecla H – pergunta raio de círculo e traça-o;

Linha 140 – tecla K – apaga círculo desenhado (só se houver);

Linha 150 – tecla O – entra em modo OVER 0 e INVERSE 0;

Linha 160 até 240 – muda cores de tintas e aciona BRIGHT 1;

Linha 260 – tecla F – traça segmentos radiais de reta;

Linha 270 – tecla S – grava desenho – modifique-o para fita cassete;

Linha 280 – tecla J – carrega desenho – modifique-o para fita cassete;

Linha 290 – tecla L – chama sub-rotina de piscar ponto (sem desenhar), a partir da linha 500;

Linha 300 – tecla CAPS SHIFT – chama sub-rotina em linguagem de máquina, armazenada a partir do endereço 64000 e move toda a tela para a esquerda (é o chamado scroll lateral);

Linha 310 – tecla SPACE – faz o scroll lateral para a direita, a partir da rotina em linguagem de máquina que começa no endereço 65000; e

Linha 320 – Retorna para plotar o ponto na nova posição.

Na sub-rotina da linha 500 até 590, mova o ponto, sem desenhar, e, através da tecla A, trace um arco de circunferência; da tecla W, trace um retângulo entre a posição anterior e a nova; da tecla Q, una os dois pontos, com um segmento de reta; e da tecla Y, retorne da sub-rotina sem fazer nada.

Da linha 9000 até a linha 9080 estão os códigos em valores decimais que significam, para o micro, as duas rotinas de scroll lateral, tanto para a direita quanto para a esquerda. Você pode perfeitamente introduzir estas rotinas em seu próprios programas.

Capítulo V

DIFERENÇAS ENTRE O TK 85 E O TK 90X

Houve uma evolução muito grande entre o TK 83 e o TK 85, principalmente em relação à aparência, já que surgiu um teclado, e o gabinete mudou de aparência.

Quanto ao hardware, a ROM do TK 85 recebeu 2K adicionais armazenando as rotinas referentes ao armazenamento rápido em fita e a RAM também cresceu, e muito.

No TK 90X, as alterações foram muitas e boas, a começar pelo teclado, aspecto de maior crítica, do ZX Spectrum, dada a sua lentidão, na digitação de textos, e não na programação (eu particularmente não acho – me acostumei com ele).

Sem considerar as rotinas de armazenamento rápido em fita cassete, residentes nos 2K adicionais da ROM original do TK 85, as duas versões de Basic oferecidas por essas duas máquinas são muito semelhantes. E, dependendo do caso, alguns programas de um podem servir para o outro (mas não ser carregado diretamente). O TK 85 possui apenas dois comandos que o TK 90X não tem, que são SCROLL e UNPLOT, mas existe uma série de comandos e funções do TK 90X que não são encontrados no TK 85. Essa listagem está na tabela 4, a seguir. Os asteriscos nessa tabela indicam quais comandos e funções não são facilmente traduzidos para a linguagem Basic do TK 85. Os comandos de cor e som podem ser omitidos, mas com relação à alta resolução e comandos de entrada e saída de arquivos existem opções.

O comando PLOT aparece em ambos os micros, mas seu efeito é um pouco diferente, portanto cuidado! Outra diferença fundamental é com relação a PEEK e POKE, pois os endereçamentos são completamente diferentes.

O comando USR "A" do TK 90X não tem similar no TK 85; ele deve ser omitido dos programas a serem convertidos.

TABELA 1 – Conversão do TK 85 para o TK 90X

TK 85	TK 90X	Comentários
SCROLL	RAND USR 3582 ou LET r = USR 3582	No TK 90X, este comando faz o SCROLL de uma linha de cada vez.
PLOT X, Y	PRINT AT 21-X/2, Y/2;	Imprime o caractere gráfico padrão do TK 90X (teclas 1 a 8).
UNPLOT X, Y	PRINT OVER 1; AT 21-X/2, y/2	Sobrepõe um caractere gráfico padrão, apagando o anterior.

TABELA 2 – Conversão do TK 90X para o TK 85

TK 90X	TK 85	Comentários
BIN LET x = BIN 0I00I001	LET x = valor decimal	BIN permite a representação binária de um número, em caracteres gráficos definidos pelo usuário.
READ/DATA/RESTORE READ x,y DATA 11,22	LET LET x = 11 LET y = 22	
DEF FN e FN DEF FN a(x) = SQR x LET y = FN a(z)	LET a$ = "SQR x" LET x = p LET y = VAL a$	O equivalente a FN necessita de pelo menos duas linhas de programa.
PLOT	Não tem equivalente	A resolução gráfica do TK 85 é baixa
SCREEN$ LET m = SCREEN$ x,y	LET m = PEEK (PEEK 16396 + 256 * PEEK 16397 + 1 + Y + 33 * X)	

TABELA 3 – Conversões generalizadas

TK 85	TK 90X	Comentários
1- QUAD POKE 16436,255 POKE 16437,255 LET t = (65535 - PEEK 16436 - 256 * PEEK 16437)/60	TVCOUNT POKE 23672,0: POKE 23673, 0: POKE 23674,0 LET t = (PEEK 23672 + 256 * PEEK 23673 + 65536 * PEEK 23674)/60	
2- Número da primeira linha = zero POKE 16510,0	POKE 23756,0 (sem alteração da RAM original, isto é, sem periféricos)	
3- RAMTOP POKE 16388, X-256 * INT (X/256) POKE 16389, INT (X/256)	CLEAR x	

TABELA 4 – Funções ou comandos do TK 90X não disponíveis no TK 85

BORDER *	FORMAT *	ATTR *
BRIGHT *	INK *	BIN
CAT *	INVERSE	FN
CIRCLE *	MERGE *	IN *
CLOSE *	MOVE *	OVER *
DATA	OPEN *	POINT *
DEF FN	OUT *	SCREEN$
DRAW *	PAPER *	SOUND *
ERASE *	READ	VAL $ *
FLASH *	RESTORE *	TRACE *
	VERIFY *	

TABELA 5 – Conversões de variáveis de sistema

TK 85		TK 90X	
NOME	ENDEREÇO	NOME	ENDEREÇO
CALREG	16414	BREGCAL	23655
BANCO	16443		Sem equivalente
ENCAR	16406	CHNXADD	23645
CORDX	16438	LSTPLOT	23677
CORDY	16439	COORDS	23678
DEST	16402	XVARADD	3629
POSPR	16398	DFPSPRT	23684
DFILE	16396		Sem equivalente
DFSZ	16418	SIZE	23659
ELINE	16404	INADD	23641
CODR	16384	ERRCD	23610
LPC	16394	CURLINE	23625
ENSP	16386	PERR	23613
BAND	16385	SFLAG 0	23611
BANDX	16429	SFLAG 3	23665
QUAD	16436	TVCOUNT	23672
ULTK	16421	KEYPRS	23560
HARG	16424		Sem equivalente
MEM	16415	MEMCADD	23656
MEMBO	16477	MEMSPCAL	23698
MODO	16390	CURSOR	23617
PXLN	16425	NEXEXC	23637
VCPB	16427	CONTJMP	23662
CPB	16391	EXCLINE	23621
PRBUFF	16444	–	23296
PRCC	16440	PRTADD	23680
RTP	16388	RAMTOP	23730
SEMT	16434	INITRND	23670
COLPR	16441	HVPOS	23688
LINPR	16442		23689
PILFUN	16410	STKCEND	23651
PILFIM	16412	ADSPFREE	23653
LTOP	16419	LISTNR	23660
LENCA	16430	STRVRLEN	23666
SXEN	16432	SYTADD	23668
VARS	16400	VARADD	23627
VERSN	16393		Sem equivalente
ENSX	16408	SYCHADD	23647

Capítulo VI

AS CORES PELO TECLADO

Apesar de tudo o que já foi dito tanto no manual quanto neste livro sobre como se obter cores para INK e PAPER, ou como entrar em modo FLASH ou BRIGHT, existe outra maneira de se colorir mensagens, gráficos criados pelo usuário, ou texto; através de códigos que são obtidos pressionando-se diretamente certas teclas.

Por exemplo, para se digitar uma linha de programa, que seja com INK 1 e PAPER 1, deve-se fazer o seguinte:

1- Digite o número da linha;

2- Entre em modo extended (cursor E), pressionando CAPS SHIFT junto com SYMBOL SHIFT;

3- Pressione a tecla 1 – pronto! Você já obteve papel azul;

4- Pressione novamente CAPS SHIFT (ainda em modo extended) e outra vez a tecla 1 – e você passará a escrever com tinta azul, ou seja, com a tinta da mesma cor que o papel, para que, por exemplo, nenhum pirata tenha acesso ao seu copyright da primeira linha do programa.

Essa seqüência de comandos vale para qualquer tecla de 1 a 7, referentes às cores.

A tecla 8 pode tanto desligar o FLASH quanto o BRIGHT, dependendo de qual tecla você pressionar antes dela.

A tecla 9, ao contrário da 8, pode tanto ligar o FLASH quanto o BRIGHT.

QUADRO – RESUMO DAS FUNÇÕES DAS TECLAS 1 ATÉ 0

MODO	SHIFT	TECLAS	DEF FN	FN	LINE	OPEN #	CLOSE *	MOVE	ERASE	POINT	CAT	FORMAT
E	SYMBOL		DEF FN	FN	LINE	OPEN #	CLOSE *	MOVE	ERASE	POINT	CAT	FORMAT
	CAPS		INK BLUE	INK RED	INK MAGENTA	INK GREEN	INK CYAN	INK YELLOW	INK WHITE	FLASH 0	FLASH 1	INK BLACK
	NENHUM		PAPER BLUE	PAPER RED	PAPER MAGENTA	PAPER GREEN	PAPER CYAN	PAPER YELLOW	PAPER WHITE	BRIGHT 0	BRIGHT 1	PAPER BLACK
G	CAPS ou SYMBOL										SAI DE GRAPHICS	DELETE
	NENHUM										SAI DE GRAPHICS	DELETE
K·L C	CAPS		EDIT	CAPS LOCK	TRUE VIDEO	INVERSE VIDEO	↓	→	←	↑	MODO GRAPHICS	DELETE
	SYMBOL		!	©	#	$	%	&	'	(	)	—
	NENHUM		1	2	3	4	5	6	7	8	9	0

E, para a tecla 0, valem as observações sobre as teclas de cores.

Portanto, nessa primeira fileira de teclas, de 1 a 0, cada uma delas pode ter até oito funções (não há teclado mais inteligente do que este), dependendo somente do modo em que se encontra o computador naquele momento, e, principalmente, da sua prática com o teclado.

Veja na página anterior um quadro ilustrativo dos diversos modos de se utilizar essas teclas, que por omissão, creio eu, não foi publicado no manual do TK 90X.

Capítulo VII

O MODO GRÁFICO DO TK 90X

Você já deve ter notado que este micro não possui página gráfica de alta resolução.

Ótimo. Porque essa página gráfica de alta resolução é apenas uma área da memória RAM reservada estritamente para desenhos e ela não se mistura com textos.

No TK 90X, em Basic mesmo, pode-se misturar texto e desenhos, como quiser. Não há necessidade de dar nenhuma instrução especial, para entrar e sair de qualquer modo gráfico. É uma grande vantagem desta máquina.

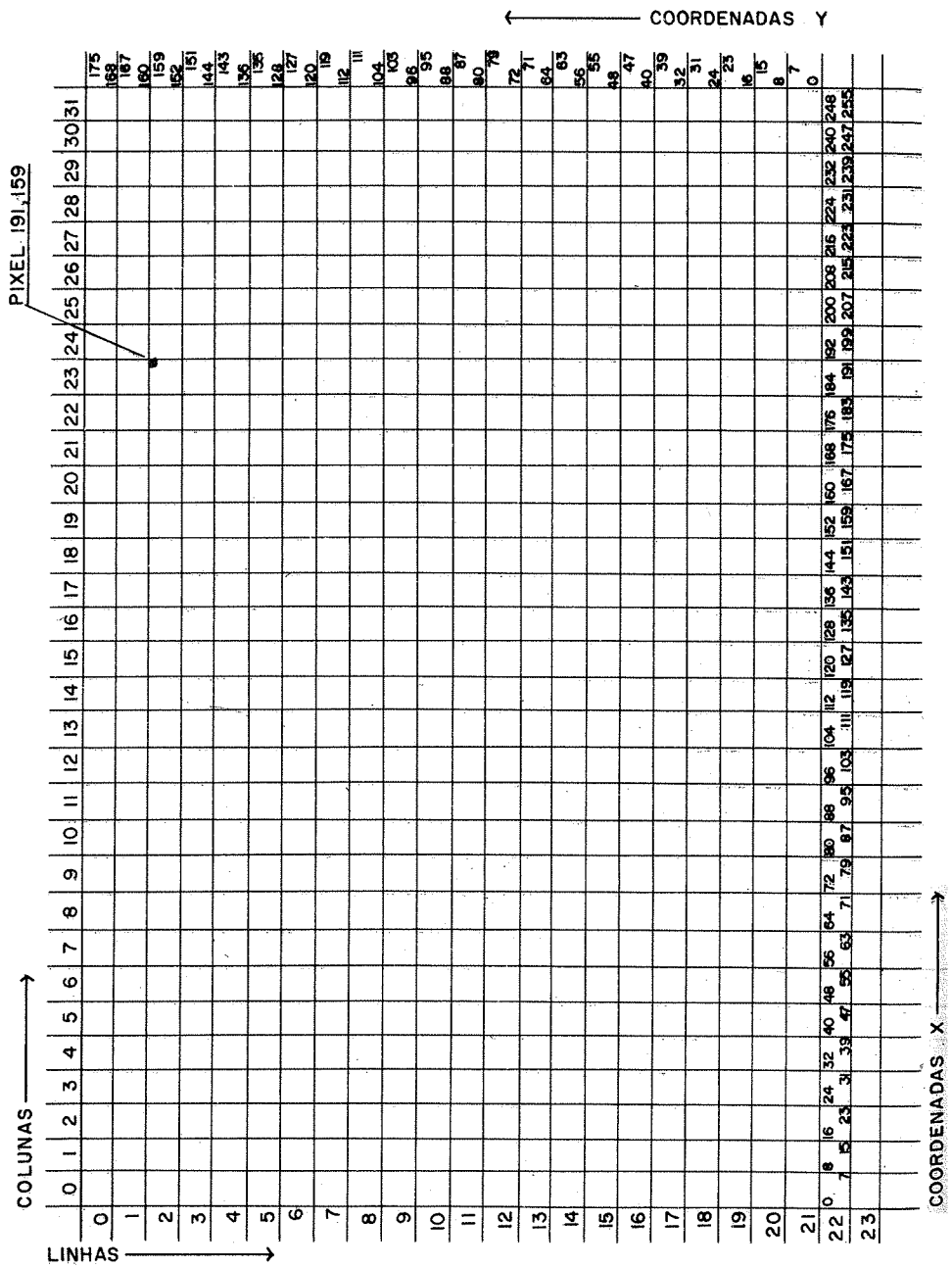
Mas, a operação de desenhar um ponto na tela apresenta algumas diferenças do modo de impressão de textos.

Um comando PRINT AT 0,0 imprime uma mensagem, ou um caractere na primeira linha do vídeo, ou seja a linha de topo (a primeira superior). Você sabe que existem 22 linhas disponíveis para texto, começando em zero e terminando em 21. Mas, se quiser mais duas linhas, lembre-se das variáveis de sistema. Basta dar POKE 23659,0 que você some com a parte inferior da tela – tome cuidado, pois estas linhas da parte inferior obrigatoriamente devem existir antes de um comando INPUT, por exemplo, ou antes de ser impressa uma mensagem lá (até mesmo SCROLL?)

Já com o comando PLOT, ou outro comando de desenho, ocorre o inverso.

Um comando PLOT 0,0 desenha um ponto no canto inferior esquerdo do vídeo. Um PLOT 255,0 desenha um ponto no canto inferior direito, enquanto PLOT 175,0 desenha um ponto no canto superior esquerdo da tela e PLOT 255,175 dese-

DISPOSIÇÃO DOS ELEMENTOS DO VÍDEO DO TK 90X



nha um ponto no canto superior direito da tela. Lembre-se de que a capacidade gráfica do TK 90X é de 256 pontos na horizontal (eixo X), e 175 pontos na vertical (eixo Y). Portanto, em computação gráfica neste micro, o eixo Y tem sentido inverso à maneira de formatar comandos de saída de textos.

Observe, pela tabela da página anterior, a orientação das posições gráficas para desenho na tela, bem como das posições de impressão de textos, que o ajudará na hora de converter gráficos em textos (coordenadas), e vice-versa.

Por exemplo, você desenha, através de PLOT e DRAW um retângulo na tela, e quer escrever alguma palavra bem no centro desse retângulo. Se os lados desse retângulo forem X e Y, teremos:

```
PRINT AT X/8, (175-Y)/8;".....
```

Note a inversão do sentido do eixo Y.

Capítulo VIII

PROGRAMAS

Neste capítulo, você encontrará diversos programas, tanto aplicativos ou utilitários quanto jogos de lazer, que visam demonstrar todo o pontencial do TK 90X.

Todas as técnicas de digitação possíveis no TK 90X foram utilizadas, inclusive as citadas no capítulo VI.

Você vai notar, através dos jogos aqui listados, a grande diferença entre este micro, e o seu irmão mais velho, o TK 85. Naquele, não é possível elaborar-se um "arcade game" em Basic, dada a sua lentidão. Neste é perfeitamente possível, visto que a sua velocidade de processamento, mesmo em Basic, é muito superior.

A utilização de um microcomputador como este não tem limite. Digite os programas a seguir, procure entendê-los, passo a passo, modifique-os, "personalize-os", mas não pare aqui. Estes programas são alguns poucos exemplos da capacidade da máquina, principalmente em função das necessidades de quem o utiliza.

Vá em frente! Comece a elaborar seus próprios programas, sem medo algum de errar, pois isso é a coisa mais normal no mundo dos programadores.

Existem diversos métodos de programação, como o diagrama de fluxos, mas o melhor deles, na minha opinião, é aquele em que nós pensamos como o micro, ou seja, escrevemos o programa, não ainda em Basic, mas a sua seqüência, a resolução do seu algoritmo, sempre objetivando aquele resultado planejado, para depois sim ser traduzido para a linguagem de computador.

Boa diversão!

NOTA: Todos os programas que requeiram algum movimento utilizam somente as teclas chamadas de "cursoras", que são:

"5" – movimenta para a esquerda

"8" – movimenta para a direita

"6" – movimenta para baixo

"7" – movimenta para cima

"0" – atirar.

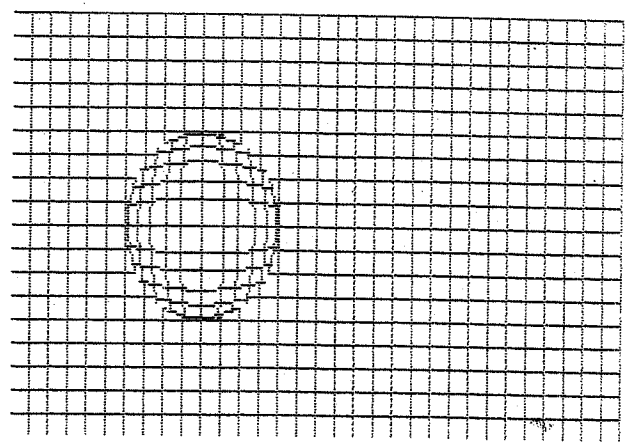
1) DEMONSTRATIVOS

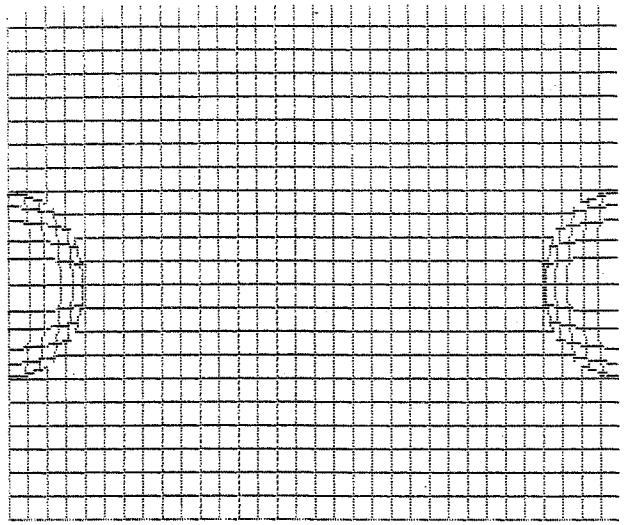
Programa "Demonstração de Bola"

Este programa é um demonstrativo da capacidade da linguagem Basic do TK 90X. Através das rotinas a partir da linha 9000, ele faz tanto SCROLL, à esquerda, quanto SCROLL, à direita, rotinas que se pode empregar em qualquer programa Basic.

A título de curiosidade, compare estes códigos, em linguagem de máquina, com os códigos do programa "DESENHANDO", e veja as diferenças.

A seguir, duas telas do programa rodando, mostrando o movimento da bola.





```
   1 REM ***      BOLA      ***
   2 REM DEMONSTRATIVO DE BASIC
   DO TK 90X
   3 REM As rotinas em linguagem
 de maquina deste programa voce
pode utilizar quando quiser
   7 GO SUB 9000
   8 BORDER 0: PAPER RND*2.8+RND
*3.5: INK 9: CLS
   9 PLOT 128,48: DRAW 0,64: PLO
T 96,80: DRAW 64,0
  10 FOR t=2 TO 10 STEP 2
  20 LET a=t/30*PI
  30 LET sx1=32*SIN a
  35 LET sy1=32*COS a
  40 LET sx2=32*SIN (PI-a)
  45 LET sy2=32*COS (PI-a)
  50 LET sx3=32*SIN (PI/2-a)
  55 LET sy3=32*COS (PI/2-a)
  60 LET sx4=32*SIN (a-PI/2)
  65 LET sy4=32*COS (a-PI/2)
 100 PLOT 128+sx1,80-sy1
 110 DRAW 0,(sy1-sy2),PI/(14-t)
 120 PLOT 128-sx1,80-sy1
 130 DRAW 0,(sy1-sy2),-PI/(14-t)
 140 PLOT 128+sx3,80+sy3
 150 DRAW -(sx3-sx4),0,PI/(14-t)
 160 PLOT 128+sx3,80-sy3
 170 DRAW -(sx3-sx4),0,-PI/(14-t
)
 180 NEXT t
 210 DEF FN q(x)=80-SQR (1024-(x
```



```
-128)*(x-128))
 220 DEF FN w(x)=80+SQR (1024-(x
-128)*(x-128))
 260 FOR x=0 TO 96 STEP 8
 270 PLOT x,0: DRAW 0,175
 280 NEXT x
 310 FOR x=104 TO 160 STEP 8
 320 PLOT x,0: DRAW 0,FN q(x)
 330 PLOT x,FN w(x): DRAW 0,175-
FN w(x)
 340 NEXT x
 360 FOR x=168 TO 255 STEP 8
 370 PLOT x,0: DRAW 0,175
 380 NEXT x
 410 DEF FN k(y)=128-SQR (1024-(
y-80)*(y-80))
 420 DEF FN h(y)=128+SQR (1024-(
y-80)*(y-80))
 460 FOR y=0 TO 48 STEP 8
 470 PLOT 0,y: DRAW 255,0
 480 NEXT y
 510 FOR y=48 TO 112 STEP 8
 520 PLOT 0,y: DRAW FN k(y),0
 530 PLOT FN h(y),y: DRAW 255-FN
h(y),0
 540 NEXT y
 560 FOR y=120 TO 175 STEP 8
 570 PLOT 0,y: DRAW 255,0
 580 NEXT y
 600 PAUSE 100
1010 FOR j=5 TO 1 STEP -1
1020 FOR f=1 TO 21: RANDOMIZE US
R 32400: PAUSE j: NEXT f
1030 FOR i=1 TO 15: RANDOMIZE US
R 32500: PAUSE j: NEXT i
1040 NEXT j
1100 FOR f=1 TO 18: RANDOMIZE US
R 32400: NEXT f
1110 FOR i=1 TO 32: RANDOMIZE US
R 32500: NEXT i
2000 DIM a$(1,704): BRIGHT 1: BO
RDER RND*6: PAPER RND*1.5+RND*4.
5: INK 9: PRINT OVER 1;AT 0,0;a$
(1)
2100 GO TO 1000
9010 FOR f=0 TO 24
9020 READ scrollsin
9030 POKE 32400+f,scrollsin
9040 NEXT f
9050 FOR f=0 TO 24
9060 READ scrollsdes
9070 POKE 32500+f,scrollsdes
9080 NEXT f
9090 RETURN
9110 DATA 6,192,17,0,64,213,225,
35,197,1,31,0,26,237,176,43
9120 DATA 119,0,35,35,19,193,16,
240,201
```

```
9210 DATA 6,192,17,255,87,213,22
5,43,197,1,31,0,26,237,184,35
9220 DATA 119,0,43,43,27,193,16,
240,201
```

Programa "Demonstração Gráfica"

São três pequenos programas, que traçam gráficos e desenhos em cores aleatórias na tela.

Tente localizar as rotinas de desenho, alterando alguns dados, em comandos PLOT ou DRAW.

Repare, no programa número três, o emprego de variáveis de sistema que armazenam as coordenadas do último ponto plotado.

```
   0>REM demonstracao grafica 1
   1 REM ************************
   2 REM JOSE EDUARDO MALUF DE
CARVALHO                ARQUITRON
   3 REM ************************
   5 BORDER RND*3+RND*4: PAPER 1
: INK RND*2+4: BRIGHT 1: OVER 1:
 CLS : PRINT #1;AT 0,0;"Demonstr
acao grafica num. 1      "
  10 LET x=0: LET y=INT (RND*175
)
  20 LET a=0: LET b=INT (RND*175
)
  30 LET g=2+INT (RND*2): LET c=
g: LET h=2+INT (RND*2): LET d=2+
INT (RND*2)
  40 IF a+g>255 OR a+g<0 THEN LE
T g=-g
  50 IF x+c>255 OR x+c<0 THEN LE
T c=-c
  60 IF b+h>175 OR b+h<0 THEN LE
T h=-h
  70 IF y+d>175 OR y+d<0 THEN LE
T d=-d
  80 LET a=a+g
  90 LET b=b+h
 100 LET x=x+c
 110 LET y=y+d
 120 PLOT a,b: DRAW x-a,y-b
 130 GO TO 40
```

```
   0>REM demonstracao grafica 2
   1 REM ************************
   2 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
```



```
   3 REM ************************
  10 LET p=RND*7
  20 LET g=RND*7
  30 LET s=RND*8+1
  40 LET l=RND*9+5
  50 LET m=RND*11+5
  60 LET f=RND*5
  65 LET c=RND*5
  68 LET z=2*PI/c
  70 LET a=2*PI/s: LET x=128: LE
T y=88: LET r=0
  80 BORDER p: PAPER p: INK g: C
LS : PRINT #1;AT 0,0;"Demonstrac
ao grafica num.2
  90 PLOT x,y
 100 FOR h=1 TO c
 110 FOR i=1 TO s
 120 FOR j=1 TO 2
 130 FOR k=1 TO m
 140 LET x=l*SIN r
 150 LET y=l*COS r
 160 DRAW x,y
 170 LET r=r+a
 180 IF r>2*PI THEN LET r=r-2*PI
 190 NEXT k
 200 LET r=r-a+PI
 210 IF r>2*PI THEN LET r=r-2*PI
 220 NEXT j
 230 LET r=r+f*a
 240 IF r>2*PI THEN LET r=r-2*PI
 250 NEXT i
 260 LET r=h*z
 270 IF r>2*PI THEN LET r=r-2*PI
 280 NEXT h
 290 GO TO 10
```

```
   0>REM demonstracao grafica 3
  10 REM ************************
  20 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
  30 REM ************************
  40 BORDER 0: PAPER RND*6: INK
9: BRIGHT RND: CLS : PRINT #1;AT
 0,0;"Demonstracao grafica num.3
     "
  50 DIM x(4)
  60 DIM y(4)
  70 DIM l(4)
  80 DIM m(4)
  90
 100 FOR x=0 TO 255 STEP 64
 110 FOR y=0 TO 175 STEP 44
 120 LET xm=INT (x/64)
 130 LET ym=INT (y/44)
 140 LET smu=.1
 150 IF (xm-INT (xm/2)*2)=(ym-IN
T (ym/2)*2) THEN LET smu=.9
 160 LET x(1)=x+63: LET y(1)=y+4
3
```

```
 170 LET x(2)=x+63: LET y(2)=y
 180 LET x(3)=x: LET y(3)=y
 190 LET x(4)=x: LET y(4)=y+43
 200 LET rmu=1-smu
 210 FOR i=1 TO 30
 220 FOR j=1 TO 4
 230 LET nj=(j-INT (j/4)*4)+1
 240 LET l(j)=rmu*x(j)+smu*x(nj)
 250 LET m(j)=rmu*y(j)+smu*y(nj)
 260 NEXT j
 270 FOR j=1 TO 4
 280 PLOT x(j),y(j)
 281 LET k=j-1
 282 IF k=0 THEN LET k=4
 283 IF smu=.1 THEN DRAW l(j)-PE
EK (23677),m(j)-PEEK (23678)
 284 IF smu=.9 THEN DRAW l(k)-PE
EK (23677),m(k)-PEEK (23678)
 290 LET x(j)=l(j)
 300 LET y(j)=m(j)
 310 NEXT j
 320 NEXT i
 330 NEXT y
 340 NEXT x
```

2) JOGOS

Programa "Alunissagem"

Muito cuidado com o solo lunar, que é basicamente areia!

Não bata nas montanhas, pois isso desintegrará a sua nave espacial.

Aterrisse exatamente na base, senão sua nave estará totalmente destruída.

Resumindo, esse é o objetivo deste jogo muito simpático.

Experimente.

```
   0 REM ALUNISSAGEM
 100 REM ************************
 105 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
 110 REM ************************
 120 RESTORE : FOR I=1 TO 16: RE
M CARACTERES GRAFICOS
 130 LET S=USR (CHR$ (144+I-1))
 140 FOR J=1 TO 8
 150 READ g: POKE (S+J-1),g
 160 NEXT J
 170 NEXT I
```



```
 180 DATA 0,0,0,1,1,7,7,31,25,0,
0,128,128,224,224,248,152,53,107
,127,243,96,96,192,192,172,214,2
54,207,66,33
 190 DATA 1,1,1,1,1,1,1,1,1,224,
176,140,134,129,135,140,176,1,1,
7,1,7,7,31,25,224,128,224,128,22
4,224,248,152
 200 DATA 0,0,0,4,5,15,6,15,0,0,
0,0,96,192,224,248,27,7,3,6,0,0,
0,0,120,168,112,32,0,0,0,0
 210 DATA 160,67,38,60,24,48,96,
97,4,154,244,124,6,4,4,6,194,96,
32,36,104,120,79,193,130,14,12,3
6,124,78,195,2
 211 REM ABCDEFGHIJKLMNOP
 212 REM ▲▚▞▙▟▛▜▝▘▗▖▐▌▀▄
 220 LET Id=0
 230 GO TO 8000
1000 PRINT #1;AT 0,0;"  ATERRISS
E     NA          BASE ! "
1020 LET bx=INT (RND*21): IF bx<
11 THEN GO TO 1020
1030 LET lt=bx*8: LET rt=(bx+4)*
8
1040 PRINT AT 21,bx;"▀": PRINT A
T 21,bx+1; PAPER 5; INK 1;"▚▞":
PRINT AT 21,bx+3;"▀"
1050 LET di=-1: GO SUB 1500
1060 LET di=1: GO SUB 1500
1070 RETURN
1500 REM
1530 LET Iy=8+(di=1)*16
1540 LET Ix=((rt+24)*(di=1))+(lt
*(di=-1))
1550 LET up=255-(di=-1)*255
1560 IF di=1 THEN PLOT rt,8: DRA
W INK 2;24,16
1570 PLOT Ix,Iy
1580 LET a=RND: LET ry=((a<=.6)-
(a>.6))*(INT (RND*48)+1)
1590 LET rx=di*(INT (RND*16)+1)
1600 IF di*(Ix+rx)>up*di THEN LE
T rx=up-Ix
1610 IF Iy+ry>144 OR Iy+ry<0 THE
N LET ry=0-ry
1620 LET Ix=Ix+rx: LET Iy=Iy+ry
1630 DRAW INK 2;rx,ry
1640 IF Ix<>up THEN GO TO 1550
1650 RETURN
2000 REM DESTRUICAO
2020 PRINT AT H,X;"  "
2030 FOR J=1 TO 5
2040 PRINT AT H+1,X;"AB": PRINT
AT H+2,X;"CD"
2050 BEEP .05,-(RND*48)
2060 REM
2070 PRINT AT H+1,X;"IJ": PRINT
```

```
AT H+2,X;"KL"
2080 BEEP .05,-(RND*48)
2090 REM
2100 PRINT AT H+1,X;"MN": PRINT
AT H+2,X;"OP"
2110 BEEP .05,-(RND*48)
2120 REM
2130 NEXT J
2140 FOR i=h+1 TO 20: PRINT AT i
-1,x;" ";AT i,x;"DC";AT i+1,x;"
AB": NEXT i
2150 RETURN
8020 BORDER 1: INK 0: PAPER 7: O
VER 0: FLASH 0: CLS
8030 GO SUB 1000:
8040 LET X=0: LET H=X
8050 LET OH=H: LET OX=X
8060 LET X=X+0.5*(X<30)-(INKEY$=
"5")
8070 LET H=H+0.5-(INKEY$="7")*(H
>0)
8080 PRINT AT OH,OX;" ": PRINT
AT OH+1,OX;" "
8090 PRINT AT H,X;"AB": PRINT AT
 H+1,X;"CD"
8100 LET cr=ATTR (H+2,X)+ATTR (H
+2,X+1)
8110 IF cr<58 OR cr=82 OR cr=112
 THEN GO TO 8140
8120 GO SUB 2000
8130 FOR f=1 TO 10: BEEP RND*.05
,RND*20: NEXT f: GO TO 210
8140 IF H>=20 THEN GO TO 8120
8150 IF cr=82 THEN LET Id=1: GO
TO 8170
8160 IF Id=0 THEN GO TO 8050
8170 FOR I=1 TO 6: PAUSE 25: BOR
DER I: NEXT I: BORDER RND*6
8180 PRINT AT 18,bx+1;"EF": PRIN
T AT 19,bx+1;"GH"
8190 BEEP .125,12: BEEP .25,19:
PAUSE 10: BEEP .125,12: BEEP .25
,19: PAUSE 10: BEEP .125,12: BEE
P .25,19: PAUSE 10: BEEP .125,12
: BEEP .25,19: PAUSE 10
8200 BEEP .25,12: BEEP .125,14:
BEEP .125,15: BEEP .125,22: BEEP
 .125,22: BEEP .125,15: BEEP .12
5,14: BEEP .125,19: BEEP .125,14
: BEEP .25,12
8210 GO TO 210
```

Programa "Aquário"

Neste jogo, o objetivo é destruir o maior número possível de sapos, cobras e peixes, que estão passando na sua frente, na água.

Interrompa o programa, quando estiver jogando, e procure entendê-lo; tente modificá-lo, introduzindo novos caracteres gráficos, ou então possibilitando um novo movimento a sua base (para cima).

Quanto mais peixes e cobras destruir, mais pontos você obterá, e talvez até ganhe bônus! Mas, cuidado com o seu total em munição.

```
  1 REM ***     AQUARIO      ***
  5 REM ************************
**
 10 REM JOSE EDUARDO MALUF DE
CARVALHO                ARQUITRON
 15 REM ************************
*
 20 LET b=0: LET hi=0: GO SUB 9
000
 30 GO SUB 8000
 40 LET s=0: LET mis=20: GO SUB
 7000
 50 PRINT AT 5,0;a$''b$
 60 LET h=h+(INKEY$="8" AND h<3
0)-(INKEY$="5" AND h>0)
 70 IF INKEY$="0" THEN LET mis=
mis-1: GO SUB 1000
 75 IF mis<1 THEN GO TO 500
 80 PRINT AT 20,h; INK 6;" ABC
"
 90 LET b$=b$(2 TO )+b$(1)
100 LET a$=a$(LEN a$)+a$( TO (L
EN a$)-1)
101 PRINT AT 0,0; PAPER 5; INK
9;"Placar ";s,"Misseis ";mis;" "
110 IF b$="
          " THEN GO SUB 2000
120 IF a$="
          " THEN GO SUB 3000
140 GO TO 50
500 PRINT AT 10,12; FLASH 1; PA
PER 5;"ACABOU"
501 PRINT AT 0,0; PAPER 5; INK
9;"Placar ";s,"Misseis ";mis;" "
510 PRINT AT 12,10;"Voce marcou
";s
520 IF s>hi THEN LET hi=s
525 PRINT ''TAB 4;"Maior placar
 de hoje ";hi
530 INPUT "Pressione 's' p/ jog
ar outra ou 'n' p/ parar ", LINE
 a$
540 IF a$="n" THEN LOAD ""
550 IF a$="s" THEN GO TO 30
1000 LET b=h+2
1010 FOR f=19 TO 4 STEP -1
1012 PRINT AT f,b;"H"
1019 BEEP .001,-(f-30)
```

```
1020 PRINT AT 5,0;a$''b$
1030 LET h=h+(INKEY$="8" AND h<3
0)-(INKEY$="5" AND h>0)
1040 PRINT AT 20,h; INK 6;" ABC
"
1050 LET b$=b$(2 TO )+b$(1)
1060 LET a$=a$(LEN a$)+a$( TO (L
EN a$)-1)
1070 IF f=5 THEN IF a$(b)<>" " T
HEN GO TO 4000
1080 IF f=7 THEN IF b$(b)<>" " T
HEN GO TO 5000
1090 PRINT AT f,b;" "
1100 NEXT f: RETURN
2000 PRINT AT 10,7; FLASH 1;"Bon
us 1000 pontos"
2010 LET b$="DEFG    O   IJK DEF
G    IJK O O "
2011 LET s=s+1000
2015 BEEP .1,20: BEEP .2,15
2020 FOR p=1 TO 150: NEXT p
2040 PRINT AT 10,7;"
     ": RETURN
3000 PRINT AT 10,7; FLASH 1;"Bon
us 2000 pontos"
3010 LET a$="DEFG    LM N    DEF
G    N LM    "
3011 LET s=s+2000
3015 BEEP .1,20: BEEP .2,15
3020 FOR p=1 TO 150: NEXT p
3030 PRINT AT 10,7;"
     ": RETURN
4000 LET a$(b-2 TO b+2)="    "
4010 LET s=s+20
4011 BEEP .05,-5
4020 RETURN
5000 LET b$(b-2 TO b+2)="    "
5010 LET s=s+10
5011 BEEP .05,-5
5020 RETURN
7000 BORDER 0: PAPER 1: INK 4: C
LS
7010 FOR a=0 TO 3: PRINT PAPER 5
;"
   ": NEXT a
7020 RETURN
8000 LET a$="DEFG    LM N    DEF
G    N LM    "
8010 LET b$="DEFG    O   IJK DEF
G    IJK O O "
8020 LET h=15
8022 LET s=0
8030 RETURN
9000 FOR a=USR "a" TO USR "o"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 0,0,0,127,255,255,255,
255
```



```
9040 DATA 0,24,60,255,255,255,25
5,255
9050 DATA 0,0,0,254,255,255,255,
255
9060 DATA 0,96,BIN 11100111,252,
127,63,15,7
9070 DATA 1,15,31,63,255,255,240
,192
9080 DATA 128,240,248,255,255,63
,31,7
9090 DATA 2,15,BIN 11100111,255,
255,252,248,192
9100 DATA 24,60,60,24,60,36,0,0
9110 DATA 0,BIN 00011111,BIN 011
00111,255,BIN 01100111,BIN 00011
111,0,0
9120 DATA BIN 11000000,BIN 11111
000,BIN 11111110,255,BIN 1111111
0,BIN 11111000,BIN 11000000,0
9130 DATA 0,BIN 0011100,BIN 0111
1000,BIN 11110000,BIN 01111000,B
IN 00111000,0,0
9140 DATA BIN 00000001,BIN 01000
011,BIN 01100111,BIN 00111111,BI
N 00011111,BIN 00111111,BIN 0110
0111,BIN 01000001
9145 DATA BIN 11100000,BIN 11110
000,BIN 11111000,BIN 11001100,BI
N 11001100,BIN 11111000,BIN 1111
0000,BIN 11100000
9150 DATA BIN 00001000,BIN 00011
100,BIN 10111110,BIN 11111101,25
5,BIN 10111010,BIN 00011110,BIN
00001100
9160 DATA BIN 01000011,BIN 10101
110,BIN 00111100,BIN 01001000,BI
N 01001000,BIN 00111100,BIN 1010
1110,BIN 01000011
9170 REM ABCDEFGHIJKLMNO
9180 REM [ilegível]
```

Programa "Aranhas"

Neste jogo, você é a aranha – aparece à esquerda do vídeo –, que deve comer o maior número possível de moscas que vão-se aproximando. Através de movimentos para cima e para baixo, a aranha captura as moscas, mas cuidado que o tempo está passando.

```
  0>REM ARANHAS
 10 REM ************************
 11 REM JOSE EDUARDO MALUF DE
CARVALHO                ARQUITRON
 12 REM ************************
```

```
  20 LET hi=0: GO SUB 9000
  30 GO SUB 8000
  35 GO SUB 6000
  40 PRINT AT 0,0;"Placar ";sc
  50 PRINT AT v,0;"  "
  60 LET v=v+(INKEY$="6" AND v<2
1)-(INKEY$="7" AND v>2)
  70 IF SCREEN$ (v,2)<>" " THEN
GO SUB 1000
  80 PRINT INK 0;AT v-1,0;"C"
  90 PRINT AT v,0; INK 9;"AB"
 100 FOR a=1 TO 6
 110 LET f(a)=f(a)-INT (RND*2):
IF f(a)<1 THEN PRINT AT a*3,1;"
 ": LET f(a)=28
 120 PRINT AT a*3,f(a); INK 1;"D
E "
 130 NEXT a
 140 PRINT AT 0,19;"Tempo perdid
o ";INT time;" ": LET time=time-
.5: IF time<0 THEN GO TO 2000
 150 GO TO 50
 990 STOP
1000 LET sc=sc+2
1010 PRINT AT v,2;"  "
1020 LET f(v/3)=28
1030 PRINT AT 0,0;"Placar ";sc
1040 RETURN
2000 CLS : PRINT AT 5,9; FLASH 1
; BRIGHT 1;"ACABOU O JOGO "
2010 PRINT '' TAB 6; FLASH 1; BRI
GHT 1;"Voce marcou  ";sc
2020 IF sc>hi THEN LET hi=sc
2030 PRINT '';"  Maior pontuacao
de hoje ";hi
2040 INPUT "Pressione"; FLASH 1;
"'s'"; INK 7;" p/ jogar de novo
 ou"; FLASH 1;"'n'"; INK 7;" par
a parar "; LINE a$
2050 IF a$="s" THEN RUN
2060 IF a$="n" THEN LOAD ""
3030 FOR a=1 TO 6: LET f(a)=28:
NEXT a
6000 BORDER 0: PAPER 6: INK 7: B
RIGHT 1: CLS
6100 RETURN
8010 LET v=2
8020 DIM f(6)
8030 FOR a=1 TO 6: LET f(a)=28:
NEXT a
8040 LET sc=0
8050 RANDOMIZE
8060 FOR a=1 TO v-1: PRINT INK 0
;AT a,0;"C": NEXT a
8070 PLOT 0,168: DRAW 255,0
8080 LET time=99
8990 RETURN
9000 FOR a=USR "a" TO USR "e"+7
```



```
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 60,126,255,255,127,127
,149,148
9040 DATA 0,108,248,220,252,248,
96,0
9050 DATA 16,16,16,16,16,16,16,1
6
9060 DATA 0,17,63,94,111,23,45,0
9070 DATA 0,252,2,2,252,224,0,0
9080 REM ABCDE
9090 REM
```

Programa "Asteróides"

Você é uma nave solitária, navegando pelo universo infinito, quando, de repente, surge a sua frente uma chuva de asteróides. Desvie-se deles, através das teclas "5" (para a esquerda) e "8" (para a direita), e, se estiverem muito próximos, acione o seu campo de defesa eletromagnético, com a tecla "0".

Analise o programa, e veja na linha 50 uma outra forma de tirar partido da lógica-padrão do micro (verdadeiro/falso).

Analise também a linha 405, onde consta uma opção para gravar o jogo. De propósito está faltando uma linha, que executa essa condição. Digite-a.

Se quiser alterar as cores-padrão deste jogo, tome cuidado, por que ele está baseado no comando ATTR (linha, coluna).

```
   0)CLEAR : BORDER 1: PAPER 1:
INK 7: CLS : GO SUB 9000: LET b=
0
   1 REM ASTEROIDES
  10 REM ************************
  20 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
  30 REM ************************
  40 LET e=0: LET b=0: INK 0: CL
S : LET t=0: LET p=5: LET l=15:
LET l1=l: LET f=6
  50 POKE 23692,255: LET l=l+(IN
KEY$="8" AND l<29)-(INKEY$="5" A
ND l>0): IF INKEY$="0" THEN LET
f=f-1: IF f>0 THEN PRINT AT 8,l;
 INK 6;"PRO": GO TO 64
  62 IF INKEY$="0" THEN LET f=f-
1: IF f>0 THEN PRINT AT 8,l; INK
 6;"PRO"
  64 IF ATTR (8,l+2)=15 OR ATTR
(8,l)=15 THEN LET p=p-1
```

```
  66 IF ATTR (6,l+1)=15 THEN LET
 p=p-2
  68 IF p<1 THEN GO TO 400
 100 PRINT AT 7,l1;"   ";AT 21,3
1;"": PRINT : PRINT AT 7,l; INK
6;"S"; INVERSE 1;p; INVERSE 0;"T
": LET t=t+1: LET l1=l: PRINT AT
 21,RND*31; INK 7;CHR$ (RND*11+1
44);AT 21,RND*31;CHR$ (RND*11+14
4): GO TO 50
 400 FOR n=1 TO 3: PRINT AT 6,l1
;"   ";AT 7,l1; INK 2;" M ";AT 8
,l1;"   ": PAUSE 2: PRINT AT 6,l
1; INK 6;"OGU";AT 7,l1;"N N";AT
8,l1;"POR": PAUSE 2: NEXT n: FOR
 n=6 TO 8: PRINT AT n,l1;"   ":
NEXT n
 402 CLS : IF b<t-13 THEN LET b=
t-13
 403 INK 7: IF e<b THEN LET e=b:
 PRINT AT 6,0;"Voce conseguiu o
melhor placar  Entre para o Hall
 da Fama ";"Digite duas iniciai
s do seu nome": LET n$="": FOR n
=1 TO 2: PAUSE 0: LET n$=n$+INKE
Y$+".": NEXT n: CLS
 405 PRINT ;"Placar= ";t-13,"Mel
hor placar ";b';"Recorde ";: PRI
NT FLASH 1;e;" de ";n$';: FOR n=
1 TO 50: NEXT n: PRINT FLASH 0;"
Pressione <P> p/ novo jogo, <S>
   para gravar ou <F> p/ termina
r": PAUSE 0: GO TO (5 AND INKEY$
="p")+(430 AND INKEY$="s")+(450
AND INKEY$="f")
 430 SAVE "asterÒides" LINE 1
 450 PRINT ';';"Tchau, tchau "
 460 GO TO 460
9020 FOR m=1 TO 21
9030 READ p$
9040 FOR n=0 TO 7
9050 READ a: POKE USR p$+n,a
9060 NEXT n: NEXT m: RETURN
9065 BORDER 1: PAPER 1: CLS : IN
K 7: RETURN
9070 DATA "a",62,33,255,129,129,
66,37,26
9080 DATA "b",112,148,138,65,129
,129,94,32
9090 DATA "c",0,0,24,36,36,56,0,
0
9100 DATA "d",0,0,48,40,40,16,0,
0
9110 DATA "e",0,24,40,40,16,0,0,
0
9120 DATA "f",112,136,136,68,40,
48,0,0
9130 DATA "g",0,0,0,56,44,16,0,0
```



```
9140 DATA "h",0,0,30,18,34,84,23
2,0
9150 DATA "i",94,177,130,228,34,
65,66,60
9160 DATA "j",0,62,65,66,34,17,1
4,0
9170 DATA "k",96,144,144,116,10,
49,65,126
9180 DATA "l",62,65,242,9,247,13
6,144,96
9190 DATA "m",145,62,16,7,244,8,
74,137
9200 DATA "n",100,24,198,0,222,0
,24,102
9210 DATA "o",0,4,34,18,1,68,50,
9
9220 DATA "p",9,50,68,1,18,34,4,
0
9230 DATA "q",0,8,170,42,73,65,1
37,137
9240 DATA "r",144,76,34,128,72,6
8,32,0
9250 DATA "s",0,3,15,15,120,200,
255,56
9260 DATA "t",0,192,240,240,30,1
9,255,28
9270 DATA "u",0,32,68,72,128,36,
76,144
9280 REM ABCDEFGHIJKLMNOPQRSTU
9290 REM
```

Programa "Balde"

O objetivo deste jogo consiste em captar o maior número de pingos de água que estão caindo da parede, dentro do seu balde, através dos movimentos com as teclas "5" e "8". Você tem no total, 99 pingos para encher o balde. Quando ouvir um som na tevê, significa bônus no placar.

Vale para a linha 100 deste programa o mesmo comentário da linha 50 do programa anterior.

O programa começa chamando as sub-rotinas das linhas 8000 e 9000.

Na sub-rotina que inicia na linha 8000 são feitas as inicializações das variáveis, e a chamada para a sub-rotina que começa na linha 7000, onde é feita a impressão da parede.

Na sub-rotina que inicia na linha 9000, são criados os caracteres gráficos, retornando à estrutura principal do programa, da linha 46 até 210.

```
   0>REM ***     BALDE     ***
  10 REM ************************
*
  20 REM JOSE EDUARDO MALUF DE
CARVALHO                ARQUITRON
  25 REM ************************
*
  30 GO SUB 8000
  40 GO SUB 7000
  45 GO SUB 9000
  50 LET p=INT (RND*26)+3: LET d
=d+1: PRINT AT 0,22;"Pingo ";d
  60 FOR l=6 TO 18
  70 PRINT AT 20,v; INK 6;"  GH
 ";AT 21,v;"  IJ  "
  80 PRINT AT l,p; INK 5; BRIGHT
 1;"  ";AT l+1,p;"CD";AT l+2,p;"
EF"
  90 IF l+3=20 AND v+1=p+1 OR v+
2=p+1 OR v+2=p OR v+1=p THEN LET
 sc=sc+2: PRINT AT 0,0;"Placar "
;sc: BEEP .008,10
 100 LET v=v+(1.5*(INKEY$="8" AN
D v<25))-(1.5*(INKEY$="5" AND v>
0))
 110 NEXT l
 120 PRINT AT l,p;"  ";AT l+1,p;
"  "
 130 IF d=99 THEN GO TO 150
 140 GO TO 50
 150 PRINT AT 5,11;"ACABOU "
 160 PRINT '''''"      Voce marco
u     ";sc
 170 IF sc>hi THEN LET hi=sc
 180 PRINT '''"Maior placar de h
oje ";hi
 190 INPUT "Pressione 's' p/ out
ro jogo ou 'n' p/parar "; LINE a
$
 200 IF a$="n" THEN LOAD ""
 210 IF a$="s" THEN GO TO 30
7000 CLS
7010 PRINT "PLACAR ";sc
7020 PRINT INK 2;"ABABABABABABAB
ABABABABABABABABAB"
7025 PRINT INK 3;"BABABABABABABA
BABABABABABABABABA"
7030 PRINT INK 2;"ABABABABABABAB
ABABABABABABABABAB"
7035 PRINT INK 3;"BABABABABABABA
BABABABABABABABABA"
7990 RETURN
8000 BORDER 0: PAPER 1: INK 7: C
LS : BRIGHT 1
8010 LET sc=0
8020 LET v=15
8030 LET d=0
8990 RETURN
```



```
9000 FOR a=USR "a" TO USR "j"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 255,255,255,255,255,25
5,0,0
9040 DATA 252,252,252,252,252,25
2,0,0
9050 DATA 0,0,0,0,0,1,3,7
9060 DATA 0,0,0,0,0,128,192,224
9070 DATA 7,7,15,15,15,7,7,3
9080 DATA 224,224,240,240,240,22
4,224,192
9090 DATA 0,0,127,63,95,111,119,
59
9100 DATA 0,0,254,254,253,251,25
1,246
9110 DATA 62,63,63,63,31,31,31,3
1
9120 DATA 14,254,254,254,252,252
,252,252
9130 REM ABCDEFGHIJ
9140 REM ▀▄ ▗ ▚ ▄▀▀▀
```

Programa "Blackjack"

Este é o conhecido jogo de cartas "21", onde você deve totalizar, com as cartas que tem na mão, 21 pontos. Se por acaso conseguir esse total com apenas duas cartas, então você fez um "Blackjack", e ganha em dobro o que apostou.

O TK 90X é muito bom, e de início, deu-lhe um crédito de Cr$ 100.

A sub-rotina da linha 8000 define os quatro naipes do baralho.

Cada carta é desenhada através da sub-rotina que começa na linha 8500.

Existem 13 cartas disponíveis, mas a solução empregada neste programa foi a de pegar na ROM a estrutura-padrão de cada caractere a ser impresso no vídeo, e então ampliá-lo duas vezes, tanto na largura quanto na altura, para ser armazenado no grupo de caracteres gráficos A B C D. Como não existe um caractere-padrão para o número 10, o caractere gráfico E foi reservado para esse propósito.

Como nós já vimos no início deste livro, para ser impresso, cada caractere consome 8 bytes, e a definição standard para este grupo de bytes está armazenada na ROM, começando no endereço 15360. Portanto, o endereço de início de cada caractere na ROM pode ser determinado pela fórmula:

15360 + 8 * código do caractere (linha 9020)

Na linha 270, você deverá notar uma notação estranha:

270 LET a = p(i) = 11

No programa, a variável a é usada para armazenar o número de AZ que existe em cada mão. O significado desta linha é:

LET a = 0

IF p(i) = 11 THEN LET a = 1

É uma outra forma de tirar partido da lógica do micro.

Divirta-se.

```
   0 REM ***   BLACKJACK   ***
  10 REM ************************
  20 REM  JOSE EDUARDO MALUF DE
CARVALHO                ARQUITRON
  30 REM ************************
  40 POKE 23561,20: POKE 23562,3
 100 BORDER 5: PAPER 6: BRIGHT 1
: INK 9: OVER 0: CLS
 110 LET b$="N23456789TAJQK"
 120 LET a=0: LET m=100
 130 GO SUB 8000
 140 DIM p(4)
 150 IF m=0 THEN PRINT AT 10,1;"
APOSTAS TERMINARAM";AT 19,1;"VOC
E NAO TEM MAIS CREDITO.      VOLT
E QUANDO TIVER DINHEIRO": STOP
 160 PRINT AT 10,1;"VOCE TEM CRE
DITO DE  $";m: PRINT "QUANTO QUE
R APOSTAR "
 170 INPUT b
 180 CLS
 190 IF b>m THEN GO TO 160
 200 LET m=m-b
 210 PRINT INK 0;AT 0,0;"Suas ca
rtas";AT 11,0;"Cartas do TK 90X"
 220 PRINT AT 9,1;"Total = ";AT
20,1;"Total = "
 230 FOR i=2 TO 1 STEP -1
 240 GO SUB 500
 250 NEXT i
 260 FOR i=1 TO 2
 270 LET a=p(i)=11
 280 IF i=1 THEN INPUT FLASH 1;
PAPER 2; INK 7;"Quer outra carta
? ";a$: LET a$=a$+" ": IF a$(1)
="n" THEN GO TO 340
 290 GO SUB 500
 300 IF p(i)>21 THEN LET i=2: GO
 TO 340
 310 IF p(i+2)=5 THEN GO TO 340
 320 IF i=2 AND p(i)>16 THEN GO
TO 340
```



```
 330 GO TO 280
 335 BEEP .005,RND*20
 340 NEXT i
 350 IF p(i)<=21 AND p(3)=5 AND
(p(2)<>21 OR p(4)<>2) THEN GO TO
 360
 360 IF p(1)=21 AND p(3)=2 AND (
p(2)<>21 OR p(4)<>2) THEN LET b=
b+1.5: PRINT FLASH 1;AT 5,18;"BL
ACKJACK": GO TO 380
 370 IF (p(2)<=21 AND p(2)>=p(1)
) OR p(1)>21 THEN LET b=0
 380 LET m=m+2*b: BEEP .05,0
 390 GO TO 130
 500 LET p(i+2)=p(i+2)+1
 510 LET c=INT (13*RND)+2
 520 IF c=11 THEN LET a=a+1
 530 LET p(i)=p(i)+c*(c<12)+10*(
c>11)
 540 IF p(i)<22 OR a=0 THEN GO T
O 570
 550 BEEP .05,0: LET a=a-1
 560 BEEP .05,10: LET p(i)=p(i)-
10
 570 BEEP .05,20: GO SUB 8500
 580 GO SUB 9000
 590 PRINT INK 0;AT 11*i-2,9;p(i
)
 600 RETURN
8000 DATA 0,78,209,81,81,81,78,0
8010 DATA 102,255,255,255,126,60
,24,24
8020 DATA 24,60,126,255,255,126,
60,24
8030 DATA 24,60,126,255,255,219,
24,60
8040 DATA 60,60,219,231,231,219,
24,24
8050 RESTORE 8000
8060 FOR k=USR "e" TO USR "e"+39
: READ x: POKE k,x: BEEP .005,RN
D*30: NEXT k
8070 RETURN
8080 REM ABCDEFGHI
8090 REM  ¬┌10▀▄▌▐
8500 LET p=11*i-7: LET q=6*p(i+2
)-5
8510 INK 0: PRINT AT p-3,q;"▛▀▀
▀▜": FOR k=1 TO 5: PRINT AT p-3+
k,q;"▌    ▐": BEEP .05,30: NEXT
k: PRINT AT p+3,q;"▙▄▄▄▄▟": INK
0
8520 LET suit=1+INT (4*RND): INK
 2*(suit<3): PRINT AT p-2,q+3;CH
R$ (148+suit)
8530 RETURN
9000 LET x=USR "a"
9010 FOR k=0 TO 31: POKE x+k,0:
```

```
BEEP .005,RND*-10: NEXT k
9020 LET s=8*CODE b$(c)+15360: I
F b$(c)="T" THEN LET s=USR "e"
9030 FOR j=0 TO 7: LET v=PEEK (s
+j): LET c=0: LET d=256
9040 FOR k=8 TO 1 STEP -1: LET d
=d/2: IF v>=d THEN LET c=c+3*d*d
: LET v=v-d
9045 BEEP .005,RND*-10
9050 NEXT k
9060 LET h=INT (c/256): LET l=c-
256*h
9070 POKE x,h: POKE x+1,h: POKE
x+8,l: POKE x+9,l
9080 LET x=x+2+8*(j=3)
9090 PRINT AT p,q+2;"AB";AT p+1,
q+2;"CD"
9095 BEEP .005,RND*30
9100 NEXT j
9110 RETURN
```

Programa "Bola"

Atenção: não deixe a bola bater nem no chão nem do seu lado, pois assim você perde uma chance.

O objetivo deste jogo consiste em rebater a bola o maior número de vezes possível, para acumular pontos.

Repare que a bola "se achata" quando bate em superfícies horizontais, e "se alonga", quando bate em superfícies verticais.

Foram definidos apenas quatro caracteres gráficos para este programa.

```
  0>REM ***     BOLA      ***
 10 REM ************************
 20 REM JOSE EDUARDO MALUF DE
CARVALHO                AR@UITRON
 30 REM ************************
*
 40 LET hi=0
 50 GO SUB 9000
 60 GO SUB 8000
 65 PRINT AT 21,15; INK 2;"CHAN
CES "; INK 7;li
 70 PRINT AT 20,v;" "
 80 LET v=v+(INKEY$="8" AND v<3
1)-(INKEY$="5" AND v>0)
 90 PRINT AT 20,v; INK 5; BRIGH
T 1;"D"
```



```
 100 PRINT AT a,b;" "
 110 LET a=a+m: IF a>19 OR a<1 T
HEN BEEP .05,10: LET m=-m
 120 LET b=b+n: IF b>30 OR b<1 T
HEN BEEP .05,15: LET n=-n
 130 PRINT AT a,b; INK c;"A"
 140 IF a=0 OR a=20 THEN PRINT A
T a,b; INK c;"B"
 150 IF b=0 OR b=31 THEN PRINT A
T a,b; INK c;"C"
 160 IF a=19 AND b=v THEN PRINT
AT a,b; INK c;"B": BEEP .05,15:
LET sc=sc+2: LET m=-m
 170 IF a=20 AND b=v THEN GO SUB
 500
 180 PRINT AT 21,0;"Placar"; INK
 7;sc
 190 GO TO 70
 500 PRINT AT 20,v; FLASH 1; BRI
GHT 1; INK 2;"D"
 510 LET li=li-1
 515 PRINT AT 21,15;"CHANCES ";
INK 7;li
 520 FOR p=1 TO 20: BEEP .006,-p
: NEXT p
 525 IF li=0 THEN GO TO 600
 526 LET a=INT (RND*5)+2: LET b=
INT (RND*27)+2
 530 RETURN
 600 PRINT AT 2,10; FLASH 1; BRI
GHT 1;"ACABOU O JOGO"
 610 PRINT '''"   VOCE MARCOU ";
 INK 7;sc
 620 IF sc>hi THEN LET hi=sc
 630 PRINT '''" Maior placar de
hoje ";hi
 640 INPUT "Pressione 's' p/ out
ro jogo ou 'n' p/ parar "; LINE
a$
 650 IF a$="n" THEN LOAD ""
 660 IF a$="s" THEN CLS : GO TO
60
8000 BORDER 1: PAPER 0: INK 7: C
LS
8005 LET v=15: LET a=INT (RND*5)
+2: LET b=INT (RND*27)+2
8010 LET sc=0: LET m=1: LET n=1
8020 LET c=6: LET li=5
8990 RETURN
9000 FOR a=USR "a" TO USR "d"+7
9010 READ ur: POKE a,ur
9020 NEXT a: RETURN
9030 DATA 60,126,255,255,255,255
,126,60
9040 DATA 0,0,0,60,126,255,126,6
0
9050 DATA 8,28,62,62,62,62,28,8
9060 DATA 255,129,129,129,129,12
```

```
9,129,255
9070 REM ■■■■□
9080 REM ABCD
```

Programa "Colecionador de Ovos"

Mais um jogo interessante, onde você é um colecionador de ovos, que deve apanhar a maior quantidade de ovos raríssimos que aparecem na tela, tomando o cuidado para não ser atingido pelos terríveis raios laser que são disparados de bases fixas.

Para apanhar os ovos, você deve dirigir-se até a base que os sustenta, onde está marcado um "X", levá-los até a porção inferior da tela, nos locais vazios, e, da mesma forma que os apanhou, depositá-los também por baixo da prateleira, onde está indicado um "-".

As teclas de movimento são as mesmas que em qualquer outro programa deste livro, ou seja, as teclas cursoras (5 - 6 - 7 - 8).

```
   1 REM ***       OVOS         ***
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   4 REM ************************
  55 LET hi=0
  60 GO SUB 9000
  70 GO SUB 8000
  80 GO SUB 7000
 150 PRINT AT h,v;" "
 160 LET h=h+(INKEY$="6" AND h<2
1)-(INKEY$="7" AND h>0): LET v=v
+(INKEY$="8" AND v<31)-(INKEY$="
5" AND v>0)
 170 IF SCREEN$ (h,v)<>" " THEN
GO SUB 6000
 180 IF RND>.5 THEN GO SUB 6500
 190 PRINT AT h,v; INK 3+ovo;"G"
 191 BEEP .008,0
 200 IF vidas<1 THEN PRINT AT 21
,0;"      Voce foi atingido.
": GO TO 1500
 210 PRINT AT 21,0;"Vidas ";vida
s
 230 PRINT AT 21,15;"Ovos coleta
dos";tot
 998 GO TO 150
 999 STOP
1500 BEEP 1,4: BEEP 1,4: BEEP .3
,4: BEEP 1,2,4: BEEP .75,7: BEEP
 .5,6: BEEP 1,6: BEEP .3,4: BEEP
 .7,4: BEEP .6,3: BEEP 1,4
1501 IF sc>hi THEN LET hi=sc
1510 INPUT "Pressione 's' p/ out
ro jogo ou 'n' p/ parar"; LINE a
$: CLS
1512 IF a$="n" THEN STOP
1514 IF a$="s" THEN GO TO 70
2000 IF ovo=1 THEN RETURN
2005 PRINT AT 1,v;"  ";AT 2,v; I
NK 6;"EF"
2010 LET ovo=1
2015 BEEP .1,10
2020 RETURN
2500 IF ovo=0 THEN PRINT AT h,v;
 INK 1;"_": LET h=h+1
2510 PRINT AT 17,v; INK 6;"AB";A
T 18,v; OVER 1; INK 6;"CD"
2515 LET ovo=0
2516 PRINT AT 19,v;" "
2517 BEEP .1,20: BEEP .15,15
2520 LET tot=tot+1: IF tot=6 THE
N GO TO 9500
2530 RETURN
6000 LET b$=SCREEN$ (h,v)
6010 IF b$="X" THEN GO TO 2000
6020 IF b$="_" THEN GO TO 2500
6030 LET vidas=vidas-1
6031 LET ovo=0
6040 PRINT AT h,v; FLASH 1; INK
3;"G": BEEP .1,-10: FOR p=1 TO 2
0: NEXT p: RETURN
6500 GO TO (INT (RND*3)+1)*100+6
500
6600 PLOT 9,m1: DRAW INK 2; BRIG
HT 1;245,0
6605 GO SUB 6900
6606 PAUSE 2
6610 DRAW OVER 1;-248,0: PLOT OV
ER 1;254,m1: RETURN
6700 PLOT m2,167: DRAW INK 2; BR
IGHT 1;0,-120
6705 GO SUB 6900
6706 PAUSE 2
6710 DRAW OVER 1;0,120: RETURN
6800 PLOT m3,167: DRAW INK 2; BR
IGHT 1;0,-120
6805 GO SUB 6900
6806 PAUSE 2
6810 DRAW OVER 1;0,120: RETURN
6900 IF SCREEN$ (h,v)<>" " THEN
GO TO 6000
6910 RETURN
7000 LET n=0: PRINT AT 0,0;"Plac
ar ";sc;TAB 23;"Record ";hi
7005 FOR a=1 TO 4
7010 PRINT AT 1,n;"  ";
7020 FOR b=1 TO e(a)
```

```
7030 PRINT INK 6;"AB";
7040 NEXT b
7050 PRINT AT 2,n;"  ";
7060 FOR b=1 TO e(a)
7070 PRINT INK 6;"CD";
7080 NEXT b
7090 OVER 1
7095 PRINT AT 2,n;"  ";
7100 FOR b=1 TO e(a)
7110 PRINT INK 6;"EF";
7120 NEXT b
7122 PRINT AT 3,n;"  ";
7125 FOR b=1 TO e(a): PRINT INK
6;"X ";: NEXT b
7130 LET n=n+(e(a)*2)+2
7140 NEXT a
7150 PRINT AT 0,9; INK 2;"I";AT
0,21;"I";4,0;"H"
7160 PRINT AT 18,8; INK 1;"EF EF
 EF EF EF EF"
7165 PRINT AT 19,8; INK 1;"_  _
  "
7170 OVER 0
7990 RETURN
8000 BORDER 0: INK 7: PAPER 0: C
LS
8010 DIM e(4)
8020 RESTORE 9120: FOR z=1 TO 4:
 READ e(z): NEXT z
8030 LET h=10: LET v=15
8040 LET sc=0: RANDOMIZE
8050 LET m1=17*8+4: LET m2=9*8+4
: LET m3=21*8+4
8070 LET ovo=0
8080 LET vidas=5
8090 LET tot=0
8100 RETURN
9000 FOR a=USR "a" TO USR "j"-1
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 1,3,7,7,15,15,15,31
9040 DATA 128,192,224,224,240,24
0,248,248
9050 DATA 31,31,31,31,15,15,7,3
9060 DATA 248,248,248,248,240,24
0,224,192
9070 DATA 192,192,192,192,192,22
4,224,255
9080 DATA 3,3,3,3,3,7,7,255
9090 DATA 56,56,16,254,16,16,40,
68
9100 DATA 128,192,240,255,255,24
0,192,128
9110 DATA 255,126,60,60,60,24,24
,24
9120 DATA 3,2,2,3
9500 LET sc=sc+100
9505 LET h=10: LET v=15: LET tot
```



```
=0: LET ovo=0
9510 PRINT AT 0,0;"Placar ";sc
9515 IF sc>hi THEN LET hi=sc
9520 FOR p=1 TO 31: BEEP .01,p:
BEEP .005,-p: NEXT p
9530 CLS
9540 GO TO 80
9550 REM ABCDEFGHI
9560 REM
```

Programa "Defesa aérea"

Aqui, você é um exímio piloto de um caça, que deve impedir, de todas as maneiras possíveis, que exterminadores alienígenas cheguem à Terra, destruindo tudo que estiver na superfície.

Além de tomar cuidado com as bombas que são lançadas contra seu avião, você não deve encostá-lo nem no limite superior da tela nem na superfície terrestre, pois aos poucos você acaba com ele dessa maneira.

Através das teclas cursoras de movimento para cima e para baixo (6 - 7), aumente ou diminua a altitude de seu avião, e, com a tecla 0, dispare raios mortais contra os exterminadores, destruindo-os.

Repare na rotina, escrita em linguagem de máquina, que faz o SCROLL lateral da imagem, com 36 bytes de comprimento, na declaração DATA da linha 3000.

O que esta rotina faz é mudar de posição lateralmente cada linha da tela, em alta resolução, isto é, o caractere que estava na posição 0,1, passará para a posição 0,0, daí para 0, 31 e assim por diante.

A rotina executa o mesmo procedimento com os atributos da tela.

Observe nas linhas 140 e 150 os comandos POKE desta rotina. Se quiser utilizá-la em outro programa, com outros endereços, basta alterar os dados destas linhas, e não é necessário que você entenda de linguagem de máquina. Se tiver curiosidade, anote os bytes da linha DATA, e, através do Apêndice D, do manual da máquina, veja o significado das instruções em linguagem de máquina. É muito mais complicado, mas também muito mais fascinante!

A listagem possui algumas linhas escritas com cores diretamente do teclado. São elas:

5120 PRINT OVER 1; AT bx, by; "esp GRA J"; AT nx, by; "EXT SHI 2 esp GRA J"

5270 IF ax = sx AND ay > sy + 4 AND ay < 30 THEN FOR k = 1 TO 11: SOUND .1,5: PRINT OVER 1; AT ax, ay - 1; "esp GRA D GRA E": NEXT k: LET ax = INT (15 * RND): LET ay = 29: PRINT AT ax, ay - 1; "EXT SHI 4 esp GRAD GRA E" EXT SHI 0: LET sc = sc + 100

5280 IF bx = sx AND by > sy + 4 AND by < 30 THEN FOR k = 1 TO 5: SOUND .1, 10: PRINT OVER 1; AT bx, by; "EXT SHI 2 esp GRA J": NEXT k: LET bx = 1 + INT (RND * 14): LET by = 30: PRINT AT bx, by; "EXT SHI 2 esp GRA J" EXT SHI 0: LET sc = sc + 20

5350 PRINT OVER 1; AT ax, ay; "esp GRA D GRA E"; AT nx, ny; "EXT SHI 4 esp GRA D GRA E" EXT SHI 0

7000 IF ABS (bx-ax) < = 1 AND ABS (by-ay) < = 1 THEN FOR k = 1 TO 11: SOUD .1, 5: PRINT OVER 1; AT ax, ay; "esp GRA D GRA E": NEXT k: LET ax = INT (15 * RND): LET ay = 29: PRINT AT ax, ay; "EXT SHI 4 GRA D GRA E" EXT SHI 0

7020 PRINT OVER 1; AT ax, ay; "esp GRA D GRA E"; AT ax-1, ay; "EXT SHI 4 esp GRA D GRA E" EXT SHI 0

7070 PRINT OVER 1; AT sx, sy-1; "esp GRA F GRA G GRA H GRA I"; AT sx, sy; "EXT SHI 6 esp GRA F GRA G GRA H GRA I" EXT SHI 0

7145 LET ay = ay + 1: PRINT OVER 1; AT ax, ay-1; "esp GRA D GRA E"; AT ax, ay; "EXT SHI 4 esp GRA D GRA E" EXT SHI 0

7300 PRINT OVER 1; AT sx, sy-1; "esp GRA F GRA G GRA H GRA I"; AT sx, sy; "EXT SHI 6 esp GRA F GRA G GRA H GRA I" EXT SHI 0

8560 PRINT OVER 1; AT sx, sy-1; "esp EXT SHI 6 GRA F GRA G GRA H GRA I" EXT SHI 0: SOUND k/50, 12-2*k: NEXT k

8710 PRINT OVER 1; AT sx, sy-1; "esp EXT SHI 6 GRA F GRA G GRA H GRA I" EXT SHI 0: SOUND .1,15-2 * k: NEXT k

A legenda utilizada significa:

esp – pular um espaço – existem outras linhas na listagem do programa onde também se deve pular um espaço;

GRA – utilizar modo gráfico (cursor em G)

EXT – utilizar modo extended (cursor em E)

SHI – pressionar a tecla CAPS SHIFT


```
   1 REM **  DEFESA AEREA     **
   2 REM **********************
   3 REM JOSE EDUARDO MALUF  DE
CARVALHO       ARQUITRON
   4 REM **********************
 100 RESTORE 2000
 110 FOR i=USR "a" TO USR "k"-1
 120 READ x: POKE i,x
 130 NEXT i
 140 RESTORE 3000: CLEAR 32500
 150 FOR i=32500 TO 32535: READ
x: POKE i,x: NEXT i
 160 BRIGHT 1: BORDER 1: PAPER 1
: INK 7: CLS
 161 PRINT AT 15,0;"   Teclas:"'
"7 - para subir";"6 - Para desce
r";"0 - para atirar": PAUSE 300:
 CLS
 170 PLOT 0,8
 180 LET y=8
 190 LET p=-1
 200 FOR j=1 TO 3
 210 LET a=3+INT (7*RND)
 220 GO SUB 8100
 230 DRAW 4,15-y: DRAW 8,0
 240 LET y=15
 250 LET p=p+a+1
 260 FOR k=1 TO INT (4*RND)+1: P
RINT INK 2; OVER 1;AT 20-k,p;"C"
: NEXT k: PRINT INK 3;AT 20-k,p;
"B";AT 19-k,p;"A"
 270 NEXT j
 280 LET a=30-p
 290 GO SUB 8100
 300 DRAW 11,8-y
 310 LET sx=3: LET sy=10
 320 LET ax=INT (15*RND): LET ay
=26
 330 LET bx=INT (RND*15): LET by
=30: IF bx=ax THEN GO TO 330
 340 LET t=0
 350 LET e=0: LET h=3
 360 LET dm=0: LET sc=0
1000 PRINT INK 1;AT sx,sy;" FGHI
"
1010 PRINT INK 4;AT ax,ay;" DE"
1020 PRINT INK 2;AT bx,by;" J"
2000 DATA 1,57,189,167,146,254,1
24,124,56,56,56,104,76,68,68,198
2010 DATA 126,36,60,24,60,102,19
5,255
2020 DATA 63,99,193,193,127,28,7
,3,252,198,131,131,254,56,224,19
2
2030 DATA 255,63,15,7,3,7,255,0,
192,240,252,255,255,255,240,3,0,
0,31,249,224,255,252,240,0,0,0,1
92,248,255,0,0
2040 DATA 28,28,244,199,227,47,5
```

```
6,56
3000 DATA 33,1,64,17,0,64,26,1,3
1,0,237,176,16,35,19,62,88,168,3
2,242,6,22,197,26,1,31,0,237,176
,16,35,19,193,16,243,201
4000 REM ABCDEFGHIJKLMNOPQRSTU
4010 REM ABCDEFGHIJ
5000 LET v=USR 32500
5010 LET x$=INKEY$
5020 LET nx=sx+2*(x$="6")*(sx<18
)-2*(x$="7")*(sx>0)
5030 IF nx=sx THEN GO TO 5060
5040 LET v$=SCREEN$ (nx,sy+1): L
ET w$=SCREEN$ (nx,sy+2): LET x$=
SCREEN$ (nx,sy+3): LET v$=v$+w$+
x$
5050 IF v$<>"    " THEN GO TO 850
0
5060 IF SCREEN$ (nx,sy+4)<>" " T
HEN GO TO 8500
5070 PRINT OVER 1; INK 9;AT sx,s
y-1;" FGHI"; INK 9;AT nx,sy;" FG
HI"
5080 LET sx=nx
5090 LET by=by-1: IF by<0 THEN L
ET by=31
5100 IF RND>.7 OR by=31 THEN GO
TO 5140
5110 LET nx=bx+(sx-1>bx)-(sx-1<b
x)+(by=sy+2)
5120 PRINT OVER 1;AT bx,by;" J";
AT nx,by;" J"
5130 LET bx=nx
5140 IF bx=sx AND ABS (by-sy-1)<
=1 THEN GO TO 8500
5200 IF x$<>"0" THEN GO TO 5290
5220 PLOT INK 3; OVER 1;8*sy+43,
170-8*sx
5230 DRAW INK 3; OVER 1;212-8*sy
,0
5240 BEEP .1,6
5250 PLOT INK 9; OVER 1;8*sy+43,
170-8*sx
5260 DRAW INK 9; OVER 1;212-8*sy
,0
5270 IF ax=sx AND ay>sy+4 AND ay
<30 THEN FOR k=1 TO 11: BEEP .1,
5: PRINT OVER 1;AT ax,ay-1;" DE"
: NEXT k: LET ax=INT (15*RND): L
ET ay=29: PRINT AT ax,ay-1;" DE"
: LET sc=sc+100
5280 IF bx=sx AND by>sy+4 AND by
<30 THEN FOR k=1 TO 5: BEEP .1,1
0: PRINT OVER 1;AT bx,by;" J": N
EXT k: LET bx=1+INT (RND*14): LE
T by=30: PRINT AT bx,by;" J": LE
T sc=sc+20
5290 LET ay=ay-1: IF ay<0 THEN L
```



```
ET ay=31
5300 IF RND>.7 OR ay>=29 THEN GO
 TO 5000
5310 LET nx=ax+(ax<18)-t: IF ax=
18 THEN GO TO 5330
5320 LET v$=SCREEN$ (nx,ay+1): L
ET w$=SCREEN$ (nx,ay+2): IF v$+w
$<>"    " THEN GO TO 7000
5330 LET ny=ay+(ax=18)
5340 IF ny<>ay AND SCREEN$ (nx,n
y+2)<>" " THEN GO TO 7000
5350 PRINT OVER 1;AT ax,ay;" DE"
;AT nx,ny;" DE"
5360 LET ay=ny: LET ax=nx
6000 GO TO 5000
7000 IF ABS (bx-ax)<=1 AND ABS (
by-ay)<=1 THEN FOR k=1 TO 11: BE
EP .1,5: PRINT OVER 1;AT ax,ay;"
 DE": NEXT k: LET ax=INT (15*RND
): LET ay=29: PRINT AT ax,ay;" D
E"
7010 IF ABS (sx-ax)<=2 AND ABS (
by-sy)<=2 THEN GO TO 8500
7020 PRINT OVER 1;AT ax,ay;" DE"
;AT ax-1,ay;" DE"
7030 LET ax=ax-1: LET ay=ay-1: L
ET by=by-1
7040 IF ay<0 THEN LET ay=31
7050 IF by<0 THEN LET by=31
7060 LET v=USR 32500
7070 PRINT OVER 1;AT sx,sy-1;" F
GHI";AT sx,sy;" FGHI"
7080 LET f=0
7090 FOR x=ax-3 TO ax+3
7095 IF x>20 THEN GO TO 7125
7100 FOR y=ay+1 TO ay+3
7105 IF y<0 OR y>31 THEN GO TO 7
120
7110 IF ATTR (x,y)=11 THEN PRINT
 AT x,y; INK 1;" ": LET f=f+1
7120 NEXT y
7125 NEXT x
7130 IF f=2 THEN GO TO 7200
7140 PRINT AT ax,ay+3; INK 1;" "
;AT ax+1,ay+2;"   ";AT ax+2,ay+2
;"
7145 LET ay=ay+1: PRINT OVER 1;A
T ax,ay-1;" DE";AT ax,ay;" DE"
7150 LET e=e+1: IF e=3 THEN GO T
O 8550
7160 GO TO 5000
7200 PRINT OVER 1;AT ax,ay;" DE"
;AT ax-1,ay;" DE"
7210 LET ax=ax+1: REM LET ay=ay-
1: IF ay<0 THEN LET ay=31
7220 LET by=by-1: IF by<0 THEN L
ET by=31
7230 LET v=USR 32500
```

```
7240 PRINT INK 4;AT ax+1,ay;" A"
;AT ax+2,ay;" B"
7250 FOR k=ax-1 TO 0 STEP -1
7260 PRINT OVER 1; INK 4;AT k+1,
ay;"DE";AT k+2,ay;" A";AT k+3,ay
;" B";AT k,ay;"DE";AT k+1,ay;" A
";AT k+2,ay;" B"
7270 BEEP .2,15-k/2
7280 NEXT k
7290 LET e=0: LET h=h-1: IF h=0
THEN GO TO 8550
7300 PRINT OVER 1;AT sx,sy-1;" F
GHI";AT sx,sy;" FGHI"
7310 PRINT INK 1;AT 1,ay;"  ";AT
 2,ay;"  "
7320 LET ax=0: LET ay=ay-1: IF a
y<0 THEN LET ay=31
7330 GO TO 5000
8100 FOR i=0 TO 2*a-2
8110 LET iy=INT (36*RND)-18
8120 IF y+iy<0 THEN LET iy=-y
8130 IF y+iy>22 THEN LET iy=22-y
8140 DRAW 4,iy
8150 LET y=y+iy
8160 NEXT i
8170 RETURN
8500 PRINT #1;AT 1,0; INK 5; FLA
SH 1;"Assim voce acaba com a nav
e"
8510 LET dm=dm+1: IF dm<5 THEN G
O TO 8700
8520 PRINT AT 21,14; FLASH 1; IN
K 7;" Irreparavel": FOR s=22944
TO 23104
8530 IF PEEK s=11 THEN BEEP .5,1
0: POKE s,9: POKE s+32,9
8540 NEXT s
8550 FOR k=1 TO 15
8560 PRINT OVER 1;AT sx,sy-1;" F
GHI": BEEP k/50,12-2*k: NEXT k
8570 PRINT AT 1,0; INK 7;"Sistem
a de suporte de vida huma na nao
 esta mais funcionando    Voce f
alhou na tentativa de de  fender
 seu planeta"
8580 PRINT INK 4;AT 10,0;"Seu pl
acar = ";sc
8600 INPUT "Quer tentar outra ve
z ? ";a$
8610 LET a$=a$+" ": IF a$(1)="s"
 THEN RUN 100
8650 STOP
8700 FOR k=1 TO 15
8710 PRINT OVER 1;AT sx,sy-1;" F
GHI": BEEP .1,15-2*k: NEXT k
8720 PRINT INK 1; FLASH 0;AT 21,
0;"                              "
8730 PRINT OVER 1;AT ax,ay-1;" D
E";AT bx,by-1;" J"
8740 LET ax=INT (15*RND): LET ay
=25
8750 LET bx=INT (15*RND): LET by
=30: IF bx=ax THEN GO TO 8750
8760 PRINT #1;AT 1,0;"
                              "
8770 GO TO 1000
8780>REM [illegible]
8790 REM ABCDEFGHIJ
```


Programa "Freeway"

Faça de conta que você é o pequeno homem que aparece na tela, tentando desesperadamente atravessar esta avenida muito movimentada. Não se esqueça que você pode tentar atravessar ou desistir, ou seja, voltar para a calçada.

Analise a rotina do programa, e veja que soluções inteligentes para a sensação de movimento da tela.

```
   1 REM ***      FREWAY      ***
   5 REM ***********************
  10 REM JOSE EDUARDO MALUF DE
CARVALHO      ARQUITRON
  15 REM ***********************
  20 GO SUB 9000: LET hi=0
  30 GO SUB 8000
  40 LET v=14: LET h=16: GO SUB
7000
  50 PRINT AT v,h; PAPER 8;" "
  60 LET v=v+(INKEY$="6" AND v<2
1)-(INKEY$="7" AND v>0)
  70 LET h=h+(INKEY$="8" AND h<3
1)-(INKEY$="5" AND h>0)
  80 PRINT AT 4,0; INK 6;b$;AT 1
1,0;a$;AT 5,0;a$;AT 12,0;b$
  90 IF SCREEN$ (v,h)="" THEN GO
 TO 1000
 100 PRINT AT v,h; PAPER 8;"E"
 110 LET b$=b$(32)+b$( TO 31)
 120 LET a$=a$(2 TO )+a$(1)
 130 IF v=2 THEN GO TO 2000
 150 GO TO 50
1000 PRINT AT v,h; FLASH 1;"E"
1010 DIM s$(1,704)
1020 PRINT AT 0,0; FLASH 8; OVER
 1; PAPER 8; INK 2;s$(1)
1025 IF ml>0 THEN LET ml=ml-1: G
O TO 40
1026 IF sc>hi THEN LET hi=sc
1030 INPUT "Pressione 's' para o
utro jogo ou 'n' p/ parar "; LIN
```

```
E b$
1040 IF b$="n" THEN STOP
1050 IF b$="s" THEN GO TO 30
2000 LET sc=sc+10
2010 GO TO 40
7000 CLS
7010 PRINT 'TAB 12;"MAIOR PLACAR
";hi;TAB 23;"PLACAR ";sc;' PAPER
 1;"  "
7020 PRINT INK 6'b$'a$
7030 PRINT ' PAPER 4;"
                      "
7050 PRINT INK 5'a$'b$
7060 PRINT ' PAPER 1;"  "
7070 PRINT AT v,h; PAPER 8;"E"
7080 FOR a=1 TO ml: PRINT AT 1,a
; INK 5;"E": NEXT a
7090 RETURN
8000 BORDER 6: PAPER 0: BRIGHT 1
: INK 9: CLS
8010 LET sc=0
8030 LET a$="AB AB  AB   AB    A
B     AB
8040 LET b$="CD     CD    CD   C
D  CD CD   CD"
8060 LET ml=6
8990 RETURN
9000 FOR a=USR "a" TO USR "e"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 0,1,2,127,235,253,25,5
9040 DATA 0,240,16,252,215,187,5
6,16
9050 DATA 0,15,8,63,235,221,26,6
9060 DATA 0,128,64,254,215,191,2
8,8
9070 DATA 28,28,8,8,62,8,28,34
9080 REM [illegible]
9090 REM ABCDE
```

Programa "Invasores"

Aqui está a versão, para o TK 90X, do famoso "arcade game" denominado "Os invasores", conhecido mundialmente por ser um dos primeiros jogos de ação para computadores.

Você vai notar nitidamente, aqui, a velocidade do processamento em Basic.

Digite o programa, e à medida que for destruindo os Invasores, note que a velocidade irá progressivamente aumentando.

Mais uma vez sugiro que tente alterar o jogo, modificando comandos como SOUND (nesta listagem = BEEP), ou comandos de cores relativas a papel e tinta, ou até as linhas DATA de criação dos caracteres gráficos (acompanhe com UDG 2).

A listagem:

```
   0 REM ***   INVASORES    ***
   1 REM ************************
   2 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   3 REM ************************
  99 REM INVASORES
 100 DATA 0,31,63,63,123,113,225
,225,0,248,252,252,222,142,135,1
35,255,127,111,111,199,192,240,2
40,255,254,246,246,227,3,15,15
 110 DATA 7,63,127,65,193,199,25
5,124,224,252,254,131,131,227,25
4,62,124,127,51,48,112,88,136,24
8,254,204,142,26,17,31,0,0
 120 DATA 31,63,63,115,113,225,2
25,255,248,252,252,206,142,135,1
35,255,127,111,103,103,96,48,60,
60,254,246,230,230,6,12,60,60
 130 DATA 7,63,127,193,193,199,1
27,124,224,252,254,130,131,227,2
55,62,127,51,113,88,136,248,0,0,
62,254,204,12,14,26,17,31
 140 DATA 24,24,24,24,60,126,231
,231,7,31,63,60,112,112,248,248,
224,248,252,60,14,14,31,31
 150 DATA 18,26,14,14,7,3,1,0,36
,44,56,56,112,96,192,128
 160 FOR i=USR "a" TO USR "a"+16
7: READ x: POKE i,x: NEXT i
 190 PRINT INK 9;AT 15,0;"Teclas
""5 - Para a esquerda""8 - Par
a a direita""0 - Para atirar":
PAUSE 300: CLS
 200 BRIGHT 1: BORDER 7: PAPER 7
: INK 9: CLS
 210 PRINT AT 0,0;"Placar";AT 0,
21;"Vidas ↑↑↑"
 220 LET f=2
 230 LET sc=0
 250 LET q=f: LET f=f+1: IF f=11
 THEN STOP
 300 LET c$="
                "
 310 LET s=144
 320 GO SUB 9000
 330 LET k$=a$+b$+c$
 340 GO SUB 9000
 350 LET i$=a$+b$+c$
 360 GO SUB 9000
```

```
 370 LET m$=a$+b$+c$
 380 GO SUB 9000
 390 LET n$=a$+b$+c$
 400 LET a$=k$+i$+k$+i$: LET a$=
a$( TO 343)
 410 LET b$=m$+n$+m$+n$: LET b$=
b$( TO 343)
 500 LET k$="  ▗▄▖  ": LET i$="
▟▀▀▀▙": LET m$="▜   ▛": LE
T n$=""
 510 PRINT INK 4;AT 16,2;k$;k$;k
$;k$;i$;i$;i$;i$;m$;m$;m$;m$
 590 LET i=0
 600 LET r=0: LET s=10: LET t=1
 610 LET u=0: LET v=0: LET x=0:
LET y=0
 615 LET d=21: LET e=12
 620 PRINT AT 20,30;c$
 625 LET li=5
1000 FOR p=r TO s STEP t
1010 BEEP .005,1: PRINT AT q,0;c
$( TO p);a$;AT 20,i;" 0
                    R▀5 "
1020 IF u<q THEN GO TO 1050
1035 IF SCREEN$ (u,v)<>" " THEN
GO SUB 5000
1040 IF u>=q THEN PRINT AT u,v;"
0";AT u+1,v;" "
1050 LET i=i+(i<28)*(INKEY$="8")
-(i>0)*(INKEY$="5")
1060 IF u>q-1 OR INKEY$<>"0" THE
N GO TO 1080
1070 LET u=20: LET v=i+1
1100 IF d>20 THEN GO TO 1200
1110 IF SCREEN$ (d,e)<>" " THEN
GO SUB 6000
1115 IF SCREEN$ (d,e+1)<>" " THE
N GO SUB 6000
1120 IF d<=20 THEN PRINT AT d-1,
e;" "; INK 2;AT d,e;"TU"
1200 LET u=u-1
2010 PRINT AT q,0;c$( TO p);b$;A
T 20,i;" 0
          R▀5 "
2020 IF u<q THEN GO TO 2050
2035 IF SCREEN$ (u,v)<>" " THEN
GO SUB 5000
2040 IF u>=q THEN PRINT AT u,v;"
0";AT u+1,v;" "
2050 LET i=i+(i<28)*(INKEY$="8")
-(i>0)*(INKEY$="5")
2060 IF u>q-1 OR INKEY$<>"0" THE
N GO TO 2080
2070 LET u=20: LET v=i+1
2100 IF d<=20 THEN GO TO 2130
2110 LET w=INT (6*RND): LET x$=a
$(LEN a$-4*w-2): IF x$<>"C" AND
```



```
x$<>"G" THEN GO TO 2130
2120 LET d=q+1+INT (LEN a$/32):
LET e=p-4*w+20+t
2125 IF e=31 THEN LET e=30
2130 IF d=20 THEN PRINT AT 20,e;
" "
2200 LET u=u-1
2210 LET d=d+1
2999 NEXT p
3000 IF t=-1 THEN GO TO 3500
3010 LET t=-1: LET r=9: LET s=0
3020 GO TO 1000
3500 IF LEN a$>55 THEN GO TO 351
0
3505 IF a$(23 TO )=c$ THEN GO TO
 250
3510 IF a$(LEN a$-32 TO )=c$ THE
N LET a$=a$( TO LEN a$-96): LET
b$=b$( TO LEN a$)
3520 IF 32*q+LEN a$>609 THEN GO
TO 8500
3525 LET r=0: LET s=10: LET t=1
3530 PRINT AT q,0;c$: LET q=q+1:
 GO TO 1000
5000 IF ATTR (u,v)=60 THEN GO TO
 5100
5005 IF ATTR (u,v)=58 THEN GO TO
 6000
5010 LET h=v-p+32*(u-q-1)
5020 IF h>=0 THEN GO SUB 5200
5030 LET h=h+32: IF h<LEN a$ THE
N GO SUB 5200
5040 LET h=h+32: IF h<LEN b$ THE
N GO SUB 5200
5050 LET h=4-INT (h/96)
5060 LET sc=sc+100*h: PRINT AT 0
,6;sc
5110 PRINT INK 6;AT u,v;"BA";AT
u+1,v;"SR": BEEP .05,10: PRINT A
T u,v;" ";AT u+1,v;" "
5120 LET u=0
5130 RETURN
5200 IF h=0 THEN LET h=1
5205 LET a$(h TO h+2)="   "
5210 LET b$(h TO h+2)="   "
5220 RETURN
6005 IF d=20 THEN GO TO 6018
6010 PRINT AT d-1,e;" "; INK 2;
AT d,e;"AB": BEEP .005,-10: PRIN
T AT d,e;" "
6012 IF (v=e OR v=e+1) AND d=u T
HEN PRINT AT u,v;" ";AT u+1,v;"
": LET u=0
6015 LET d=21: RETURN
6020 PRINT AT d-1,e;" "; INK 2;
AT d,e;"AB": BEEP .005,10: PRINT
 AT d,e;" "
6030 FOR k=1 TO 10: PRINT OVER 0
```

```
;AT 21,i;"0": BEEP .01,0: NEXT k
6040 LET li=li-1
6050 PRINT AT 0,27;"      ";AT 0,
27;"↑↑↑↑↑"( TO li)
6060 IF li=0 THEN GO TO 8510
6510 IF d<=20 THEN PRINT INK 6;A
T d,e;"BA";AT d+1,e;"SR": BEEP .
05,10: PRINT AT d,e;"  ";AT d+1,
e;"  "
6520 LET u=0: LET d=21
6530 RETURN
8500 PRINT AT q,0;c$;a$
8510 PRINT INK 9;AT 5,0;"Os inva
sores chegaram.": PRINT "Tudo es
ta perdido"
8520 INPUT "Quer tentar outra ve
z (s/n)? ";n$
8530 IF LEN n$=0 THEN GO TO 8520
8540 IF n$(1)="s" THEN RUN 200
8550 STOP
9000 LET a$=CHR$ (s)+CHR$ (s+1)+
"  ": LET a$=a$+a$+a$+a$+a$+a$+"
  "
9010 LET b$=CHR$ (s+2)+CHR$ (s+3
)+"  ": LET b$=b$+b$+b$+b$+b$+b$+b$
+"  "
9020 LET s=s+4
9030 RETURN
9040 REM [illegible]
9050 REM ABCDEFGHIJKLMNOPQRSTU
```

Programa "Invasores espaciais"

Neste jogo muito conhecido, você não deve, em hipótese alguma, deixar que nenhum invasor aterrisse.

Tome cuidado, pois nem sempre eles descem na vertical...

```
   0 REM *** INVASORES ESPACIAIS
   1 REM ************************
   2 REM JOSE EDUARDO MALUF DE
CARVALHO      ARQUITRON
   3 REM ************************
   5 GO SUB 520
  40 FOR n=0 TO 7: READ j: POKE
USR "a"+n,j: NEXT n
  50 FOR n=0 TO 7: READ k: POKE
USR "b"+n,k: NEXT n
  60 FOR n=0 TO 7: READ i: POKE
USR "c"+n,i: NEXT n
  70 LET a=10: LET s=0: LET q=1
```



```
  80 LET ss=0: LET z=31: LET y=1
20
  90 PRINT AT 12,4;"Quer instruc
oes?"
  95 BEEP .1,20: LET a$=INKEY$:
IF a$="s" THEN GO TO 380
  96 IF a$="n" THEN GO TO 100
  97 IF a$="" THEN GO TO 95
 100 BRIGHT 1: BORDER 1: PAPER 1
: INK 6: CLS
 110 PRINT AT 1,0; BRIGHT 1; INK
3;"
    ";AT 19,0; INK 6;"
"
 120 PRINT AT 0,2; BRIGHT 1; INK
5;"Placar ";AT 0,16; BRIGHT 1;
INK 5;"Recorde";ss
 150 LET c=INT (RND*30)+1: LET b
=3
 152 IF c<=5 THEN LET w=2000
 153 IF c>5 AND c<26 THEN LET w=
3000
 154 IF c>=26 THEN LET w=1000
 155 LET ch=INT (RND*2)+1
 160 PRINT AT 18,a; INK 6;" A ";
AT b-1,c-1;"   ";AT b,c; INK 4;C
HR$ (ch+144)
 170 LET x=(a*8)+12
 180 LET a=a+(INKEY$="8" AND a<=
28)-(INKEY$="5" AND a>=1)
 190 PLOT x,z: PLOT OVER 1;x,z
 200 IF INKEY$="0" THEN DRAW INK
5;0,y: BEEP .025,20: PLOT x,z:
DRAW OVER 1;0,y: GO TO 260
 205 GO SUB w
 210 LET b=b+1: BEEP .01,19-b: I
F b=19 THEN GO TO 230
 212 PRINT AT 18,a; INK 6;" A "
 214 LET a=a+(INKEY$="8" AND a<=
28)-(INKEY$="5" AND a>=1)
 220 GO TO 160
 230 BEEP .5,-20
 240 PRINT AT 18,c-1;"   ": PRIN
T AT 21,q*2; INK 7;CHR$ (ch+144)
: IF q=5 THEN GO TO 300
 250 LET q=q+1: GO TO 150
 260 IF x=(c*8)+4 THEN GO TO 280
 270 GO TO 205
 280 FOR n=-3 TO -1: PRINT AT b,
c; INK 1; INVERSE 1;CHR$ (ch+144
); BEEP .04,5-n: PRINT AT b,c; I
NK 2;CHR$ (ch+144): BEEP .04,n:
NEXT n
 290 LET s=s+(b+(ch*2)): PRINT A
T 0,8; BRIGHT 1; INK 5;s: PRINT
AT b,c;" ": GO TO 150
 300 PRINT AT 10,12; INK 7;"ACAB
OU": IF s>=ss THEN LET ss=s: PRI
```

```
NT AT 0,21; BRIGHT 1; INK 5;ss
 305 GO SUB 600
 310 PRINT AT 12,6; INK 6;"Press
ione s p/ jogar outra".
 320 IF INKEY$="s" THEN CLS : BE
EP .3,30: GO TO 70
 330 IF INKEY$="n" THEN CLS : ST
OP
 340 GO TO 320
 350 DATA BIN 00000000,BIN 00011
000,BIN 00011000,BIN 10011001,BI
N 10011001,BIN 11111111,BIN 1000
0001,BIN 10000001
 360 DATA BIN 00111100,BIN 01111
110,BIN 11011011,BIN 01111110,BI
N 01011010,BIN 10000001,BIN 0100
0010,BIN 00100100
 370 DATA BIN 00111100,BIN 00100
100,BIN 00111100,BIN 01111110,BI
N 11111111,BIN 00100100,BIN 0100
0010,BIN 10000001
 380 BORDER 2: PAPER 2: INK 7: C
LS
 390 PRINT "INVASORES  ": PRINT
AT 0,0; OVER 1;"_____"
 400 PRINT AT 2,0;"Os invasores
estao invadindo";AT 4,1;"seu pla
neta e sua unica esperan-ca e ex
termina-los antes que aterrissem
e invadam a super-ficie"
 410 PRINT AT 10,1; INVERSE 1; I
NK 7; FLASH 1;"CUidado..."
 420 PRINT AT 12,0;"Somente cinc
o deles podem aterrissar";AT 14,
1;"Nao permita  !!!"
 430 PRINT AT 21,1;"Pressione qu
alquer tecla "
 440 BEEP .1,16: BEEP .1,21: IF
INKEY$="" THEN GO TO 440
 450 CLS
 455 PRINT "INVASORES  ": PRINT
 AT 0,0; OVER 1;"_____"
 460 PRINT AT 2,0;"Mover base a
direita pressione"; INVERSE 1;"'
8'"
 470 PRINT AT 4,0;"Mover base a
esquerda pressione"; INVERSE 1;"
'5'"
 480 PRINT AT 6,0;"Pressione"; I
NVERSE 1;"'0'"; INVERSE 0;"P/ at
irar": PRINT AT 9,0;"Sao dois ti
pos de invasores, um valendo mai
s pontos que o outro"
 487 PRINT AT 15,5; INK 5;"B...D
roid": PRINT AT 17,5; INK 4;"C..
.Droid"
 490 PRINT AT 21,1;"Pressione EN
TER p/ comecar"
```

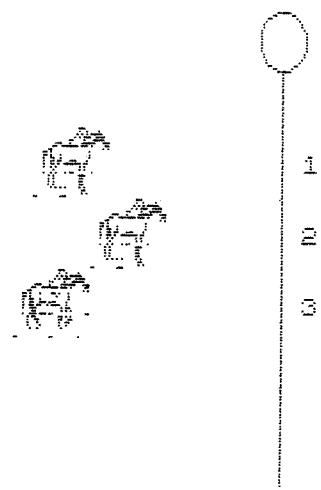


```
 500 BEEP .2,5: IF INKEY$<>"" TH
EN GO TO 100
 510 GO TO 500
 520 BORDER 2: PAPER 2: INK 7: C
LS
 530 PRINT AT 10,9;"INVASORES
"
 540 FOR n=-10 TO 40
 550 BEEP .1,40-n
 560 NEXT n: PAUSE 2: BEEP .5,20
: BEEP 1,15: RETURN
 600 FOR n=0 TO 2: BEEP .1,10: B
EEP .1,15: NEXT n
 610 FOR n=20 TO 0 STEP -2
 620 BEEP .1,n: NEXT n
 630 PAUSE 3: BEEP .25,15: BEEP
.25,15
 640 RETURN
1000 LET c=c-1: RETURN
2000 LET c=c+1: RETURN
3000 RETURN
4000 REM ###
5000 REM ABC
```

Programa "Jóquei Clube do TK 90X"

Outro belo jogo, envolvendo caracteres gráficos que simulam o movimento de cavalos correndo.

Após sua digitação, você deve apenas seguir as orientações e perguntas da tela, para que passe alguns momentos neste "Jóquei".

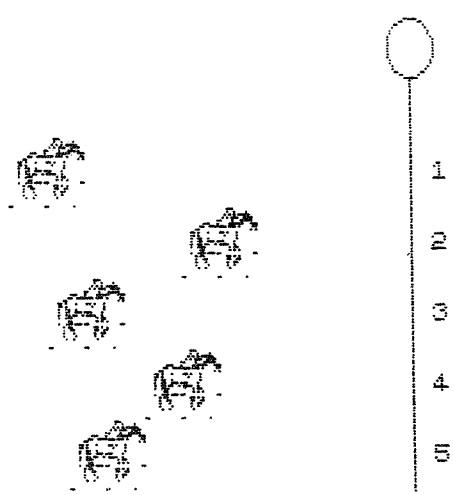


```
   0 REM JOQUEI CLUBE DO TK90X
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   4 REM ************************
  20 GO SUB 9000
  30 GO SUB 8000
  40 FOR r=1 TO 1+2+3+4/2
  50 CLS
  60 PRINT AT 0,0;"Numero do par
eo ";r
  70 DIM d(p)
  80 FOR a=1 TO p: LET d(a)=INT
(RND*10)+1
  90 PRINT '"A cotacao de  ";h$(
a)
 100 PRINT "Se vencer e de  ";d(
a);":1"
 110 NEXT a
 120 GO SUB 5000
 130 PRINT #1;AT 1,0;"Pressione
qualquer tecla"
 140 PAUSE 0
 150 CLS : CIRCLE INK 2;240,165,
10: PLOT 240,155: DRAW INK 2;0,-
140: FOR a=1 TO p: PRINT AT a*3+
3,31;a: NEXT a
 155 DIM c(p)
 160 FOR a=1 TO p
 170 PRINT AT a*3+2,c(a);a$;AT a
*3+3,c(a);b$;AT a*3+4,c(a);c$
 180 LET c(a)=c(a)+(1/d(a)+INT (
RND*2))
 185 IF c(a)>25 THEN GO TO 250
 190 NEXT a
 200 FOR a=1 TO p: BEEP .008,c(a
)
 210 PRINT AT a*3+2,c(a);a$;AT a
*3+3,c(a);d$;AT a*3+4,c(a);e$
 220 LET c(a)=c(a)+(1/d(a)+INT (
RND*2))
 225 IF c(a)>25 THEN GO TO 250
 230 NEXT a
 240 GO TO 160
 250 LET w$=h$(a)
 260 PAUSE 100: CLS
 270 PRINT "O vencedor e ";w$
 280 PRINT "Joquei  ";n$(a)
 290 LET ws=d(a)*s(a)
 300 PRINT '"Venceu com ";ws: LE
T m(a)=m(a)+ws+s(a)
 310 FOR z=1 TO 50: BEEP .008,z:
 BEEP .008,-z: NEXT z
 320 NEXT r
 330 CLS
 340 LET tot=0: PRINT "No final
do jogo  "
 350 FOR a=1 TO p
```



```
 360 PRINT n$(a);" tem ";m(a)
 370 IF m(a)>tot THEN LET tot=m(
a): LET win=a
 380 NEXT a
 390 PRINT '"O campeao e ";n$(wi
n)
 400 PRINT "com ";h$(win)
 410 PRINT "com Cr$";m(win)
 420 INPUT "Pressione 's' p/ out
ro jogo ou 'n' p/ parar"; LINE a
$
 430 IF a$="n" THEN STOP
 440 IF a$="s" THEN CLS : GO TO
30
 990 STOP
5000 FOR z=1 TO 50: BEEP .002*p,
z: BEEP .008,-z: NEXT z
5010 FOR a=1 TO p
5020 CLS
5030 PRINT "Cavalo:_ ";h$(a)
5040 PRINT "Joquei:_ ";n$(a)
5050 PRINT '"Voce tem Cr$ ";m(a)
5055 IF m(a)<1 THEN PRINT "Voce
nao tem dinheiro e portanto nao
pode apostar ": PAUSE 150: NEXT
a: RETURN
5060 INPUT "Quanto voce quer apo
star no pro-xximo pareo ";s(a): I
F s(a)>m(a) THEN GO TO 5060
5070 LET m(a)=m(a)-s(a)
5080 NEXT a
5090 RETURN
8000 BORDER 0: PAPER 0: INK 7: C
LS : BEEP .1,10: BEEP .2,15
8010 LET a$="   ABCD"
8020 LET b$="   EFGH"
8030 LET c$="   IJKL"
8040 LET d$="   EMNO"
8050 LET e$="   IP@ "
8060 PRINT AT 1,6;"Joquei Clube
do TK 90X"
8070 INPUT "Quantos jogadores (m
ax 5)"; LINE p$: IF p$<"1" OR p$
>"5" THEN GO TO 8070
8080 LET p=VAL p$: DIM s(p): DIM
 m(p): DIM n$(p,10): DIM h$(p,15
)
8090 FOR a=1 TO p
8100 PRINT AT 1,3;"Digite o seu
nome, jogador  #";a
8110 INPUT "Max 10 letras "; LIN
E n$(a)
8120 LET m(a)=50: NEXT a
8125 CLS
8130 RESTORE 9200: FOR a=1 TO p
8140 READ h$(a): PRINT n$(a);" a
gora dono "'h$(a)''
8150 NEXT a
```

```
8160 PRINT #1;AT 1,0;"Pressione
qualquer tecla": PAUSE 0
8990 RETURN
9000 FOR a=USR "a" TO USR "q"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 0,0,0,0,0,0,0,3
9040 DATA 0,0,0,0,01,113,207
9050 DATA 0,28,18,42,87,132,15,2
38
9060 DATA 0,0,16,248,60,124,199,
131
9070 DATA 5,13,13,9,9,9,9,0
9080 DATA 9,15,0,144,157,178,192
,192
9090 DATA 221,209,33,1,194,242,1
1,10
9100 DATA 128,0,0,0,0,0,0,128
9110 DATA 3,2,2,2,2,1,0,0
9120 DATA 192,192,96,32,16,32,0,
0
9130 DATA 1,26,25,19,18,8,0,0
9140 DATA 128,128,128,0,0,0,0,0
9150 DATA 16,0,16,144,177,167,64
,128
9160 DATA 69,1,1,1,194,246,20,20
9170 DATA 128,0,0,0,0,0,0,0
9180 DATA 128,128,128,128,128,64
,0,0
9190 DATA 28,28,24,24,24,20,12,1
0,0
9200 DATA "Off the way","Manquit
ola","Azarao","Georgino","Prince
sa"
9210 REM
9220 REM ABCDEFGHIJKLMNOPQ
```



Programa "Monstros Voadores"

Neste jogo, você é o único defensor da estação espacial. Eis que, de repente, surgem sobre você alguns monstros voadores, tentando acertá-lo com bombas de poder mortífero. Você também deve atirar seu raio laser mortal, para se defender. Não é tarefa difícil.

Este é mais um magnífico jogo em linguagem Basic, demonstrando a incrível capacidade deste micro.

Atenção para a digitação das seguintes linhas:

180 FOR i = USR "GRA A" TO USR "GRA A" + 167

290 PRINT "esp EXT SHI 2 GRA C GRA D esp esp"

330 PRINT "EXT SHI 4 GRA A EXT SHI 2 GRA E GRA F EXT SHI 4 GRA B esp"

510 PRINT AT v, g; "esp EXT SHI 2 GRA Q GRA R GRA S esp"; AT y, x; "esp esp esp esp"; AT y + 1,x; "esp esp esp esp"

540 PRINT AT y, x; "EST SHI 0 GRA G EXT SHI 1 GRA C GRA D EXT SHI 0 GRA H"; AT y + 1,x; "GRA I EXT SHI 1 GRA E GRA F EXT SHI 0 GRA J"

570. . . . . AT i - 1, x - 1; "GRA T GRA U": . . .

660 PRINT AT v,g; "esp EXT SHI 2 GRA Q GRA R GRA S esp"; AT y,x; "esp esp esp esp"; AT y + 1,x; "esp esp esp esp"

690 PRINT AT y,x; "EXT SHI 0 GRA K EXT SHI 1 GRA C GRA D EXT SHI 0 GRA L"; AT y - 1,x; "GRA M EXT SHI 1 GRA E GRA F EXT SHI 0 GRA N"

720. . . AT i + 1, x + 1; "GRA T GRA U": . . .

1010 PRINT AT y-1,x; "esp esp esp esp"; AT y,x; "EXT SHI 2 GRA 0 EXT SHI 6 GRA C GRA D EXT SHI 2 GRA P EXT SHI 0"; AT y + 1,x; "esp EXT SHI 6 GRA E GRA F EXT SHI 0 esp"

1300 PRINT AT v, g + 1; "GRA Q GRA R GRA S"; OVER 1; AT v, x + 1; "GRA T GRA U": FOR i = 1 TO 11: SOUND .1, -10:PRINT OVER 1; AT v, g; "esp GRA Q GRA R GRA S": SOUND .02,4: NEXT i

1420 PRINT AT v, g + 1; "EXT SHI 2 GRA Q GRA R GRA S"; OVER 1; AT y + 1, x, "EXT SHI 0 GRA G EXT SHI 1 GRA C GRA D EXT SHI 0 GRA

H"; AT y + 2, x; "EXT SHI 1 GRA I GRA E GRA F GRA J": FOR i = 1 TO 11: SOUND .1,10: PRINT OVER 1; AT v, g; "EXT SHI esp GRA Q GRA R GRA S": SOUND .02, 4: NEXT i

Os símbolos acima querem dizer o seguinte:

– esp – pular um espaço

– EXT – inserir em modo extended

– SHI – apertar também CAPS SHIFT

– GRA – inserir em modo gráfico

```
  0 REM MONSTROS VOADORES
  10 REM ************************
  20 REM JOSE EDUARDO MALUF DE
CARVALHO        ARQUITRON
  30 REM ************************
 100 DATA 14,31,28,24,16,16,16,1
6,112,248,56,24,8,8,8,8
 110 DATA 3,15,63,127,127,127,25
5,227,192,240,252,254,254,254,25
5,199
 120 DATA 227,227,99,99,127,63,7
,0,199,199,198,198,254,252,224,0
 130 DATA 240,112,56,248,124,62,
255,30,15,14,28,31,62,124,255,12
0,7,0,0,0,0,0,0,0,224,0,0,0,0,0,
0,0
 140 DATA 0,0,0,0,0,0,0,7,0,0,0,
0,0,0,224,30,255,62,124,248,56
,112,240,120,255,124,62,31,28,14
,15
 150 DATA 16,16,16,16,24,28,31,1
4,8,8,8,8,24,56,248,112
 160 DATA 0,0,12,15,31,63,127,25
5,24,60,60,255,255,255,231,129,0
,0,48,240,248,252,254,255
 170 DATA 66,66,38,27,15,7,3,1,6
6,66,100,216,240,224,192,128
 180 FOR i=USR "A" TO USR "A"+16
7
 190 READ x: POKE i,x
 200 NEXT i
 210 BORDER 1: PAPER 6: INK 0: B
RIGHT 1: OVER 0: CLS
 215 PRINT AT 15,0;"Teclas"'"5 -
 Para a esquerda"'"6 - Para a di
reita"'"0 - Para atirar": PAUSE
300: CLS
 220 LET v=21
 230 LET s=1
 240 LET g=5
```



```
 250 LET x=7
 260 LET y=0
 270 PRINT TAB x;
 280 FOR i=1 TO 5-s
 290 PRINT "  CD  ";
 300 NEXT i
 310 PRINT : PRINT TAB x;
 320 FOR i=1 TO 5-s
 330 PRINT "AEFB ";
 340 NEXT i
 500 LET g=g+(g<27)*(INKEY$="8")
-(g>1)*(INKEY$="5")
 510 PRINT AT v,g;" QRS ";AT y,x
;"     ";AT y+1,x;"     "
 520 LET x=x+(y>3)-(y<4)+2*SGN (
g-x)*(RND>.5): IF x>28 THEN LET
x=0
 530 LET y=y+1: IF y>20 THEN LET
 y=20
 540 PRINT AT y,x;"GCDH";AT y+1,
x;"IEFJ"
 550 IF y=v-2 THEN GO TO 1400
 560 IF RND>.15 THEN GO TO 600
 570 FOR i=y+2 TO v-1: BEEP .02,
v-i: PRINT AT i,x+1;"  ";AT i+1,
x+1;"TU": NEXT i
 580 PRINT AT i,x+1;"  "
 590 IF g=x OR g=x-1 THEN GO TO
1300
 600 IF INKEY$<>"0" THEN GO TO 6
50
 610 LET m=150: IF g=x OR g=x-1
THEN LET m=157-8*y
 620 PLOT 19+8*g,8: DRAW 0,m: BE
EP .1,0: IF m<150 THEN GO TO 100
0
 630 DRAW INVERSE 1;0,1: DRAW IN
VERSE 1;0,-m-1
 650 LET g=g+(g<27)*(INKEY$="8")
-(g>1)*(INKEY$="5")
 660 PRINT AT v,g;" QRS ";AT y,x
;"     ";AT y+1,x;"     "
 670 LET x=x+(y>3)-(y<4)+2*SGN (
g-x)*(RND>.5): IF x>28 THEN LET
x=0
 680 LET y=y+1: IF y>20 THEN LET
 y=20
 690 PRINT AT y,x;"KCDL";AT y+1,
x;"MEFN"
 700 IF y=v-2 THEN GO TO 1400
 710 IF RND>.15 THEN GO TO 750
 720 FOR i=y+2 TO v-1: BEEP .01,
v-i: PRINT AT i,x+1;"  ";AT i+1,
x+1;"TU": NEXT i
 730 PRINT AT i,x+1;"  "
 740 IF g=x OR g=x-1 THEN GO TO
1300
 750 IF INKEY$<>"0" THEN GO TO 5
00
```

```
 760 LET m=150: IF g=x OR g=x-1
THEN LET m=157-8*y
 770 PLOT 19+8*g,8: DRAW 0,m: BE
EP .1,0: IF m<150 THEN GO TO 100
0
 780 DRAW INVERSE 1;0,1: DRAW IN
VERSE 1;0,-m-1
 790 GO TO 500
1000 FOR y=y+1 TO v-2
1005 BEEP .02,y/2
1010 PRINT AT y-1,x;"    ";AT y,
x;"OCDP";AT y+1,x;" EF "
1015 NEXT y
1020 PRINT AT y-1,x;"    "
1200 LET s=s+1
1210 PRINT AT y,x;"    ";AT y+1,
x;"    ";AT 0,0;: IF s<5 THEN GO
 TO 250
1220 IF s=5 THEN GO TO 1500
1300 PRINT AT v,g+1;"0RS";: OVER
1;AT v,x+1;"TU": FOR i=1 TO 11:
BEEP .1,-10: PRINT OVER 1;AT v,g
;" 0RS": BEEP .02,4: NEXT i
1320 CLS
1330 RUN 220
1400 PRINT AT y,x;"    ";AT y+1,
x;"    ";AT 0,0;
1410 IF ABS (x-g)>1 THEN GO TO 2
50
1420 PRINT AT v,g+1;"0RS";: OVER
1;AT y+1,x;"GCDH";AT y+2,x;"IEFJ
": FOR i=1 TO 11: BEEP .1,10: PR
INT OVER 1;AT v,g;" 0RS": BEEP .
02,4: NEXT i
1440 RUN 100
1500>PRINT AT 5,0;"Parabens-voce
 destruiU os inimi-gos!
                    Pressione <5>
 para a esquerda              <8>
 para a direira      e        <0>
 para atirar"
1510 BEEP .3,1: BEEP .05,0: BEEP
 .05,-1: BEEP .5,9: BEEP .05,0:
BEEP .05,-1: BEEP .6,1: BEEP .5,
10
1520 PRINT FLASH 1; PAPER 2; INK
 7;AT 14,0;"CUIDADO -  AI VEM EL
ES OUTRA VEZ"
1530 FOR i=1 TO 3: BEEP 1,20: BE
EP .1,0: NEXT i
1540 CLS : GO TO 220
1550 REM
1560 REM ABCDEFGHIJKLMNOPQRSTU
```


Programa "Xadrez"

Este programa permite que você jogue xadrez com o computador. Tome cuidado, porque ele é um grande jogador, mas terrivelmente lento.

Os caracteres gráficos são muito bonitos, e, como alternativa, você pode alterar o programa, no sentido de que duas pessoas joguem xadrez. A escolha é sua.

Este programa é um exemplo excelente do que é possível fazer com o TK 90X, já que xadrez é considerado um jogo muito difícil para computadores, dada sua lógica. Ainda mais considerando-se a linguagem Basic, e suas limitações perante a linguagem de máquina (velocidade de processamento principalmente).

Uma única nota: seja honesto em suas jogadas, porque o programa não verifica se a sua jogada é ou não permitida. Arme-se de uma boa dose de paciência, e enfrente a "fera".

Os caracteres gráficos:

```
ABCDEFGHIJKLMNOPQRSTU
```

A tela:

```
  A B C D E F G H
1
2
3
4
5
6
7
8
                    Sua vez
                    DE    e 2
                    Para e
```

```
   1 PAPER 6: BORDER 6: CLS
   2 LET p=248: LET t=31: LET a=
128: LET g=13: LET h=160
```

```
    3 LET b=192: LET d=224: LET e
=240: LET f=15
    4 LET x=0: LET y=1: LET z=3:
_ET c=7: LET i=160
    5 LET f$="": DIM a$(1,1)
    6 LET kx=5: LET ky=8: LET q=0
   10 FOR m=USR "a" TO USR "t"+7
   20 READ n: POKE m,n: NEXT m
   30 DATA x,x,x,x,x,x,y,z
   40 DATA x,x,x,x,x,x,a,b
   50 DATA c,c,z,y,z,c,x,x
   60 DATA d,d,b,a,b,d,x,x
   70 DATA y,c,c,y,g,f,f,c
   80 DATA a,d,d,a,176,e,e,d
   90 DATA z,z,z,y,c,f,31,x
  100 DATA b,b,b,a,d,e,p,x
  110 DATA 4,5,2,y,z,c,c,c
  120 DATA 32,h,64,a,b,d,d,d
  130 DATA z,y,y,y,z,c,f,x
  140 DATA b,a,a,a,b,d,e,x
  150 DATA y,z,c,c,14,g,c,z
  160 DATA a,b,h,112,e,d,d,b
  170 DATA x,x,x,x,x,5,5,c
  180 DATA x,x,x,x,x,h,h,d
  190 DATA x,x,x,y,z,c,c,z
  200 DATA x,x,a,b,d,e,p,152
  210 DATA y,y,y,z,c,c,f,x
  220 DATA a,a,a,b,d,d,e,x
  230 DIM t$(7,2): DIM b$(7,2)
  240 FOR x=1 TO 7: READ t$(x),b$
(x): NEXT x
  250 DATA "AB","CD","QR","ST"
  260 DATA "MN","KL","OP","CD"
  270 DATA "IJ","KL","EF","GH","
","","
  280 LET i=2: LET p=4: PRINT "
A B C D E F G H"
  290 PRINT : FOR x=0 TO 7: FOR y
=0 TO 1: PRINT " ";
  300 FOR z=0 TO 3: PRINT PAPER p
;" "; PAPER i;" ";
  310 NEXT z: PRINT : NEXT y: LET
 t=i: LET i=p: LET p=t: NEXT x
  315 FOR y=1 TO 8: PRINT AT y+y,
0;y: NEXT y
  320 DIM b(8,8): FOR y=1 TO 8: F
OR x=1 TO 8
  330 LET z=(y=8)*-b(x,1)+(y=2)*-
1+(y=7)
  340 IF y=1 THEN READ z
  350 LET b(x,y)=z: NEXT x: NEXT
y
  360 DATA -4,-2,-3,-5,-6,-3,-2,-
4
  370 GO SUB 2000
  380 DIM d(8,8,2): DIM p(6)
  390 FOR x=1 TO 6: LET p(x)=2: F
OR y=1 TO 8
```



```
  400 LET d(x,y,2)=((y<3) OR (y=8
))+((y>3) AND (y<7))*-1
  410 LET d(x,y,1)=((y>1) AND (y<
5)+(y>5))*-1
  420 NEXT y: IF x<>2 THEN GO TO
460
  430 FOR y=1 TO 4: READ d(x,y,1)
,d(x,y,2): LET d(x,y+4,1)=-d(x,y
,1): LET d(x,y+4,2)=-d(x,y,2)
  440 NEXT y
  450 DATA -2,1,-1,2,1,2,2,1
  460 IF x=1 THEN LET p(x)=8
  470 IF x>4 THEN LET p(x)=1
  480 NEXT x
  490 DIM c$(5): RESTORE 500: FOR
 x=1 TO 5: READ c$(x): NEXT x
  500 DATA "p","n","b","r";"q"
  600 GO SUB 3000
  610 PRINT AT 8,22; FLASH 1;"Pen
sando";
  620 GO SUB 4000
  630 GO TO 600
  800 LET a$(1)=INKEY$: IF a$(1)=
"p" THEN GO TO 900
  805 IF (a$(1)="") OR (a$(1)<"a"
) OR (a$(1)>"h") THEN GO TO 800
  810 PRINT a$(1);" ";
  820 LET x1=CODE a$(1)-96
  830 LET a$(1)=INKEY$: IF (a$(1)
="") OR (a$(1)<"1") OR (a$(1)>"8
") THEN GO TO 830
  840 PRINT a$(1);
  850 LET y1=CODE a$(1)-48
  860 RETURN
  900 LET x1=999: LET y1=0: RETUR
N
1000 LET z1=INT ((ATTR (y+y,x+x)
)/8): IF z1>=8 THEN LET z1=z1-8
1010 PRINT BRIGHT 1; PAPER z1; I
NK i;AT y+y,x+x;t$(c);AT y+y+1,x
+x;b$(c);
1020 RETURN
2000 FOR y=1 TO 8: FOR x=1 TO 8
2010 LET z=b(x,y): IF z=0 THEN G
O TO 2040
2020 LET i=7: IF z>0 THEN LET i=
0
2030 LET c=ABS z: GO SUB 1000
2040 NEXT x: NEXT y
2050 RETURN
3000 IF f$="1" THEN PRINT AT 20,
22;"CHEQUEMATE": STOP
3010 PRINT AT 3,22;"Sua vez";: P
RINT AT 4,22;"DE   ";: GO SUB 80
0: IF x1=999 THEN STOP
3020 LET x=x1: LET y=y1
3030 PRINT AT 5,22;"Para ";: GO
SUB 800: LET xb=x1: LET yb=y1
```

```
3040 IF b(xb,yb)>0 THEN LET p(b(
xb,yb))=p(b(xb,yb))-1
3050 LET b(xb,yb)=b(x,y)
3060 LET b(x,y)=0: LET i=7
3070 IF b(xb,yb)<-1 OR yb<8 THEN
 GO TO 3120
3080 PRINT AT 6,21;"Peca ";
3090 LET a$(1)=INKEY$:
3100 LET a=0: FOR z=1 TO 5: IF c
$(z)=a$(1) THEN LET a=-z
3110 NEXT z: IF a=0 THEN GO TO 3
090
3115 PRINT a$(1);: LET b(xb,yb)=
a
3120 LET c=7: GO SUB 1000
3130 LET c=-b(xb,yb): LET x=xb:
LET y=yb: GO SUB 1000
3140 PRINT AT 3,22;"          ";A
T 4,22;"          ";AT 5,22;"
 ";AT 6,21;"          ";
3150 RETURN
4000 LET xb=0: LET yb=xb: LET db
=xb: LET c1=1: LET bp=xb: LET x=
9
4010 LET ax=kx: LET ay=ky: LET g
$="": GO SUB 5010: LET g$=f$
4020 FOR y=1 TO 8: FOR x=1 TO 8:
 LET tb=0: LET cz=1: LET ty=0: L
ET tx=0: LET d=0
4030 IF b(x,y)<1 THEN GO TO 4370
4040 LET p=b(x,y): FOR i=1 TO 8:
 LET dy=d(p,i,2): LET dx=d(p,i,1
): LET ax=kx: LET ay=ky: LET c1=
1: LET po=0
4050 IF x+dx<1 OR x+dx>8 OR y+dy
<1 OR y+dy>8 THEN GO TO 4300
4060 IF p>2 AND p<6 THEN GO TO 4
210
4070 IF dy>-1 AND p=1 THEN GO TO
 4300
4080 IF y>1 OR p>1 THEN GO TO 41
30
4090 LET p(5)=p(5)+1
4100 LET b(x,y)=5
4120 GO TO 4300
4130 IF dx<>0 AND b(x+dx,y+dy)>-
1 THEN GO TO 4300
4140 IF p=6 THEN LET kx=x: LET k
y=y: LET ax=x+dx: LET ay=y+dy: G
O SUB 5000: IF f$="1" THEN GO TO
 4300
4150 IF dx=0 AND b(x,y+dy)<>0 TH
EN GO TO 4300
4160 LET po=8-p-b(x+dx,y+dy)*3
4170 IF y<>7 OR dx<>0 OR b(x,5)<
>0 OR p>1 THEN GO TO 4300
4180 IF g$="1" THEN GO SUB 5000:
 IF f$="" THEN GO TO 4300
```



```
4190 IF g$="1" OR RND>.3 THEN LE
T c1=2
4200 GO TO 4300
4210 LET x1=x: LET y1=y
4220 IF p=3 AND INT (i/2)*2<i TH
EN GO TO 4300
4230 IF p=4 AND INT (i/2)*2=i TH
EN GO TO 4300
4240 IF x1+dx<1 OR x1+dx>8 OR y1
+dy<1 OR y1+dy>8 THEN LET c1=c1-
1: GO TO 4300
4250 IF b(x1+dx,y1+dy)>0 THEN LE
T c1=c1-1: GO TO 4300
4260 LET po=8-p+INT (RND*3)-b(x1
+dx,y1+dy)*3
4270 IF g$="1" THEN GO SUB 5000:
 IF f$="" THEN LET po=po+50: GO
TO 4340
4280 IF b(x1+dx,y1+dy)<>0 THEN G
O TO 4300
4290 LET c1=c1+1: LET x1=x1+dx:
LET y1=y1+dy: GO TO 4240
4300 IF po=0 THEN GO TO 4340
4310 IF g$="1" THEN GO SUB 5000:
 IF f$="" THEN LET po=po+50: GO
TO 4340
4320 IF g$="" THEN GO SUB 5000:
IF f$="1" THEN LET po=0: GO TO 4
340
4330 IF po>=tb THEN LET ax=x+dx*
c1: LET ay=y+dy*c1: GO SUB 5000:
 IF f$="1" THEN LET po=po-p*2
4340 IF po>tb THEN LET tb=po: LE
T tx=x: LET ty=y: LET d=i: LET c
z=c1
4350 NEXT i
4360 IF tb>bp OR (tb=bp AND (RND
>.9 OR (ty<yb AND RND>.5))) THEN
 LET bp=tb: LET xb=tx: LET yb=ty
: LET c1=cz: LET db=d
4370 NEXT x: NEXT y: LET p=b(xb,
yb): LET x1=xb+d(p,db,1)*c1: LET
 y1=yb+d(p,db,2)*c1: IF p=6 THEN
 LET kx=x1: LET ky=y1
4380 LET ax=kx: LET ay=ky: LET b
(x1,y1)=b(xb,yb): LET b(xb,yb)=0
: LET q=1: GO SUB 5010: LET q=0
4385 PRINT AT 8,22;"          ";AT
 9,22;"MInha vez ";AT 10,22;"De
 ";CHR$ (xb+95);"-";CHR$ (yb+48)
;
4386 PRINT AT 11,22;"Para ";CHR$
 (x1+95);"-";CHR$ (y1+48);
4390 LET i=0: LET c=7: LET x=xb:
 LET y=yb: GO SUB 1000
4400 LET c=b(x1,y1): LET x=x1: L
ET y=y1: GO SUB 1000
4410 RETURN
```

```
5000 LET pc=b(x+dx*c1,y+dy*c1):
LET b(x+dx*c1,y+dy*c1)=b(x,y): L
ET b(x,y)=0
5010 LET f$="": FOR r=1 TO 8: FO
R j=1 TO 8: LET d1=d(r,j,1): LET
 d2=d(r,j,2): LET l=1
5020 IF ax+d1*l<1 OR ax+d1*l>8 O
R ay+d2*l<1 OR ay+d2*l>8 THEN GO
 TO 5060
5030 IF r=1 AND d2<1 AND q=0 THE
N GO TO 5060
5040 IF b(ax+d1*l,ay+d2*l)=-r TH
EN LET f$="1": GO TO 5070
5050 IF r>2 AND r<6 AND b(ax+d1*
l,ay+d2*l)=0 THEN LET l=l+1: GO
TO 5020
5060 NEXT j: NEXT r
5070 IF x>8 THEN GO TO 5090
5080 LET b(x,y)=b(x+dx*c1,y+dy*c
1): LET b(x+dx*c1,y+dy*c1)=pc
5090 RETURN
```

3) UTILITÁRIOS

Programa "Agenda Telefônica"

Este programa, da maneira como está listado a seguir, serve para armazenar até 100 nomes, endereços, telefones, CEP, cidade e bairro, mas é o típico programa que eu costumo chamar de multiuso, pois basta modificar o nome das matrizes, para dar-lhe outra utilização (por exemplo, catálogo de discos, de livros, de filmes etc.).

Procure determinar qual o limite da memória de seu micro, para dimensionamento máximo das matrizes.

Você pode armazenar essas matrizes, através de SAVE "nome" DATA x$( ), vantagem que nem todo Basic possui.

Se você esqueceu o telefone e o endereço de um amigo, basta digitar no micro o nome dessa pessoa, que ele lhe mostrará todos os dados armazenados sobre aquela pessoa.

Experimente criar uma rotina, para classificar em ordem, por exemplo, alfabética, os nomes das pessoas (você não vai digitar necessariamente em ordem alfabética), fazendo-o comparar os itens, dois a dois, em ordem crescente e decrescente, e substituindo a ordem quando for necessário.


```
  0 REM * AGENDA TELEFONICA *
  2 REM ************************
  3 REM JOSE EDUARDO MALUF   DE
 CARVALHO      ARQUITRON
  4 REM ************************
 100 DIM n$(100,32)
 110 DIM a$(100,32)
 120 DIM b$(100,20)
 130 DIM c$(100,20)
 140 DIM d$(100,15)
 145 DIM e$(1,32)
 200 PRINT AT 0,0;"_____________
INDICE_____________"
 210 PRINT AT 5,0;"Arquivo de en
derecos"; FLASH 1;"Pressione <1
> para entrar dados"
 220 PRINT AT 8,0;"Listagem de n
omes"; FLASH 1;"Pressione <2> p
ara ver listagem"
 230 PRINT AT 11,0;"Listagem de
enderecos"; FLASH 1;"Pressione
<3> p/ ver enderecos"
 240 PRINT AT 14,0;"Gravacao";
FLASH 1;"Pressione <4> p/ gravar
 memoria"
 245 PAUSE 0
 250 IF INKEY$="1" THEN GO SUB 1
000
 260 IF INKEY$="2" THEN GO SUB 1
500
 270 IF INKEY$="3" THEN GO SUB 2
000
 280 IF INKEY$="4" THEN GO SUB 4
000
 300 GO TO 200
 998
 999 REM [illegible]
1000 CLS
1010 PRINT "_____Arquivo de end
erecos_____"
1020 PRINT AT 5,0;"Apos iniciado
, escolha um numerode 1 a 100 e
siga as instrucoes"
1030 PRINT AT 15,0;"Pressione <1
> para comecar";AT 17,0;"Pressio
ne <0> para  indice"
1035 PAUSE 0
1040 IF INKEY$="1" THEN CLS : GO
 TO 1060
1045 IF INKEY$="0" THEN CLS : RE
TURN
1050 GO TO 1040
1060 PRINT AT 0,0;"Digite o nume
ro do endereco"
1065 PRINT "(de 1 a 100)"
1070 INPUT x
1080 IF 1>x OR x>100 THEN CLS :
GO TO 1060
```

```
1090 POKE 23658,8
1095 CLS
1100 PRINT AT 10,0;"Entre com o
nome (max.32 caract)"
1110 INPUT n$(x)
1120 PRINT AT 10,0;"Entre com o
nome da rua e o num.""(max. 32
caracteres)"
1130 INPUT a$(x)
1135 CLS
1140 PRINT AT 10,0;"Entre com no
m da cidade e bairro"
1150 INPUT b$(x)
1155 CLS
1160 PRINT AT 10,0;"Entre com o
telefone
1170 INPUT c$(x)
1175 CLS
1180 PRINT AT 10,0;"Entre com co
digo postal
1200 INPUT d$(x)
1205 CLS
1210 PRINT AT 0,0;"Esta tudo cor
reto (s/n) ?"
1220 GO SUB 5000
1300 POKE 23658,0
1320 IF INKEY$="s" THEN GO TO 10
00
1330 IF INKEY$="n" THEN GO TO 10
90
1340 GO TO 1320
1500 CLS
1510 PRINT "_____Escolha de n
ome_____"
1515 PRINT AT 3,0; FLASH 1;"Pres
sione qualquer tecla p/ sair"
1520 PRINT AT 5,0;"Pressione <1>
 p/ num. de 1 a 20"
1521 PRINT AT 7,0;"Pressione <2>
 p/ num. 21 a 40"
1522 PRINT AT 9,0;"Pressione <3>
 p/ num. 41 a 60"
1523 PRINT AT 11,0;"Pressione <4
> p/ num. 61 a 80"
1524 PRINT AT 13,0;"Pressione <5
> p/ num. 81 a 100"
1530 PRINT AT 20,0;"Pressione <0
> p/  indice"
1535 PAUSE 0
1540 IF INKEY$="1" THEN LET f=1:
 LET g=20: GO TO 1800
1550 IF INKEY$="2" THEN LET f=21
: LET g=40: GO TO 1800
1560 IF INKEY$="3" THEN LET f=41
: LET g=60: GO TO 1800
1570 IF INKEY$="4" THEN LET f=61
: LET g=80: GO TO 1800
1580 IF INKEY$="5" THEN LET f=81
```



```
: LET g=100: GO TO 1800
1590 IF INKEY$="0" THEN CLS : RE
TURN
1600 GO TO 1540
1800 CLS : PRINT "_Numero_______
__Nome_____________
"
1810 FOR x=f TO g
1820 PRINT "Nome      ";x
1830 PRINT n$(x)
1840 NEXT x
1850 PAUSE 0
1860 IF INKEY$="z" THEN COPY
1870 GO TO 1500
2000 CLS : PRINT "_____Listagem
de enderecos_____"
2010 PRINT AT 5,0;"Siga instruco
es apos ter inicia-do o programa
"
2020 PRINT AT 17,0;"Pressione <1
> para comecar"
2030 PRINT AT 20,0;"Pressione <0
> p/ indice"
2040 PAUSE 0
2050 IF INKEY$="0" THEN CLS : RE
TURN
2060 IF INKEY$="1" THEN GO TO 20
75
2070 GO TO 2050
2075 CLS
2080 PRINT AT 10,0;"Digite o nom
e da pessoa cujo en-dereco voce
esta querendo"
2085 POKE 23658,8
2090 INPUT e$(1)
2100 FOR x=1 TO 100
2170 IF n$(x)=e$(1) THEN GO TO 2
200
2180 NEXT x
2190 CLS : PRINT "O nome ";e$(1)
;"nao consta do arquivo. Cheque
a sua listagem "
2195 POKE 23658,0: PAUSE 200: GO
 TO 2000
2200 POKE 23658,0: CLS : GO SUB
5000
2210 PRINT AT 20,0;"Pressione qu
alquer tecla "
2900 PAUSE 0
3000 GO TO 2000
4000 CLS
4010 PRINT "_____Rotina de grava
cao_____
"
4020 PRINT AT 5,0;"Quando voce c
arregar o programa da proxima ve
z, digite LOAD ""ender"". O prog
rama automaticamente entrara rod
ando, juntamente com os arquivos
 gravados"
```

```
4030 PRINT AT 17,0;"Pressione <0
> para  indice "
4040 PRINT AT 18,0;"Pressione <1
> p/ iniciar e siga instrucoes"
4045 PAUSE 0
4050 IF INKEY$="0" THEN CLS : RE
TURN
4060 IF INKEY$="1" THEN GO TO 40
80
4070 GO TO 4050
4080 CLS
4200 SAVE "ender" LINE 190
4300 STOP
5000 PRINT
5010 PRINT n$(x)
5020 PRINT
5030 PRINT a$(x)
5040 PRINT
5050 PRINT b$(x)
5060 PRINT
5070 PRINT c$(x)
5080 PRINT
5090 PRINT d$(x)
5100 RETURN
```

Programa "Ampliação"

Programa auto-explicativo, onde se deve digitar alguma palavra com no máximo 16 caracteres, para que o micro possa ampliá-la até sete vezes.

```
   0>REM AMPLIACAO DE CARACTERES
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   4 REM ************************
  50 PAPER 0: BORDER 0: INK 7: C
LS
  60 PRINT "Este programa amplia
 uma mensa -gem de 2 a 7 vezes.
            O numero maximo de c
aracteres namensagem e:
                2 vezes          -16
 caracteres 3 vezes              -10
 caracteres 4 vezes              - 8
 caracteres 5 vezes              - 6
 caracteres 6 vezes              - 5
 caracteres 7 vezes              - 4
 caracteres"
  70 INPUT "Digite o numero de c
aracteres";nch
  80 PRINT AT 11,2;nch
  90 INPUT "Digite a mensagem
```



```
              ";a$
 100 PRINT AT 13,2;a$
 110 INPUT "Digite o numero para
 ampliacao  ";n: PRINT AT 15,2;n
 120 PRINT #1;AT 0,0;"Pressione
qualquer tecla          ": PAUSE,0
: BORDER 1: INK 9: CLS : PRINT A
T 0,0;a$
 130 FOR x=0 TO nch*8-1
 140 FOR y=175 TO 168 STEP -1
 150 LET z=175-y
 160 IF POINT (x,y)=1 THEN GO SU
B 300
 170 NEXT y
 180 NEXT x
 190 INPUT "Outra mensagem (s/n)
-<Z> copia  ";b$
 200 IF b$="s" THEN RUN
 210 IF b$="z" THEN COPY : GO TO
 190
 220 NEW
 300 FOR c=0 TO n-1
 310 FOR d=0 TO n-1
 320 PLOT INK 7;n*x+c,175-(80+d+
z*n)
 330 NEXT d
 340 NEXT c
 350 RETURN
```

Programa "Análise de Vendas"

Este programa utiliza o histórico de vendas de uma empresa por um determinado número de anos, para mostrar as flutuações ocorridas, bem como as tendências detectadas no período.

Os resultados deste programa podem ser muito úteis para qualquer empresa de vendas cujas atividades sejam afetadas por períodos anuais (as chamadas estações do ano e do programa).

As informações necessárias para que o programa rode são os valores das vendas por estação anual, de cada ano do período determinado pelo usuário (o programa aceita um período máximo de 18 anos).

Após a entrada das informações, o programa calculará a proporção de cada estação, pela divisão das vendas atuais pela média obtida entre as estações de todos os anos medidos.

O programa também traça um gráfico contínuo das vendas anuais da empresa, permitindo uma imediata visualização das informações sobre as vendas naquele período.

A listagem:

```
    1 REM * ANALISE DE VENDAS *
    2 REM ************************
    3 REM JOSE EDUARDO MALUF DE
CARVALHO        ARQUITRON
    4 REM ************************
  110 REM indice por periodo
  120 BORDER 7: PAPER 7: INK 1: F
LASH 0: OVER 0: CLS
  130 GO TO 8000
1000 REM
1010 REM entrada de dados
1020 PRINT AT 0,9; INVERSE 1;"In
dice por periodo"
1030 PRINT AT 4,0;"Historico de
vendas"
1040 PRINT AT 7,6;"Do ano "
1050 PRINT AT 10,6;"Ao ano "
1060 PRINT AT 7,17; FLASH 1; INV
ERSE 1;" > "; FLASH 0; INK 2; IN
VERSE 1;"  "
1070 INPUT s$: LET y1=INT (VAL s
$): IF y1<1900 OR LEN (STR$ y1)>
4 THEN GO TO 1070
1080 PRINT AT 7,17;" ";y1
1090 PRINT AT 10,17; FLASH 1; IN
VERSE 1;" > "; FLASH 0; INK 2; I
NVERSE 1;"  "
1100 INPUT s$: LET y2=INT (VAL s
$): IF y2<1900 OR LEN (STR$ y2)>
4 OR y2<y1 OR y2>y1+17 THEN GO T
O 1100
1110 PRINT AT 10,17;" ";y2
1120 LET ny=y2-y1+1
1130 DIM y$(ny,4)
1140 DIM h$(ny,4,4)
1150 DIM a$(ny,7)
1160 DIM r$(ny,4,6)
1170 DIM d$(4,6)
1172 DIM q(4)
1174 DIM c(ny)
1180 DIM c(ny)
1190 DIM c(ny)
1200 FOR i=y1 TO y2: LET y$(i-y1
+1)=STR$ i: NEXT i
2000 REM
2010 REM entrada de vendas
2020 CLS : PRINT AT 0,10; INVERS
E 1;"Historico de vendas"
2030 PRINT AT 1,26; INVERSE 1;"A
VR.";AT 2,0; INVERSE 1;"ANO ";AT
 2,5; INVERSE 1;"EST1";AT 2,10;
INVERSE 1;"EST2";AT 2,15; INVERS
E 1;"EST3";AT 2,20; INVERSE 1;"E
ST4";AT 2,26; INVERSE 1;"VENDAS"
2040 IF hm=1 THEN GO TO 2500
2050 FOR i=1 TO ny
2060 LET in=i+3-1: PRINT AT in,0
;y$(i)
```



```
2070 LET hp=4: LET ts=0
2080 FOR j=1 TO 4
2090 PRINT AT in,hp; FLASH 1;" >
 "; INVERSE 1; FLASH 0; INK 2;"
 "
2100 INPUT s$: IF VAL s$<0 THEN
GO TO 2100
2110 LET n1=INT (VAL s$+.5): LET
 ts=ts+n1: LET s$=STR$ n1
2120 LET z$="     ": LET z$(5-LEN
 s$ TO )=s$
2130 PRINT AT in,hp;" ";z$
2140 LET h$(i,j)=z$: LET hp=hp+5
2150 NEXT j
2160 LET s$=STR$ (ts/4): LET ns=
4: LET nd=2
2170 GO SUB 7000
2180 PRINT AT in,hp+1;s$
2190 LET a$(i)=s$
2200 NEXT i
2210 LET hm=1: GO TO 2550
2500 REM
2510 REM display
2520 FOR i=1 TO ny
2530 PRINT AT i+3-1,0;y$(i);" ";
h$(i,1);" ";h$(i,2);" ";h$(i,3);
" ";h$(i,4);" ";a$(i)
2540 NEXT i
2550 INPUT "Copio na impressora
(s/n)";k$
2560 IF k$="s" THEN COPY : GO TO
 2550
2570 IF k$="n" OR k$="n" THEN RE
TURN
2580 IF k$="e" THEN CLEAR : STOP

2590 GO TO 2550
3000 REM
3010 REM calculos
3020 CLS : PRINT AT 0,9; INVERSE
 1;"Proporcoes  "
3030 PRINT AT 2,0; INVERSE 1;"An
o ";AT 2,6; INVERSE 1;"EST1";AT
2,13; INVERSE 1;"EST2";AT 2,20;
INVERSE 1;"EST3";AT 2,27; INVERS
E 1;"EST4"
3040 PRINT AT 21,0; INVERSE 1;"I
NDICE"
3050 IF cm=1 THEN GO TO 3500
3060 PRINT AT 1,11; FLASH 1; INK
 3;"Calculando"
3070 FOR i=1 TO 4: LET q(i)=0: N
EXT i
3080 FOR i=1 TO ny
3090 LET n1=VAL a$(i)
3100 FOR j=1 TO 4
3110 LET n2=VAL h$(i,j): LET n3=
n2/n1
```

```
3120 LET s$=STR$ n3: LET ns=1: L
ET nd=4
3130 GO SUB 7000
3140 LET r$(i,j)=s$: LET q(j)=q(
j)+VAL s$
3150 BEEP .03,10: REM coloque SO
UND AQUI
3160 NEXT j
3170 NEXT i
3180 FOR i=1 TO 4
3190 LET n1=q(i)/ny
3200 LET s$=STR$ n1: LET ns=1: L
ET nd=4
3210 GO SUB 7000
3220 LET d$(i)=s$
3230 BEEP .05,30: REM coloque SO
UND aqui
3240 NEXT i
3250 PRINT AT 1,11;" "; INVERSE
1; INK 3;"Terminado"; INVERSE 0;
 INK 1;" "
3260 LET cm=1
3270 NEXT i
3280 PRINT AT 1,11;" "; INVERSE
1; INK 3;"Terminado"; INVERSE 0;
 INK 1;" "
3290 LET cm=1
3300 REM
3310 REM display
3320 FOR i=1 TO ny
3330 PRINT AT i+3-1,0;y$(i);" ";
r$(i,1);" ";r$(i,2);" ";r$(i,3);
" ";r$(i,4)
3340 NEXT i
3350 PRINT AT 21,5;d$(1);" ";d$(
2);" ";d$(3);" ";d$(4)
3370 INPUT "Copio na impressora
(s/n)?";k$
3380 IF k$="s" THEN COPY : GO TO
 3370
3390 IF k$="n" THEN RETURN
3400 IF k$="e" THEN CLEAR : STOP

3410 GO TO 3370
4000 REM
4010 REM plotagem
4020 CLS : PRINT AT 0,10; INVERS
E 1;"Grafico de vendas"
4030 PRINT AT 2,0; INVERSE 1;"An
o ";AT 1,26; INVERSE 1;"AVE";AT
2,26; INVERSE 1;"Vendas"
4040 IF gm=1 THEN GO TO 4500
4050 LET gm=1
4060 PRINT AT 2,12; INVERSE 1; I
NK 3; FLASH 1;"Plotando"
4080 LET max=INT (VAL a$(1)): LE
T min=max
4090 FOR i=2 TO ny
```



```
4100 LET n1=INT (VAL a$(i))
4110 IF n1>max THEN LET max=n1
4120 IF n1<min THEN LET min=n1
4130 BEEP .05,0
4140 NEXT i
4150 LET ran=max-min
4160 LET rat=INT (ran/152+.5)
4170 LET ox=40
4180 FOR i=1 TO ny
4190 LET n1=INT (VAL a$(i)): LET
 nx=INT ((n1-min)/rat+.5)+40
4200 LET c(i)=nx-ox: LET ox=nx
4210 BEEP .05,30
4220 NEXT i
4230 PRINT AT 2,12;"          "
4500 REM
4510 REM redisplay curvas
4520 FOR i=1 TO ny
4530 LET in=i+3-1: PRINT AT in,0
;y$(i);AT in,25;a$(i)
4540 NEXT i
4550 PLOT 40,155: DRAW 151,0: PL
OT 40,155: DRAW 0,-155
4560 PLOT 40+c(1),147
4570 FOR i=2 TO ny
4580 DRAW c(i),-8
4590 NEXT i
4600 INPUT "Copio na impressora
(s/n)?";k$
4610 IF k$="s" THEN COPY : GO TO
 4600
4620 IF k$="n" THEN RETURN
4630 IF k$="e" THEN STOP
4640 GO TO 4600
7000 REM
7010 REM
7020 REM
7030 LET x$="          "
7040 LET w$="1000000"
7050 LET t$=STR$ (INT VAL s$)
7060 IF ns>0 THEN LET t$=x$(1 TO
 ns-LEN t$)+t$+".": GO TO 7080
7070 LET t$=".0"
7080 LET n3=VAL w$(1 TO nd+1)
7090 LET u$=STR$ (INT ((VAL s$-V
AL t$)*n3+.5))
7100 IF VAL u$<VAL w$(1 TO nd) T
HEN LET u$=w$(2 TO nd-LEN u$+1)+
u$
7110 IF ns=0 THEN LET t$="."
7120 LET s$=t$+u$
7130 RETURN
8000 REM
8010 REM
8020 LET hm=0: LET cm=hm: LET gm
=hm
8030 GO SUB 1000
8040 GO SUB 2000
```

```
8050 GO SUB 3000
8060 GO SUB 4000
8070 GO TO 8040
```

Programa "Avaliação de Patrimônio"

Programa financeiro que auxilia as pessoas na itemização e avaliação de seu patrimônio pessoal. Esta avaliação é útil em casos, por exemplo, de cobertura de apólices de seguros. Pode ser utilizada também para despesas de capital, em pequenas empresas.

Cada item do patrimônio é avaliado pelo seu custo original, valor de revenda e custo atualizado.

O valor de revenda é calculado com base na depreciação conforme tempo de vida, e o custo é atualizado de acordo com a inflação especificada.

O número máximo de itens é 100.

Note-se que todos os valores manipulados pelo programa são arredondados, e serão sempre números inteiros.

A listagem:

```
   1 REM  ANALISE DE PATRIMONIO
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   4 REM ************************
 120 GO TO 9000
1040 CLS : PRINT AT 1,6;"Avaliac
ao de patrimonio": PRINT AT 2,6;
"----------------------"
1050 PRINT AT 4,3;"Nome : "
1060 PRINT AT 6,3;"Ano corrente
: "
1070 PRINT AT 8,3;"Inflacao lota
l :     %"
1080 PRINT AT 10,3;"Num. de iten
s : "
1090 PRINT AT 12,1;"Total": PRIN
T AT 13,1;"-----"
1100 PRINT AT 14,3;"Valor altera
do "
1110 PRINT AT 15,3;"Valor origin
al "
1120 PRINT AT 16,23;"---------"
```



```
1130 PRINT AT 17,3;"Diferenca ";
AT 17,22;" "
1150 PRINT AT 19,2;"Dif Perc.";A
T 19,14;"   .   %"
1160 PRINT AT 21,3;"Total de val
ores "
1170 RETURN
5000 REM
5010 REM
5020 REM
5030 FOR i=1 TO ni
5040 CLS : PRINT AT 1,8;"Descric
ao do item": PRINT AT 2,8;"-----
-----------"
5050 PRINT AT 4,1;"Descricao :
"
5060 PRINT AT 6,1;"Situacao :"
5070 PRINT AT 8,1;"Data aquisica
o ";AT 8,22;"Vida : "
5080 PRINT AT 10,1;"Custo origin
al :"
5090 PRINT AT 13,1;"Corrente": P
RINT AT 14,1;"---------"
5100 PRINT AT 16,3;"Val revenda
";AT 16,21;" "
5110 PRINT AT 18,3;"Custo altera
do "
5120 PRINT AT 4,14; INVERSE 1;"
"
5130 PRINT AT 6,11; INVERSE 1;"
"
5140 PRINT AT 8,16; INVERSE 1;"
"
5150 PRINT AT 8,27; INVERSE 1;"
"
5160 PRINT AT 10,16; INVERSE 1;"
"
5170 REM
5180 PRINT AT 4,0; FLASH 1;">"
5190 INPUT d$(i): IF d$(i,1)="*"
 THEN GO TO 5410
5200 PRINT AT 4,14; INVERSE 1;d$
(i)
5210 PRINT AT 4,0;" ": PRINT AT
6,0; FLASH 1;">"
5220 INPUT l$(i): IF l$(i)="*" T
HEN GO TO 5410
5230 PRINT AT 6,11; INVERSE 1;l$
(i)
5240 PRINT AT 6,0;" ": PRINT AT
8,0; FLASH 1;">"
5250 INPUT t$(i): IF VAL t$(i,1
TO 4)>VAL c$(1 TO 4) THEN GO TO
5250
5260 PRINT AT 8,16; INVERSE 1;t$
(i)
5270 PRINT AT 8,0;" ": PRINT AT
8,21; FLASH 1;">"
```

```
5280 INPUT l: IF l<0 THEN GO TO
5280
5290 LET l=INT l: PRINT AT 8,27;
 INVERSE 1;l
5300 PRINT AT 8,21;" ": PRINT AT
 10,0; FLASH 1;">"
5310 INPUT o: IF o<0 THEN GO TO
5310
5320 LET o=INT o: PRINT AT 10,16
; INVERSE 1;o
5330 PRINT AT 10,0;" "
5340 REM
5350 LET yu=VAL c$-VAL t$(i): LE
T yu=(yu>=l)*l: LET rv=INT ((l-y
u)*o/l)
5360 LET tc=tc+rv: PRINT AT 16,2
3+7-LEN (STR$ rv);rv
5370 LET rc=INT (o*(1+li/100)↑yu
): LET tr=tr+rc: PRINT AT 18,23+
7-LEN (STR$ rc);rc
5380 LET to=to+o
5390 IF INKEY$="" THEN GO TO 539
0
5395 IF INKEY$="z" THEN COPY
5400 NEXT i
5410 RETURN
6000 REM
6010 REM
6020 PRINT AT 4,9; INVERSE 1;n$
6030 PRINT AT 6,17; INVERSE 1;c$
6040 PRINT AT 8,20; INVERSE 1;IN
T (li*100)/100
6050 PRINT AT 10,20; INVERSE 1;n
i
6060 PRINT AT 14,24+8-LEN (STR$
tr);tr
6070 PRINT AT 15,24+8-LEN (STR$
to);to
6080 LET di=ABS (tr-to): PRINT A
T 17,24+8-LEN (STR$ di);di
6090 IF tr<to THEN PRINT AT 17,2
1;"-": PRINT AT 19,12;"-"
6100 PRINT AT 19,14;INT ((di/to)
*10000)/100;"  %"
6110 PRINT AT 21,24+8-LEN (STR$
tc);tc
6120 IF INKEY$<>"*" THEN GO TO 6
120
6125 IF INKEY$="z" THEN COPY
6130 RETURN
8000 REM
8010 REM
8020 DIM c$(4): DIM n$(20): LET
n$=" ": LET cy=0: LET li=0: LET
ni=0: LET tr=0: LET to=0: LET tc
=0
8030 GO SUB 1000
8040 PRINT AT 4,9; INVERSE 1;"
```



```
8050 PRINT AT 6,17; INVERSE 1;"
 "
8060 PRINT AT 8,20; INVERSE 1;"
 ";AT 8,23;" "
8070 PRINT AT 10,20; INVERSE 1;"
 "
8080 REM
8090 PRINT AT 4,2; FLASH 1;">"
8100 INPUT n$: IF n$=" " THEN GO
 TO 8100
8110 PRINT AT 4,9; INVERSE 1;n$(
1 TO 20)
8120 PRINT AT 4,2;" ": PRINT AT
6,2; FLASH 1;">"
8130 INPUT c$: IF VAL c$<1980 TH
EN GO TO 8130
8140 PRINT AT 6,17; INVERSE 1;c$
8150 PRINT AT 6,2;" "
8160 PRINT AT 8,2; FLASH 1;">"
8180 INPUT li: IF li>300 OR li<-
10 THEN GO TO 8180
8190 PRINT AT 8,20;""+STR$ (INT
((li*100)/100)+""
8210 PRINT AT 8,2;" ": PRINT AT
10,2; FLASH 1;">"
8220 INPUT ni: PRINT AT 10,20;""
+STR$ ni+""
8230 PRINT AT 10,2;" "
8240 PRINT AT 12,9; FLASH 1; INK
 3;"Inicializando"
8250 DIM d$(ni,18)
8260 DIM l$(ni,10)
8270 DIM t$(ni,4)
8280 DIM l(ni)
8290 DIM o(ni)
8300 FOR i=1 TO ni
8310 LET d$(i)=" ": LET l$(i)="
": LET t$(i)=" "
8320 LET l(i)=0: LET o(i)=0
8330 NEXT i: PAUSE 300
8340 PRINT AT 12,9; INK 3; FLASH
 1;"Terminado     ": PAUSE 500
8350 LET tr=0: LET to=tr: LET tc
=tr
8360 RETURN
9000 REM
9010 REM
9020 GO SUB 8000
9030 GO SUB 5000
9040 REM
9050 GO SUB 1000
9060 GO SUB 6000
9070 PRINT AT 12,9; FLASH 1; INK
 1;"Programa terminado "
9080 STOP
```

Programa "Calendário Permanente"

Programa muito útil, que lhe dá o calendário do mês bem como a diferença entre duas datas e que dia da semana cairá a data que você digitar.

O algoritmo do programa é um pouco complicado; fica aqui, como "lição de casa", o entendimento dele. (Não é tão complicado – simplesmente está baseado numa única fórmula de conversão de datas – encontre-a no programa.)

Mas o programa está de propósito "sujo", ou seja existem diversas linhas inúteis. É um bom exercício a tentativa de "limpá-lo".

Existe uma matriz que não serve para nada. . .

A sub-rotina que começa na linha 5000 é a que explica o funcionamento do programa.

A sub-rotina da linha 3700 é a de definição de matrizes. . .

As sub-rotinas das linhas 1000 e 1500 são iguais e armazenam datas, mas executam tarefas diferentes.

As sub-rotinas das linhas 2000 e 3000 fazem os cálculos das datas propriamente ditos, para serem mostrados na tela.

```
    0 REM CALENDARIO PERMANENTE
    1 REM JOSE EDUARDO MALUF DE
CARVALHO          ARQUITRON
    2 DIM a(50)
    3 DIM d$(7,7)
    4 DIM m$(12,9)
    5 DIM g(24)
    6 DIM l(31)
    7 CLS : BORDER 1
    8 PAPER 1
    9 INK 7
   16 GO SUB 5000
   17 GO SUB 3700
   18 BORDER 0: PAPER 0: INK 6: B
RIGHT 1: CLS
   20 INPUT "Modo:"'"1 - Calendar
io"'"2 - Intervalo entre dois di
as"'"3 - Dia da semana";v: IF v<
1 OR v>3 THEN GO TO 20
   30 IF v=1 THEN GO TO 400
   31 IF v=2 THEN GO TO 45
   32 IF v=3 THEN GO TO 200
   33 IF v=4 THEN GO TO 3600
   45 PRINT '"Qual a primeira da
ta?"
   50 GO SUB 1000
   60 GO SUB 2000
```



```
   70 LET a=f
   80 PRINT '"Qual a segunda dat
a?"
   90 GO SUB 1000
  100 GO SUB 2000
  110 LET a=f-a
  120 PRINT '"A diferença e ";a;
" dias": PRINT "
": PAUSE 300: CLS : GO
TO 20
  200 GO SUB 1000
  202 GO SUB 2000
  204 GO SUB 3000
  216 LET d$(1)="Domingo"
  217 LET d$(2)="Segunda"
  218 LET d$(3)="Terca"
  219 LET d$(4)="Quarta"
  220 LET d$(5)="Quinta"
  221 LET d$(6)="Sexta"
  222 LET d$(7)="Sabado"
  230 PRINT AT 18,1;"O dia ";d;"
do mes ";m$(m);" do ano ";y;" e
";d$(c): PRINT "
": PAUSE 300: CL
S : GO TO 20
  400 GO SUB 1500
  410 LET d=1
  420 GO SUB 2000
  430 GO SUB 3000
  489 LET l=0
  490 LET l=a(m)
  491 LET a(m)=l(m)
  500 IF m<>2 THEN GO TO 550
  510 IF INT (y/4)<>y/4 AND m=2 T
HEN LET l=28 AND a=28
  512 IF INT (y/4)<>y/4 THEN GO T
O 550
  520 IF INT (y/400)=y/400 THEN G
O TO 540
  530 IF INT (y/100)=y/100 THEN L
ET l=28: GO TO 550
  540 IF INT (y/100)=y/100 THEN L
ET l=29
  550 FOR x=1 TO c-1: LET a(x)=0:
 NEXT x
  560 LET s=0
  565 IF m=2 AND INT (y/4)<>y/4 T
HEN LET l=28: IF m=2 AND INT (y/
4)=y/4 THEN LET l=29
  566 IF m=1 OR m=3 OR m=5 OR m=7
 OR m=8 OR m=10 OR m=12 THEN LET
 l=31
  567 IF m=4 OR m=6 OR m=9 OR m=1
1 THEN LET l=30
  568 IF m=2 AND INT (y/4)=y/4 TH
EN LET l=29
  570 FOR x=c TO c+l-1: LET s=s+1
  LET a(x)=s: NEXT x
```

```
 580 FOR x=c+1 TO 42: LET a(x)=0
: NEXT x
 582 IF m=1 OR m=3 OR m=5 OR m=7
 OR m=8 OR m=10 OR m=12 THEN LET
 l=31
 584 IF m=4 OR m=6 OR m=9 OR m=1
1 THEN LET l=30
 590 CLS
 600 PRINT AT 3,11;" Mes - ";m$(
m)
 610 PRINT AT 4,11;" Ano - ";y
 740 PRINT
 750 PRINT '''  DOM SEG TER QUA
QUI SEX SAB"
 760 FOR j=0 TO 35 STEP 7
 770 PRINT
 775 LET a=1
 780 PRINT TAB 3;a(j+1);TAB 7;a(
j+2);TAB 11;a(j+3);TAB 15;a(j+4)
;TAB 19;a(j+5);TAB 23;a(j+6);TAB
 27;a(j+7)
 800 NEXT j
 810 PLOT INK 3;0,0: DRAW 255,0:
 DRAW 0,175: DRAW -255,0: DRAW 0
,-175
 811 PLOT 1,1: DRAW 253,0: DRAW
0,173: DRAW -253,0: DRAW 0,-173
 830 LET a$=INKEY$: IF a$="" THE
N GO TO 830
 840 CLS : GO TO 20
1000 INPUT "Mes  -  ";m: IF m<1
OR m>12 THEN GO TO 1000
1010 INPUT "Dia  -  ";d: IF d<1
OR (d>29 AND m=2) OR (d>31 AND m
<>2) THEN GO TO 1010
1020 INPUT "Ano  -  ";y: IF y<10
0 THEN LET y=y+1900
1030 RETURN
1500 INPUT "Mes  -  ";m: IF m<1
OR m>12 THEN GO TO 1500
1510 INPUT "Ano  -  ";y: IF y<10
0 THEN LET y=y+1900
1520 RETURN
2000 IF m>2 THEN GO TO 2030
2010 LET f=365*y+d+31*(m-1)+INT
((y-1)/4)-INT (.75*(INT (((y-1)/
100)+1)))
2020 GO TO 2040
2030 LET f=365*y+d+31*(m-1)-INT
(.4*m+2.3)+INT (y/4)-INT (.75*(I
NT (y/100)+1))
2040 RETURN
3000 LET c=f+(INT (f/7)*-7): IF
c=0 THEN LET c=7: RETURN
3010 RETURN
3700 LET a=0: LET l=0: LET c=0:
LET m=0
3800 LET l(1)=31
```



```
3801 LET l(2)=28
3802 LET l(3)=31
3803 LET l(4)=30
3804 LET l(5)=31
3805 LET l(6)=30
3806 LET l(7)=31
3807 LET l(8)=31
3808 LET l(9)=30
3809 LET l(10)=31
3810 LET l(11)=30
3811 LET l(12)=31
3820 LET m$(1)="JANEIRO"
3821 LET m$(2)="FEVEREIRO"
3822 LET m$(3)="MARCO"
3823 LET m$(4)="ABRIL"
3824 LET m$(5)="MAIO"
3825 LET m$(6)="JUNHO"
3826 LET m$(7)="JULHO"
3827 LET m$(8)="AGOSTO"
3828 LET m$(9)="SETEMBRO"
3829 LET m$(10)="OUTUBRO"
3830 LET m$(11)="NOVEMBRO"
3831 LET m$(12)="DEZEMBRO"
3860 RESTORE 3870: READ g(24)
3870 DATA 16*3,24*3,30*3,34*3,37
*3,40*3,43*3,46*3,48*3,50*3,52*3
,53*3,54*3,55*3,56*3,57*3,57*3,5
8*3,58*3,59*3,59*3,60*3,60*3,60*
3
3880 RETURN
5000 BORDER 1: PAPER 1: BRIGHT 1
: INK 5: CLS : PRINT '''"PROGRAMA
 DE DATAS E CALENDARIO "
5010 PRINT 'TAB 13;"1.0.0"
5030 PRINT '"Este programa execu
ta o seguinte"
5040 PRINT "1 - Dado o mes e o a
no, o mes a-parece no video"
5050 PRINT "2 - Dadas duas datas
, o Spectrum calcula o numero de
 dias entre elas"
5060 PRINT "3 - Dado um dia do m
es, encontra o respectivo dia da
 semana"
5090 PRINT '''''"Voce pode omitir
 os dois primei-ros algarismos d
e qualquer ano  deste seculo"
5100 IF INKEY$="" THEN GO TO 510
0
5110 RETURN
```

Programa "Carregador de Códigos de Máquina"

Este é um programa utilitário, para aqueles que desejam iniciar em linguagem de máquina.

Ele pergunta a partir de que endereço você vai armazenar os códigos, para, em seguida, possibilitar a entrada desses códigos e seu posterior armazenamento, através de POKEs.

Se errar um código, digite "s" para interromper entrada de dados, anote o seu erro, e reinicie o programa, através de GOTO, e não RUN.

Compare o conteúdo da linha 1000 com os códigos do Apêndice D do Manual.

```
   1 REM  CARREGADOR DE CODIGOS
HEXADECIMAIS
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO        ARQUITRON
   4 REM ************************
  10 POKE 23658,8
  20 INPUT "Endereco inicial ";e
nd: LET inicio=end
  30 INPUT "Entre com o valor em
 hexa ";h$
  32 IF h$(1)="*" THEN LET end=e
nd-1: GO TO 30
  34 IF h$(1)="s" THEN STOP
  36 IF h$(1)="l" THEN INPUT "No
me do arquivo ";y$: SAVE y$CODE
inicio,(end-inicio)
  40 PRINT end;TAB 10;h$;TAB 20;
  50 LET byte=16*(CODE h$(1)-48-
(7 AND h$(1)>"9"))+(CODE h$(2)-4
8-(7 AND h$(2)>"9"))
  60 PRINT byte
  70 POKE end,byte
  80 LET end=end+1
  90 GO TO 30
```

Programa "Deleta Blocos de Linhas"

Este programa simula o comando da linguagem Basic Applesoft, denominado DEL x,y, que deleta as linhas x até y, inclusive.

O emprego deste programa é interessante, e deve ser feito após o outro programa deste livro, o "RENUMERADOR DE LINHAS". Você está digitando o seu programa, testou-o, e quer uma numeração de linhas mais adequada. Através do comando MERGE" ", carregue o renumerador de linhas, acione-o, e, de posse da nova numeração, novamente com o comando MERGE" ", carregue este programa para deletar as linhas inúteis. É um programa utilitário muito interessante.

Convém que se grave o programa após um primeiro cancelamento de linhas, para que, no caso de cancelar linhas erradas, você possa chamá-las de volta para o computador.

```
9974 REM deleta blocos de linhas
9975 OVER 0: FLASH 0: PAPER 7: I
NK 1: CLS
9976 PRINT AT 0,8; INVERSE 1;"De
letador de linhas"
9977 PRINT AT 4,0;"Deleta inclus
ive"
9978 PRINT AT 6,9;"Da linha "
9979 PRINT AT 8,9;"Ate a linha"
9980 PRINT AT 6,19; FLASH 1;" >
"; FLASH 0; INK 2; INVERSE 1;"
 "
9981 INPUT s$: IF VAL s$<0 OR VA
L s$>9973 THEN GO TO 9981
9982 LET sqd=INT (VAL s$)
9983 PRINT AT 6,19;"      ";AT 6,
20+4-LEN (STR$ sqd);sqd
9984 PRINT AT 8,19; FLASH 1;" >
"; FLASH 0; INK 2; INVERSE 1;"
 "
9985 INPUT s$: IF VAL s$<0 OR VA
L s$>9973 THEN GO TO 9985
9986 LET eqd=INT (VAL s$)
9987 PRINT AT 8,19;"      ";AT 8,
20+4-LEN (STR$ eqd);eqd
9988 PRINT AT 11,10; FLASH 1; IN
K 2; INVERSE 1;"Reformatando"
9989 LET n=23755
9990 IF 256*PEEK n+PEEK (n+1)>=s
qd THEN GO TO 9992
9991 LET n=(n+3+PEEK (n+2)+256*P
EEK (n+3)+1): GO TO 9990
9992 LET da=n+2: LET dl=-4
9993 LET n1=PEEK (n+2)+256*PEEK
(n+3): LET dl=dl+n1+4
9994 LET n2=(256*PEEK n+PEEK (n+
1)): IF n2<eqd THEN LET n=n+3+n1
+1: GO TO 9993
9995 IF n2>eqd THEN LET dl=dl-n1
-4
9996 LET n1=INT (dl/256): POKE (
da+1),n1: POKE da,dl-n1*256: POK
E (da+2),234
9997 PRINT AT 11,10;"
";AT 11,12; INK 2; INVERSE 1;"Te
rminado "
9998 PRINT AT 15,0;"Digite"; FLA
SH 1;sqd;" ENTER "; FLASH 0;" pa
ra cancelar linhas"
9999 STOP
```

Programa "Folha de Pagamentos"

Este é outro programa utilitário deste capítulo do livro.

Ele tem capacidade de armazenar dados sobre funcionários de empresa, cujo número está limitado apenas à capacidade de memória do micro (ultrapassa 300 funcionários). A versão original do programa está limitada a 50 funcionários, mas dependendo da sua necessidade, amplie-a à vontade, que o micro acusa até quanto ele agüentará.

Existe a possibilidade, também, de se emitir automaticamente contracheques para os funcionários.

NOTAS:

Os detalhes sobre funcionários em uma matriz estão estruturados da seguinte forma:

– primeira gravação – número de funcionários contidos na matriz;

– gravações seguintes – custo/hora de cada funcionário.

Listagem do programa:

```
   0>REM FOLHA DE PAGAMENTOS
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO       ARQUITRON
   4 REM ************************
 120 LET rm=0: LET pm=rm: LET pr
=rm
 130 OVER 0: PAPER 7: INK 1: FLA
SH 0: CLS : GO TO 9000
1000 REM rotina
1010 REM inicializacao
1020 CLS : PRINT AT 0,6; INVERSE
 1; INK 1;"Informacoes de empreg
ados"
1030 PRINT AT 6,1; INK 1;"Digite
 nova informacao"
1040 PRINT AT 8,1; INK 1;"Recupe
rar informacao  R"
1050 PRINT AT 15,8; INK 1;"Press
ione""; FLASH 1;"E"; FLASH 0;"
ou "; FLASH 1;"R"
1060 LET k$=INKEY$: IF k$<>"e" A
ND k$<>"r" THEN GO TO 1060
1070 IF k$="e" OR k$="r" THEN LE
T k$=CHR$ (CODE k$-32)
1080 PRINT AT 15,15; INK 1; INVE
RSE 1;"E";AT 15,22;"R": PRINT AT
 18,12;
```



```
1090 IF k$<>"r" THEN GO TO 1150
1100 PRINT INVERSE 1; INK 2;"Rec
upera"; : LET rm=1
1110 INPUT "Pressione SPACE e EN
TER quando  estiver pronto";k$
1120 IF k$<>" " THEN GO TO 1110
1130 LOAD "miniconta" DATA e$()
1140 LET ne=VAL e$(1,1 TO 2): LE
T f=VAL e$(1,15 TO 21): GO TO 11
90
1150 PRINT INVERSE 1; INK 2;" EN
TRE ": PRINT AT 21,2; INK 1; INV
ERSE 1; FLASH 1;" > "; FLASH 0;
INVERSE 0;"Num. empreg. (1 a 50)
"; INVERSE 1;"
1160 INPUT ne: IF ne>50 OR ne<1
THEN GO TO 1160
1170 PRINT AT 21,30;"  ";CHR$ 8;
CHR$ 8; INK 1; INVERSE 1;ne
1180 DIM e$(ne+1,21)
1190 DIM r$(ne,6)
1200 DIM o$(ne,6)
1210 DIM g$(ne,8)
1220 REM
1230 INK 1: PAPER 7: CLS : PRINT
 AT 0,8; INVERSE 1;"Detalhes
"
1240 PRINT AT 2,2; INVERSE 1;"No
mes      ";AT 2,21;"Custo/hora"
1250 IF rm=1 THEN GO TO 1500
1260 REM detalhes de empregados
1270 LET rm=1
1280 FOR i=1 TO ne+1
1290 LET e$(i,1 TO 15)=" ": LET
e$(i,16 TO 21)="000000"
1300 LET in=i+3-2
1310 PRINT AT in,1; : IF i>10 THE
N PRINT AT in,0;
1320 PRINT i-1; FLASH 1; INVERSE
 1;" T ";AT in,3; FLASH 0; INK 2
;"                "
1330 INPUT s$: IF s$="*" THEN RE
TURN
1340 LET e$(i,1 TO 15)=s$: PRINT
 AT in,2;" ";e$(i,1 TO 15)
1350 PRINT AT in,20; INVERSE 1;
FLASH 1;" > ";AT in,23; FLASH 0;
 INK 2;"                "
1360 INPUT s$: IF VAL s$<0 THEN
GO TO 1360
1370 LET ns=3: GO SUB 3000:
1380 PRINT AT in,20;" ";AT in,23
;"          ";AT in,23;s$: LET e$(i,
16 TO 21)=s$
1390 NEXT i
1400 PRINT AT 21,9; INVERSE 1;"F
ator extra";AT 21,25; FLASH 1;"
> "; FLASH 0; INK 2;AT 21,27;"
```

```
1410 INPUT s$: IF VAL s$<0 THEN
GO TO 1410
1420 LET f=VAL s$: PRINT AT 21,2
5;"      ";AT 21,25;f: LET e$(1
,16 TO 21)=s$
1430 LET e$(1,1 TO 2)=STR$ ne
1450 INPUT "SAVE dados (s/n)";k$
1460 IF k$="" THEN GO TO 1450
1470 IF k$(1)="s" OR k$(1)="S" T
HEN INPUT "Nome do arquivo ";x$:
 SAVE x$ DATA e$(): RETURN
1480 IF k$(1)<>"n" AND k$(1)<>"N
" THEN GO TO 1450
1490 RETURN
1500 REM display
1510 LET ne=VAL (e$(1,1 TO 2)):
LET f=VAL (e$(1,16 TO 21))
1520 FOR i=2 TO ne+1
1530 LET in=i+3-2
1540 PRINT AT in,2;e$(i,1 TO 15)
;AT in,23;e$(i,16 TO 21)
1550 NEXT i
1560 PRINT AT 21,9; INVERSE 1;"F
ator extra "
1570 PRINT AT 21,25;f
1580 LET k$=INKEY$: IF k$="" THE
N GO TO 1580
1590 IF k$="z" THEN COPY : GO TO
 1580
1600 RETURN
2000 REM
2020 CLS : PRINT AT 0,6; INVERSE
 1;"Registros de pagamentos "
2030 PRINT AT 1,17; INVERSE 1;"-
--- HORAS ----"
2040 PRINT AT 2,0; INVERSE 1;"Em
pregados";AT 2,19;"REG";AT 2,27;
"HE"
2050 IF pm=1 THEN GO TO 2500
2060 LET pm=1: LET tr=0: LET to=
tr
2070 FOR i=2 TO ne+1
2080 LET in=i+3-2
2090 PRINT AT in,0;e$(i,1 TO 15)
2100 PRINT AT in,16; FLASH 1;" >
 "; FLASH 0; INVERSE 1; INK 2;"
     "
2110 INPUT n1: IF n1<0 THEN GO T
O 2110
2120 LET s$=STR$ n1: LET ns=3: G
O SUB 3000
2130 PRINT AT in,16;" ";s$: LET
tr=tr+n1: LET r$(i-1)=s$
2140 PRINT AT in,24; FLASH 1;" >
 "; FLASH 0; INK 2; INVERSE 1;"
     "
2150 INPUT n1: IF n1<0 THEN GO T
O 2150
```



```
2160 LET s$=STR$ n1: LET ns=3: G
O SUB 3000
2170 PRINT AT in,24;" ";s$
2180 PRINT to=to+n1: LET o$(i-1)
=s$
2190 NEXT i
2200 LET s$=STR$ tr: LET ns=4: G
O SUB 3000
2210 PRINT AT 21,6; INVERSE 1;"T
otais";AT 21,16; INVERSE 0;s$
2220 LET s$=STR$ to: GO SUB 3000
2230 PRINT AT 21,24;s$
2240 RETURN
2500 REM display horas
2510 FOR i=1 TO ne
2520 LET in=i+3-1
2530 PRINT AT in,0;e$(i+1,1 TO 1
5);AT in,17;r$(i,1 TO 6);AT in,2
5;o$(i,1 TO 6)
2540 NEXT i
2550 PRINT AT 21,6; INVERSE 1;"T
otais"
2560 LET s$=STR$ tr: LET ns=4: G
O SUB 3000
2570 PRINT AT 21,16;s$
2580 LET s$=STR$ to: GO SUB 3000
2590 PRINT AT 21,24;s$
2600 LET k$=INKEY$: IF k$="" THE
N GO TO 2600
2610 IF k$="z" THEN COPY : GO TO
 2600
2620 RETURN
3000 REM
3020 LET t$=STR$ (INT VAL s$)
3030 FOR j=1 TO (ns-LEN t$): LET
 t$=" "+t$: NEXT j
3040 LET u$=STR$ (INT ((VAL s$-V
AL t$)*100+.5)): LET t$=t$+"."
3050 IF VAL u$=0 THEN GO TO 3090
3060 FOR j=1 TO LEN u$: IF u$()
TO j)="." THEN GO TO 3080
3070 NEXT j
3080 LET u$=u$(1 TO (j-1))
3090 IF VAL u$<10 THEN LET u$="0
"+u$
3100 LET s$=t$+u$
3110 RETURN
4000 REM
4010 REM calculo
4020 IF pr=1 THEN GO TO 4100
4030 LET pr=1: LET tp=0
4040 FOR i=1 TO ne
4050 LET n1=(VAL r$(i)+VAL o$(i)
*f)*(VAL e$(i+1,16 TO 21))
4060 LET s$=STR$ n1: LET ns=5: G
O SUB 3000
4070 LET g$(i,1 TO 8)=s$
4080 LET tp=tp+n1
```

```
4090 NEXT i
4100 CLS : PRINT AT 0,8; INVERSE
 1;"Registro de pagamentos"
4110 PRINT AT 2,0; INVERSE 1;"Em
pregado";AT 2,21;"Bruto";AT 3,19
;" "
4120 FOR i=1 TO ne
4130 LET in=i+3-1
4140 PRINT AT in,0;e$(i+1,1 TO 1
5);AT in,22;g$(i,1 TO 8)
4150 NEXT i
4160 PRINT AT 21,7; INVERSE 1;"T
otal de salarios";AT 21,19;" ";
INVERSE 0;
4170 LET s$=STR$ tp: LET ns=6: G
O SUB 3000
4180 PRINT " ";s$
4190 LET k$=INKEY$: IF k$="" THE
N GO TO 4190
4200 IF k$="z" THEN COPY : GO TO
 4190
4210 RETURN
9000 REM
9010 REM rotina principal
9020 GO SUB 1000
9030 GO SUB 2000
9040 GO SUB 4000
9050 GO SUB 1230
9060 GO TO 9030
```

Programa "Histogramas em Três Dimensões"

Neste programa utilitário, você simula histogramas, ou gráficos de barras (máximo 25) tridimensionais, através da utilização dos caracteres gráficos definidos pelo usuário A e B.

```
   1 REM ***     HISTOGRAMAS   ***
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO        ARQUITRON
   4 REM ************************
  30 GO SUB 9000
  40 GO SUB 8000
  50 GO SUB 7000
  60 LET d=1
  70 FOR a=10 TO 0 STEP -1
  80 PRINT AT d,0;a
  90 LET d=d+2
 100 NEXT a
 110 REM
 120 REM
 130 INPUT "Quantas barras (max.
```



```
 25)  ";i: IF i>25 THEN GO TO 13
0
 135 IF i<14 THEN LET p=2
 137 IF i>=14 THEN LET p=1
 140 FOR a=4-(1 AND p=1) TO ((i-
1)*p)+4-(1 AND p=1) STEP p
 150 INPUT "Insira valor(<100) p
/ ";(INT (a-2)/p);h: IF h>100 TH
EN LET h=h/100
 155 IF h>10 THEN LET h=h/10
 160 LET h1=(h*2)-1
 165 PRINT AT 21,a; INK 0; PAPER
 4;"A"; PAPER 2;"B"
 170 FOR b=20 TO 21-(h1+1) STEP
-1
 180 PRINT AT b,a; INK 4; PAPER
6;"A"; INK 2;"B"
 190 PRINT AT b,a; PAPER 4;" ";
PAPER 2;" "
 200 NEXT b
 210 PRINT AT b,a; PAPER 8; INK
6;"BA"
 220 LET b=b+1
 230 PRINT AT b,a; INK 4; PAPER
6;"A"; INK 2;"B"
 240 NEXT a
 250 PRINT #1; INK 9;AT 1,0;"<z>
 p/ copiar ou """" p/rodar"
 260 PAUSE 0: IF INKEY$="z" THEN
 COPY
 270 RUN
6999 STOP
7000 PLOT 24,175: DRAW 0,-175
7010 PRINT AT 0,0; PAPER 1;"Y"
7020 PRINT AT 21,31; PAPER 1;"X"
7030 RETURN
8000 INPUT "Cor da borda ";b: BO
RDER b: PAPER 0: INK 9: BRIGHT 1
: CLS
8010 LET x=0
8020 RETURN
9000 FOR a=USR "a" TO USR "b"+7
9010 READ us: POKE a,us
9020 NEXT a: RETURN
9030 DATA 128,192,224,240,248,25
2,254,255
9040 DATA 1,3,7,15,31,63,127,255
```

Programa "Mudança de Tipos de Caracteres"

Observe a listagem abaixo, e veja como os caracteres do micro estão diferentes em relação aos anteriores.

Se quiser saber como fazer isso, digite este pequeno programa auto-explicativo que ele lhe mostrará os truques necessários.

```
          POKE 23607 ,250
Para obter estes caracteres

          POKE 23607, 60

Para retornar aos caracteres
normais

Gravar:SAVE nome CODE 64000,1024
Pode dar NEW, mantendo codigos
depois da RAMTOP
```

```
   0>REM MUDANCA DE TIPOS DE
         CARACTERES
   1 REM ************************
   2 REM JOSE EDUARDO MALUF DE
CARVALHO                 ARQUITAON
   3 REM ************************
  10 POKE 23609,100
  20 BORDER 6: PAPER 6: INK 0: B
RIGHT 1: CLS
  30 LET a=PEEK 23606+256*PEEK 2
3607
  40 PRINT AT 21,2;"Digite taman
ho de RAM "
  50 INPUT " 16 ou 48 K RAM ";ra
m
  60 IF ram<>16 AND ram<>48 THEN
 GO TO 50
  70 CLS : PRINT AT 19,6;" Aguar
de 2 minutos                Novos
 caracteres estao           sendo
 gerados"
  80 IF ram=16 THEN LET c=31488
  90 IF ram=48 THEN LET c=64000
 100 REM ************************
 110 FOR n=c TO c+1024: LET b=PE
EK a: POKE n,b
 120 IF b/4=INT (b/4) THEN POKE
n,b+2
 130 IF b/8=INT (b/8) THEN POKE
n,b+4
 140 IF b/16=INT (b/16) THEN POK
E n,b+8
 150 IF b/32=INT (b/32) THEN POK
E n,b+16
 160 IF b/64=INT (b/64) THEN POK
E n,b+32
```



```
 170 IF b=66 THEN POKE n,b+32
 180 IF b=0 THEN POKE n,0
 190 LET a=a+1: NEXT n: BEEP .1,
25
 200 REM ************************
 210 CLS : POKE 23607,c/256
 220 PRINT FLASH 1;AT 2,9;"POKE
23607 ,"; c/256
 230 PRINT FLASH 1;"Para obter e
stes caracteres          "
 240 POKE 23607,60
 250 PRINT AT 8,9;"POKE 23607, 6
0"
 260 PRINT '"Para retornar aos c
aracteres    normais"
 270 PRINT AT 14,0;"Gravar:SAVE
nome CODE ";c;",1024"
 280 PRINT "Pode dar NEW, manten
do codigos  depois da RAMTOP"
 290 PAUSE 0: CLEAR c: STOP
 300 SAVE "caracteres" LINE 1
```

Programa "Processador de Texto"

Este é outro programa utilitário, que permite a criação de documentos, cartas etc.

Se o seu micro tiver 16K de RAM, você terá espaço para criar até quatro páginas de texto. Mas se tiver 48K de RAM, pode-se armazenar até 35 páginas.

Assim que se roda o programa, aparece na tela um menu de opções.

Selecione primeiramente a opção 6, para dimensionar quantas páginas você vai necessitar. A opção 8 transforma o teclado do seu micro em uma máquina de escrever. Para apagar algum caractere, digite CAPS SHIFT e uma tecla cursora (5 - 6 - 7 - 8), para que o cursor intermitente fique sobre o caractere a apagar. Para mudar o cursor de posição, utilize o mesmo método.

Assim que você preencher essa página, ela então deve ser copiada em alguma daquelas páginas dimensionadas anteriormente, através da opção 3. Estas páginas também podem ser gravadas em fita cassete ou impressas. Usando a opção de remover espaços, não haverá mais espaços entre as páginas consecutivas.

Com a opção 7, pode-se utilizar qualquer símbolo do teclado, como por exemplo os símbolos gráficos.

Você pode também limpar o conteúdo de apenas uma página, ou de todas. Basta digitar a opção desejada, e o número da página a apagar.

Outras funções especiais, acessadas através de CAPS SHIFT:

Tecla 1 – EDIT – abre um espaço à direita do cursor, no texto;

Tecla 0 – DELETE – remove espaços do texto, ou apaga palavras;

Tecla 3 – TRUE VIDEO – move o texto uma linha para baixo;

Tecla 4 – INVERSE VIDEO – move o texto uma linha para cima.

A listagem:

```
   1 REM PROCESSADOR DE TEXTOS
   2 REM ************************
   3 REM JOSE EDUARDO MALUF DE
CARVALHO          ARQUITRON
   4 REM ************************
  10 LET s=1: CLS : DIM l(1): PR
INT "Processador de texto ©1985
    "; OVER 1; BRIGHT 1;AT 0,0;
"
"
  20 BEEP .1,0,1: BEEP .2,.5: BE
EP .1,.10: POKE 23562,1: : PAPER
6: BORDER 1: INK 0
 100 DIM a$(704): LET p=5: GO TO
 3000
 900 LET k$=" ": GO TO 1010
1000 LET k$=INKEY$: IF CODE k$=k
 THEN LET s=s+1
1004 IF k<>CODE k$ THEN LET s=1
1005 LET k=CODE k$
1006 IF p>704 OR p<0 THEN LET p=
(p>704)*704+(p<0): GO TO 1000
1010 IF k$="" THEN GO TO 1000
1020 IF k>=3 AND k<13 THEN BEEP
.01,50: GO SUB 2000: GO TO 1000
1030 IF k=13 THEN GO TO 3000
1035 LET x=INT (p/32): LET y=p-1
-INT (p/32)*32: IF y=-1 THEN LET
 y=31: LET x=x-1
1040 BEEP .04,40: POKE 22527+p,5
5: LET a$(p)=k$: PRINT AT x,y;k$
: LET p=p+1: POKE 22527+p,187: G
O TO 1000
2000 LET p=p+((k=9)-(k=8))*s: PO
KE 22527,8*7: LET p=p+((k=10)*32
-(k=11)*32)*s: PRINT AT 0,0;a$:
POKE 22527+p,187
2020 IF k=7 THEN LET a$(p+1 TO 7
04)=" "+a$(p+1 TO 704): PRINT AT
 0,0;a$: POKE 22527+p,187: RETUR
N
2030 IF k=12 THEN LET a$(p TO 70
```



```
4)=a$(p+1 TO 704): PRINT AT 0,0;
a$: POKE 22527+p,187: RETURN
2040 IF k=4 THEN POKE 22527+p,8*
7: LET a$(p TO 704)="
                    "+a$(p TO 7
04-33): LET p=p+32: PRINT AT 0,0
;a$: POKE 22527+p,187
2050 IF k=5 THEN POKE 22527+p,8*
7: LET a$(p-32 TO 704)=a$(p TO 7
04): LET p=p-32: PRINT AT 0,0;a$
: POKE 22527+p,187
2090 RETURN
3000 CLS : PRINT "Tamanho do arq
uivo ";l(1);" paginas ": PRINT "
1- Carregar arquivo "'"2- Gravar
 arquivo "'"3- Mudar pagina "'"4
- Imprimir "'"5- Apagar pagina "
'"6- Apagar tudo "'"7- Usar codi
gos do teclado "'"8- Retornar ao
 texto"
3010 LET k=CODE INKEY$-48: IF k<
1 OR k>8 THEN GO TO 3010
3020 GO SUB 3000+(k*100): GO TO
3000
3100 CLS : PRINT "Carregando arq
uivo ": INPUT "Nome do arquivo "
; LINE n$: LOAD n$ DATA l(): BEE
P 2,0: CLS : PRINT AT 0,0;"O arq
uivo tem ";l(1);" paginas de com
primento": LOAD n$ DATA t$(): BE
EP 2,40: RETURN
3200 CLS : PRINT "Gravando arqui
vo ": INPUT "Nome do arquivo ";
LINE n$
3205 PRINT AT 10,5;"Gravando ";n
$;"'": SAVE n$ DATA l(): BEEP 1,
20: SAVE n$ DATA t$(): BEEP 1,40
3210 RETURN
3300 CLS : PRINT "Mudando pagina
 corrente ": INPUT ("Pagina corr
ente de (0_";l(1);")0-nenhuma");
n: IF n<>0 AND n<=l(1) THEN LET
t$(n)=a$
3310 INPUT ("Nova pagina corrent
e (1-";l(1);") ");n: IF n<=l(1)
THEN LET a$=t$(n,1 TO 704)
3315 IF n>l(1) THEN PRINT "Numer
o de pagina muito grande ": BEEP
 1,-10: PAUSE 0
3316 RETURN
3400 CLS : PRINT "Imprimindo ":
INPUT "Numero de pagina (1-";(l(
1));")";n: INPUT "Remover espaco
s (s/n) ";n$: IF n>l(1) THEN GO
TO 3400
3410 LET p$=t$(n): IF n$="s" OR
n$="S" THEN FOR a=704 TO 1 STEP
-1: IF p$(a)<>" " THEN LET p$=t$
```

```
(n,1 TO a): GO TO 3420
3415 IF n$="s" THEN NEXT a
3420 LPRINT p$: RETURN
3500 CLS : PRINT "Apagar uma pag
ina ": INPUT "Pagina a ser apaga
da (1-";(l(1));") ";n: IF n<=l(1
) THEN LET t$(n)=""
3510 RETURN
3600 CLS : PRINT "Apagar tudo ":
 INPUT "Numero de paginas ";l(1)
: PRINT "Pressione uma tecla p/
apagar ": PAUSE 0: DIM t$(l(1),7
04): RETURN
3700 CLS : PRINT "Usar todo o te
clado ": INPUT "Entre com o que
voce quer "; LINE m$: IF LEN m$>
=0 THEN LET a$(p)=m$(1): LET p=p
+1
3710 RETURN
3900 GO TO 900
```

Programa "Relógio do TK 90X"

Este programa simula um relógio, muito bem ajustado, que possui até alarme.

Digite o programa e dê RUN.

A seguir, deve-se dizer a hora, entre 0 e 23, os minutos e os segundos. O programa checa a validade do dado digitado.

Se quiser alarme, repete-se o processo anterior.

Se quiser experimentar e aproveitar a garantia do seu micro, rode o programa antes de dormir, e ligue o alarme, para a sua hora de despertar; não se esqueça de ligar apenas um alto-falante na saída do seu micro (a tevê ligada a noite inteira é um alarme muito caro).

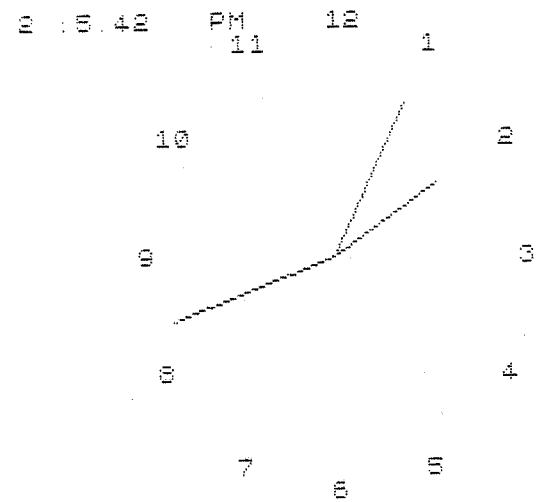
```
DIGITE A HORA ?
(0 - 23)
13
QUANTOS MINUTOS
(0 - 59)
38
QUANTOS SEGUNDOS?
(0 - 59)
23
QUER LIGAR ALARME?
(S ou N)
```



```
HORA? 14

MINUTOS? 0

SEGUNDOS?
```



Se ele estiver atrasando ou adiantando, ajuste o valor de t1, na linha 140, aumentando-o ou diminuindo-o, dependendo do caso.

```
  0)REM *** RELOGIO TK 90X  ***
  10 REM ************************
  20 REM JOSE EDUARDO MALUF DE
CARVALHO        ARQUITRON
  30 REM ************************
  40 BORDER 1: PAPER 1: BRIGHT 0
: INK 9: CLS
 100 GO SUB 360
 110 LET c=PI/30
 120 DEF FN t()=INT ((65536*PEEK
 23674+256*PEEK 23673+PEEK 23672
)/50)
 140 LET t1=FN t()
 150 LET h1=INT (t1/720)
 160 LET h=h1*c
```

```
 170 LET hx=50*SIN h: LET hy=50*
COS h
 180 LET mi=INT (t1/60)
 190 LET m=mi*c
 200 LET mx=60*SIN m: LET my=60*
COS m
 210 LET a=t1*c
 220 LET sx=72*SIN a: LET sy=72*
COS a
 230 PLOT 131,91: DRAW sx,sy
 240 PLOT 131,91: DRAW mx,my
 250 PLOT 131,91: DRAW hx,hy
 260 GO SUB 740
 270 LET t=FN t()
 280 IF t<=t1 THEN GO TO 270
 290 LET t1=t
 300 PLOT 131,91: DRAW OVER 1;sx
,sy
 310 IF INT (t1/60)<=mi THEN GO
TO 210
 320 PLOT 131,91: DRAW mx,my: PL
OT 131,91: DRAW OVER 1;mx,my
 330 IF INT (t1/720)<=hi THEN GO
 TO 180
 340 PLOT 131,91: DRAW hx,hy: PL
OT 131,91: DRAW OVER 1;hx,hy
 350 GO TO 150
 360 BORDER 1: PAPER 1: INK 6: B
RIGHT 1: OVER 0: FLASH 0: CLS :
PRINT "DIGITE A HORA ?": PRINT "
(0 - 23)"
 370 INPUT h: IF h<0 OR h>23 THE
N GO TO 370
 380 PRINT FLASH 1; INK 2;h
 390 LET d=0: IF h>=12 THEN LET
d=1: LET h=h-12*(h<>12)
 400 PRINT "QUANTOS MINUTOS": PR
INT "(0 - 59)"
 410 INPUT m: IF m<0 OR m>59 THE
N GO TO 410
 420 PRINT FLASH 1; INK 1;m
 430 PRINT "QUANTOS SEGUNDOS?":
PRINT "(0 - 59)"
 440 INPUT s: IF s<0 OR s>59 THE
N GO TO 440
 450 PRINT FLASH 1; INK 3;s
 470 PRINT "QUER LIGAR ALARME?":
 PRINT "(S ou N)"
 480 INPUT a$: IF a$<>"s" AND a$
<>"n" THEN GO TO 480
 490 IF a$="n" THEN LET al=0: BO
RDER 0: PAPER 0: INK 7: CLS : GO
 TO 630
 500 LET al=1: PRINT : PRINT "HO
RA?";
 510 INPUT ah: IF ah<0 OR ah>24
THEN GO TO 510
 520 PRINT ah: LET ad=0: IF ah>1
```



```
1 THEN LET ad=1: LET ah=ah-(ah<>
12)*12
 530 PRINT : PRINT "MINUTOS? ";
 540 INPUT am: IF am<0 OR am>59
THEN GO TO 540
 550 PRINT am
 560 PRINT : PRINT "SEGUNDOS? ";
 570 INPUT as: IF as<0 OR as>59
THEN GO TO 570
 580 PRINT as
 600 PAPER 0: BORDER 0: INK 7: C
LS : PRINT AT 20,0;"Alarme": PRI
NT ah;" :";am;" :"; as;" ";
 610 IF ad=0 THEN PRINT "AM": GO
 TO 630
 620 IF ad=1 THEN PRINT "PM"
 630 FOR n=1 TO 12
 640 PRINT AT 10-10*COS (n/6*PI)
,16+10*SIN (n/6*PI);n
 650 NEXT n: PRINT #1;AT 0,0;"RE
LOGIO TK 90X"
 660 LET dh=h: LET dm=m: LET ds=
s
 670 LET t=(h*3600+m*60+s)*50
 680 LET b1=t-INT (t/256)*256
 690 LET b2=INT (t/256)-INT (t/2
56↑2)*256
 700 LET b3=INT (t/256↑2)-INT (t
/256↑3)*256
 720 POKE 23674,b3: POKE 23673,b
2: POKE 23672,b1
 730 RETURN
 740 LET ds=t1-INT (t1/60)*60: L
ET dm=INT (t1/60)-INT (t1/3600)*
60: LET dh=INT (t1/3600)
 750 IF ds=0 THEN PRINT AT 0,0;"
"
 760 IF dh>=13 THEN LET dh=dh-12
 770 IF dh<>12 OR ds<>0 OR dm<>0
 THEN GO TO 810
 780 IF d=0 THEN LET d=1: LET dh
=12: GO TO 810
 790 IF d=1 THEN LET d=0: LET dh
=0
 800 LET dh=dh-(dh=12 AND d=0)*1
2
 810 PRINT AT 0,0;dh;" :";dm;":"
; FLASH 1;ds
 820 PRINT AT 0,9;" ";
 830 IF d=0 THEN PRINT "AM"
 840 IF d=1 THEN PRINT "PM"
 850 IF al=0 THEN RETURN
 860 IF ah<>dh OR am<>dm OR as<>
ds OR ad<>d THEN RETURN
 880 PRINT #1;AT 1,0;"Pressione
qualquer tecla": BEEP .5,27: BEE
P .5,20: IF INKEY$="" THEN GO TO
 870
 890 RETURN
```

Programa "Renumerador de Linhas"

Este é outro programa utilitário, que simula o comando de Basic Applesoft, AUTO, numerando automaticamente as linhas do programa, a partir do número 10.

Mas, atenção: ele não numera de forma automática os comandos GOTO e GOSUB, mas mostra na tela as linhas que contenham estes comandos, para facilitar a alteração numérica posterior manual.

```
9987 REM renumerador de linhas
9988 LET new=10: LET inc=10: LET
 bob=100: LET eob=9988
9989 LET l=new: LET n=23755
9990 IF (256*PEEK n+PEEK (n+1))>
=bob THEN GO TO 9992
9991 LET n=n+3+PEEK (n+2)+256*PE
EK (n+3)+1: GO TO 9990
9992 LET n1=(256*PEEK n+PEEK (n+
1)): IF n1>eob THEN STOP
9993 POKE n,INT (l/256)
9994 POKE n+1,l-256*INT (l/256)
9995 FOR i=n+4 TO n+3+PEEK (n+2)
+256*PEEK (n+3)
9996 IF PEEK i=236 OR PEEK i=237
 THEN PRINT l;" - ";CHR$ PEEK i
9997 NEXT i
9998 LET n=i: LET l=l+inc
9999 GO TO 9992
```

Programa "Scroll Lento"

Esta pequena rotina de códigos de máquinas tem um efeito muito interessante, similar a apresentação de filmes que utilizam letreiros de apresentação da ficha técnica e que fazem aquelas letras subirem a tela, como SCROLL, só que mais lentamente.

Digite esta rotina, de um endereço inicial, por exemplo, 65000, se o seu micro tiver 48K RAM.

Em seguida, faça um pequeno programa, como por exemplo:

10 PRINT AT 21, 10; "O nome"

20 PAUSE 0: IF INKEY$ = "P" THEN RAND USR 65000

E veja que efeito maravilhoso obterá, após digitar a linha 30 GOTO 20

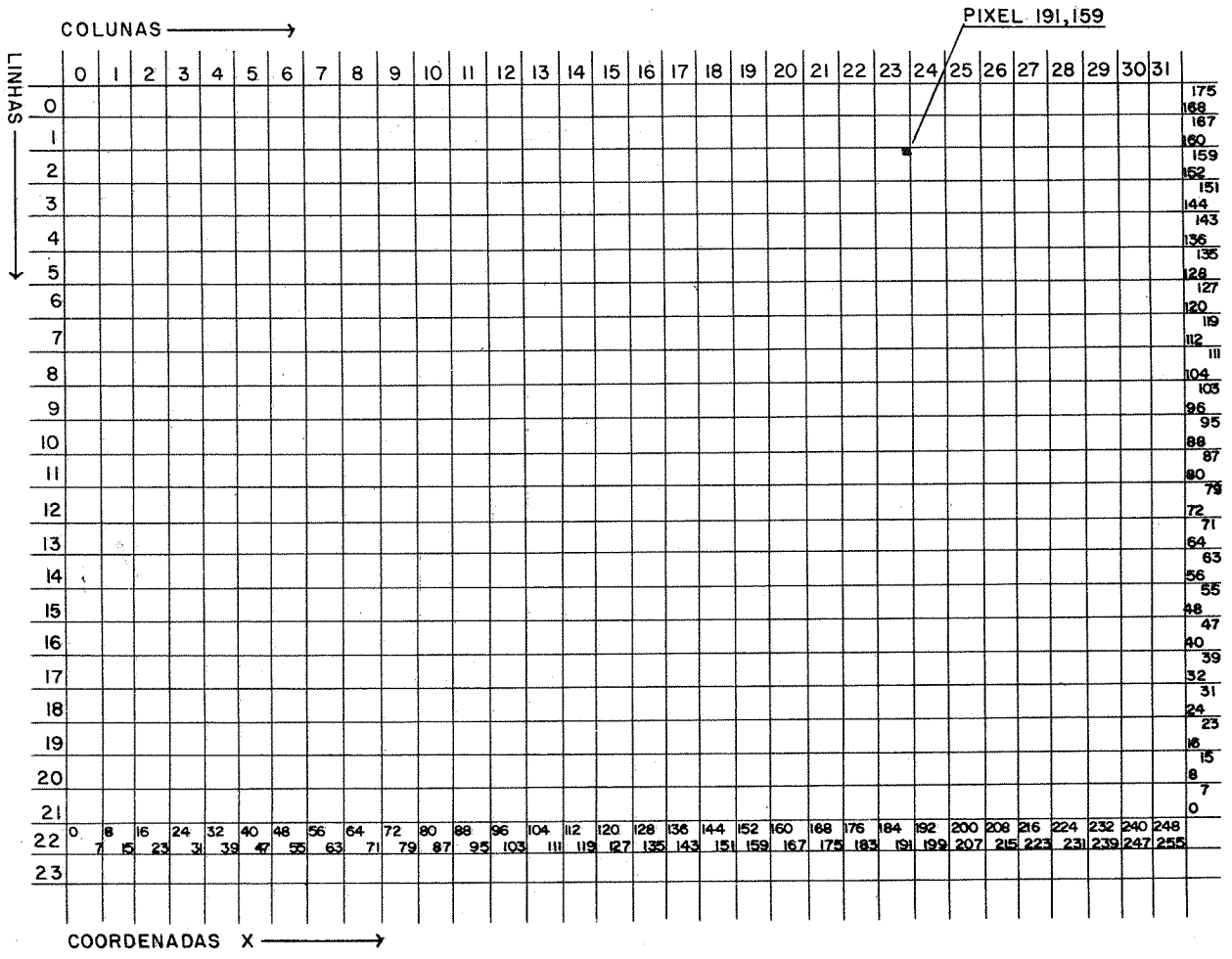
Não se esqueça de conferir atentamente os códigos do programa, pois apenas um errado põe tudo a perder.

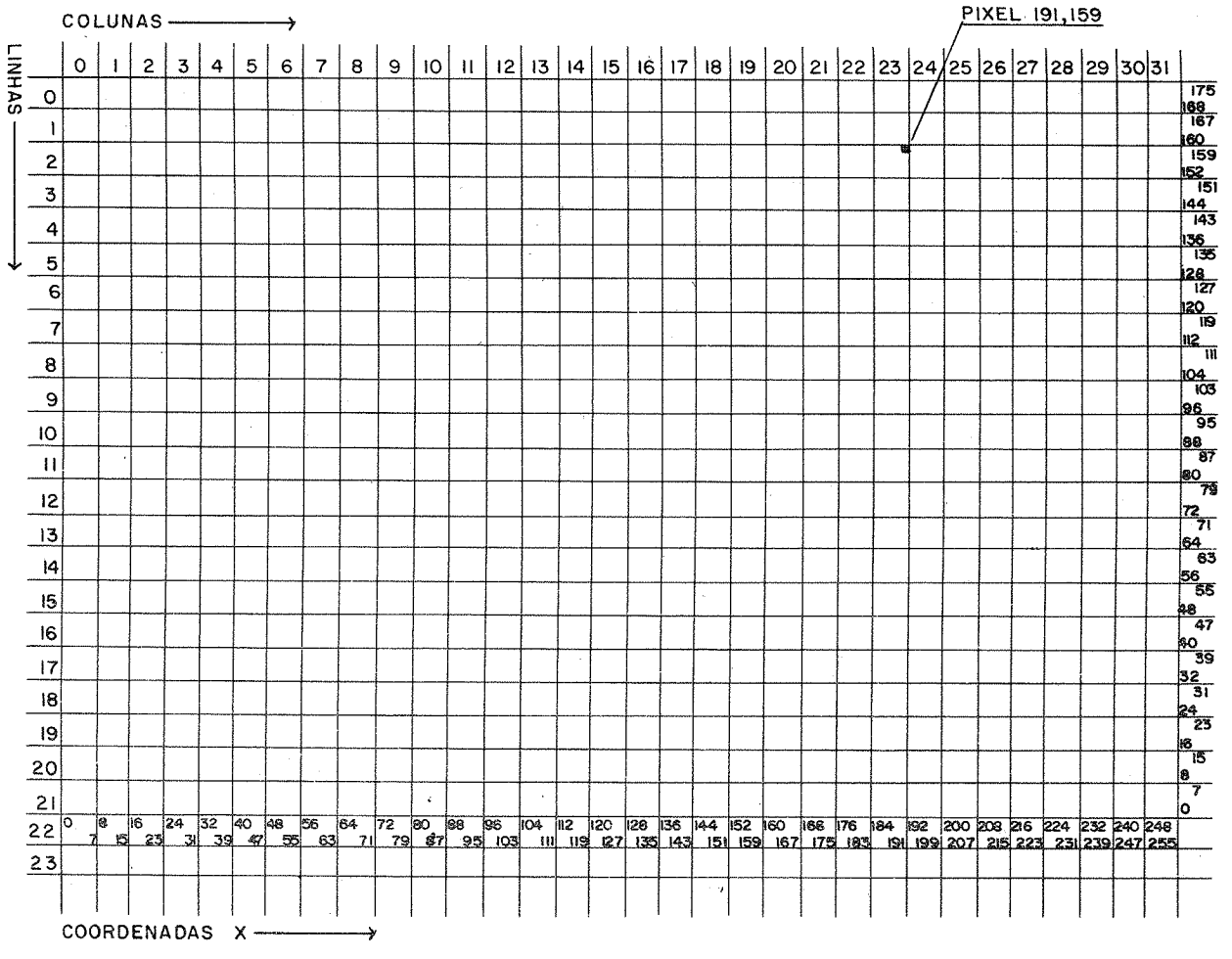
```
9992 REM scroll lento
9993 DATA 33,0,65,17,0,64,6,3,19
7,6,8,197,6,7,197,1,32,0,237,176
,1,224,0,9,229,213,225,9,229,209
,225,193,16,236,1,224,7,237,66,1
,32,0,237,176,1,32,0,237,66,229
9994 DATA 209,1,0,1,9,193,16,209
,1,0,7,213,225,9,1,32,0,229,237,
66,229,209,225,1,32,0,237,176,1,
0,1,213,225,9,193,16,177,33,224,
87,1,0,32,113,35,16,252,201
9995 INPUT "Endereco inicial ";e
9996 FOR f=0 TO 97
9997 READ a
9998 POKE e+f,a
9999 NEXT f
```

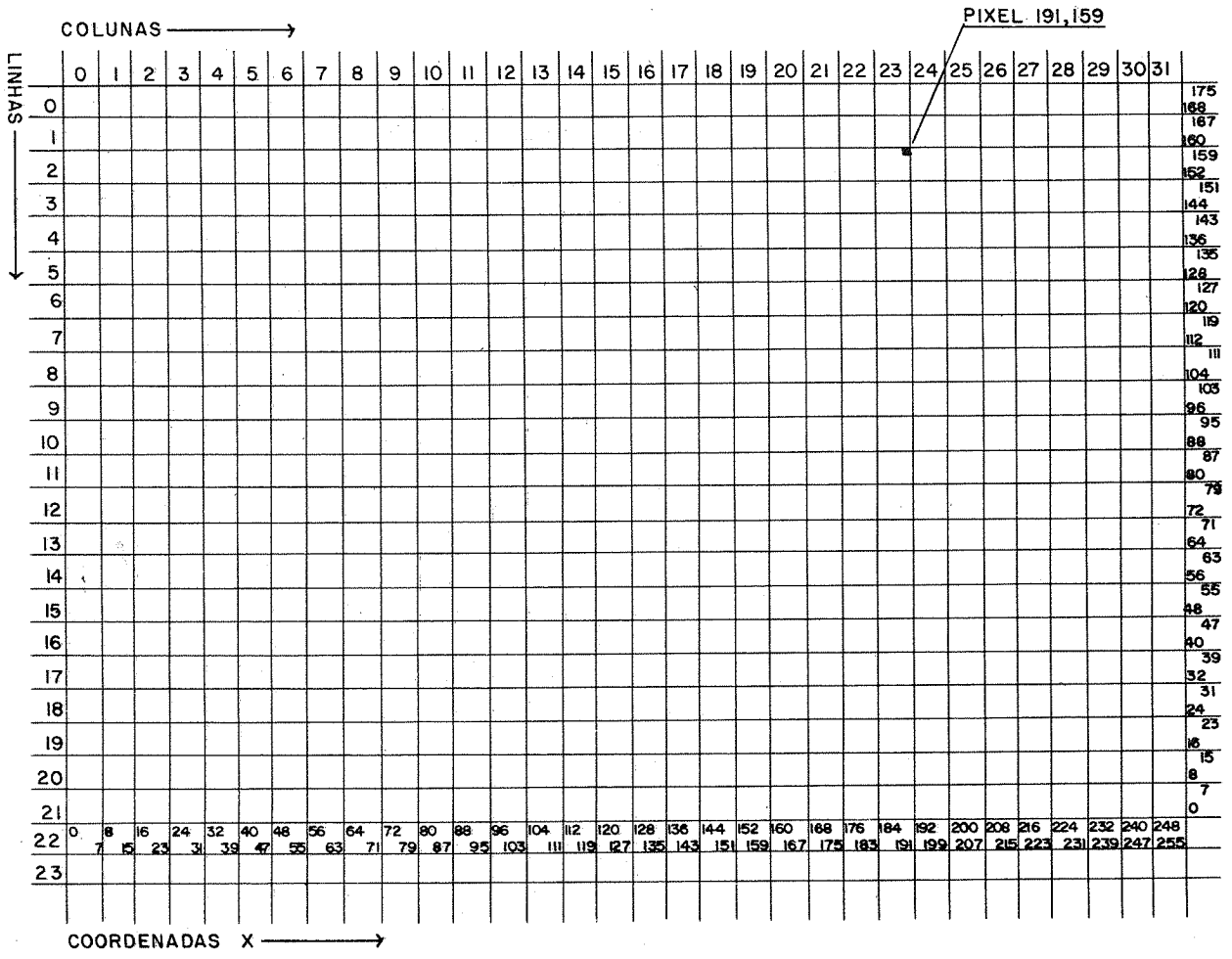
```
   COMO EXEMPLO DE APLICACAO,
EXPERIMENTE DIGITAR ESTE PEQUENO
 PROGRAMA, QUE DEMONSTRA A UTILI
ZACAO DESTA ROTINA.
   LEMBRE-SE DE DIGITA-LA, AN  -
TES DE USAR ESTE EXEMPLO.
```

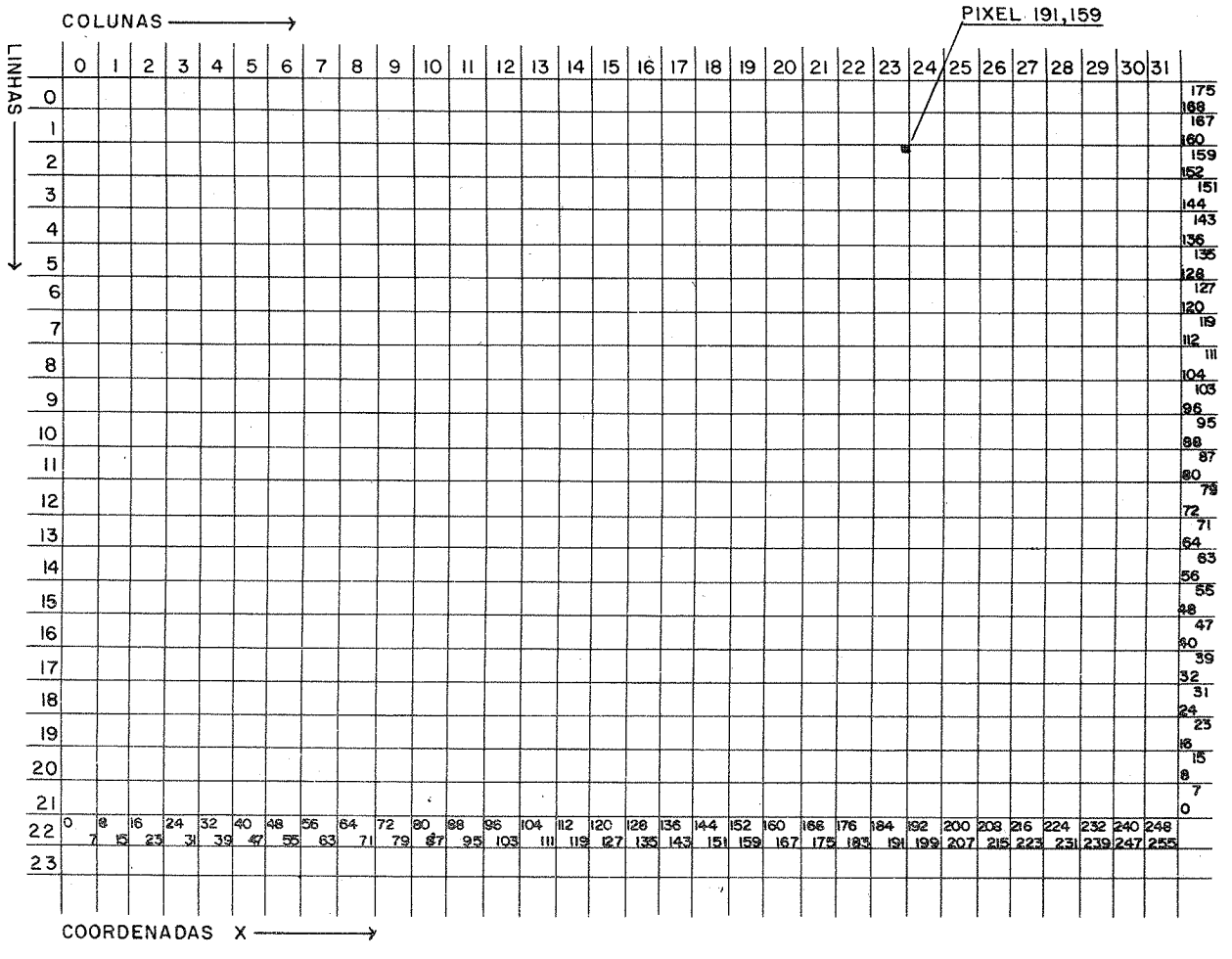
```
NOTE QUE O ENDERECO INICIAL PEDI
DO PELO PROGRAMA, EQUIVALE A
65400, CONFORME LINHA 10 A SE  -
GUIR
```

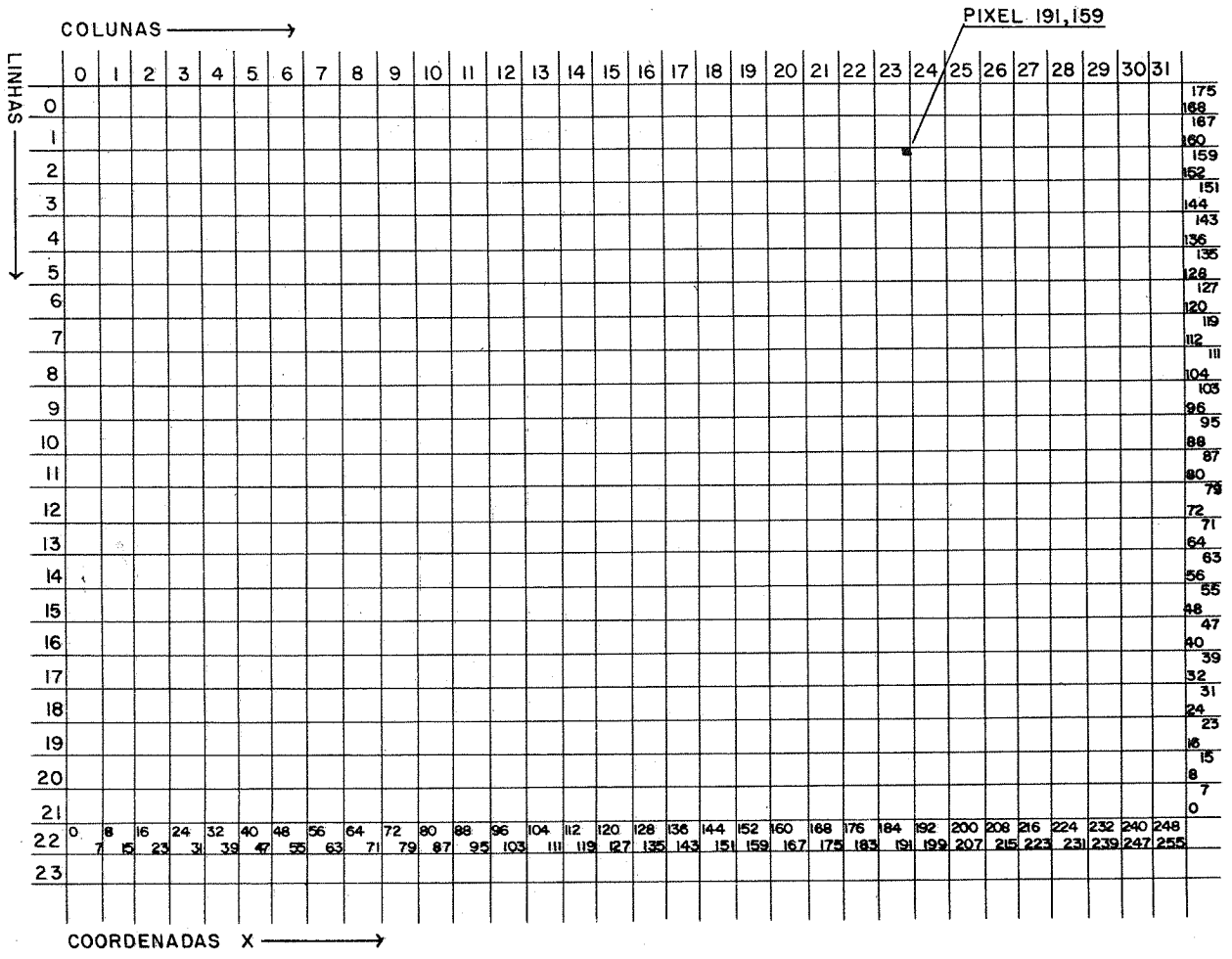
```
  10 FOR f=1 TO 10: PRINT AT 21,
12;" TK 90X ";: FOR g=1 TO 20: R
ANDOMIZE USR 65400: NEXT g: NEXT
 f
  20 GO TO 10
```

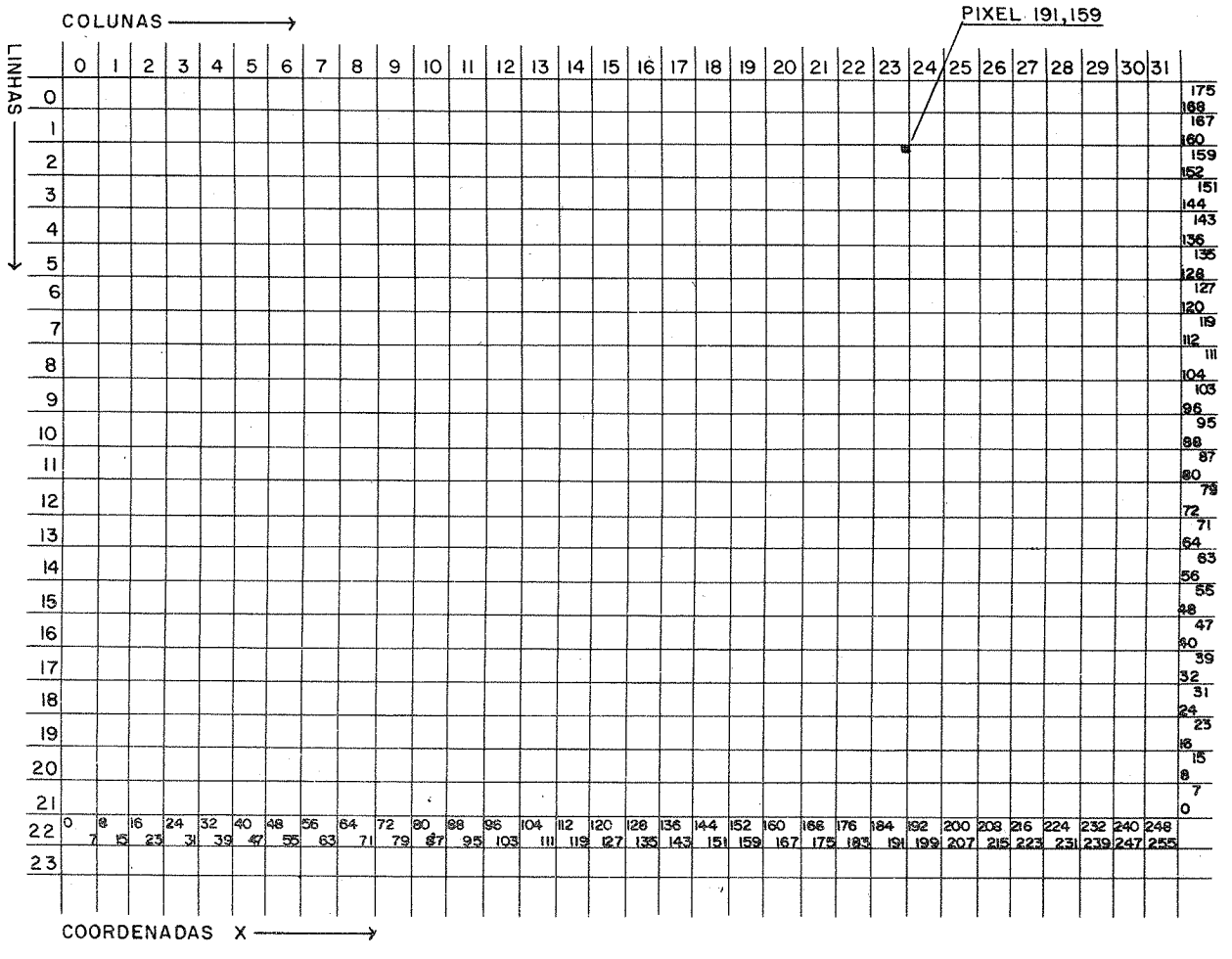

COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y

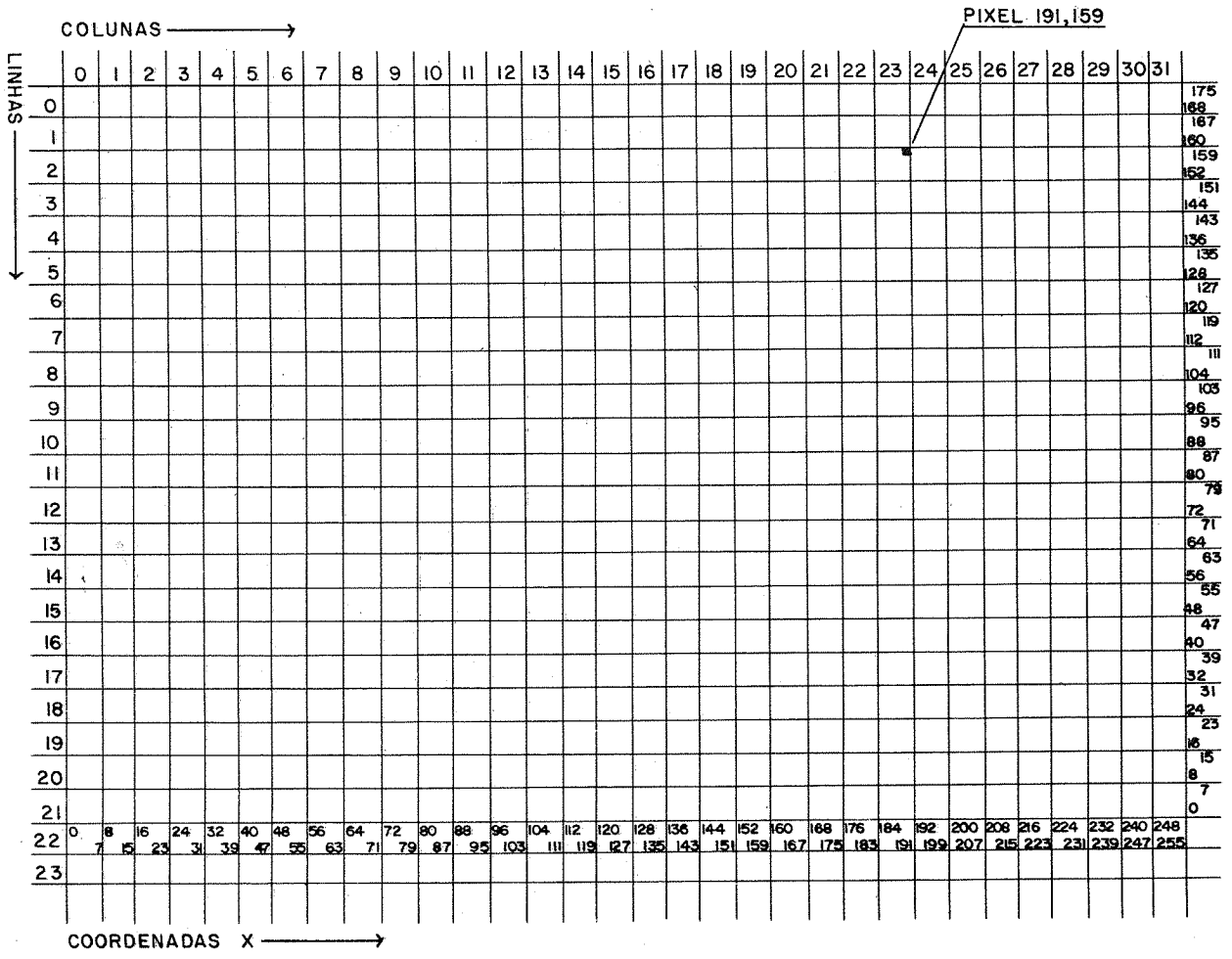

COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y

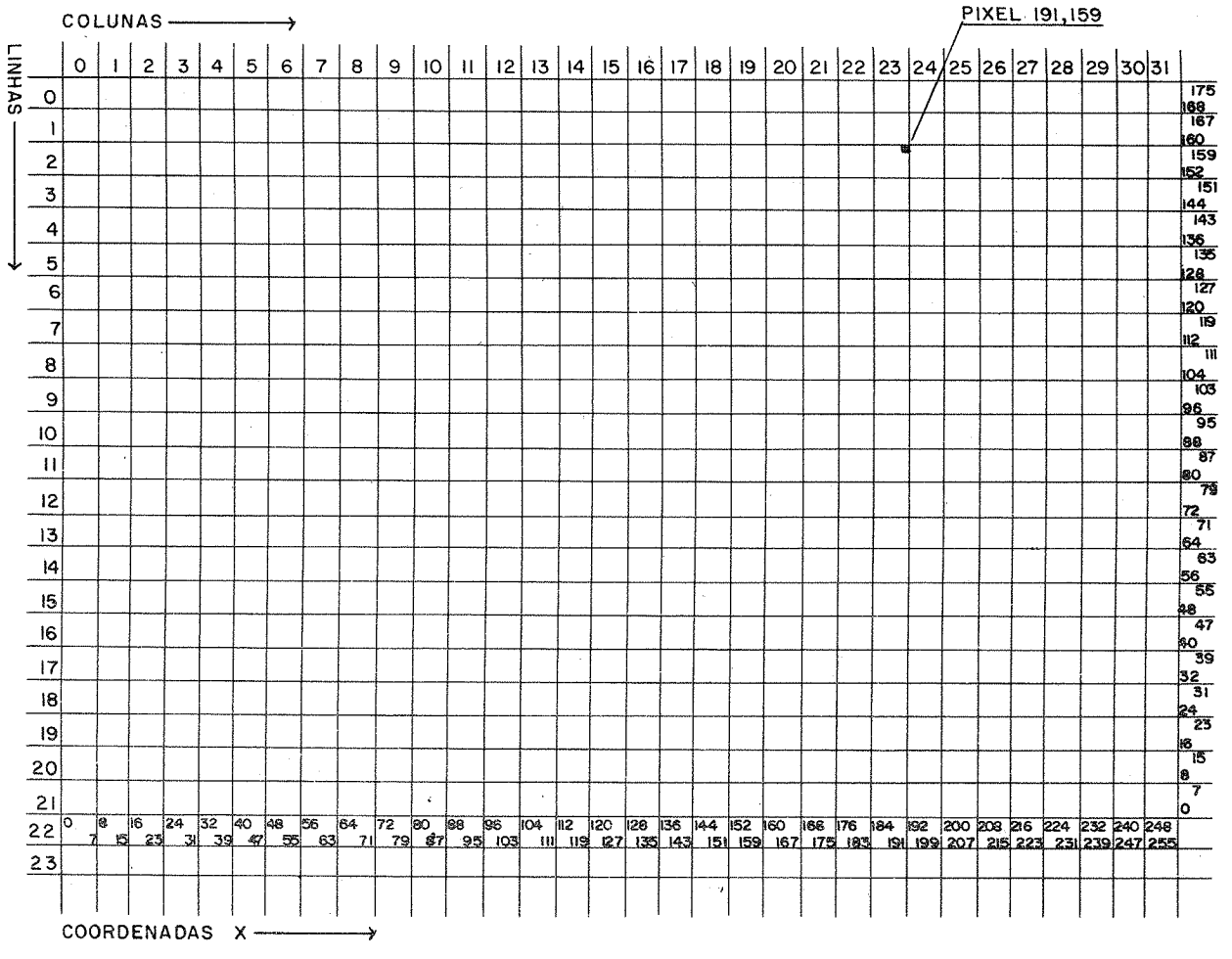

COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y

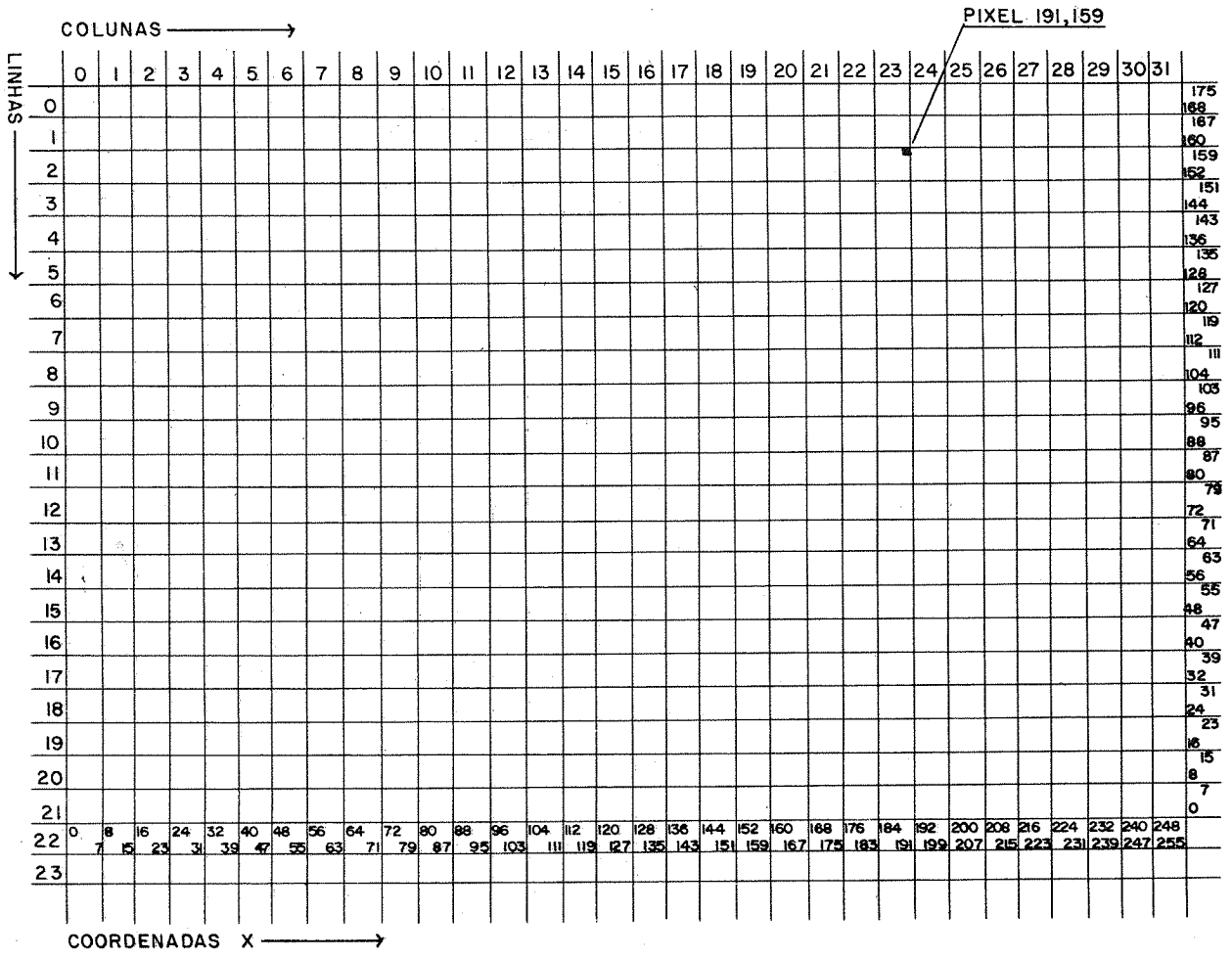

COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y

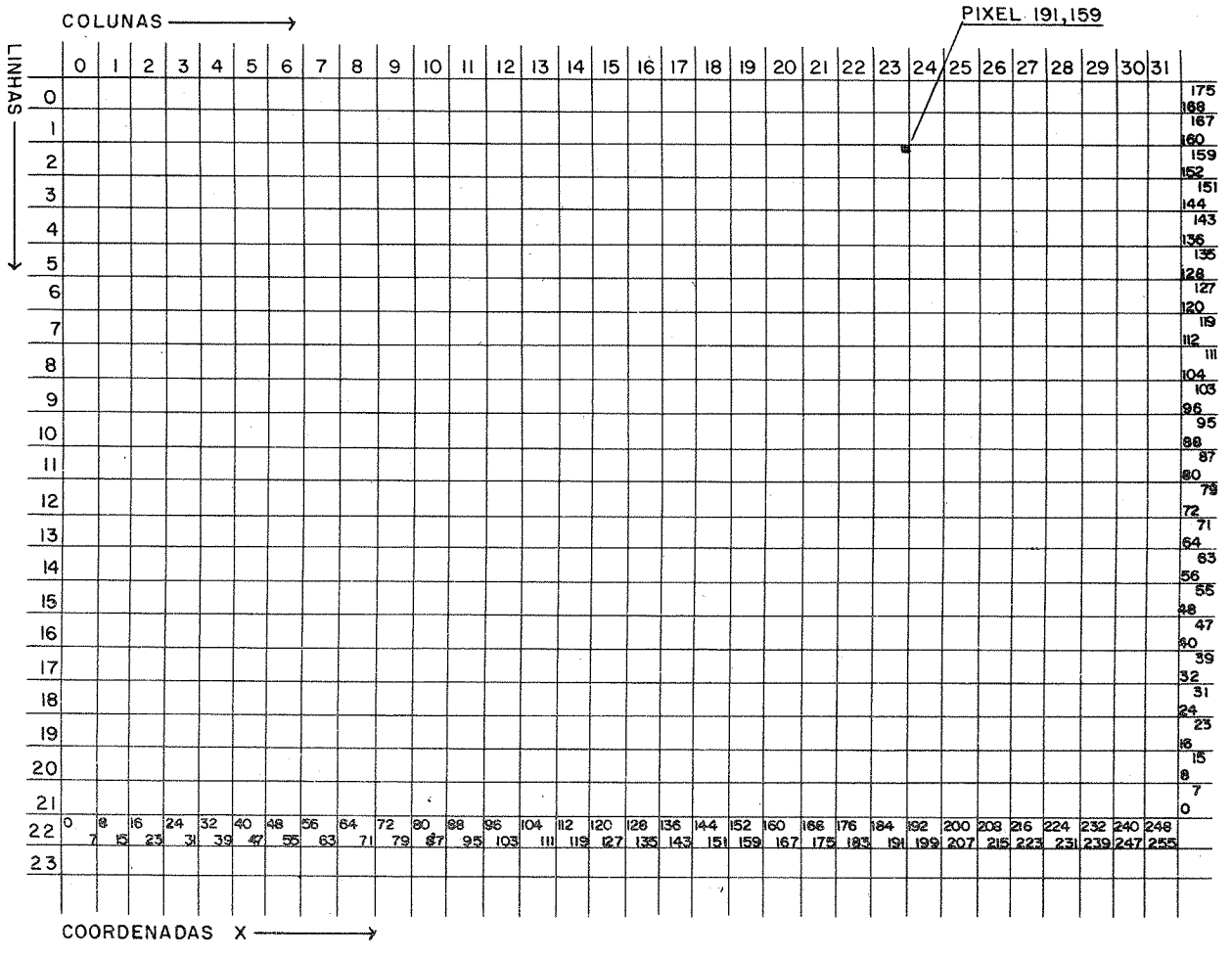

COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y


COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y


COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y


COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y


COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y


COLUNAS
LINHAS
PIXEL 191,159
COORDENADAS X
COORDENADAS Y

Composição:
JAG Composições e Artes Gráficas Ltda.

Impresso na Prol editora gráfica ltda.
03043 Rua Martim Burchard, 246
Brás - São Paulo - SP
Fone: (011) 270-4388 (PABX)
com filmes fornecidos pelo Editor.