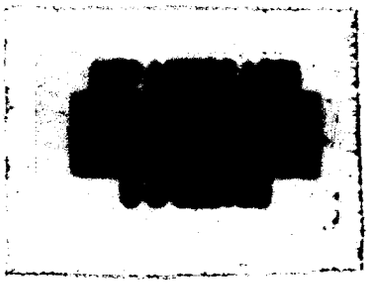
UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FISICA E QUIMICA DE SÃO CARLOS

Estudo e Implementação de uma Arquitetura para Terminais de Baixo Custo

Alvaro Garcia Neto



Dissertação apresentada ao Instituto de Física e Química de São Carlos da Universidade de São Paulo para a obtenção do título de Mestre em Física Aplicada.

Orientador: Prof. Dr. Jan F. W. Slaets.

Departamento de Física e Ciência dos Materiais
Abril de 1986

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE ____

Alvaro Garcia Neto

APRESENTADA AO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 27 DE maio DE 1986.

COMISSÃO JULGADORA:

Dr. Jan F.W.Slaets - Orientador

Dr. Valentin Obac Roda

Dr. Arthur João Catto

`A minha melhor amiga.

Agradecimento

Um trabalho dessa envergadura sempre reflete o esforço abnegado de muitos amigos. E' impossivel enumera'-los completamente; registro aqui meu agradecimento `aqueles que no momento relembro e meu constrangimento com os omitidos. A todos, a certeza de que sem o sorriso, a conversa de corredor e a presença nas horas dificeis, não teriamos chegado aqui.

`A Iara, por caminhar comigo,

a meus pais e irmãos, por tudo que fizeram e fazem,

aos sogros pela Iara.

ao prof. Slaets pela orientação, pelo exemplo profissional e sobretudo pela amizade,

ao prof. Valentin pelo tempo empregado com os problemas dessa dissertaç`ao, e pela consideração,

aos profs. Ruggiero e Moreira pela grande amizade e pelas madrugadas dos tempos de implantação do L.I.E.,

ao Eng. Gonzalo pela inestimável ajuda na depuração e programação,

ao prof. Caetano pelos comentários úteis e presença sempre afável.

aos técnicos do laboratório: Ailton, Marcos, Ivanilda, Dô e especialente Lirio, pelas montagens e pela agradável convivência diária,

`a Nêuri pelo apoio na interminável burocracia,

`a Benê pelos desenhos "prá ontem",

ao Chico e Gilda, Samuel e Raquel e Marcos e Nice pela amizade,

ao casal Lineu A. Floret pelo apoio no início do casamento,

ao casal Eunice e Teixeira pela amizade não somente para conosco mas também para com nossos queridos, e

às comunidades presbiterianas de S. Carlos e Sta. Bárbara d'Oeste pela esperança e luta comum,

o meu mais profundo obrigado.

Alvaro
Abril, 1986

APRESENTAÇAO

Indice de Figuras i
Resumo iii
Abstract v
Siglas e Convenções vii
Caracterização do trabalho xi
Introdução xv

CAPITULO I CONCEITOS BASICOS

I.1 O CINESCOPIO COMUM (CRT) 1
I.1.1 Principio de Funcionamento 2
I.1.2 Formação de Imagens por Varredura de Rastro . . 3
I.1.3 Arquitetura de um Sistema com CRT 6
I.2 O CINESCOPIO COM MEMORIA (DVST) 9
I.2.1 Principio de Funcionamento 9
I.2.2 Formação de Imagens por Varredura Aleatória . . 11
I.2.3 Arquitetura de um Sistema com DVST 14
I.3 COR E INTENSIDADE 16
I.3.1 Imagens em Preto e Branco 16
I.3.2 Imagens Coloridas 17
I.4 OUTROS TIPOS DE DISPOSITIVOS DE EXIBIÇAO 18

CAPITULO II EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

II.1 EVOLUÇAO DOS TERMINAIS DE VIDEO 19
II.1.1 Modelo VT05 21
II.1.2 Modelo VT50 22
II.1.3 Modelo VT52 25
II.1.4 Modelo VT55 27
II.1.5 Modelo VT62 28
II.1.6 Modelo VT100 32
II.1.7 Modelo VT125 35
II.1.8 Modelo PDT-11 35
II.1.9 Modelo VT200 37
II.2 SITUAÇAO DOS TERMINAIS NAS ARQUITETURAS CONVENCIONAIS 38
II.3 CRITICA `A EVOLUÇAO DOS TERMINAIS 42
II.4 NOVAS FUNÇOES PARA OS TERMINAIS 44
II.4.1 Interpretação da Linguagem de Controle 44
II.4.2 Editor de Textos 47
II.4.3 Compilação Cruzada 49

CAPITULO III CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.1 ARQUITETURA CONVENCIONAL 52
III.1.1 Subsistemas de Memória, Teclado e Comunicação 54
III.1.2 Subsistema Controlador de Video 56
III.1.2.1 Bloco de Acesso `a Memória 58
III.1.2.2 Bloco de Temporização 61

III.1.2.3 Bloco de Geração do Sinal de Video 62
III.1.2.4 Bloco de Controle e Interface 64
III.2 ARQUTETURA DO TERMINAL CONSTRUIDO 64
III.2.1 Simulação de ADM por Programação 66
III.2.2 Microprocessador Central 69
III.2.3 Demais Subsistemas 71
III.2.4 Caracteristicas do Terminal Construido 72

CAPITULO IV EXPANSOES AO TERMINAL PROPOSTO

IV EXPANSOES AO TERMINAL PROPOSTO 75
IV.1 COMPARAÇAO ENTRE O TERMINAL CONSTRUIDO E O PROPOSTO 76
IV.2 ALTERAÇOES NOS CIRCUITOS 78
IV.2.1 Expansão de Memória RAM e ROM 78
IV.2.2 Expansão Gráfica 79
IV.2.2.1 Opção 1 - Mapeamento por Bits 79
IV.2.2.2 Opção 2 - Blocos Gráficos 86
IV.2.3 Saida para Video Composto 91
IV.2.4 Coprocessador 92
IV.2.5 Conexão Serial para o Teclado 94
IV.2.6 Utilização de Componentes da última Geração 95

CAPITULO V PROGRAMAÇAO BASICA

V.1 OS PROBLEMAS DOS PROGRAMAS DE CONTROLE 98
V.2 DESCRIÇAO DAS ROTINAS 100
V.2.1 Rotina de Inicialização 100
V.2.2 Rotina de Simulação de ADM 103
V.2.2.1 Tratamento das Interrupções 104
V.2.2.2 Simulação da Memória de Tela 111
V.2.2.3 Estrutura da Rotina de Atendimento `a Interrupção 111
V.2.3 Programação do Subsistema de Teclado 113
V.2.4 Estruturação da Memória de Tela 116
V.2.4.1 Posicionamento Fixo 116
V.2.4.2 Linhas Relocáveis 117
V.2.4.3 Registro Composto 122
V.2.4.3.1 Inicialização 123
V.2.5 Conexão com o Teclado 125
V.2.5.1 Rotina de Gerenciamento do Teclado 126
V.2.5.1.1 Procedimento Salva_bcd 131
V.2.6 Controlador Simples 136

CAPITULO VI CONCLUSOES

VI CONCLUSOES 143

APENDICES

A	Sequencias de Controle VT52	147
B	Comandos de contrple do ReGis	149
C	Sequencias de controle do VT100	151
D	Corpo da Rotina Teste_U	155
E	Corpo da Rotina Teste_E	161
F	Teste de Integracao Teclado-Video	171
G	Programa Controlador do Terminal	203
H	Esquemas Eletricos Atuais do Terminal	253
I	Sinais de Controle do CRTC 8275	255

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

INDICE DE FIGURAS

1. Componentes de um CRT 2
2. Padrões de varredura 4
3. Imagens por varredura de rastro 5
4. Arquitetura de um terminal com CRT. 7
5. Componentes de um DVST. 10
6. Formação de imagens caligráficas. 13
7. Arquitetura de um terminal empregando DVST. 15
8. Simulação de intensidade por adjacência . . 17
9. Controle centralizado 38
10. Canais de controle do disco 38
11. Interpretador parcialmente no terminal . . 44
12. Interpretador conscrito ao terminal 45
13. Arquitetura de um sistema controlado por μP 52
14. Detalhamento da arquitetura com μP. 53
15. Técnica de varredura. 55
16. Subsistema de comunicação 55
17. Sinais de controle. 56
18. Arquitetura interna do CRTC 8275. 57
19. Relógio de quadratura 60
20. Memórias Separadas. 60
21. Temporização do sistema 62
22. Geração dos sinais de sincronismo 63
23. Geração do sinal de ADM 69
24. Expansão gráfica. 74
25. Mapeamento da memória gráfica 80
26. Tela alfanumérica 81
27. Circuito ampliador de retraço 82
28. Formas de onda. 82
29. Mapeamento dos blocos na tela 89
30. Correlação blocos - numerais binários . . . 90
31. Mixador de vídeo composto 92
32. Expansão para co-processador. 93
33. Mascaramento das interrupções 106
34. Colocando um CTC. 107
35. Colocando um 8259 107
36. Combinação das interrupções 108
37. Combinação e registro das interrupções. . .108
38. Subsistema de varredura 114
39. Posicionamento fixo 116
40. "Buffer" circular 117
41. Descontinuidade 119

42. Lista circular encadeada. 121
43. Registro composto 122

RESUMO

Neste trabalho projetamos e construimos um terminal de baixo custo para o sistema de automação de laboratórios em desenvolvimento no L.I.E. Para tanto, discutimos as diversas arquiteturas empregadas nos terminais buscando obter subsídios com vistas `a escolha da mais adequada `as aplicações cientificas que são típicas desse microcomputador. Também e' feito um estudo comparativo dos diversos dispositivos empregados como terminais, bem como das funções que desempenham nas disposições convencionais dos centros computacionais.

Procurou-se implementar um terminal de custo extremamente reduzido, mas capaz de ser facilmente expandido pela incorporação de módulos adicionais. Também foi objetivo do trabalho prover um projeto sólido para permitir facilmente uma vasta gama de alterações, variando desde a substituição de alguns componentes ate' a troca de todo o sistema operacional residente.

Para tanto houve grande cuidado na documentação dos esquemas e da programação. O dispositivo construido mostrou-se bastante versátil, confiável e de baixíssimo custo.

"ABSTRACT"

In the present work we compare several devices used as terminals, as well as the roles they play on a conventional computing centre. Their various architectural aspects are also investigated, providing the ground basis for the choice of the most effective arrangement for a terminal biased towards scientific aplications, which are typical of the microcomputer system developed at L.I.E..

We wanted a very low cost terminal that still retained the capability of beeing expanded by the inclusion of additional modules. A solid project, able to handle the most severe alterations, ranging from the replacement of some IC's to the replacement of the entire resident operational system was also wanted. To accomplish these goals a great care was taken on the project documentation of the software as well as of the hardware.

The device that was eventually built proved to be very versatile, reliable and low cost one.

SIGLAS E CONVENÇOES

Tentamos evitar nesta dissertação os barbarismos infelizmente tão comuns neste tipo de redação, empregando sempre que possivel um termo em português ao invés da palavra estrangeira. Nas ocasiões em que tal tradução deu margens `a dúvidas o termo estrangeiro e' aposto entre parêntesis logo a seguir. A maioria das siglas não foi traduzida por fazer parte da documentação original dos manuais consultados. Caso as alterássemos haveria grande desconforto por parte daqueles que lerão esta publicação consultando os manuais originais dos circuitos integrados mencionados. Nos casos possiveis a tradução da sigla foi feita. A seguir são listadas todas as siglas e convenções adotadas no texto:

1. A/D - Analógico/Digital.
2. ADM - Acesso Direto `a Memória.
3. ANSI - "American National Standard Institute", Instituto Nacional Americano de Padrões.
4. ASCII - "American Standard Code for Information Interchange", código padrão norte-americano para troca de informações.
5. bps - Bits por segundo.
6. BS - "Back Space", retrocesso.
7. CAD - "Computer Aided Design", projeto auxiliado por computador.
8. CCLK - "Character CLocK", relógio de caracteres.
9. CI - Circuito Integrado.
10. CR - "Carriage Return", retorno de carro.
11. CRT - "Cathode Ray Tube", tubo de raios catódicos.

12. CRTC - "Cathode Ray Tube Controller", controlador de tubo de raios catódicos.
13. CS - "Chip Select", seleção do integrado.
14. CTC - "Counter Timer Circuit", circuito temporizador e contador.
15. D/A - Digital/Analógico.
16. DACK - "Device ACKnowledge", reconhecimento de dispositivo.
17. DCL - "Digital Control Language", linguagem de controle da Digital.
18. DEC - Digital Equipment Corporation.
19. DFCM - Departamento de Física e Ciência dos Materiais.
20. DIN - "Deutsches Institut für Normung", instituto alemão para normalização.
21. DRQ - "Device ReQuest", requisição de dispositivo.
22. DVST - "Direct View Storage Tube", tubo de armazenamento com visão direta.
23. EEPROM - "Electricaly Erasable Programmable Read Only Memory", memória apenas de leitura programável e apagável eletricamente.
24. EIA - "Electronic Industries Association", associaçãso das indústrias eletrônicas.
25. EPROM - "Erasable Programmable Read Only Memory", memória apenas de leitura programável e apagável.
26. EPROM - "Erasable Programmable Read Only Memory", memória apenas de leitura programável e apagável.
27. E/S - Entrada e Saida.
28. ESC - Escape.
29. FIFO - "First In First Out", o primeiro que entra e' o primeiro que sai.
30. HP - Hewlett Packard Inc.
31. HRTC - "Horizontal ReTraCe", retraço horizontal.
32. IBM - International Business Machines.
33. IFQSC - Instituto de Física e Química de São Carlos.
34. INT - INTerrupção.
35. IO-M/ - "Input Output-Memory/", entrada e saida ou memória/.
36. IORQ - "Input Output ReQuest", requisição para entrada e saida.
37. IRQ - "Interrupt ReQuest", requisição de interrupção.
38. JCL - "Job Control Language", linguagem de controle de trabalhos.
39. LAN - "Local Area Network", rede local.
40. LED - "Light Emiting Diode", diodo emissor de luz.
41. LF - "Line Feed", avanço de linha.
42. LIE - Laboratório de Instrumentação Eletrônica.

43. LPEN - "Light PEN, entrada para caneta de luz.
44. LTEN - "LighT ENable", habilitador do canhão de luz.
45. LSI - "Large Scale Integration", integração em larga escala.
46. MRD - "Memory ReaD", leitura de memória.
47. MRQ - "Memory ReQuest", requisição de memória.
48. Modem - Modulador/Demodulador.
49. MOS - "Metal Oxide Semiconductor", semicondutor a metal e óxido.
50. MT - Memória de Tela.
51. MV - Memória de Vídeo.
52. OS - "Operating System" (S/O da IBM).
53. PAC - Projeto Auxiliado por Computador.
54. PDP - "Programmable Data Processor", processador de dados programável.
55. PDT - "Programmable Data Terminal", terminal de dados programável.
56. PROM - "Programmable Read Only Memory", memória apenas de leitura programável.
57. RAM - "Random Acess Memory", memória de acesso aleatório.
58. RD - "Read", sinal de leitura.
59. RMN - Ressonância Magnética Nuclear.
60. ROM - "Read Only Memory", memória apenas de leitura.
61. ReGis - "Remote Graphics instruction set", conjunto remoto de instruções gráficas.
62. RGB - "Red, Green, Blue"; vermelho, verde, azul.
63. RxC - Receiver Clock, relógio de recepção.
64. S/O ou SO - Sistema Operacional.
65. TAB - "Tabulate", tabulação.
66. TTL - "Transistor-Transistor Logic", lógica transistor-transistor.
67. TV - Televisão.
68. TRAX - (S/O da DEC).
69. TxC - "Transmiter Clock", relógio de transmissão.
70. UCP - Unidade Central de Processamento.
71. USART - "Universal Sincronous and Assincronous Receiver Transmitter", receptor e transmissor síncrono e assíncrono universal.
72. USP - Universidade de São Paulo.
73. UV-EPROM - "Ultra Violet Erasable Programmable Read Only Memory", memória apenas de leitura apagável por ultra violeta.
74. VAX - "Virtual Address eXtension", extensão virtual dos endereços.
75. VM - "Virtual Memory", memória virtual (S/O da IBM).
76. VMS - "Virtual Memory System", sistema de memória virtual (S/O da DEC).

77. VRTC - "Vertical ReTraCe", retraço vertical.
78. VSP - "Video Supression", sinal dde supressão de video.
79. WR - "Write", sinal de escrita.
80. Z80-SIO - "Z80 Serial Input Output", entrada e saida serial do Z80.

Os sinais elétricos ativos em nivel baixo (0 lógico) são seguidos de uma barra, como WR/, RD/, etc. Os numerais estão escritos na base decimal salvo nas listagens e quando explicitamente indicado por um H ou B finais, indicando respectivamente base hexadecimal ou binária.

CARACTERIZAÇÃO DO TRABALHO

Esta dissertação não e' um trabalho isolado, estando profundamente inserida num plano em pleno andamento no Laboratório de Instrumentação Eletrônica. Esse projeto prevê a construção de um sistema de processamento modular e altamente funcional para processamento de sinais, aquisição de dados e automação de laboratórios de pesquisas físicas, tendo ja' gerado vários trabalhos de iniciação cientifica, seis dissertações de mestrado e uma tese de livre-docência, enquanto várias outras teses de mestrado e doutorado encontram-se em andamento.

Ja' foram construídos e encontram-se plenamente operacionais: duas familias de microcomputadores de oito bits para uso geral (com projeto e programação básica totalmente desenvolvidos localmente), um gerador de pulsos programável, um terminal gráfico específico para visualização de imagens tomográficas geradas por RMN, um sistema de aquisição e controle para difração de raios X a baixo ângulo, um subsistema de processamento de FFT ultra-rápido e diversos módulos para automação de espectrografia através de RMN. Várias discussões conceituais sobre aspectos arquitetônicos específicos

também foram encetadas.

O trabalho ora apresentado e' o terceiro e último estágio de uma cadeia de desenvolvimento que remonta aos primórdios do plano de pesquisas do L.I.E. e a um dos primeiros protótipos construídos (um microcomputador CP/M empregando barramento S-100 e microprocessador 8085). Como esse plano prevê a capacitação e domínio do ciclo tecnológico completo, desde o início das atividades do laboratório os equipamentos desenvolvidos são totalmente projetados localmente. O desenvolvimento de um sistema de processamento de dados a partir do papel (ao invés da "adaptação") de um projeto ja' pronto) era uma tarefa bastante ambiciosa e árdua, e, procurando compatibiliza'-la com os recursos e prazos de que dispúnhamos, optamos por projetar e construir os subsistemas de memória, de processamento e de controle de disquetes. O terminal de vídeo alfanumérico foi implementado segundo um projeto proposto pela Intel em [58].

Tal opção, embora necessária, foi bastante distoante da filosofia de desenvolvimento integral "in loco". Assim, estando o microcomputador depurado e funcional pensou-se em projetar o subsistema restante. Nessa época, entretanto, o grupo de Ressonância Magnética Nuclear, empregando os dispositivos construídos no L.I.E. (inclusive o microcomputador), acusava uma necessidade premente de um terminal gráfico para visualização das

imagens espectroscópicas que começavam a ser obtidas. Desenvolveu-se então esse aparelho [28] empregando ainda o terminal da Intel como base.

Esses dois terminais (alfanumérico-Intel e Alfanumérico/Gráfico-[28]) apresentam alguns inconvenientes, o principal dos quais e' o emprego de um projeto relativamente ineficiente para a parte alfanumérica. A programação de controle do módulo alfanumérico publicada em [58] tem vários defeitos e inconsistências, sendo muito pouco apropriada para as modificações certamente necessárias num projeto de pesquisa. Além disso, ambos apresentavam um preço e complexidade muito acima do desejável para um sistema de baixo custo como o microLIE. O emprego de um terminal comercialmente disponivel e' ainda mais descabido: seu preço iguala-se ao do microcomputador inteiro!

A melhor solução e' sem dúvida o projeto e construção de um terminal de video de baixo custo e alta praticidade. Também e' desejável que esse dispositivo seja compativel com os computadores do centro de processamento de dados do Departamento de Fisica e Ciência dos Materiais, do qual o L.I.E faz parte. Este e' o momento apropriado também para uma discussão mais cuidadosa sobre o papel dos terminais num sistema de processamento, quiça' procurando um tipo de relacionamento mais eficaz entre o hospedeiro e seu instrumento de interface com o usuário. E sobretudo, e'

importante que o terminal implementado seja bem documentado, tanto em seus circuitos como em sua programação, e também seja suficientemente flexível e aberto para os futuros experimentos acadêmicos que, espera-se, este trabalho proporcione.

INTRODUÇÃO

Desde seus primórdios ha' aproximadamente 50 anos, a ciência da computação eletrônica vem apresentando notável desenvolvimento. Neste interregno componentes eletrônicos crescentemente aperfeiçoados têm permitido a construção de processadores cada vez mais avançados [1], [2] cuja programação sempre mais elaborada [3], [4], [5] vem por sua vez exigir computadores mais e mais complexos. Essa realimentação positiva - melhores processadores permitindo maiores programas, que por sua vez pedem computadores mais poderosos - ampliou consideravelmente os horizontes das aplicações atuais. Hoje ambicionamos sistemas capazes de reconhecer imagens e sons para aplicações em robótica, sistemas peritos que acumulem experiência e cheguem a conclusões lógicas para pesquisa em inteligência artificial e máquinas que operem com matrizes cuja dimensão se aproxima das centenas de milhares, como as usadas em mapeamento meteorológico. Essas e muitas outras aplicações exigem capacidade de processamento muito maior que a proporcionada pelo estágio atual da tecnologia, e como não sera' possível manter indefinitamente o aumento de capacidade valendo-se apenas do avanço dos componentes, procura-se hoje

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

arquiteturas mais eficientes que, sem tecnologia mais elaborada, permitam construir computadores de maior rendimento. Esta e' a causa primária do interesse atual pelos vários tipos de arquiteturas não convencionais tais como processadores de arranjo, multiprocessadores paralelos, sistemas de fluxo de dados, etc. [6], [7], [8].

Nesse contexto insere-se o estudo dos terminais de vídeo, visto poder esse dispositivo desempenhar um significativo papel no desempenho de tarefas ate' agora relegadas a seus hospedeiros. Uma solução capaz de aumentar significativamente a velocidade de processamento numa vasta gama de aplicações, a concepção de um terminal que execute tarefas complexas sob o comando do hospedeiro exige um novo projeto e novo tipo de relacionamento com o sistema operacional. Por não comprometer a arquitetura Von Neumann, esse avanço e' franqueado `as instalações ja' existentes, possibilitando uma significativa melhora na performance de vários sistemas apenas dotando-os de periféricos mais adequados.

Note-se que os primeiros terminais de vídeo começaram a ser populares ha' não mais que quinze anos. Modelos que definiram padrões históricos (que estudaremos a seguir) foram fabricados ate' 1983. Dada a pouca idade do tema desconhecemos obras que discutam em profundidade a atuação desses periféricos num sistema de processamento de dados. Neste trabalho, partindo do principio de

funcionamento e da evolução arquitetônica dos terminais de video discutimos sua função nos sistemas atuais, criticamos o papel a que esta' hoje relegado e propomos uma nova função. Definimos também as caracteristicas necessárias `a tarefa proposta, verificando qual o "hardware" necessário para cumpri-la. Fazemos um acompanhamento bastante cuidadoso de todos os passos dados na elaboração do terminal cuja construção serviu de base para nossas propostas, bem como dos raciocinios que nortearam as diversas decisões e compromissos que tiveram que ser feitos no decorrer do projeto.

Também investigamos brevemente qual a programação necessária `a integração e bom funcionamento do sistema, bem como documentamos todas as rotinas implementadas. A metodologia adotada para a confecção dos programas foi a de refinamento sucessivo ("top-down"). E' feita uma análise das principais alternativas de programação disponiveis e qual o impacto de seu emprego no desempenho do dispositivo como um todo.

Comparamos a proposta inicial, o realizado e o factivel e finalizamos procurando levantar conclusões a respeito de todo o trabalho, propondo expansões e possiveis melhoramentos, tentando fazer desta dissertação uma base para futuras discussões.

CAPITULO I

CONCEITOS BASICOS

I.0 CONCEITOS BáSICOS

Ha' dois tipos de dispositivos de exibição de imagens tradicionalmente usados em terminais de vídeo, cada um com características e exigências próprias como sera' explicitado:

I.1 O CINESCóPIO COMUM (CRT)

Desenvolvido para o emprego em aparelhos de televisão e de instrumentação, o tubo de raios catódicos (cinescópio) e' constituido das seguintes partes (figura 1):

CONCEITOS BASICOS

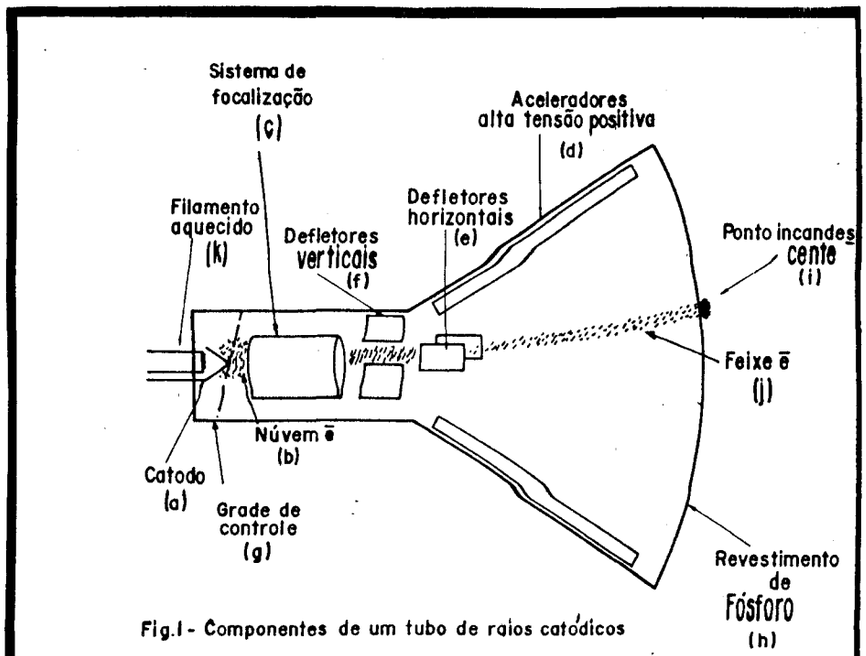


Fig.I - Componentes de um tubo de raios catódicos

I.1.1 Principio De Funcionamento -

O principio de funcionamento do CRT e' o mesmo da válvula termoiônica [9], [10] e [11]: Um filamento aquecido (a) emite elétrons por efeito termoiônico criando um feixe (b) de intensidade controlada por uma grade (g), que e' focalizado por lentes eletro-magnéticas ou eletrostáticas (c), desviado pelos defletores horizontais (e) e verticais (f) para o ponto desejado na tela e fortemente acelerado por um revestimento metálico ligado a uma alta tensão positiva (d). A tela e' um anteparo recoberto de fósforo (h) que emite luz visivel (i) na transição de retorno dos elétrons aos niveis eletrônicos normais após serem excitados com a energia transferida pelo feixe eletrônico (j).

A luminosidade de um ponto decresce exponencialmente com o tempo a partir do instante em que o feixe se move para outro local, tornando necessária a constante reescrita da informação na razão de 30 ou 60 vezes por segundo. E' necessário portanto guardar em algum outro local as informações referentes `a imagem exibida. O custo da memória de acesso direto e rápido empregada nesse armazenamento era, ate' pouco tempo, bastante elevado, dificultando a difusão desse tipo de terminais.

I.1.2 Formação De Imagens Por Varredura De Rastro -

Os dispositivos que empregam CRT's formam imagens por varredura de rastro. Esse sistema e' o empregado nas transmissões comerciais de TV, e também muito utilizado ultimamente em terminais de vídeo alfanuméricos. Sua introdução no campo dos terminais gráficos so' se tornou comercialmente viável depois do barateamento das memórias RAM ocorrido no início da década de 80. Atualmente a maioria dos terminais gráficos emprega varredura de rastro.

Nesse sistema a eletrônica associada ao CRT está sempre deslocando o feixe de elétrons segundo o padrão da figura 2.

Se quisermos acender um ponto nas coordenadas (x,y) cumpre-nos esperar ate' que o feixe passe por essa posição e então liga'-lo. Para se desenhar o clássico exemplo do triângulo de vértices A, B e C teremos:

CONCEITOS BASICOS

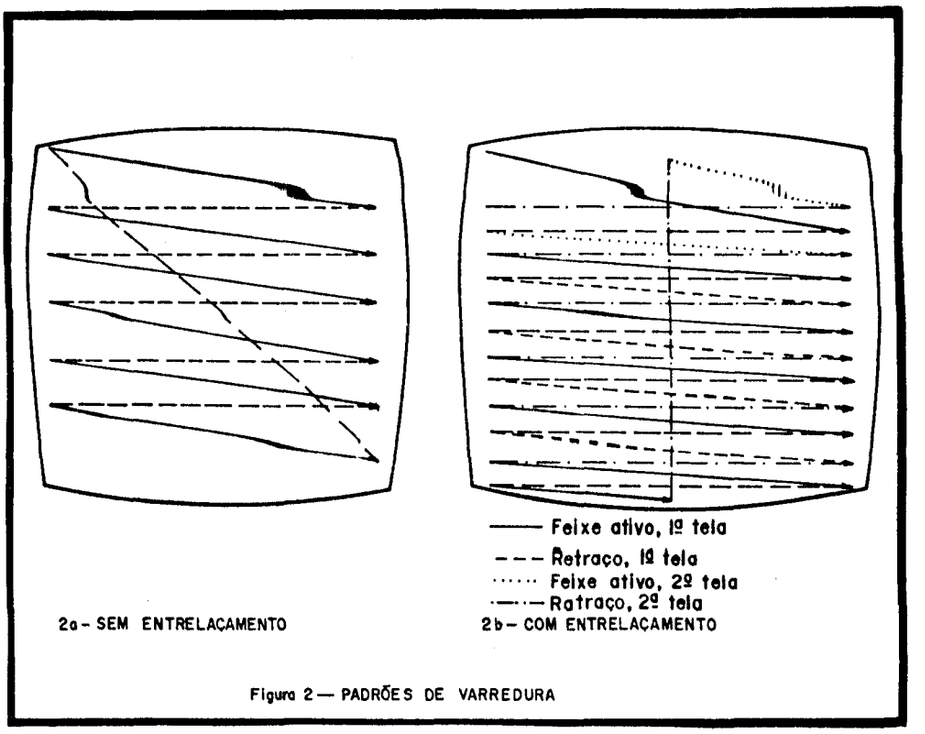


Figura 2 — PADRÕES DE VARREDURA

1. Esperar que o feixe de elétrons passe pelo vértice A (figura 3a).

2. Acionar o feixe, formando um ponto (figura 3b).

3. Esperar que o feixe passe pelo ponto adjacente em direção ao vértice C na linha inferior (figura 3c).

4. Acionar o feixe, formando outro ponto (figura 3d).

5. Assim sucessivamente ate' os vértices B e C (figura 3e).

6. Unir os vértices B e C pela linha que passar mais próxima (figura 3f).

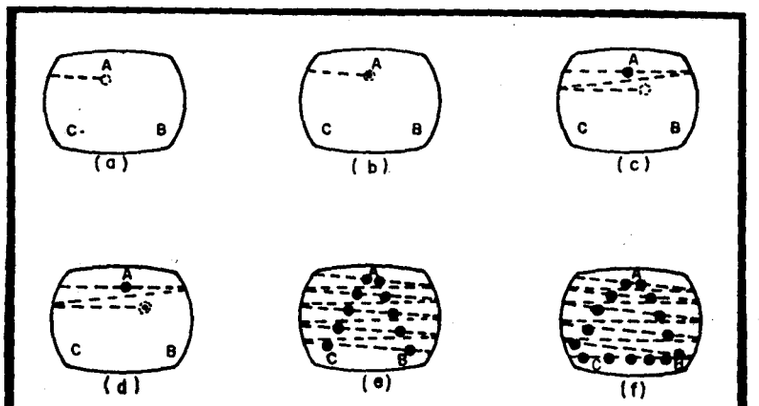


Figura 3 - IMAGENS POR VARREDURA DE RASTRO

Este tipo de processo e' ideal para videos que necessitem de refrescamento devido ao evanescimento da imagem, pois o feixe de elétrons nunca cessa sua movimentação. Também presta-se bastante bem a aplicações com alterações muito rápidas na tela. Seus inconvenientes são a exigência de circuitos que façam a continua movimentação do feixe e de memórias necessárias para informar, a cada posicão do mesmo, a existência ou não de um ponto.

Note-se que para se escrever qualquer dado num terminal desse tipo basta alterar a memória de onde e' tirada a informação para o canhão. Isso torna a modificação de imagens bastante simples. Entretanto, para se formar uma imagem e' necessária a transferência de tantos bits quantos forem os pontos da tela, geralmente um número bastante elevado , tornando a carga de imagens um processo relativamente lento.[1]

Outro inconveniente dos CRT's e' a variação de luminosidade das figuras em relação a sua inclinação. Como os pontos não são igualmente espaçados na direção horizontal e vertical, uma reta com um dado número de pontos desenhada na horizontal parecera' mais brilhante que uma feita na vertical com a mesma densidade de pontos devido `a sua maior concentração. Essa caracteristica e' bastante inconveniente para a maioria das aplicações, razão pela qual várias técnicas de compensação têm que ser adotadas, sobrecarregando os circuitos e a programação desses terminais.

I.1.3 Arquitetura De Um Sistema Com CRT -

A maior dificuldade num sistema baseado em CRT e' a temporização. A varredura de rastro (figura 2) e' feita por dois geradores de função "dente de serra" cuja sincronização controla todo o sistema, como explicitado adiante. O ato de trazer o feixe desligado até o canto superior da tela e' chamado retraço vertical, e o de trazer o feixe até o inicio da linha, retraço horizontal.

Os dados a serem exibidos no video são escritos numa memória de tela de onde são retirados em tempo oportuno

---

1 - No entanto, os dados para um terminal mapeado por bit nem sempre precisam ser em número grande pois o terminal pode ser capaz de responder a comandos de alto nivel enviados pelo hospedeiro. A quantidade de informação transferida quase sempre e' inversamente proporcional `a inteligência do terminal.

pelo circuito controlador de video, sob o comando dos sincronizadores horizontal e vertical. Caso sejam letras, esses dados são traduzidos em código apropriado por uma ROM decodificadora. Se forem imagens, são passados diretamente para um circuito conversor que os serializa, transforma em grandeza analógica e injeta no eixo Z (liga/desliga do feixe de elétrons) do tubo. As letras, depois de codificadas, também são apresentadas ao circuito decodificador (figura 4).

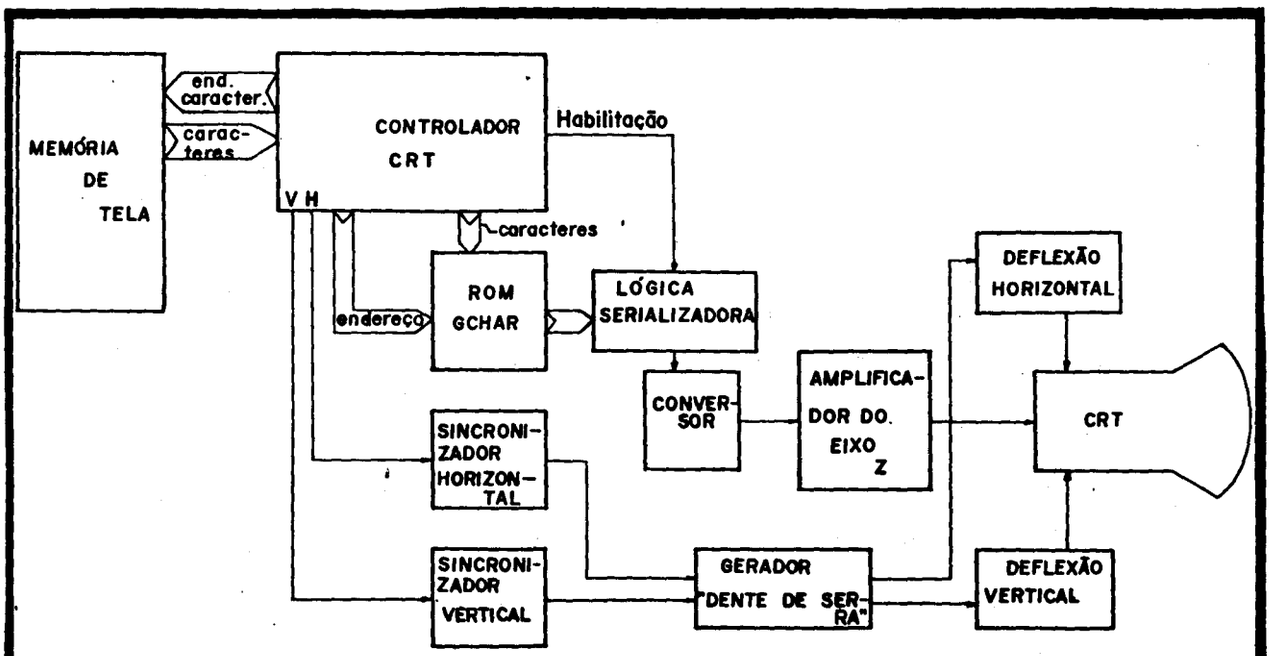


Fig. 4 Arquitetura de um terminal empregando CRT.

Note-se que este e' um processo continuo. Ao chegar ao fim da tela, o controlador simplesmente retorna ao inicio e recomeça um ciclo idêntico. A interface com o processador hospedeiro também e' altamente dependente do grau de inteligência embutida no terminal, mas o modo mais simples de se escrever na tela consiste em alterar

direta e sincronamente a memória gráfica.

O tempo torna-se uma grandeza critica em CRT's. Devido ao decaimento da luminosidade do fósforo, pelo menos 30 vezes por segundo a informação deve ser re-escrita. Se o terminal tiver 512 linhas por 512 colunas (uma resolução média) o tempo máximo gasto numa linha sera' [28]:

$$\frac{1\ s}{30 \times 512\ \text{linhas}} \cong 65\ \mu s/\text{linha} \qquad (1)$$

Entretanto, devemos levar em conta também o tempo de retraço vertical, não despreziveis 1,5 ms tipicamente. Nossa expressão fica então:

$$\frac{\frac{1}{30}\ s - 1{,}5 \times 10^{-3} s}{512\ \text{linhas}} \cong 61\ \mu s/\text{linha} \qquad (2)$$

Em 61 microssegundos precisamos escrever toda a informação de uma linha e trazê-la de volta ao inicio, com o feixe desligado. O tempo de retraço normalmente e' da ordem de 10 microssegundos. Temos então 51 microssegundos para escrever 512 pontos, ou seja, temos:

$$\frac{61\ \mu s - 10\ \mu s}{512\ \text{pontos}} \cong 100\ ns/\text{ponto} \qquad (3)$$

um tempo bastante pequeno. Se o sistema tiver uma resolução de 1024 x 1024, teremos por ponto o tempo de 21 nanossegundos. A eletrônica necessária para chavear o

feixe nessa freqüência está longe de ser trivial.

I.2 O CINESCÓPIO COM MEMÓRIA (DVST)

Um aperfeiçoamento do CRT foi a introdução do tubo de raios catódicos com memória, também chamado tubo de visão direta (em inglês, "storage cathode ray tube" ou "direct view storage tube", DVST) [12], [13], usado nos primeiros terminais gráficos antes mesmo do advento dos terminais de vídeo alfanuméricos.

Algumas aplicações como meteorologia e sistemas militares desde ha' muito necessitavam de saídas gráficas, e antes que os terminais de vídeo se apresentassem como alternativas populares ja' havia subsistemas gráficos para aplicações específicas. Esses subsistemas usavam DVST's devido `a sua alta resolução, facilidade de construção, e principalmente por não exigirem o refrescamento da imagem como os CRT's, permitindo a montagem de imagens extremamente detalhadas.

I.2.1 Princípio De Funcionamento -

Num DVST, uma esparsa nuvem de elétrons esta' continuamente fluindo dos alimentadores (e) para a tela em perpendicularmente `a mesma pela ação de colimadores (a) internos ao tubo. Uma superfície (b) paralela `a tela (chamada superfície de armazenamento) funciona como barreira para essa nuvem, impedindo que a mesma excite o

fósforo. Entretanto, quando o feixe eletrônico atinge um ponto, ele o faz com intensidade suficiente para tornar uma pequena região da superfície de armazenamento ligeiramente positiva (c), fazendo com que mesmo na ausência do feixe, haja naquele local um fluxo de elétrons suficiente para excitar o fósforo e produzir um ponto luminoso (d) (figura 5).

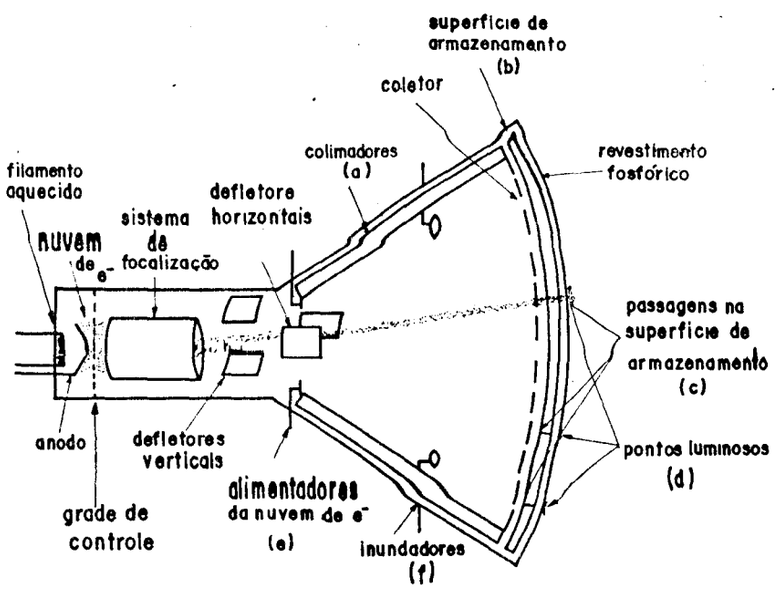


figura 5 - TUBO DE ARMAZENAMENTO (DVST)

O inconveniente desse sistema e' sua incapacidade de apagamento seletivo, pois as regiões do anteparo tornadas positivas não podem ser individualmente alteradas. O apagamento e' efetuado produzindo uma descarga de elétrons na tela inteira pelos inundadores (f). Isso impede sua aplicação em sistemas animados e na maioria dos processos interativos onde haja edição e mapeamento da tela, uma limitação séria. Além disso, os DVST são inerentemente menos brilhantes que os CRT's comuns, porém, como não exigem refrescamento da imagem, os

circuitos necessários ao seu funcionamento são muito mais simples[2].

Tradicionais fábricas de partes de computador como a Tektronix [18], a IBM [14] e Hewlett Packard [15], [16] e [17] empregam DVST's em muitos de seus produtos. Esse fato influencia em diversos aspectos da arquitetura e do protocolo de comunicações de diversos terminais.

I.2.2 Formação De Imagens Por Varredura Aleatória -

A varredura aleatória (também chamada caligráfica, randômica ou vetorizada) usada nos DVST forma a imagem pela concatenação de pontos adjacentes a partir de um de seu extremos. Retornando ao exemplo do triângulo temos:

1. O feixe de elétrons e' trazido para o vértice A (figura 6a).

2. Aciona-se o feixe, formando um ponto (figura 6b).

3. Desloca-se o feixe para o ponto adjacente em direção ao vértice B (figura 6c).

4. Aciona-se o feixe, formando outro ponto (figura

---

2 - Embora esta seja uma idéia bastante difundida (cf. [28]), com o barateamento das memórias RAM e a integração dos controladores de video em CI's de integração em larga escala, os circuitos com CRT são atualmente mais baratos que os que empregam DVST's.

6d).

5. Assim sucessivamente até o vértice B (figura 6e).

6. Desloca-se o feixe para o ponto adjacente em direção ao vértice C (figura 6f).

7. Aciona-se o feixe, formando um ponto (figura 6g).

8. Repetem-se os dois passos anteriores até o vértice C (figura 6h).

9. Desloca-se o feixe em direção ao vértice A (figura 6i).

10. Aciona-se o feixe, formando um ponto (figura 6j).

11. Repetem-se os dois passos anteriores até o vértice A (figura 6k).

E estara' formado o triângulo. Note-se que como os pontos não evanescem, o desenho permanecera' estático na tela, dispensando o armazenamento de informações a seu respeito em RAM como no caso dos CRT's.

A maioria dos terminais caligráficos não exige o nivel de detalhe do exemplo anterior porque a geração de retas lhes e' uma ação primitiva. Para traça'-las basta transferir para o periférico as coordenadas iniciais,

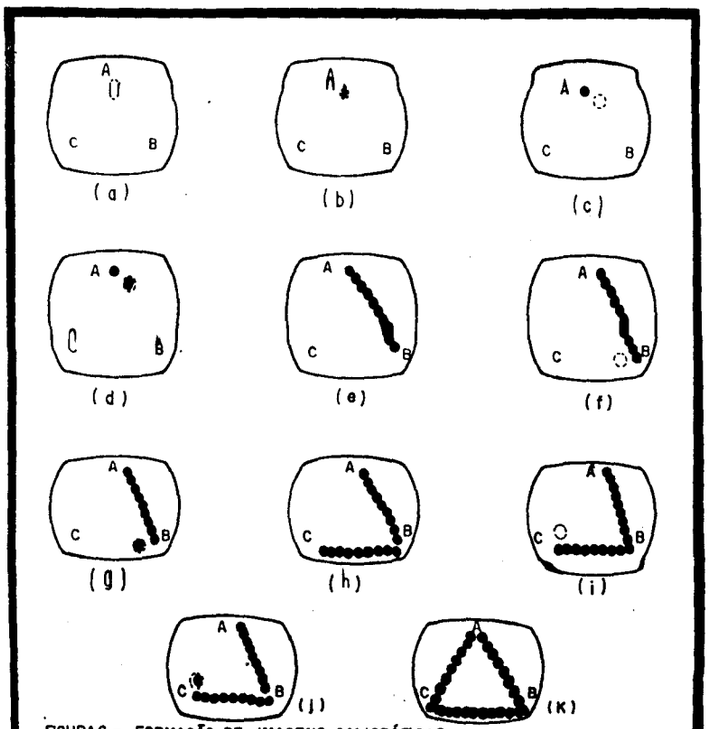


FIGURA 6 – FORMAÇÃO DE IMAGENS CALIGRÁFICAS

finais e a intensidade. Um circuito gerador de rampas (também chamado gerador de vetores) levara' o feixe para os pontos apropriados e acionara' o canhão de elétrons [16], [28]. Para se traçar um triângulo teremos então as seguintes instruções:

1. Traçar uma reta com intensidade X de (a , a) a (b , b).

2. Traçar uma reta com intensidade X de (b , b) a (c , c).

3. Traçar uma reta com intensidade X de (b , b) a (c , c).

(a , a), (b , b) e (c , c) são respectivamente as coordenadas dos pontos A, B e C. E' fácil perceber que a quantidade de informações necessárias para a manipulação de figuras num terminal com DVST e' bastante reduzida[3] . Apenas as coordenadas, a intensidade e o tipo de figura são necessários .

I.2.3 Arquitetura De Um Sistema Com DVST -

Um subsistema tipico de visualização usando DVST e' construido ao redor dos circuitos do gerador de retas (figura 7). Nele estão conectados os 4 registradores de inicio e fim do vetor e o registrador de intensidade. O gerador de retas, a partir desses dados, gera os sinais para os circuitos de deflexão horizontal, deflexão vertical e de acionamento da grade (eixo z - intensidade do feixe de elétrons). Também e' comum um somador que define as coordenadas do traçado relativas `a origem armazenadas em dois registradores, permitindo facilmente o deslocamento da figura para outras regiões da tela.

Dada a dificuldade em escrever caracteres alfanuméricos nesse tipo de tubos, os terminais com DVST

---

3 - Deve-se ressaltar que nem todos os terminais têm geradores de vetores eficientes. Além disso, contornos muito intrincados fazem desaparecer essa vantagem dos DVSTs, podendo exigir transferências de dados maiores que o método de mapeamento por bits, pois o desenho tem que ser fragmentado em um número desmesuradamente grande de segmentos de reta.

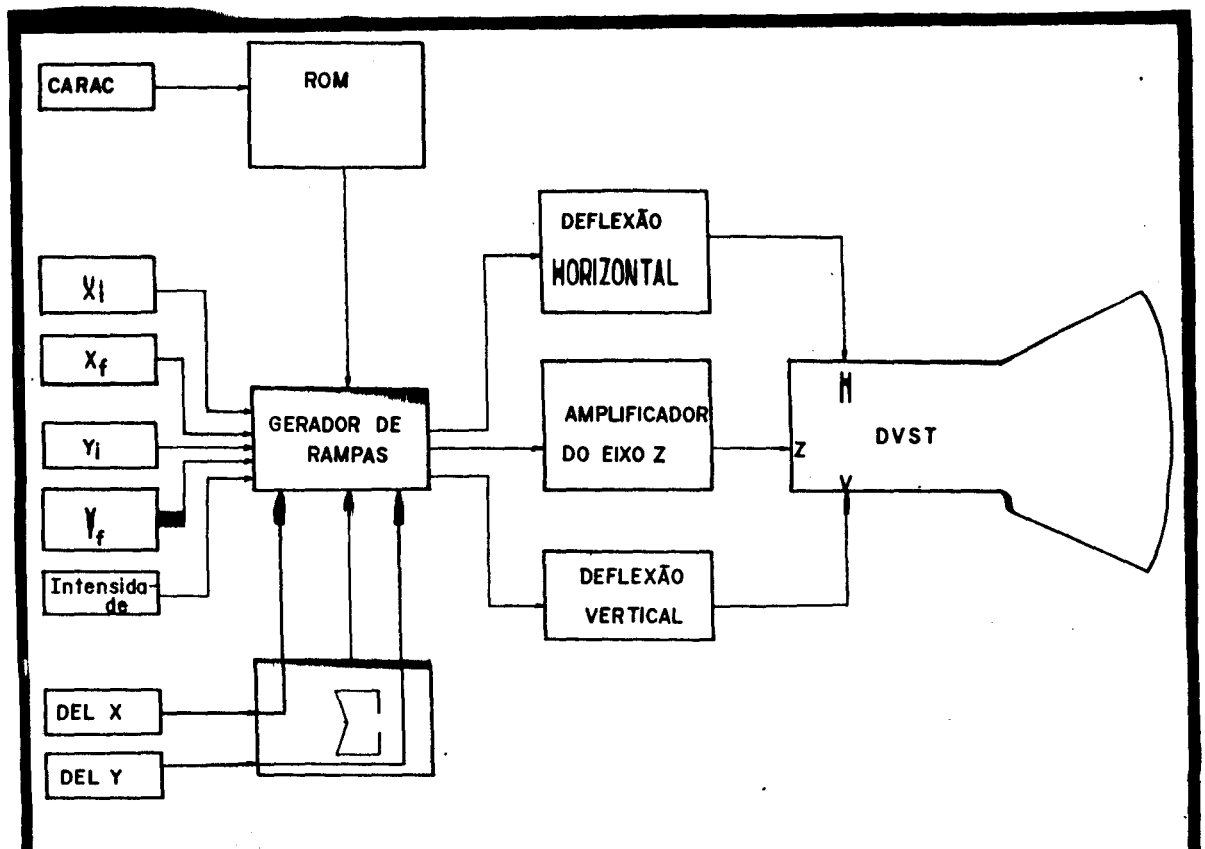


Fig. 7 Arquitetura de um terminal empregando DVST

incorporam também uma ROM geradora de caracteres e circuitos necessários para permitir a escrita de letras e números. E como o gerador de caracteres quase sempre age como controlador de todo o circuito, e' comum que os sinais para limpeza do tubo também sejam por ele gerados.

A comunicação com o computador hospedeiro sempre depende do grau de inteligência do terminal. Em seu modo mais simples, o hospedeiro escreve nos registros x , x , y , y , Int, Desl_X, Desl_Y e CARAC os dados necessários.

I.3 COR E INTENSIDADE

CONCEITOS BASICOS

I.3.1 Imagens Em Preto E Branco -

Os CRT's se prestam muito bem para para a exibição de imagens continuas tais como fotos e imagens, enquanto os DVST's são melhor empregados em contornos complexos, dada sua maior resolução. Em imagens num terminal monocromático (ou seja, com fósforo de apenas uma côr), a questão de quantas tonalidades de cinza são necessárias para uma sensação de continuidade e' complexa, envolvendo o grau de sensibilidade do olho humano, grandeza nem sempre facilmente aferivel, nem tampouco homogênea. Ate' com 16 diferentes graduações de cinza e' detectável a mudança de um ponto para outro, gerando contornos indesejàveis. A linha divisória situa-se em torno dos 32 niveis, considerada por alguns como aceitàvel e por outros como ainda passivel de critica [29]. 64 niveis, espaçados logaritmicamente são aceitos pela quase totalidade dos autores como um excelente padrão [30]. Médicos treinados, entretanto, conseguem distinguir cerca de 600 niveis em imagens radiológicas [62].

Para terminais incapazes de modular a tensão no eixo Z, criando niveis de intensidade em um único ponto, a solução e' valer-se da tendência de fusão do olho humano. Cada pedaço de informação não será mais exibido num único ponto, mas num conjunto de quatro pontos adjacentes [29], [30].

Essa aproximação divide por dois a resolução em cada eixo, mas permite que, em um sistema onde o ponto somente pode estar ligado ou desligado, haja, por exemplo, cinco niveis de intensidade distintos, a saber (figura 8):

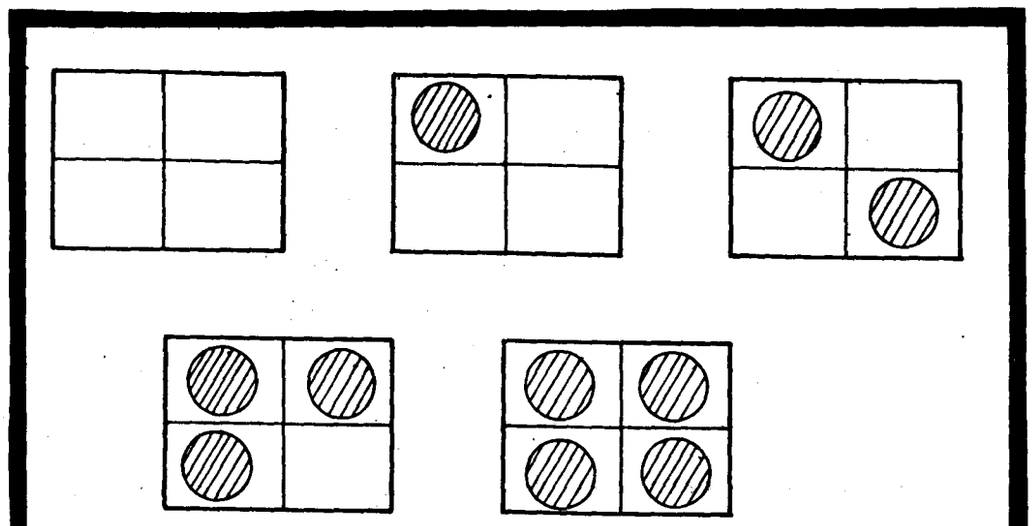


Fig. 8 Simulação de intensidade por adjacência

I.3.2 Imagens Coloridas -

A análise de cor infelizmente não pode ser feita tão rapidamente. Verificamos que, em imagens acromáticas, 64 niveis são suficientes, ou seja, 6 bits por ponto. Já em imagens coloridas, os estudos de [31] a [35] mostram que o padrão aceitável para a recepção de uma imagem policromática envolve três cores fundamentais (são adotados o vermelho, o verde e o azul formando o padrão Red Green Blue - RGB).

CONCEITOS BASICOS

Supondo que o mesmo número de intensidades necessárias a uma imagem em preto e branco seja necessário para cada cor (uma ilação não necessariamente válida) teremos 18 bits por ponto. Os sistemas comerciais de boa resolução empregam 24 bits por ponto, enquanto os de alta resolução usam até 10 bits por cor, num total de 30 bits por ponto [36], [40].

I.4 OUTROS TIPOS DE DISPOSITIVOS DE EXIBIÇÃO

Diversos outros tipos de dispositivos de exibição de dados tem sido desenvolvidos, tais como painéis a gás (plasma), matrizes de diodos emissores de luz, matrizes de cristal liquido, vários tipos de CRT's coloridos, etc [18] a [27]. O potencial dessas novas tecnologias e' muito grande; seu estudo, entretanto, foge do escopo deste trabalho. Consideramos o CRT e o DVST apenas pelo impacto que têm na arquitetura dos terminais que os incorporam.

CAPITULO II

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

II.0 EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

II.1 EVOLUÇÃO DOS TERMINAIS DE VIDEO

A evolução dos terminais de computador esta' intimamente associada ao desenvolvimento dos sistemas operacionais. Durante as duas primeiras gerações a interação usuário-processador foi essencialmente através de lotes de tarefas [43] e [44]; o único terminal interativo, quando existia, era destinado ao operador, sendo a E/S para os usuários composta por periféricos tipo leitora/perfuradora de cartões ou de fita de papel e impressora. Somente com a introdução dos S.O. baseados em compartilhamento de tempo ("time sharing") passou o terminal interativo a ser empregado em larga escala [43].

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

Os primeiros dispositivos empregados para E/S interativa foram as máquinas de escrever modificadas então usadas para comunicação por telex ("flexowriters"). Os teletipos entretanto apresentam sérias limitações de velocidade, nivel de ruido e fadiga mecânica, tendo sido substituidos pelos terminais de video.

Não e' nosso intento descrever toda a história desses periféricos, porém uma breve investigação sobre seu desenvolvimento permite conclusões bastante singulares. Para acompanhar a evolução dos terminais ateremo-nos apenas aos desenvolvidos por um particular fabricante de computadores (DEC) pelas seguintes razões:

1. Foi uma das primeiras empresas a empregar maciçamente terminais de video em sua linha de computadores. Foi a primeira a fabricar um computador cuja console era um terminal de video [51].

2. As seqüências de controle do VT52, um dos seus modelos, foram o primeiro padrão de linguagem de controle para terminais. Adotada por vários fabricantes, ainda e' empregada na industria de microcomputadores, como por exemplo, nos microcomputadores Intellec (Intel) [46], CP600 e CP700 (Prológica) [47].

3. As seqüências de controle do modelo VT100 (que sucedeu ao VT52) tiveram tamanho impacto que foram adotadas como padrão para todos os terminais assincronos como norma ANSI X-3.64.

4. Temos acesso a variada literatura sobre esses dispositivos.

II.1.1 Modelo VT05 -

Um dos primeiros terminais de video disponiveis no mercado, o VT05 possuia tela de 20 x 72 caracteres e taxa máxima de comunicação de 2.400 bps pelo padrão EIA de circuito de 20 mA. O comprimento da palavra de comunicação era fixado em 7 bits havendo dois tipos de terminal: um sem paridade e um com paridade par. Apenas letras maiúsculas podiam ser representadas na tela, num total de 63 simbolos distintos.

Os dados recebidos pelo teclado eram passados diretamente ao hospedeiro sem qualquer pre'-processamento. Assim, caracteres especiais como CR (ASCII 13) ou BS (ASCII 8) precisavam ser tratados pelo computador, empregando para isso a caracteristica mais avançada do VT05: a capacidade de posicionar o cursor em qualquer posição da tela sob o comando do processador hospedeiro. Nenhum outro protocolo de comunicação era suportado.

II.1.2 Modelo VT50 -

Provavelmente um dos primeiros terminais a incorporar um microprocessador, o modelo VT50 representou um significativo avanço por introduzir o conceito de comandos como extensão do código ASCII.

Várias ações primitivas a um terminal são descritas pelo código ASCII, como CR, LF, TAB, etc. O VT50 interpretava os seguintes caracteres de comando desse código:

1. BELL (ASCII 7) - Toca a campainha interna do terminal.

2. BS (Back Space ASCII 8) - Volta o cursor uma posição.

3. TAB (Tabulate, ASCII 9) - Avança o cursor ate' a próxima marca de tabulação, fixada de 8 em 8 colunas a partir da coluna 9.

4. LF (Line Feed, ASCII 10) - Avança o cursor para a linha inferior. Se na última linha, desloca toda a tela uma linha para cima.

5. CR (Cariage Return, ASCII 13) - Traz o cursor para o inicio da linha.

Entretanto, ha' várias outras ações inerentes apenas aos terminais de video para as quais não existe uma representação ASCII, como por exemplo avançar o cursor uma posição `a direita ou apagar a tela. Ao invés de empregar códigos ASCII definidos para outras ações ou extender unilateralmente o padrão optou-se por definir um protocolo baseado em modos de comando: Ao receber um código ESC (Escape, ASCII 27) o terminal entra em modo "escape", do qual so' sai após o recebimento de outro ESC. Enquanto estiver no modo "escape", algumas letras são interpretadas como comandos especiais, a saber:

1. A - Mover o cursor uma linha para cima ou ignorar se ja' esta' na primeira linha da tela.

2. C - Mover o cursor uma coluna para a direita ou ignorar se ja' esta' na última coluna da tela.

3. H - Trazer o cursor para a primeira linha e primeira coluna (canto superior esquerdo da tela).

4. J - Apagar todos os caracteres desde a posição do cursor ate' o fim da tela.

5. K - Apagar todos os caracteres desde a posição do cursor ate' o fim da linha (margem direita).

6. Z - Solicitar ao terminal que se identifique (O terminal envia um código que permite ao hospedeiro identificar qual modelo de terminal esta a ele conectado).

7. [- Ligar modo de travamento da tela (sem deslocamento ao atingir a última linha).

8. ? - Desligar o modo travamento (o terminal desloca toda a tela uma linha para cima, perdendo a primeira ao atingir a última linha).

Para manter a compatibilidade com programas escritos para E/S através de cartões perfurados a tela foi extendida para 80 colunas, embora para tanto fosse necessário reduzir o número de linhas para 12. O VT50 era capaz de comunicar-se em até 9.600 bps pelo padrão RS 232c ou, opcionalmente, circuito de 20 mA. A palavra de transmissão tinha obrigatoriamente 7 bits, podendo ou não ser incluido um oitavo bit de paridade par, opção definida por uma chave no terminal. O gerador de caracteres continha apenas letras maiúsculas, num total de 64 símbolos. O VT50 não era capaz de posicionar diretamente o cursor em qualquer local da tela, deslocando-o através de comandos ESC A, ESC C, BS ou LF.

II.1.3 Modelo VT52 -

Introduzido em 1975 o VT52 teve um grande impacto, vindo a ter muitas de suas caracteristicas consideradas como padrões para terminais. Um melhoramento do modelo anterior, este terminal interpreta uma seqüência de controle iniciada por ESC cujos comandos definem uma interface padronizada entre o hospedeiro e o terminal permitindo:

1. Movimentar o cursor uma posição para cima, baixo, direita ou esquerda, sem deslocar a tela ou trazê-lo até o canto superior esquerdo.

2. Apagar os caracteres desde a posição do cursor ate' o fim da linha ou da tela.

3. Identificar qual o tipo do terminal conectado ao hospedeiro.

4. Deslocar a tela uma linha para cima ou para baixo ("scroll").

5. Escolher os códigos gerados por algumas teclas entre dois possiveis conjuntos de caracteres mutuamente exclusivos.

6. Posicionar o cursor em qualquer local da tela.

Quatorze diferentes seqüências de controle compõem o repertório do VT52. Mais importante que o conjunto de comandos propriamente dito foi a definição de uma linguagem de comando consistente, abandonando o conceito de modo normal e modo "escape" do VT50. Em qualquer instância uma seqüência ESC A faz o cursor subir uma linha, uma seqüência ESC J apaga a tela, e assim por diante.

Essa linguagem continua a ser empregada por muitos equipamentos digitais, sendo interpretada por uma série de terminais e microcomputadores de outros fabricantes. No Brasil citamos entre outros, o TDA, Scopus TVA 1052, Scopus TVA 2052 e o CP600. O conceito de seqüência de controle iniciada por ESC foi preservado na norma ANSI, embora a linguagem tenha sido alterada.

Os comandos de um byte como CR, LF, TAB, BELL e BS também são interpretados. Um protocolo de comunicação baseado em XON (ASCII 17) e XOFF (ASCII 19) permite que tanto o terminal como o hospedeiro possam interromper e reiniciar o fluxo de dados entre si.

A tela foi extendida para 24 x 80 caracteres maiúsculos e minúsculos, a comunicação pode atingir 9.600 bps com palavras de 5, 6, 7 ou 8 bits, paridade par, impar ou sem paridade (opções selecionáveis por chaves) e as velocidades de transmissão e recepção foram totalmente separadas. Algumas caracteristicas ergonômicas para

facilitar o uso intensivo do dispositivo foram pela primeira vez incorporadas a um terminal, como por exemplo teclado numérico separado para a entrada de grandes volumes de dados numéricos, tecla de repetição, confirmação sonora do pressionamento de teclas ("keyclick") e um algoritmo especial que habilita a recepção de uma tecla pressionada sem que a anterior tenha sido ainda liberada ("rollover"). Encontram-se no apêndice A as seqüências de controle da linguagem do VT52.

II.1.4 Modelo VT55 -

O modelo VT55 acrescenta `as caracteristicas do VT52 uma capacidade gráfica limitada, podendo desenhar histogramas e funções simples. Para cada um dos 512 pontos no eixo X, o VT55 pode representar 2 pontos no eixo Y. Assim, a curva mais complexa que pode ser desenhada e' um circulo. Não e' possivel exibir uma reta interceptando um circulo em sua tela.

O gerador de vetores do VT55 e' por "hardware", permitindo a construção de histogramas com vários tipos de sombreados, hachuras e intensidades. E' um produto da mesma geração do VT52, tendo sido lançado praticante na mesma época, com arquitetura semelhante e componentes da mesma tecnologia.

II.1.5 Modelo VT62 -

O modelo VT62 implementa o primeiro terminal de modo bloco deste fabricante, sendo a primeira (e malograda) tentativa de transferir para os periféricos algumas das responsabilidades do hospedeiro.

Num terminal de modo bloco os dados entrados no teclado não são enviados ao hospedeiro logo que digitados, mas sim em conjuntos de caracteres (telas ou blocos). O computador central define primeiramente o formato da tela especificando campos numéricos, campos que necessariamente devem ser preenchidos e mensagens padrão de erro. Após as definições, o terminal armazena localmente os dados que recebe, verificando se os mesmos preenchem os requisitos necessários e emitindo as mensagens de erro que julgar apropriadas sem a intervenção do hospedeiro. Ao término da tela, todo o bloco e' transmitido, podendo a operação ser repetida com apenas um comando do computador central.

A idéia básica e' atraente: sendo toda validação dos dados feita localmente, os programas aplicativos ficam muito mais simples por dispensarem estes testes sobre as entradas fornecidas, e o hospedeiro com muitos terminais fica bastante aliviado porque essa validação e' feita paralelamente em cada um dos periféricos.

Para facilitar a conexão de vários terminais a um único computador, o VT62 suporta um protocolo de comunicação bastante elaborado do tipo "multidrop": vários dispositivos partilham uma linha comum, as mensagens do processador central somente são reconhecidas se forem precedidas pelo endereço do terminal, definivel por chaves internas ao periférico. Essa arquitetura voltou recentemente a ser empregada nas redes locais de processadores (LAN - "Local Area Networks") tipo Ethernet e similares [52]. Outras capacidades do VT62 incluem:

1. Tela logicamente dividida em duas partes: área de exibição e área de formulários, com tamanho definivel por programação.

2. Diferentes atributos para cada área de tela, alguns pre'-definidos (área de exibição protegida, área de formulários em video reverso, etc.) e alguns definiveis por programação.

3. Primitivas para a confecção de menus.

4. Mensagens padrão de erro definiveis pelo hospedeiro, permitindo que o terminal possa dialogar com o usuário sem a intervenção do processador central.

5. Várias primitivas para a definição de formulários:

    1. Aceitar apenas entradas numéricas.
    2. Aceitar apenas entradas numéricas com sinal.
    3. Aceitar apenas entradas alfanuméricas.
    4. Definir campo cujos caracteres devem ser ecoados.
    5. Definir campo cujos caracteres devem ser invisiveis.
    6. Definir campo protegido.
    7. Definir campo onde pelo menos um caracter deve ser colocado.
    8. Definir campo que precisa ser preenchido completamente.
    9. Marginar `a esquerda.
    10. Marginar `a direita.
    11. Definir campo com vídeo reverso.
    12. Definir campo protegido.

6. Teclas programáveis pelo hospedeiro, fazendo com que entradas completas possam ser efetuadas com apenas um toque.

7. Autoteste diagnóstico.

8. Porto para impressora.

A filosofia de trabalho deste terminal e' um passo na direção da proposta desta dissertação. Entretanto, o VT62 padece de um defeito capital, a nosso ver a causa do seu malogro comercial: Não se trata de um terminal convencional, sendo incapaz de comunicar-se fora do modo bloco. Isso exige um sistema operacional exclusivo e excludente (TRAX) não havendo conexão possivel entre o VT62 e os demais sistemas da própria Digital. Embora alguns bancos norte-americanos tenham chegado a

adota'-lo, o projeto foi logo abandonado devido a seu fraco desempenho comercial.

Este episódio parece ter fornecido lições valiosas: O modelo subseqüente, talvez o mais bem sucedido da série, embora apresente uma linguagem de comando bem mais elaborada e complexa, inclui um modo de compatibilidade com o VT52 permitindo que a programação anteriormente feita possa ser ainda utilizada.

II.1.6 Modelo VT100 -.

Produzido a partir de 1977, o modelo VT100 tem um conjunto de seqüências de controle que terminou por transformar-se no padrão adotado pela ANSI. Esses comandos permitem:

1. Movimentar o cursor qualquer número de posições para cima, baixo, direita ou esquerda, sem deslocar a tela.
2. Trazer o cursor até o inicio da tela.
3. Apagar os caracteres desde a posição do cursor ate' o fim da linha.
4. Apagar os caracteres desde a posição do cursor ate' o inicio da linha.
5. Apagar os caracteres desde a posição do cursor ate' o fim da tela.
6. Apagar os caracteres desde a posição do cursor ate' o inicio da tela.
7. Identificar qual o modelo de terminal conectado ao hospedeiro.
8. Deslocar a tela uma linha para cima ou para baixo ("scroll").

9. Trancar um número de linhas na tela, protegendo-as do deslocamento ("scroll") para cima ou para baixo.

10. Algumas teclas com dois significados mutuamente exclusivos selecionáveis por comando.

11. Definir linhas com caracteres de largura dupla.

12. Definir linhas com caracteres de altura dupla.

13. Definir o formato do cursor (bloco ou sublinhado, piscante ou fixo).

14. Definir atributos especiais caracter a caracter (vídeo reverso, sublinhado, intensificado, piscante).

15. Ligar e desligar indicadores luminosos (LED's) do painel.

16. Efetuar testes de confiabilidade do terminal.

17. Chavear entre dois conjuntos de seqüências de controle, ANSI e um modo compativel com VT52.

18. Definir algumas caracteristicas do terminal (80/132 colunas, barulho das teclas, repetição automática das teclas mantidas pressionadas, posição das marcas de tabulação, etc.) .

19. Fazer o deslocamento de tela de modo suave para permitir a leitura continua ("smooth scroll")[4].

Uma das caracteristicas mais populares desse terminal e' o tamanho intercambiável de tela: 24 x 80 ou 24x132 caracteres maiúsculos e minúsculos, além de simbolos matemáticos e caracteres semi-gráficos. A velocidade de comunicação pode atingir 9.600 bps com palavras de 5, 6, 7 ou 8 bits, paridade par, impar ou sem paridade, sendo essas opções definiveis por programação e

---

4 - A seqüência completa dos comandos ANSI X3.64 esta' no apêndice C.

armazenadas em memória não volátil. As velocidades de transmissão e recepção são totalmente separadas, ha' um teclado numérico separado e um mecanismo de repetição controlado por programação que repete automaticamente uma tecla mantida pressionada por mais de meio segundo.

As caracteristicas básicas do VT100 foram estendidas a toda uma familia de terminais como o VT101 (mais barato, sem possibilidade de expansão), VT102 (com conexão para modem, interface malha de 20 mA e expansão de memória), VT105 (compativel com VT55), terminais com encapsulamento resistente `a corrosão, etc. Esses dispositivos não serão descritos em pormenores.

II.1.7 Modelo VT125 -

O modelo VT125 foi lançado para suprir uma grave falha do VT100: incapacidade gráfica. E' um terminal funcionalmente idêntico ao VT100 com um matriz gráfica de 230 linhas por 400 colunas, mapeado por bit (tecnologia de varredura de rastro).

Este terminal incorpora tecnologia apenas ligeiramente mais avançada que a do VT100, podendo ser considerado apenas um apêndice do mesmo. Não ha' nenhuma alteração na arquitetura, nem nos principios básicos de funcionamento. A única diferença na linguagem de comando e' a inclusão do protolocolo ReGis [44] para transferência de imagens para a tela gráfica. Os

comandos Regis estão descritos no apêndice B.

II.1.8 Modelo PDT-11 -

O PDT-11 foi a mais séria tentativa de prover um novo papel para os terminais num centro de processamento de dados. Trata-se de um VT100 que incorpora em seu interior um microcomputador LSI-11 com sistema operacional residente de tempo real. Três diferentes modelos dessa familia foram produzidos: sem memória de massa (PDT-11/130), com cartuchos de fitas magnéticas (PDT-11/200) e com disquetes de 8''(PDT-11/350).

Foi a reencarnação do modelo VT62 com suas principais deficiências corrigidas. O PDT-11, além de ser um terminal convencional se a aplicação assim o exigisse, podia receber do hospedeiro um programa e executa'-lo localmente. Como o conjunto de instruções do hospedeiro (PDP-11 ou VAX na maioria dos casos) era o mesmo do terminal, várias situações interessantes poderiam surgir, pois seria possivel executar o mesmo programa no hospedeiro ou no terminal. Essa possibilidade permitiria que boa parte da depuração de um programa fosse feita fora de linha e sem sobrecarregar o hospedeiro, embora não haja evidências de que essa capacidade tenha sido utilizada.

Não obstante todas essas vantagens o PDT-11 também fracassou, tendo sido retirado de fabricação por volta de 1982. As razões parecem ter sido estritamente comerciais, pois esse terminal era bastante caro. A tecnologia empregada para a construção do microcomputador LSI-11 ja' era obsoleta quando o mesmo foi incorporado ao PDT-11. Circuitos ultrapassados são necessariamente mais caros e menos competitivos; um microcomputador baseado no microprocessador 68020 com 32 bits e gerenciamento de memória virtual custa atualmente menos que a metade de um LSI-11 do tipo que equipava o PDT-11, com UCP de 16 bits e limitado a 64k bytes.

Cremos que se este periférico tivesse um processador com melhor relação custo/beneficio seria hoje o padrão para terminais.

II.1.9 Modelo VT200 -

A última geração de terminais Digital e' a familia VT200, com três modelos: Um terminal alfanumérico, um terminal gráfico e um gráfico colorido. Embora o "design" tenha sofrido mudanças radicais com um teclado maior, mais baixo (de acordo com a norma DIN) e com diversas outras caracteristicas ergonômicas, este terminal e' funcionalmente muito similar ao VT100.

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

O projeto foi radicalmente alterado, mas apenas para aproveitar os circuitos integrados de última geração, com maior densidade e recursos. A arquitetura continua muito parecida `a de seu famoso antecessor. Nenhum grande avanço foi proporcionado por esses terminais. Suas principais caracteristicas são:

1. Emulação dos terminais VT100 e VT52.

2. Capacidade de auto-teste estendida.

3. Teclas redefiniveis por programação.

4. Gerador de caracteres redefinivel por programação.

5. Emulação do protocolo do Tektronix 4044[5]

6. Interpretação do protocolo ReGis[5]

7. Definição das caracteristicas do terminal (velocidade de transmissão, protocolo interpretado, tipo de terminal emulado, etc.) através de menus e armazenamento das mesmas em memória não volátil.

II.2 SITUAçAO DOS TERMINAIS NAS ARQUITETURAS CONVENCIONAIS.

---

5 - Para os terminais gráficos VT240 e VT241.

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

Os primeiros computadores foram construidos com grandes dificuldades. A tecnologia disponivel somente permitia circuitos caros, pouco confiáveis e de grande dissipação, criando tantos problemas de alimentação, consumo e manutenção que tornava os projetistas extremamente preocupados com o número de componentes. Para reduzir a complexidade dos circuitos, ou talvez devido `a ausência de mecanismos adequados para o gerenciamento de multiprogramação) uma única unidade exercia controle sobre todas as sub-unidades.

Era a unidade central de processamento (UCP) quem controlava as transferências de e para os discos, quem calculava o endereço dos arquivos, quem gerava os pulsos necessários para o subsistema de memória de massa, quem monitorava e respondia aos sinais gerados em todas as linhas de periféricos, etc. Com isso, boa parte do tempo total do processador era empregado em gerenciamento (figura 9).

Com o barateamento e aumento de confiabilidade dos componentes os subsistemas periféricos foram ficando cada vez mais inteligentes e autônomos, liberando a UCP para outras tarefas. O primeiro desses dispositivos foi o canal de acesso direto `a memória (ADM), que permite a transferência de dados diretamente dos periféricos para o subsistema de disco sem a intervenção da UCP, como mostra a figura seguinte.

Unidade de acesso direto `a memória

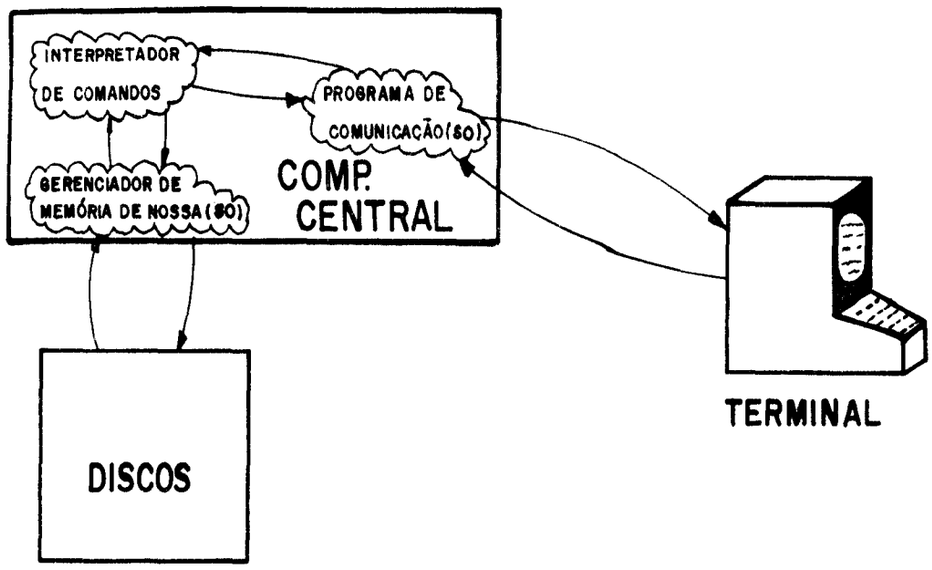


Figura 9 — CONTROLE CENTRALIZADO

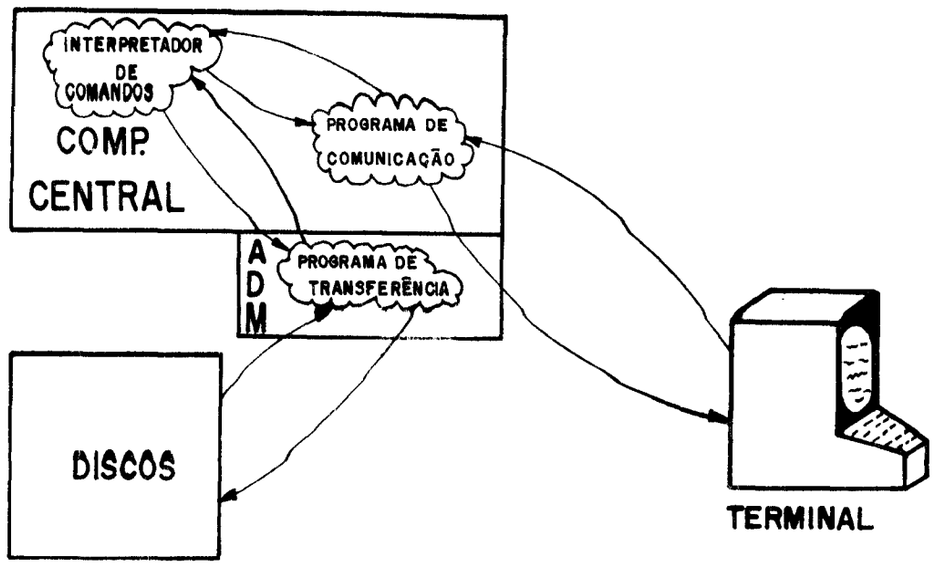


Figura 10 — CANAL DE CONTROLE DO DISCO

Em seguida foram aperfeiçoados os periféricos de E/S (com a introdução dos processadores dedicados a E/S chamados canais de E/S na terminologia da IBM, seu primeiro fabricante) e os subsistemas de memória de massa. Atualmente a maioria das unidades de armazenamento de massa e' bastante autônoma, exigindo pouquissima atenção da UCP. Recentemente foram introduzidas unidades que fazem todo o gerenciamento da

transferência e controle do posicionamento dos arquivos localmente [54].

Embora os dispositivos de E/S tenham sido aperfeiçoados, ainda requerem excessiva atenção da UCP. Essa carga e' em grande parte devida `a arquitetura ineficiente dos terminais de video empregados para E/S. Como a capacidade de processamento local desses periféricos e' limitadissima, as UCP's centralizam tarefas facilmente executàveis localmente. Um exemplo tipico e' o processador VAX 11 [51], [55], bastante popular: Os circuitos ("hardware") e a programação ("software) que compõem seu sistema de E/S são baseados em interrupção, ou seja, ao se editar um arquivo cada caracter pressionado pelo usuàrio interrompe o processador. Na maioria dos sistemas, uma vasta gama de tarefas perfeitamente executàveis por microprocessadores pouco poderosos são efetuadas por caras e sobrecarregadas UCP's. Não e' razoàvel empregar uma màquina de 64 bits capaz de operações em ponto flutuante e sofisticados mecanismos de endereçamento e interrupção para interpretar comandos triviais de uma linguagem de controle ou efetuar uma compilação simples. No entanto, como os dispositivos de E/S não realizam essas tarefas o trabalho fica a cargo do processador central na maioria dos casos.

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

O fraco desempenho dos terminais afeta muito profundamente o desempenho dos hospedeiros. Se executassem localmente todas as tarefas a seu alcance certamente o hospedeiro ficaria aliviado de muitos trabalhos triviais, diminuindo a parcela do tempo dedicada a eles e `a troca de contexto dos processos ativos. Cumpre notar que uma porcentagem bastante grande da capacidade de processamento de uma UCP multiprogramada e' gasta no gerenciamento dos processos. No caso da interpretação de comandos da linguagem de controle (JCL) o esforço computacional para ativação do processo e' muito maior que o necessàrio `a execução do processo propriamente dito. A sobrecarga ("overhead") que um sistema operacional multiusuàrio impõe ao processador e' quase sempre muito grande, crescendo em função direta do número de processos independentes que precisa controlar. Transferindo alguns deles para os periféricos (como fazem alguns sistemas de grande porte através dos canais de E/S) ganham-se não somente os recursos que esses processos empregariam mas, acima de tudo, o tempo de gerenciamento a eles associado. Cumpre notar que a sobrecarga ("overhead") de o sistema e' um emprego de recursos sem efetuar nenhum trabalho útil para os usuàrios, devendo ser minimizada ao máximo.

II.3 CRITICA 'A EVOLUçãO DOS TERMINAIS

Uma comparação critica entre os primeiros terminais disponiveis e os atuais revela uma evolução bastante lenta em relação aos demais componentes de um sistema de processamento de dados. A grosso modo três gerações de terminais de video ja' passaram, com as seguintes caracteristicas:

1. Primeira Geração: Tela de tamanho reduzido, controle fixo por circuitos, sem linguagem de controle, ausencia de caracteres de controle, caracteristicas de comunicação fixas.

2. Segunda Geração: Tela de tamanho usual, controle por microprocessador, linguagem de controle primitiva, interpretação da maioria dos caracteres de controle do código ASCII, comunicação flexivel (através de chaves), algumas caracteristicas ergonométricas, capacidade gráfica limitada.

3. Terceira Geração: Tela usual e expandida, controle através de microprocessador bastante poderoso, linguagem de controle elaborada, interpretação da maioria dos caracteres de controle do código ASCII, comunicação flexivel, caracteristicas e parâmetros armazenados em RAM não volàtil, projeto ergonométrico, capacidade gráfica

EVOLUÇAO ARQUITETONICA E FUNCIONAL DOS TERMINAIS

Embora pareça ser uma evolução bastante acelerada, os terminais mais recentes são ainda funcionalmente muito próximos aos da primeira geração. Obviamente incorporam os avanços tecnológicos do periodo, mas a parte realmente critica desses dispositivos, a linguagem de comando e a capacidade de aliviar o hospedeiro de tarefas simples quase não foram alteradas. Alguns dos processadores atualmente empregados para o controle interno dos terminais são quase tão poderosos quanto os computadores aos quais alguns dos terminais de primeira geração foram conectados. Não obstante essa capacidade embutida, pouquissimas vêzes o microprocessador e' chamado a auxiliar o hospedeiro numa tarefa.

Uma categoria de periféricos próxima `a dos terminais, a das estações gráficas, apresenta evolução mais rápida. Atualmente a maioria delas e' capaz de processar localmente grandes quantidades de dados, armazenando-os fora do computador hospedeiro e manipulando-os conforme os pedidos do usuário. Esse parece ser o caminho mais lógico a ser trilhado também pelos terminais de video.

II.4 NOVAS FUNçõES PARA OS TERMINAIS

Além da validação de dados (explicada na descrição do VT62), um terminal inteligente poderia aliviar o hospedeiro executando localmente, por exemplo, as seguintes tarefas:

II.4.1 Interpretação Da Linguagem De Controle -

Todos os sistemas interativos têm liguagens de controle relativamente elaboradas, procurando facilitar a interface usuário-processador. Na quase totalidade dos casos a mesma e' implementada através de um interpretador (DCL no VMS [48], Shell no UNIX [49], JCL no OS e VM da IBM [50]) que e' ativado mediante a leitura de uma cadeia de símbolos lida de um terminal. Esse interpretador e' um processo que inclui quase sempre um analisador sintático e um analisador léxico que analisam a correção do comando e invocam a rotina do núcleo do sistema operacional que efetua a particular ação pedida pelo usuário. Para efetuar a análise do comando o interpretador compete com os demais processos ativos naquele instante pelos recursos do sistema. A interpretação e' feita caracter a caracter, com a formação de palavras reservadas, pesquisa em tabelas de símbolos globais ao sistema, tradução de nomes e pesquisa `as regras sintáticas da linguagem.

Embora não seja um processo muito rápido, o interpretador de linguagem de controle não exige um processador de muitos recursos. Não e' necessário efetuar operações em ponto flutuante, não e' necessária uma área de dados muito grande e, se os símbolos globais ao sistema não forem traduzidos e as regras da gramática forem residentes no interpretador, não são necessários muitos acessos a dados armazenados em memórias

secundárias. O microprocessador normalmente empregado para controle do terminal pode perfeitamente fazer esta interpretação passando para o hospedeiro apenas um comando codificado que inclua os símbolos globais e a ação desejada[6] (figura 11). Sem dúvida isto aliviaria o hospedeiro, pois apenas comandos sintaticamente corretos seriam recebidos, estando também ja' codificados numa forma muito mais eficiente do que numa cadeia de caracteres. Um terminal mais poderoso poderia ate' mesmo solicitar por ADM a tabela de símbolos globais ao sistema; desta forma o interpretador invocaria diretamente a rotina apropriada do núcleo do sistema operacional. Os ganhos nesse caso seriam muito maiores pois o processo intepretador não competiria de forma alguma pela UCP do hospedeiro (figura 12).

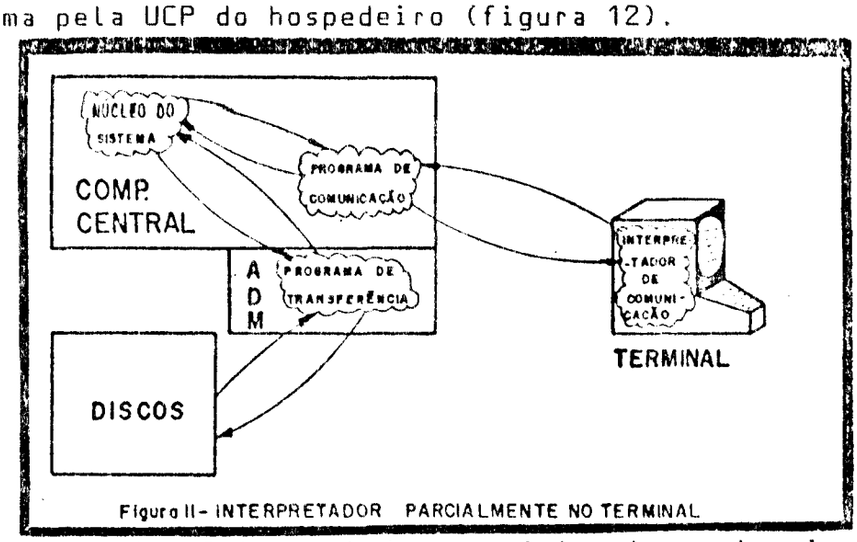


Figura 11 - Interpretador parcialmente no terminal.

Figura 12 - Interpretador totalmente local ao terminal.

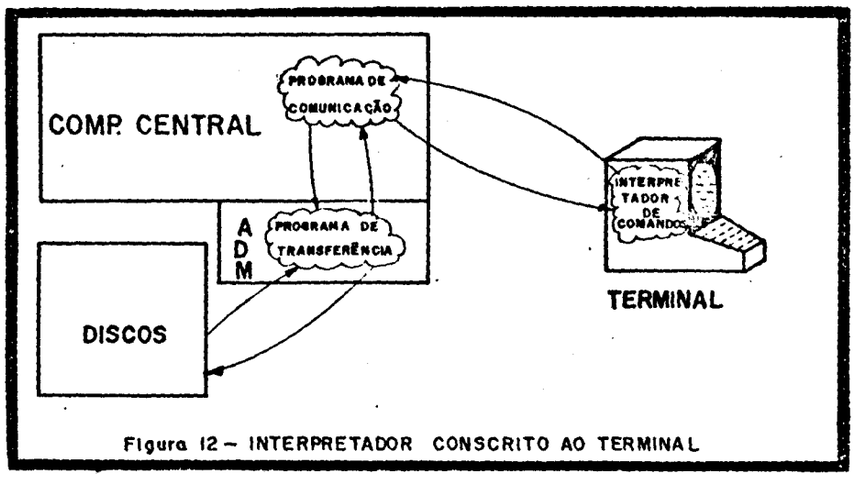


Figura 12 – INTERPRETADOR CONSCRITO AO TERMINAL

II.4.2 Editor De Textos -

Grande parte do processamento interativo de alguns centros compõe-se de algum tipo de edição de textos. Esta tarefa também não exige um processador muito poderoso, havendo editores extremamente elaborados disponiveis para os mais limitados microprocessadores [60].

Neste caso a dificuldade reside no armazenamento dos dados. Um terminal com 256 kbytes de memória volátil (RAM) e' econômicamente viável[7] , sendo este tamanho de memória mais que razoável para armazenamento de textos. O problema reside na transferência de dados do subsistema de memória de massa para o terminal, podendo inclusive

---

6 - Durante a revisão bibliográfica que fizemos para estre trabalho encontramos terminais controlados pelos seguintes microprocessadores: 8080, 8085, Z80, 6801, 6809 (microprocessadores de 8 bits), 8088 e 80188 (microprocessadores de 16 bits) e 6803 e 8048 (microcontroladores de 8 bits). Desses, cremos "a priori" que apenas o 8048 não seria capaz de executar o processo interpretador da linguagem de controle.

tomar um tempo tão excessivamente grande que não se justifique caso a alteração a ser feita no arquivo seja pequena. Por exemplo, um arquivo com 20 páginas datilografadas tem aproximadamente 60 Kbytes e transmitir essa quantidade de dados a 9.600 bps demora 51 segundos. Se a comunicação for através de modem's a velocidade de transmissão certamente não ultrapassara' 2.400 bps, demorando a carga do arquivo então 3,5 minutos . Como após a edição devemos gravar novamente o arquivo alterado no disco esses tempos são duplicados; e se desejarmos modificar uma única palavra neste arquivo perderemos perto de dois minutos a 9.600 bps e entre sete e oito minutos a 2.400 bps. Certamente um mecanismo melhor precisa ser proposto.

Como a maioria dos editores atuais e' do tipo "editor de tela", quer o editor seja executado no hospedeiro, quer seja localmente, pelo menos uma tela de dados (aproximadamente 2 kbytes) precisa ser transferida para o terminal. Pode-se-ia então implementar um editor que fosse executável localmente, trabalhando sobre a tela transferida do subsistema de armazenamento de massa do hospedeiro. Como a maioria dos controladores dos

---

7 - O preço das memórias de acesso direto (RAM's) tem diminuido exponencialmente em anos recentes; sua densidade tem crescido na mesma proporção. Um subsistema de 256 kbytes requereria 128 ci's e custaria mais de 500 dólares ha' cinco anos. Hoje pagariamos menos de 50 dólares e empregariamos 8 ci's.

subsistemas de armazenamento de massa dos computadores de grande porte e' capaz de atender a pedidos de alto nivel, essa transação seria feita sem o envolvimento da UCP central. Mesmo que esse tipo de transferência não seja suportado, certamente o controlador de memória de massa e' capaz de transferências diretas, ficando o papel do computador central reduzido `a inicialização dessa transferência. As telas editadas poderiam ser devolvidas ao hospedeiro ou armazenadas no terminal para outras modificações. Embora não seja uma edição tão eficiente como as feitas centralmente, ja' que algum tipo de re-transferência de dados sempre sera' necessário, esse mecanismo poderia reduzir significativamente a carga de alguns computadores, especialmente em centros de ensino e pesquisa.

II.4.3 Compilação Cruzada -

Se algum mecanismo de edição local for implementado nada impede que a compilação dos programas do hospedeiro seja efetuada localmente. Embora seja um processo mais complexo que a interpretação de uma linguagem de controle ou uma edição, uma compilação e' perfeitamente viável em microprocessadores de 8 e 16 bits, fato facilmente constatável verificando-se a gama de compiladores disponiveis para esses dispositivos.

Outras capacidades poderiam ser incluidas no terminal. Entretanto, se o mesmo for capaz de receber do hospedeiro um programa e executá-lo sob o comando deste com os dados que o mesmo lhe fornecer cremos que grande parte da carga atual dos centros de processamento sera' feita localmente, em paralelo e sem grandes alterações arquitetônicas nos processadores.

Apenas uma pergunta resta: Quais as caracteristicas do terminal capaz de efetuar estas ações? Não seria este periférico por demais caro e complexo? Nos capitulos subseqüentes investigaremos uma proposta de arquitetura de terminal deste tipo, verificando que o mesmo tem um nivel de complexidade muito pouco superior ao dos terminais convencionais.

significativamente a carga de alguns computadores, especialmente em centros de ensino e pesquisa.

II.4.3 Compilação Cruzada -

Se algum mecanismo de edição local for implementado nada impede que a compilação dos programas do hospedeiro seja efetuada localmente. Embora seja um processo mais complexo que a interpretação de uma linguagem de controle ou uma edição, uma compilação e' perfeitamente viável em microprocessadores de 8 e 16 bits, fato facilmente constatável verificando-se a gama de compiladores disponiveis para esses dispositivos. Como o sistema operacional do hospedeiro ja' teria sido modificado para prover a transferência de dados para a edição local bastaria desenvolver o compilador.

Outras capacidades poderiam ser incluidas no terminal. Entretanto, se o mesmo for capaz de receber do hospedeiro um programa e executá-lo sob o comando deste com os dados que o mesmo lhe fornecer cremos que grande parte da carga atual dos centros de processamento sera' feita localmente, em paralelo e sem grandes alterações arquitetônicas nos processadores.

Apenas uma pergunta resta: Quais as caracteristicas do terminal capaz de efetuar estas ações? Não seria este periférico por demais caro e complexo? Nos capitulos subseqüentes investigaremos uma proposta de arquitetura

de terminal deste tipo, verificando que o mesmo tem um nivel de complexidade muito pouco superior aos terminais convencionais.

CAPITULO III

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.0 CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

No capitulo I descrevemos genericamente as arquiteturas de terminal empregando CRT's e DVST's. Apresentaremos agora com detalhes a arquitetura de um terminal convencional com CRT, mostrando em seguida quais as alterações necessàrias para transforma'-lo num terminal inteligente como o descrito no capitulo 2. O dispositivo que descreveremos foi construido no Laboratório de Instrumentação Eletrônica como parte do trabalho prático desta dissertação.

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.1 ARQUITETURA CONVENCIONAL

Todos os terminais estudados[8] foram constuidos em torno de microprocessadores. A arquitetura convencional de terminal controlado por microprocessador e' um sistema orientado através de um barramento comum `a UCP e aos subsistemas de memória, controle de vídeo, comunicação e varredura de teclado, como pode ser visto nas figuras 13 e 14. A maioria dos projetos inclui também um segundo duto para comunicação entre a memória de tela e o subsistema de vídeo, controlado por um multiplexador de memória de porto dual ou unidade de acesso direto `a memória (ADM).

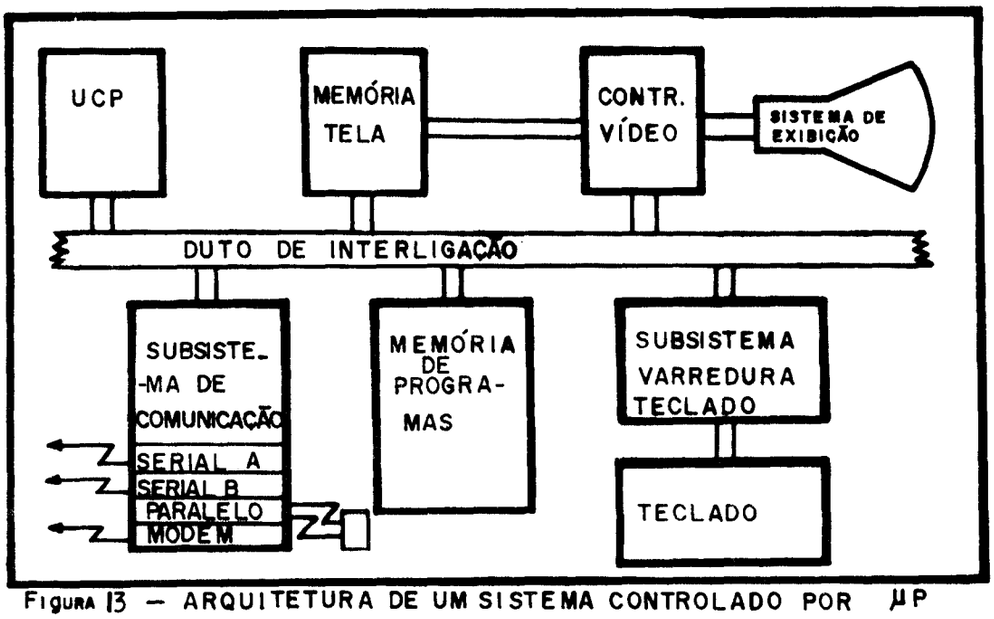


Figura 13 — ARQUITETURA DE UM SISTEMA CONTROLADO POR µP

Na maior parte das implementações o funcionamento do

---

8 - Investigamos os projetos de vários terminais, o mais antigo dos quais data de 1976 e o mais recente, 1984. A maior parte dos projetos e' de periféricos efetivamente comercializados, a maioria no exterior.

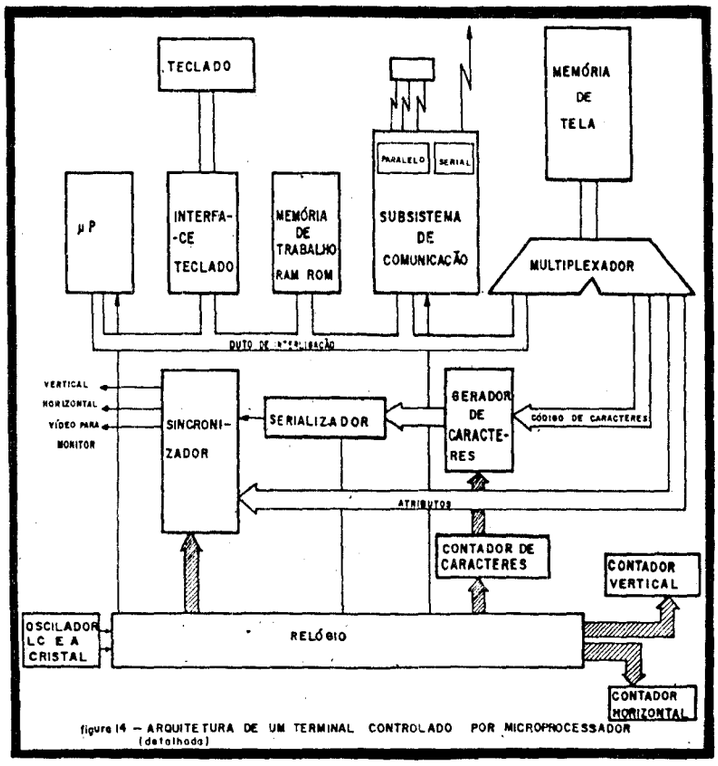


figura 14 — ARQUITETURA DE UM TERMINAL CONTROLADO POR MICROPROCESSADOR (detalhada)

sistema e' muito semelhante: O microprocessador da unidade de controle ativa o subsistema de varredura de teclado; quando uma tecla e' detectada, transforma-a na letra correspondente e, caso não seja um comando da linguagem de controle, envia-a pelo porto serial para o hospedeiro. Os dados recebidos pelo porto serial são interpretados de acordo com a sintaxe da linguagem de comando e se forem letras comuns são inseridos numa porção da memória associada ao subsistema controlador de video (chamada por isso memória de tela, MT, ou memória de video, MV). Se forem comandos a ação solicitada e' efetuada. O subsistema controlador de video no instante apropriado retira essas letras da MV e as transforma na seqüência de pulsos necessária para a representaçao das mesmas na tela. O subsistema de exibição recebe esses

pulsos e os condiciona para os níveis e temporização exigidos pelo tubo de raios catódicos (CRT), conforme as figuras 13 e 14.

III.1.1 Subsistemas De Memória, Teclado E Comunicação -

A maioria dos subsistemas de um terminal comum e' muito similar aos equivalentes encontrados num microcomputador. O subsistema de memória e' composto por memórias não voláteis (quase sempre do tipo EPROM e, eventualmente, EEPROM) e memórias voláteis (do tipo RAM, dinâmicas ou estáticas). Na memória são armazenadas o programa de controle dos demais subsistemas, o interpretador da linguagem de controle, variáveis de trabalho e a tela do terminal. No subsistema de varredura de teclado ha' portos paralelos cujas linhas varrem a matriz de teclas ou um controlador dedicado, que as varre e serializa como mostra a figura 15. Este subsistema interfaceia o microprocessador e o teclado do terminal.

O subsistema de comunicação inclui geradores de velocidade de comunicação ("baud rate"), portos seriais sincronos e assincronos ("USART's"), portos paralelos e moduladores/demoduladores (modem's). A figura 16 descreve uma configuração genérica para o subsistema de cominicação. Pelo subsistema de comunicação o terminal e' conectado ao hospedeiro e a outros periféricos como impressora, "plotter", outros terminais, etc.

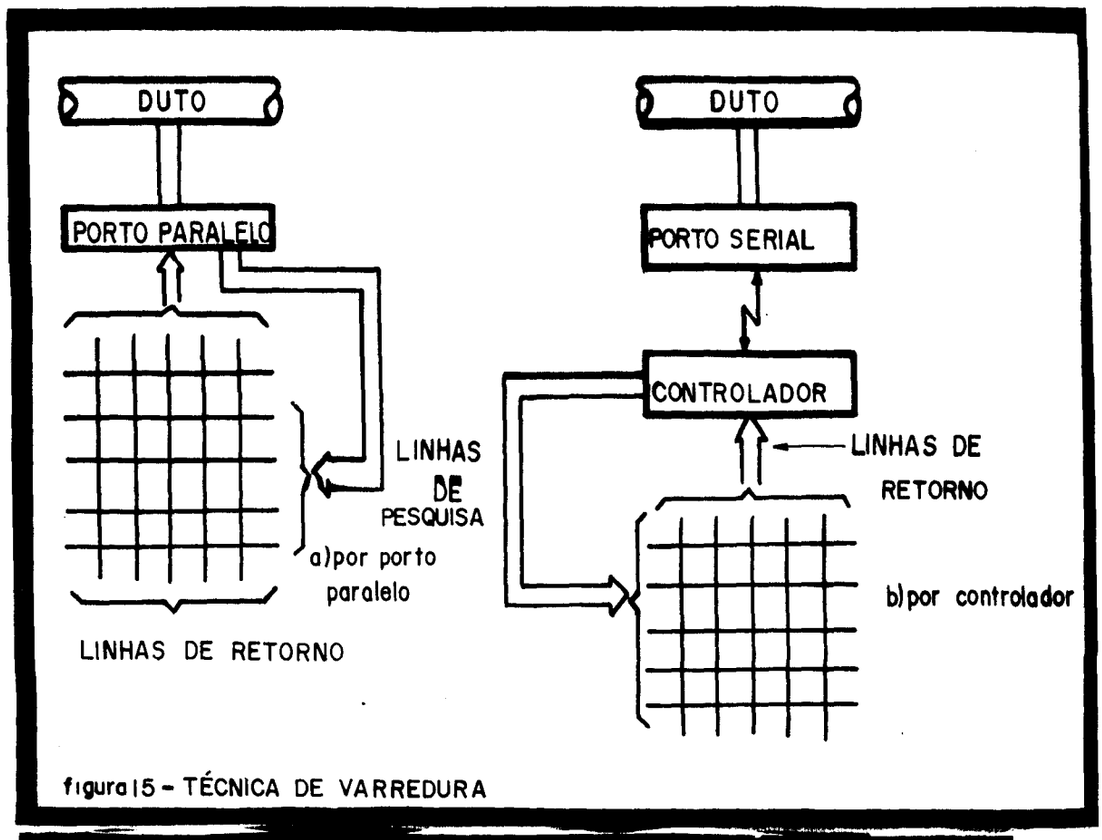


figura 15 - TÉCNICA DE VARREDURA

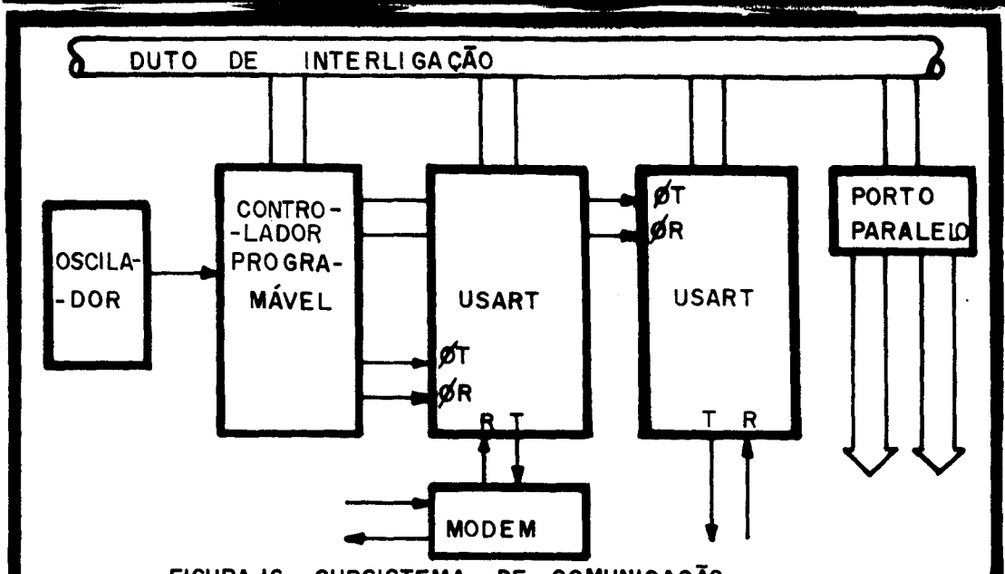


FIGURA 16 - SUBSISTEMA DE COMUNICAÇÃO

III.1.2 Subsistema Controlador De Video -

O mais complexo dos subsistemas do terminal e' o controlador de video (CRTC), responsável pelo controle do CRT mediante a geração dos pulsos de sincronismo horizontal e vertical que fazem a varredura da tela. Como o feixe esta' sob seu comando, o CRTC sabe qual o

instante apropriado para liga'-lo e desliga'-lo de tal forma que apareça na tela uma figura correspondente ao texto armazenado na memória de vídeo. São pelo menos três os sinais gerados pelo CRTC: sinal de sincronismo horizontal, sinal de sincronismo vertical e sinal de ativação do feixe. Na prática, outros sinais podem ser gerados como sinal de inversão dos dados, sinal de intensificação do feixe, etc. Esses sinais são todos gerados a partir dos dados lidos na memória de vídeo e de contadores internos ao controlador (figura 17).

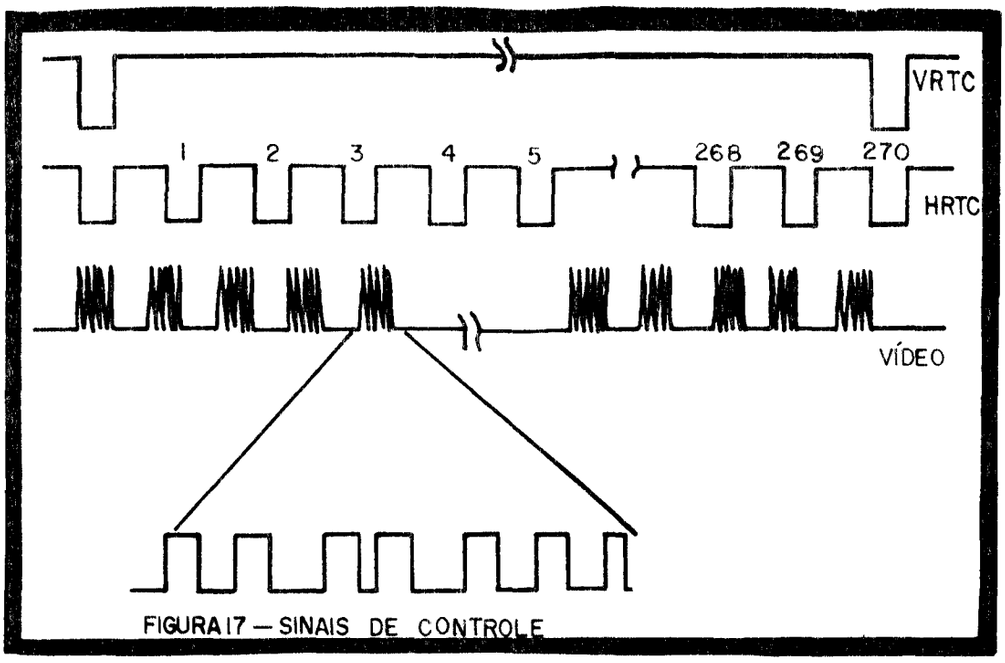


FIGURA 17 — SINAIS DE CONTROLE

O CRTC para gerar corretamente esses sinais com temporização tão restrita tem uma arquitetura interna bastante elaborada, como descrito na figura 18.

CONSIDERAÇÕES SOBRE A ARQUITETURA ADOTADA

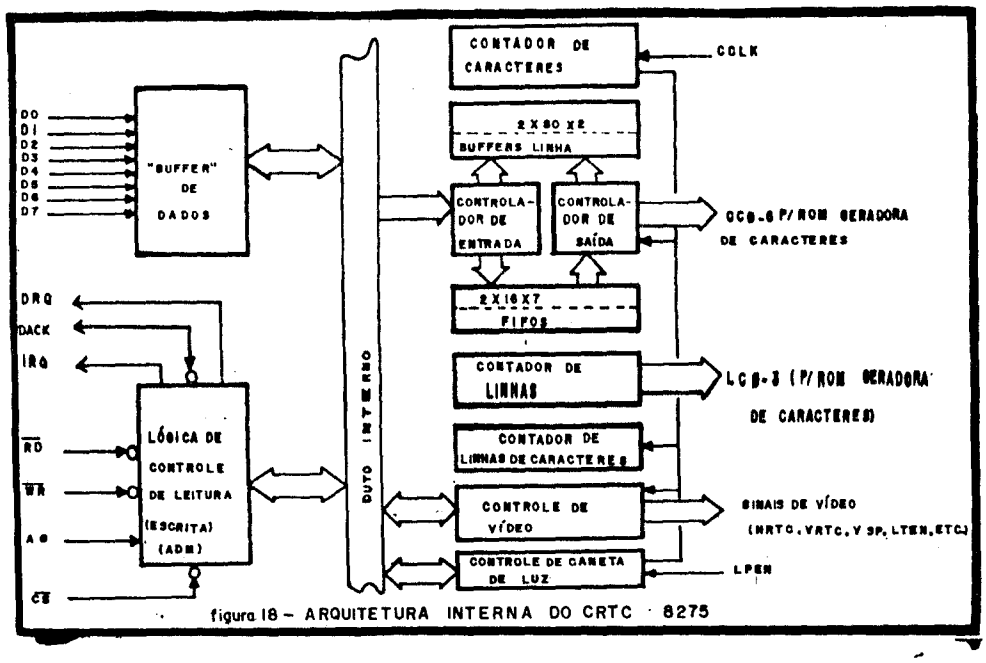


figura 18 - ARQUITETURA INTERNA DO CRTC 8275

Esses componentes podem ser didaticamente divididos em quatro blocos: bloco de acesso `a memória, composto pela lógica de contenção, pelo endereçamento da MV, pela lógica de deslocamento de tela ("scroll") e pelos "buffers" de linha; bloco de temporização, composto pelos contadores de linha e coluna, pelo relógio de ponto e pelo gerador de sincronismo; bloco de geração do sinal de videoçomposto pelo gerador de caracteres, pelo serializador, pela lógica de inibição e criação do sinal e pela lógica de cursor e finalmente, bloco de controle composto pelo barramento interno, pela unidade central de controle e pela interface com o microprocessador.

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.1.2.1 Bloco De Acesso `a Memória -

Este bloco permite que o CRTC possa ler os dados da MV. Como os dados presentes na MV são escritos pela UCP e lidos pelo CRTC, algum mecanismo de sincronização precisa existir para permitir que dois mestres diferentes possam usar o mesmo escravo. Trata-se de uma sincronização bastante dificil, pois o CRTC não pode esperar pelos dados em hipótese alguma visto ser sua temporização bastante rigida. Entretanto, se a UCP não tiver algum tipo de prioridade a performance do terminal sera' extremamente baixa pois e' ela quem controla as ações do mesmo (limpar a tela, poisicionar o cursor, etc.). E' devido principalmente aos problemas de sincronização que os terminais mais antigos não trabalham com taxas de comunicação muito elevadas.

Os mecanismos de sincronização mais comumente encontrados são:

1. Controle do duto pelo CRTC - Neste tipo de contenção a MV e' logicamente pertencente a CRTC. Como ha' um periodo fixo de inatividade do controlador inerente `a lógica do CRT, o

---

9 - Tempo necessário para que a eletrônica de direção do feixe de elétrons o traga da'ultima para a primeira linha da tela, tipicamente entre 800 e 1.000 uS.

10 - Nos CRT's comuns ha' 30 ou 60 retraços verticais por segundo.

tempo de retração vertical[9] , apenas durante este periodo a UCP pode acessar a MV, ficando bloqueada quando o CRTC volta `a atividade. E' a contenção mais segura, porém exige circuitos de separação entre os dutos da UCP, memória e CRTC (integrados de alta impedância ou multiplexadores) e permite que a UCP trabalhe apenas 1 ms a cada 16 ms[10] ou seja, menos que 7% do tempo total. Arquiteturas com essa contenção são normalmente limitadas a 2.400 bps.

2. Acesso entrelaçado - Alguns processadores [56], [57] usam um relógio de quadratura controlando o acesso `a memória: leituras e escritas somente ocorrem em um dos niveis do sinal, como na figura 19. Nestes casos pode-se implementar a contenção de memória permitindo o acesso do CRTC apenas nos ciclos inativos da UCP. O inconveniente desse método reside no fato de que pouquissimas UCP's e CRTC's tem sua temporização baseada num relógio de quadratura. Exceções são os CI's 6800, 6809, 68000 e 6845.

3. Memória auxiliar para o CRTC - Normalmente a exibição uma linha de caracteres requer entre 9 e 11 varreduras horizontais. E' possivel aliviar bastante as restrições de funcionamento da UCP empregando uma memória de video separada com apenas uma linha de caracteres destinados ao

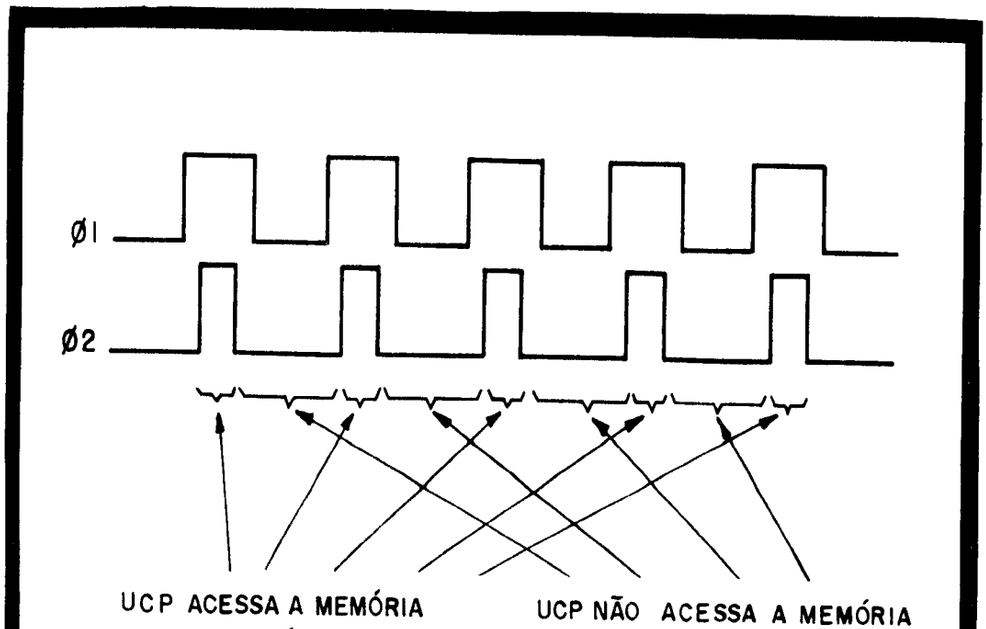


figura 19 — RELÓGIO DE QUADRATURA

vídeo. Dessa forma, durante o período das 9 a 11 varreduras que uma linha demora para ser composta na tela a UCP fica livre para acessar a MV, ja' que o trafego de dados fica confinado entre o CRTC e esta memória separada. Após a última varredura uma nova linha e' copiada da MV para o "buffer" de linha, sendo este o único instante no qual se requer inatividade da UCP.

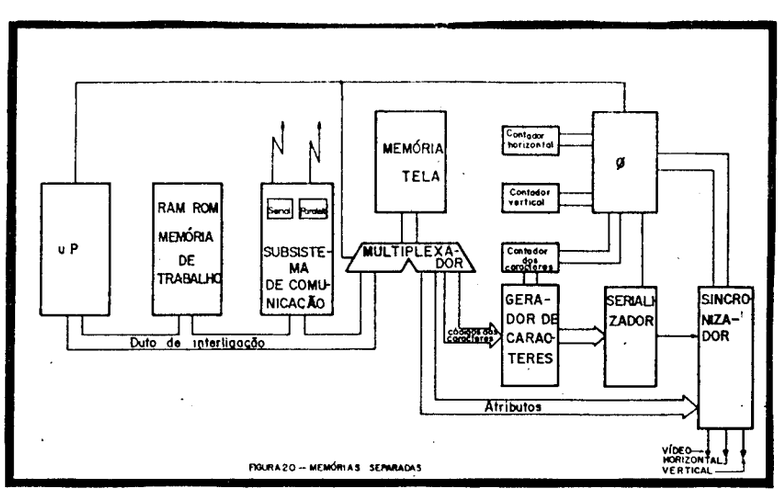


FIGURA 20 — MEMÓRIAS SEPARADAS

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

Se a transferência entre MV e a memória auxiliar for efetuada por ADM a UCP possui o controle do duto entre 65%e 80% do tempo, permitindo terminais de alta eficiência. Essa técnica também apresenta a vantagem de ser aplicável a qualquer tipo de UCP que suporte transferências de ADM e qualquer controlador de vídeo. Sua desvantagem esta' na quantidade de CI's que exige: um gerenciador de acesso direto `a memória e memórias separadas para o CRTC.

III.1.2.2 Bloco De Temporização -

Para que o tubo de raios catódicos possa operar os sinais de sincronismo horizontal e vertical devem ser enviados repetidamente. Como cada CRT possui características diferentes, normalmente os sincronismos são gerados por contadores programáveis, que dividem o relógio básico do sistema por constantes definíveis pela UCP. O relógio de ponto (sinal cuja borda de subida e' empregado para ligar o feixe e para deslocar o código do caracter no registro de deslocamento) também e' gerado neste bloco, bem como o relógio de caracter, usado pelo CRTC para endereçar a ROM geradora de caracteres e pelo sincronizador inibidor/criador do sinal. Na página seguinte encontra-se a temporização completa gerada por esse bloco (figura 21).

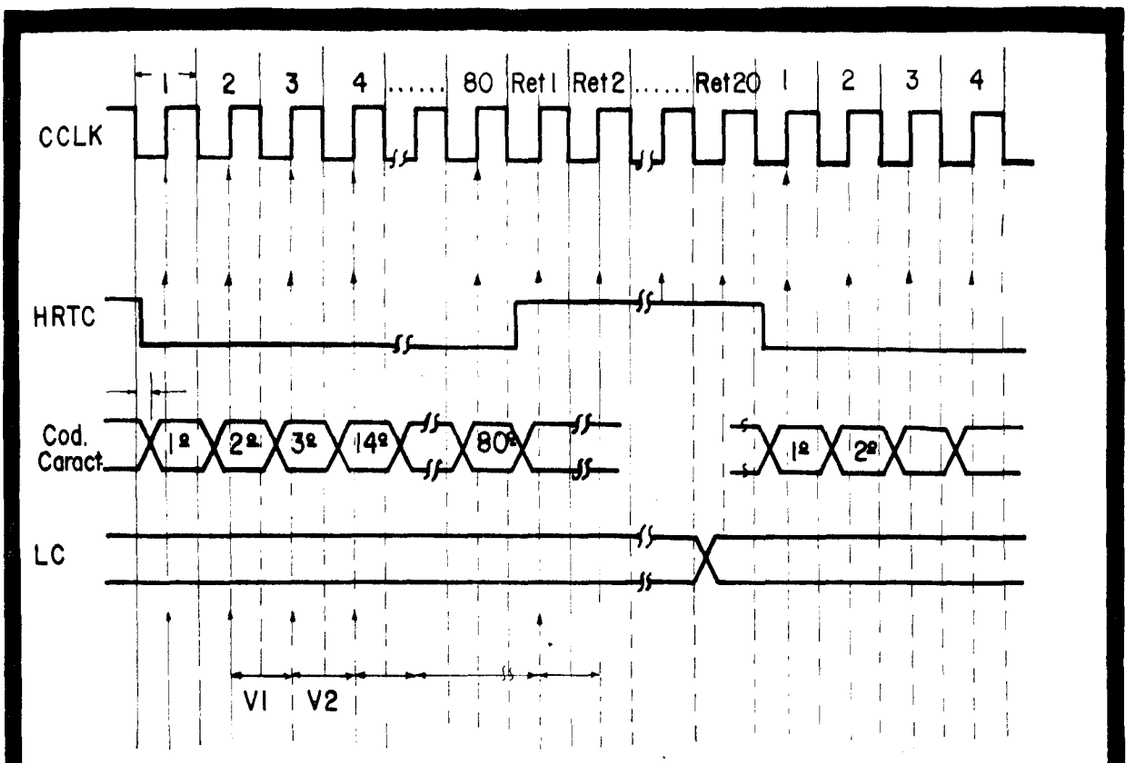


FIGURA 21 — TEMPORIZAÇÃO DO SISTEMA

III.1.2.3 Bloco De Geração Do Sinal De Video -

Esta parte do controlador transforma os caracteres armazenados em código na MV numa cadeia de pulsos para o CRT. Para tanto primeiramente os caracteres são retirados um a um da MT (ou do "buffer" de linha) e apresentados ao gerador de caracteres ao mesmo tempo que os contadores de varredura são zerados. A cadeia de bits que representa então a primeira linha da matriz desse caracter aparece na saida do gerador, sendo transferida para o deslocador que a serializa sob o comando do gerador de sincronismo. O sinal serializado passa ao sincronizador inibidor/criador do sinal de video que, no momento apropriado, apresenta o sinal ao CRT. O sincronizador e' chamado inibidor/criador porque a saida do serializador pode ser suprimida caso o sincronizador

verifique que o feixe não se encontra numa posição apropriada. Alguns sinais extras como sublinhado ou cursor são também criados neste elemento. A lógica do cursor indica os momentos nos quais o mesmo deve aparecer na tela.

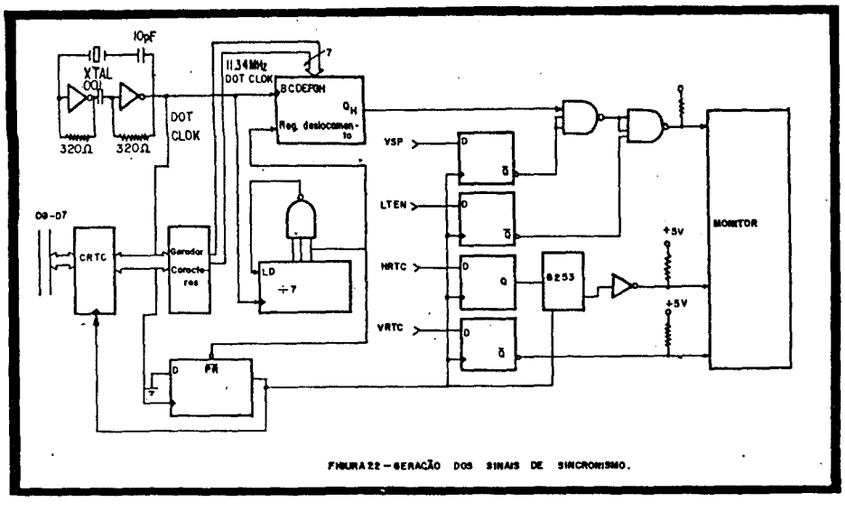


FIGURA 22 — GERAÇÃO DOS SINAIS DE SINCRONISMO.

III.1.2.4 Bloco De Controle E Interface -

Todos os outros elementos são controlados por este bloco, também responsável pela programação do CRTC, normalmente via registradores que podem ser lidos e escritos pela UCP. Ha' alguns dispositivos, entretanto, cujos parâmetros não são programáveis mas definidos pela máscara de ROM usada. Um exemplo e' o CRTC DP8350 da National [60].

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.2 ARQUTETURA DO TERMINAL CONSTRUIDO

Construimos um terminal convencional para verificarmos em sequida quais as modificações necessárias para que o mesmo tivesse as caracteristicas de inteligência que julgamos necessárias. No apêndice C encontra-se o diagrama de blocos do projeto, que agora comentaremos.

A decisão mais critica a ser tomada era a escolha do mecanismo de contenção de acesso `a memória, ja' que esta opção traz em seu bojo a decisão sobre o tipo de microprocessador e de controlador de video a ser empregado. Desde o inicio do projeto havia uma forte tendência a construir este terminal convencional com um minimo de componentes e com o menor custo possivel, o que excluia o emprego de um microprocessador mais elaborado ou de um CRTC pouco conhecido. Esses fatores praticamente eliminaram a possibilidade de empregarmos a arbitração de memória por acesso entrelaçado pois, embora seja a mais eficiente, forçaria o emprego dos componentes da Motorola (microprocessador 68xx e CRTC 6845). Descartamos o esquema de controle do duto pelo CRTC por levar a um baixo desempenho da UCP, ja' que pretendiamos em seguida ampliar o projeto para executar localmente funções mais elaboradas. Restou-nos a opção de memórias separadas, com "buferização" de uma linha para o CRTC. A desvantagem desse tipo de contenção e' o encarecimento do projeto por exigir memórias separadas e algum tipo de

controle de transferência de ADM entre elas.

O primeiro inconveniente pode ser sanado optando-se pelo controlador 8275 ou 8276 da Intel, que inclui internamente a memória necessária para armazenar uma linha. Infelizmente esta decisão fixa a largura da tela em 80 caracteres, visto ser este o comprimento do "buffer" interno desses CRTC's. Seria possivel expandi-lo para 160 colunas cascateando-se dois controladores, o custo e complexidade do circuito neste caso são quase triplicados, razão pela qual não consideramos a possibilidade. O emprego do 8275 foi também considerado visto ser um dos mais baratos controladores disponiveis no mercado[11] .

Na figura 18 mostramos a arquitetura interna do CRTC 8275. Note-se os buffers internos, as fifos usadas para armazenar caracteres de controle, os contadores programáveis e a interface com o microprocessador. No apêndice E descrevemos cada um de seus sinais.

Mesmo empregando-se o 8275 o projeto ainda ficaria bastante caro por empregar um controlador de ADM. A solução foi simular o controlador de ADM por programação, como veremos.

---

11 - Em teoria o 8276, que e' um 8275 sem a interface de ADM e outras sofisticações deveria ser mais barato. Pelo menos no Brasil entretanto, e' sensivelmente mais caro, cremos que por ser mais recente.

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

III.2.1 Simulação De ADM Por Programação -

A função do ADM no circuito e' transferir uma linha de dados da MT para os "buffers" internos ao 8275. O modo convencional de efetuar esta transferência e' esperar uma solicitação de dados proveniente do 8275 e gerar um sinal de pedido de duto. Ao fim do ciclo, a UCP reconhece que outro mestre deseja o duto e responte afirmativamente, colocando suas linhas em estado de alta impedância ate' que a transferência termine. O controlador de ADM, de posse do dúto, endereça a porção de memória que deseja transferir, acionando simultaneamente um sinal de leitura para a memória e de escrita para o 8275. A transferência de cada byte dura quatro ciclos [58].

Pode-se simular esta ação através de uma rotina de atendimento rápido disparada por interrupção e alguns circuitos adicionais. Desta forma, quando o 8275 requisita dados uma interrupção e' gerada. A rotina de interrupção liga um bit especial de transferência que faz com que os circuitos transformem cada acesso de leitura na memória em ciclo de escrita no 8275 e, em seguida, lê os dados da MV através de sucessivas instruções de leitura na pilha (POP's). Desta forma dois bytes são transferidos a cada 10 ciclos, numa taxa de transferência de um byte a cada cinco ciclos, bastante próxima `a velocidade do ADM. Cada byte lido da memória e' escrito no CRTC e no acumulador da UCP. Como a unidade central

de processamento não necessita desses dados eles são descartados. O CRTC consegue armazena'-los porque seus "buffers" internos são do tipo "FIFO", ou seja, os vários dados escritos no mesmo endereço são diferenciados por sua ordem de chegada ao dispositivo.

Após algumas otimizações conseguimos implementar os circuitos necessários para essa simulação com duas portas "NAO", e duas portas "E", a um custo mais de cinqüenta vêzes menor que o de um controlador de ADM. Obviamente esta técnica exige que o programa de controle seja capaz de atender rapidamente a rotinas de interrupção, como entretanto esta e' uma capacidade que o terminal inteligente teria necessariamente que possuir (como veremos no capitulo 5) o inconveniente não e' tão grande. A figura seguinte mostra o simulador de ADM que transforma qualquer acesso de leitura `a memória em posição superior ao endereço 8000H em transferência de ADM.

Para tanto, o sinal de pedido de acesso direto `a memória gerado pelo CRTC (DRQ) e' convertido num sinal de interrupção apropriado para a UCP através de um inversor ligado `a entrada INT/ do Z80. Em atendimento `a interrupção o microprocessador passa a ler os dados na MT. A decodifição de memórias do projeto não leva em conta o bit 15 do duto de endereços, fazendo com que os endereços na faixa 0000H-7FFFH sejam mapeados sobre os da faixa 8000H-FFFFH.

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

A função do bit 15 e' indicar ao simulador de ADM que os dados lidos pelo processador devem ser transferidos também para o CRTC. Isto e' feito através das portas 2 e 3, que geram um sinal de reconhecimento de ciclo de acesso direto `a memória (DACK/) sempre que o bit 15 for 1 e o sinal de leitura da memória for ativo (MRD/). Note-se que devido ao fato do programa sempre residir na metade inferior da memória o ciclo de busca de instrução ("opcode fetch") não gera DACK/ para o controlador. Entretanto, o 8275 exige também que seja ativo o sinal de escrita (WR/) durante o ciclo de ADM. Para tanto multiplexamos o sinal de escrita da UCP (WR/) com o sinal DACK/. Isto e' necessário porque para a programação normal do 8275 precisamos fazer chegar ate' o mesmo os sinais RD/ e WR/ como se fosse um periférico comum do duto. Por economia de CI's empregamos duas portas OU-EXCLUSIVO que estavam inativas num CI 74LS86 para implementar uma porta "E".

III.2.2 Microprocessador Central -

Esta técnica foi descrita numa publicação técnica da Intel [58] num projeto de terminal que usava uma PROM para a simulação de ADM e o microprocessador 8085. Como desejávamos manter o menor custo possivel substituimos este microprocessador pelo Z80, muito mais barato e poderoso. Outra vantagem dessa substituição reside no fato de que o conjunto de instruções mais poderoso deste

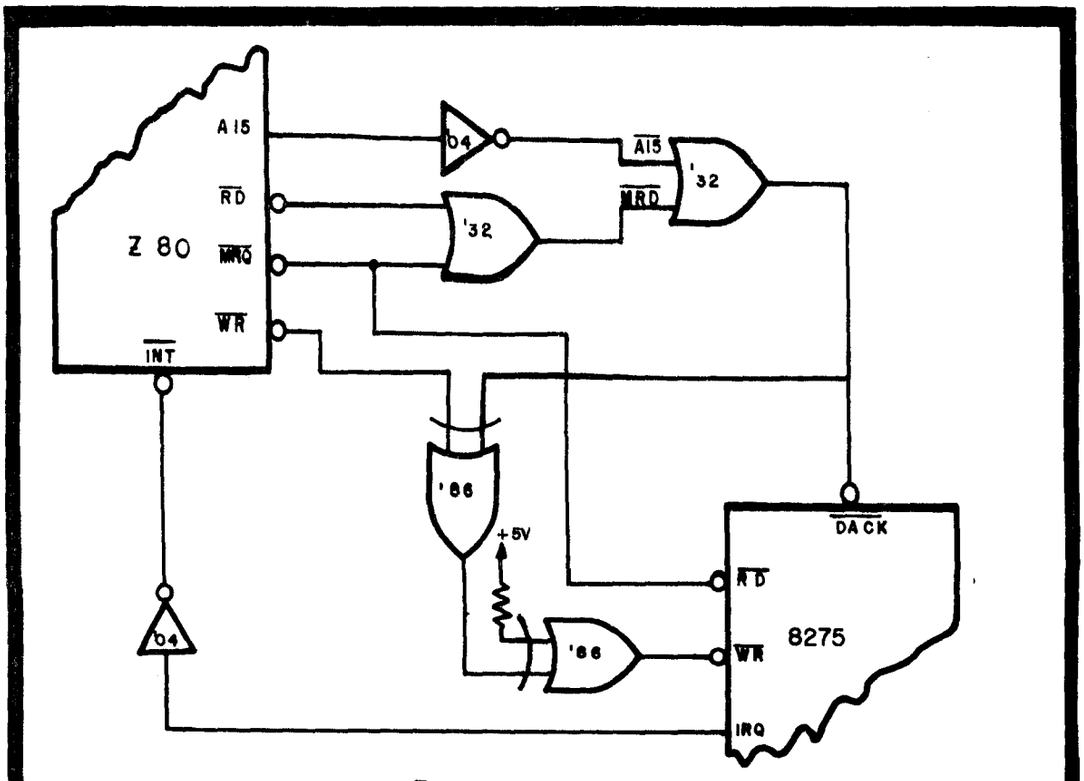


figura 23 – GERAÇÃO DO SINAL DE ADM

último permite que aplicações mais ambiciosas, como era nosso intento. O projeto final emprega o microprocessador (Z80) e o controlador (8275) mais baratos do mercado.

Ao empregarmos o Z80 também economizamos os circuitos necessários `a demultiplexação do duto, necessários ao 8085. Os sinais da interface de leitura e escrita do Z80 (RD/, WR/, MRQ/ e IORQ/) também permitiram a economia de um multiplaxador, pois como pode ser visto no projeto o espaço de endereçamento de E/S não foi decodificado. Com os sinais disponiveis no 8085 (RD/, WR/ e IO-M/) este esquema não teria sido possivel.

CONSIDERAÇÕES SOBRE A ARQUITETURA ADOTADA

Considerável parcela de tempo foi economizada alocando os registradores duplicados do Z80 apenas para o tratamento de interrupção. Desta forma, ao invés das cinco instruções PUSH e cinco instruções POP necessárias para salvar o estado da máquina a cada inicio de interrupção e restaura'-lo ao final apenas duas instruções de troca de banco de registradores ativos e' necessária. No primeiro caso são necessários 105 ciclos de máquina, no segundo, apenas 16 ciclos.

III.2.3 Demais Subsistemas -

A memória e' composta por 12K de EPROM e 6K de RAM estática, o que permite programas de razoável complexidade. O terminal suporta carga de programas pelo hospedeiro. Na sua configuração máxima o terminal endereça ate' 70K de EPROM (CI's de 4K bytes) e 44K de RAM (CI's de 2K bytes). O subsistema de comunicação tem dois portos seriais assincronos bidirecionais com velocidade programável entre 50 e 9.600 bps, o susbsistema de teclado e' composto por apenas um CI e o subsistema de geração do sinal de vídeo e sincronização e' feito com 8 CI's TTL e MOS de integração em pequena e média escala. No apêndice D encontra-se o esquema elétrico do terminal.

CONSIDERAÇOES SOBRE A ARQUITETURA ADOTADA

Note-se que o terminal em sua configuração mínima tem apenas 23 CI's. Nenhum outro projeto de nosso conhecimento tem tão poucos componentes, havendo alguns como o TDA e o SCOPUS 2052 que apresentam mais que o dobro desse número, com componentes de mesma tecnologia e capacidades semelhantes. O terminal VT-100 da Digital, embora ligeiramente mais antigo e com maiores capacidades, tem perto de 100 componentes.

III.2.4 Caracteristicas Do Terminal Construido -

O terminal que projetamos tem as seguintes caracteristicas:

1. Formato de tela

    1. 80 colunas, 96 letras por linha (16 caracteres de controle e 80 caracteres de exibição).

    2. 25 linhas no máximo (24 para compatibilidade com VT52).

2. Formato do caracter

    1. Matriz de caracteres 5x7 num campo 7x10.

2. Quatro tipos de cursos definiveis por programação.

3. Cursor na nona linha quando for do tipo sublinhado.

4. Ate' oito diferentes geradores de caracter.

5. Ate' quatro geradores diferentes podem ser misturados na mesma tela.

6. Quatro independentes atributos de campo: intensificado, video reverso, piscante, sublinhado.

3. Teclado

1. 105 teclas.

2. Disposição semelhante ao VT-240.

3. Teclas re-definiveis.

4. Ate' 256 caracteres gerados por uma única tecla.

4. Programação

1. Linguagem de comando VT52.

2. Linguagem de comando VT100.

3. Monitor reduzido (Menos de 2 K bytes).

4. Sistema operacional residente em tempo real.

5. Comunicação

1. Compativel com RS 232c.

2. Velocidade de comunicação programável entre 50 e 9.600 bps.

3. Sinais para Modem.

4. Comunicação "full duplex".

Trata-se de um terminal bastante poderoso, pois alia um baixissimo custo e extrema simplicidade à caracteristicas de programação bastante especiais, como explicitado no capitulo seguinte.

Entretanto, quão distante esta' este terminal do dispositivo inteligente descrito no capitulo anterior? No capitulo seguinte tentaremos responder a esta pergunta comparando as caracteristicas dos dois dispositivos e discutindo as modificações necessárias nos os casos em que ha' divergências.

CAPITULO IV

EXPANSÕES AO TERMINAL PROPOSTO

IV.0 EXPANSÕES AO TERMINAL PROPOSTO

Embora a proposta original fosse construir um primeiro protótipo do modo mais simples possível, apenas para testar o CRTC escolhido e procurando manter mínimo o número de componentes, verificamos surpresos que o terminal construído não distava muito da proposta feita anteriormente. Como procuraremos mostrar neste capítulo, com pequenas adições nos circuitos e uma programação apropriada, o projeto inicial transforma-se num dispositivo bastante próximo ao desejado, mantendo ainda características bastante atrativas de simplicidade e baixíssimo custo.

IV.1 COMPARAçÃO ENTRE O TERMINAL CONSTRUIDO E O PROPOSTO

Para aquilatar as diferenças entre o terminal construido e o proposto faremos uma comparação analitica entre as principais caracteristicas de cada um deles, cujo resultado e' mostrado na tabela abaixo:

Terminal Proposto	Terminal Construido
Capacidade de processamento local.	Mais de 40% do tempo do processador livre.
Capacidade de carregamento de programas.	Capacidade de carregamento de programas
Memória RAM suficiente para edição e compilação local (256K minimo).	Memoria RAM reduzida. Expansão de memória descrita a seguir.
RAM para pelo menos duas telas.	RAM para pelo menos duas telas (4K bytes).
Capacidade gráfica.	Expansão para capacidade gráfica reduzida descrita a seguir.
S/O em tempo real permitindo tarefas múltiplas.	Monitor simples. S/O em tempo real discutido no próximo capitulo.
Vários conjuntos de caracteres.	Vários conjuntos de caracteres.

EXPANSÕES DO TERMINAL PROPOSTO

Conjunto de caracteres programável.	Conjunto de caracteres fixo.
Teclado redefinivel por programação.	Redefinição do teclado possivel.
Operação no modo bloco com diversos modos de definição de tela e validação de entradas.	Operação no modo bloco possivel, ja' que o processador dispoe de tempo livre.
Primitivas para a emulação de diversas linguagens de comando.	Primitivas para a emulação de diversas linguagens de comando.
Espaco de ROM para Firmware (pelo menos 64K).	Expansão para ROM descrita a seguir.

Verifica-se que o terminal construido possui caracteristicas bastante próximas das propostas para um terminal inteligente. As grandes diferenças residem na programação; as poucas modificações nos circuitos são:

IV.2 ALTERAçÕES NOS CIRCUITOS

IV.2.1 Expansão De Memória RAM E ROM. -

Para velocidade de atendimento da interrupção que simula ADM usamos o bit 15 como gatilho de simulação do sinal DACK/, mapeando os 32K superiores sobre os 32K inferiores do espaço de endereçamento do microprocessador. O projeto então fica limitado ao acesso direto de 32 Kbytes de memória RAM/ROM, claramente insuficiente para as necessidades do terminal inteligente.

Este problema pode ser superado através de bancos de memória selecionáveis por programação. Nos esquemas abaixo usamos um 74LS137 como definidor do banco, possibilitando a conexão de ate' 8 bancos de 32 Kbytes num total de 256 Kbytes de RAM. Empregamos CI's 64Kx1 de memória dinâmica, os de menor custo por bit atualmente[12]. Outro 74LS137 permite expandir o espaço de ROM para outros 8 bancos de 16 Kbytes de ROM, totalizando 128 kbytes de ROM se empregarmos CI's UV-EPROM 27128 ou 8 bancos de 32 Kbytes totalizando 256 Kbytes caso optemos pelos CI's UV-EPROM 27256. Esta capacidade de memória e' mais que suficiente para as necessidades do terminal, mesmo se incluirmos capacidade de edição e compilação local. O fato da memória ser segmentada em bancos de 32K não apresenta muitos inconvenientes pois não se pretende que o terminal execute programas de terceiros, nem um sistema operacional convencional.

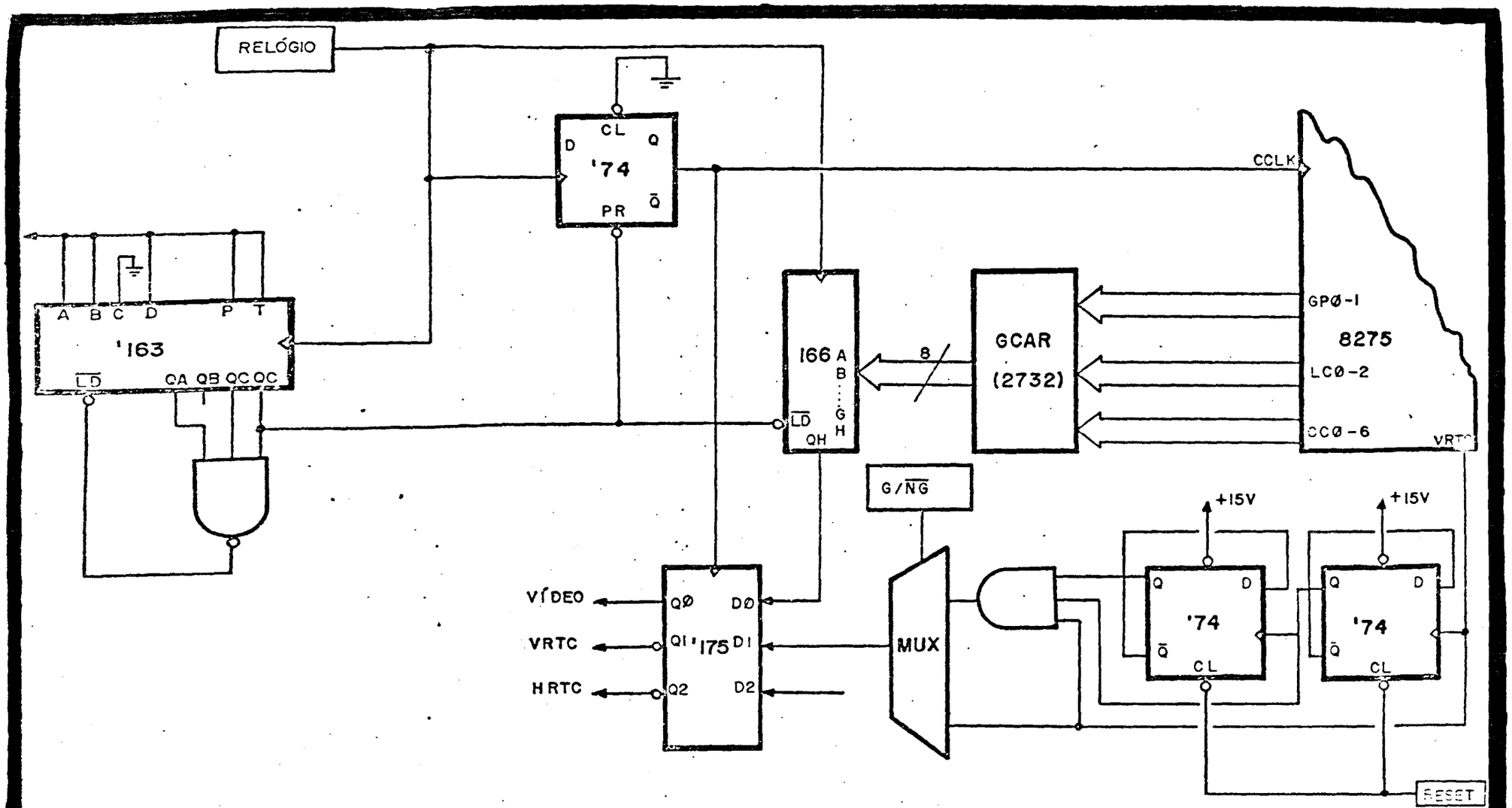


figura 24 – EXPANSÃO GRÁFICA

EXPANSOES DO TERMINAL PROPOSTO

IV.2.2 Expansão Gráfica -

Uma das caracteristicas ergonométricas mais apreciadas recentemente e' a capacidade de interfaceamento com o operador através de comandos gráficos. Desta forma, algum tipo de apresentação gráfica torna-se quase imprescindivel num terminal que se pretende, seja de uso geral. Uma possibilidade e' acoplar um módulo gráfico ja' desenvolvido no L.I.E. para aplicações em reconstrução de imagens [28]. Este módulo, entretanto, possui 128K de memória e ate' oito planos com dois niveis de intensidade numa tela de 512x250 pontos e ocupa três placas de circuito impresso, o que certamente e' demais para nossas necessidades. Para aplicações mais modestas propomos duas modificações que transformam o terminal alfanumérico num dispositivo gráfico de baixa resolução.

IV.2.2.1 Opção 1 - Mapeamento Por Bits -

O pricipio de funcionamento desta expansão gráfica esta' baseado no sacrifico de um dos quatro geradores de caracteres disponiveis no terminal. Neste gerador de caracteres gravamos uma tabela identidade, ou seja, a configuração de bits usada para endereça'-lo pelas linhas

---

12 - Embora ja' estejam disponiveis os CI's de 256Kx1, seu custo ainda e' maior que 4 CI's 64Kx1, pelo menos no pais.

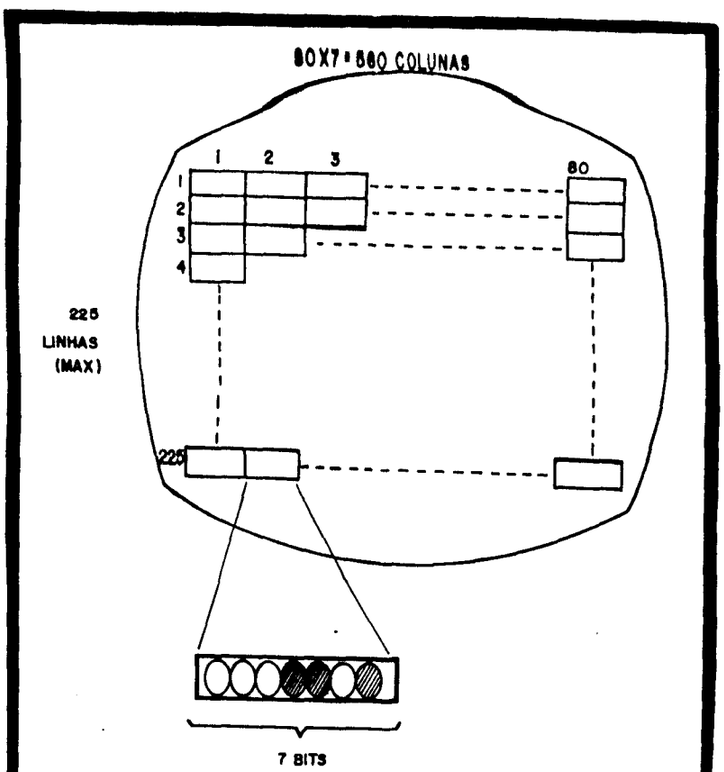


FIGURA 25 - MAPEAMENTO DA TELA GRÁFICA

CC0-CC6 e' a mesma que o gerador fornece ao registrador de deslocamento em suas saidas D0-D6. Se programamos o CRTC para caracteres de matriz 7x1 (sete colunas e uma única linha), teremos representado na tela uma cópia dos bits da memória de tela, como mostrado na figura 25.

Esta opção apresenta dois problemas de solução não trivial, um decorrente das limitações de programação do CRTC adotado e outro devido aos tempos envolvidos.

A primeira dificuldade surge pelo fato de que o tempo de retraço vertical gerado pelo 8275 e' definido como múltiplo do número de linhas de caracteres na tela. Esse contador aceita programação entre 1 e 64, uma variação bastante razoável para aplicações alfanuméricas pois como nelas normalmente os caracteres são compostos

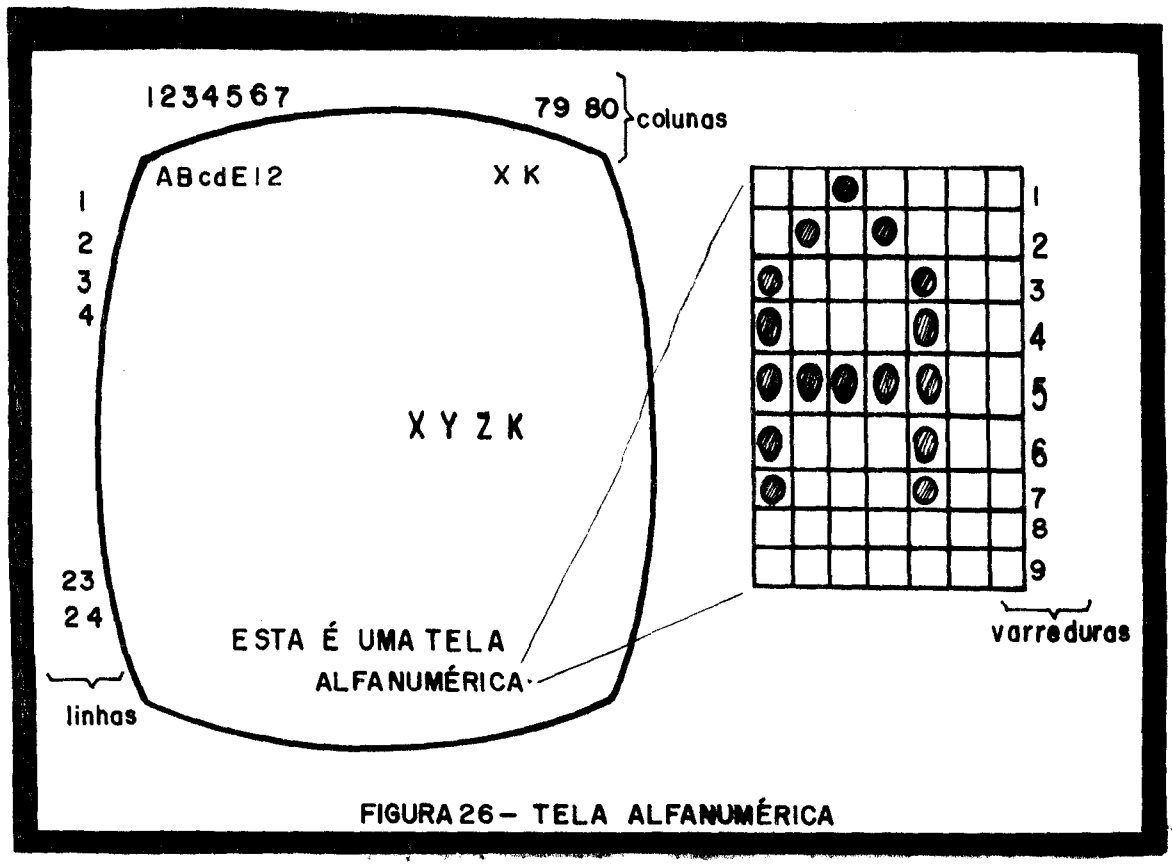


FIGURA 26 – TELA ALFANUMÉRICA

de pelo menos 7 varreduras (figura 26), a freqüência de retraço vertical pode variar entre 7 e 448 vêzes o tempo de retraço horizontal.

Infelizmente em aplicações gráficas o caracter não e' composto por 7 varreduras mas por apenas uma, (como mostra a figura 25) ja' que cada varredura não tem nenhuma vinvulação com a anterior. Assim, devido `a limitação do contador de retraço vertical, apenas 64 varreduras seriam possíveis, no máximo, em cada tela. Como este limite e' bastante restritivo projetamos o circuito da figura seguinte, cuja função e' suprimir três a cada quatro pulsos de retraço.

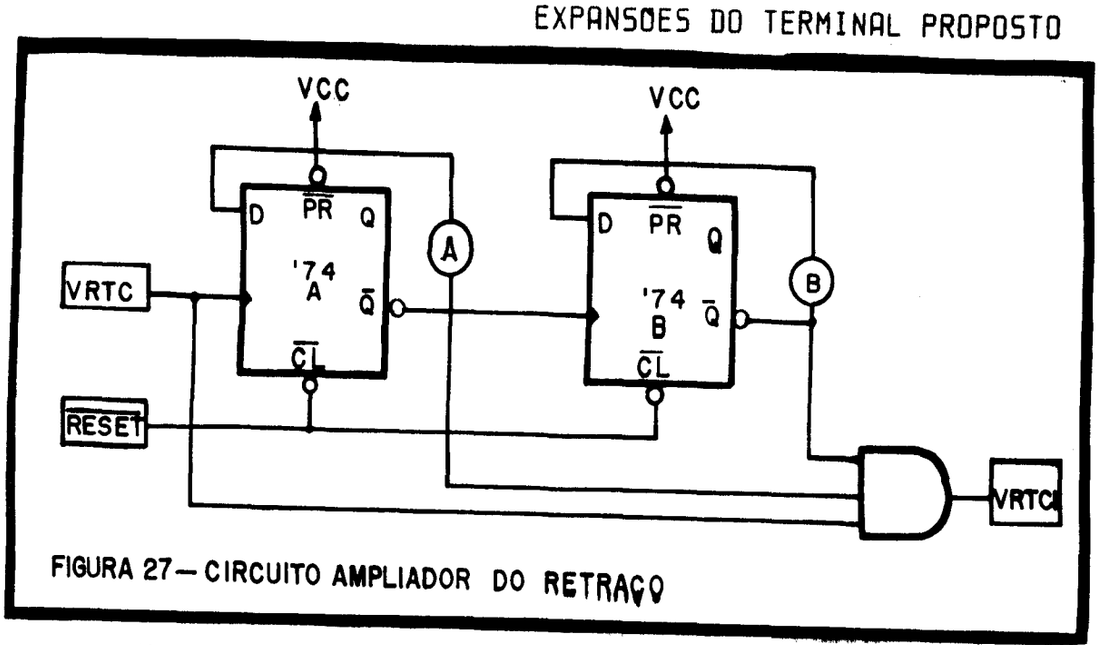


FIGURA 27— CIRCUITO AMPLIADOR DO RETRAÇO

Neste circuito usamos biestáveis tipo D sensiveis `a borda de subida como divisores binários. Assim, a cada dois pulsos na entrada de relógio uma transição e' verificada nas saidas Q e Q/. A porta "E" final faz com que o comprimento do pulso final (dividido por 4) seja igual `a largura do pulso original. As entradas CL/ são ligadas ao RESET/ do microprocessador para que os divisores trabalhem sincronizados. Na figura 28 mostramos as formas de onda do circuito.

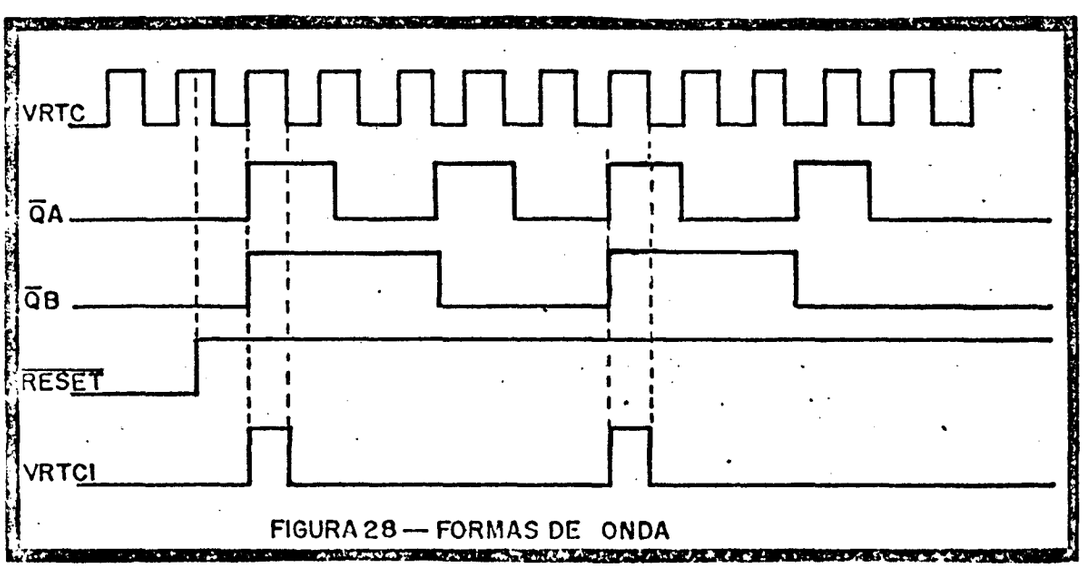


FIGURA 28 — FORMAS DE ONDA

EXPANSOES DO TERMINAL PROPOSTO

O segundo problema, embora bastante relaciondo ao primeiro, e' de solução muito mais complexa. Em aplicações alfanuméricas, apenas a cada 7 ou mais varreduras horizontais necessitamos de novos dados ao contrário das aplicações gráficas, onde dados são necessários para cada varredura horizontal. Desta forma a UCP e' pelo menos 7 vezes mais solicitada, pois ao fim da cada vsrredura os dados fornecidos ja' foram consumidos e outros se fazem necessários.

Apenas alguns cálculos são necessários para se verificar que uma UCP do porte da empregada não apresenta vazão suficiente para cobrir as necessidades do sistema:

Usamos uma técnica de simulação de ADM que permite uma velocidade de transferência máxima de um caracter a cada 5 ciclos de relógio[13] . Pode-se então calcular que com um relógio de 3,072 MHz como o inicialmente empregado o tempo gasto para a transferência de um caracter e':

$$\frac{1}{3,072} \times 5 = 1,63 \text{ us/byte} \qquad (4)$$

Considerando que a taxa máxima de repetição do pulso de retraço horizontal de um monitor tipico (Ball Bros TV-12) e' de 15.700 ± 500 Hz e que a freqüência do pulso de retraço vertical e' necessàriamente 60 Hz para evitar

---

13 -- Velocidade máxima possivel, desprezando-se o tempo de atendimento `a interrupção, empilhamento, atualização e desempilhamento de registros.

interferência com a oscilação da rede elétrica verificamos que para mostrarmos 25 linhas de caracteres com 10 retraços por caracter (no máximo) teremos uma tela com 250 retraços. Das especificações do monitor sobre a duração do pulso de retraço vertical e das restrições do 8275 (que somente aceita múltiplos do número de varreduras necessárias para montar uma linha na especificação da duração do pulso de retraço vertical) definimos que o pulso de retraço vertical tera' a duração de duas linhas de caracteres. O número total de varreduras de cada tela sera' então:

(núm. linhas de caracter + 2) * núm. varr./linha

$$(25 + 2) * 10 = 270 \text{ varreduras/tela} \qquad (5)$$

Este número esta' dentro dos limites de freqüência horizontal exigidos pelo monitor pois:

freqüência horizontal * número de varreduras/tela

$$60 * 270 = 16.200 \text{ Hz} \qquad (6)$$

que esta' dentro do limite de 15.750 ± 500 Hz. O tempo de duração de cada retraço sera' então:

$$\frac{1}{16.200} = 61,7 \text{ microssegundos} \qquad (7)$$

Considerando que cada caracter e' transferido em 1,63 us no máximo (4) e que cada varredura demora 61,7 us (7), o maior número possivel de ser transferido nesse interregno e':

61,7

$$\frac{}{1,63} = 37 \text{ caracterees} \qquad (8)$$

que e' um valor bastante ruim, ja' que devido a limitações no projeto cada caracter e' composto por 7 pontos. A dimensão da tela fica reduzida a 259 pontos por 250 linhas (no máximo) pelo simples fato de que a UCP não consegue fornecer mais rapidamente dados ao CRTC.

A única solução viável seria o aumento da velocidade da UCP, ja' que descartamos de antemão qualquer possibilidade do emprego de um dispositivo de ADM. O microprocessador empregado (Z80A) tem uma velocidade máxima de 4 MHz; 25% maior que a inicialmente empregada (3.072 MHz). Se alterarmos o gerador do relógio para 4 MHz a máxima quantidade de caracteres que pode ser transmitida sera':

$$\frac{1}{4 \text{ MHz}} \times 5 \text{ ciclos} = 1,25 \ \mu s/\text{byte transferido} \qquad (9)$$

$$\frac{61,7}{1,25} = 49 \text{ caracteres} \qquad (10)$$

$$49 \text{ caracteres} \times 7 \text{ bits} = 343 \text{ pontos/linha} \qquad (11)$$

Seria possivel ainda empregar um microprocessador Z80B, cuja velocidade máxima e' 6 MHz. Embora essa opção provavelmente exigisse memória mais rápida (150 ns), a máxima dimensão da tela passaria a ser:

$$\frac{1}{6 \text{ MHz}} \times 5 \text{ ciclos} = 830 \text{ ns/byte transferido} \qquad (12)$$

$$\frac{61,7}{0,83} = 74 \text{ caracterees} \qquad (13)$$

$$74 \text{ caracteres} \times 7 \text{ bits} = 518 \text{ pontos/linha} \qquad (14)$$

IV.2.2.2 Opção 2 - Blocos Gráficos -

A opção anterior peca por exigir uma UCP de pelo menos 8 MHz para que a tela apresente uma dimensão aceitável, destoando completamente das demais soluções propostas nesta dissertação por exigir um aumento de força bruta de processamento enquanto todo o restante do trabalho e' dirigido no sentido de se empregar os componentes mais simples e baratos disponiveis. Procuramos portanto outro modo de implementar a expansão gráfica.

Uma caracteristica bastante comum em terminais alfanuméricos e' um gerador de caracteres de blocos semi-gráficos. Com esses blocos e' possivel desenhar figuras geométricas simples e também exibir alguns caracteres especiais. Pensamos na possibilidade de usar blocos semi-gráficos para exibir imagens, pois como cada um deles pode possuir várias linhas, a demanda de transferência de dados da memória de video para o controlador e' reduzida na mesma proporção[14]. Procuramos no entanto uma arquitetura que permita a efetiva exibição de imagens e não apenas de uns poucos contornos.

EXPANSOES DO TERMINAL PROPOSTO

Não se trata de uma tarefa fácil pois o CRTC adotado (8275) e' fortemente orientado para controle alfanumérico, empregando um duto de 8 bits inclusive para os dados da tela, ou seja, as transferências de dados entre memória de vídeo e o CRTC possuem no máximo um byte de largura. Trata-se de um arranjo conveniente para um controlador alfanumérico onde normalmente o código de caracteres exibíveis e' o ASCII, de 7 bits. No 8275 o bit mais significativo e' um indicador de byte de controle, havendo portanto apenas 128 caracteres diferentes representáveis na tela. Trata-se de um número excessivamente pequeno para distinguir todas as possíveis combinações de blocos alfanuméricos do tamanho das letras, pois com as dimensões de 5x7 ha' 2 diferentes matrizes de bits. Seriam necessários 35 bits para numera'-las biunivocamente.

Com sete bits os maiores blocos que podem ser representados são 7x1, 3x2, 2x3 e 1x7. Note-se que a opção 1 era simplesmente o caso particular com o bloco 1x7. A opção de 7x1 não se mostra conveniente devido ao

---

14 - Esta e' uma afirmação de fácil comprovação lembrando-se de que o CRTC pede um caracter para a memória de vídeo assim que termina de exibir o anterior. Como na opção anterior cada caracter ocupava apenas uma única varredura, um pedido de ADM era feito a cada varredura. Usando-se blocos de 7 linhas (por exemplo), a transferência de um novo dado somente sera' solicitada a cada 7 varreduras; durante esse período o tráfego de dados sera' restrito ao CRTC, gerador de caracteres e subsistema de geração do sinal de vídeo.

pequeno número de colunas que a tela gráfica teria, pois com esse tamanho de blocos no máximo 448 linhas por 80 colunas seriam possíveis. Como o monitor não consegue mostrar mais que 250 linhas teríamos com esse tamanho de bloco uma tela de 250x80 pontos, um formato bem pouco usual. Os blocos de 1x7 são impraticáveis sem um processador mais rápido, como discutimos no ítem anterior. Resta-nos então escolhermos entre blocos de 2x3 ou 3x2.

Ambos os tamanhos apresentam a séria desvantagem de desperdiçar um bit, pois com apenas 6 bits pode-se representar os 64 diferentes blocos dessa dimensão. Como o bit 7 ja' não e' empregado por ser usado pelo 8275, se optarmos por blocos dessa dimensão cerca de 25% das informações na memória de vídeo não serão empregadas pois apenas 6 bits em cada byte serão significativos. Mesmo assim esses blocos apresentam vantagens indiscutíveis sobre os outros pois permitem uma tela gráfica de 240x128 ou 160x192.

A tela do terminal fica então mapeada como mostrado na figura 29.

Os limites de 80 colunas e 64 linhas são impostos pelo CRTC, cujos "buffers" de linha armazenam 80 bytes e cujo contador de retraços tem 6 bits. Esta seria uma solução satisfatória não fosse pela perda de 25% da informação armazenada. Procuramos uma solução que

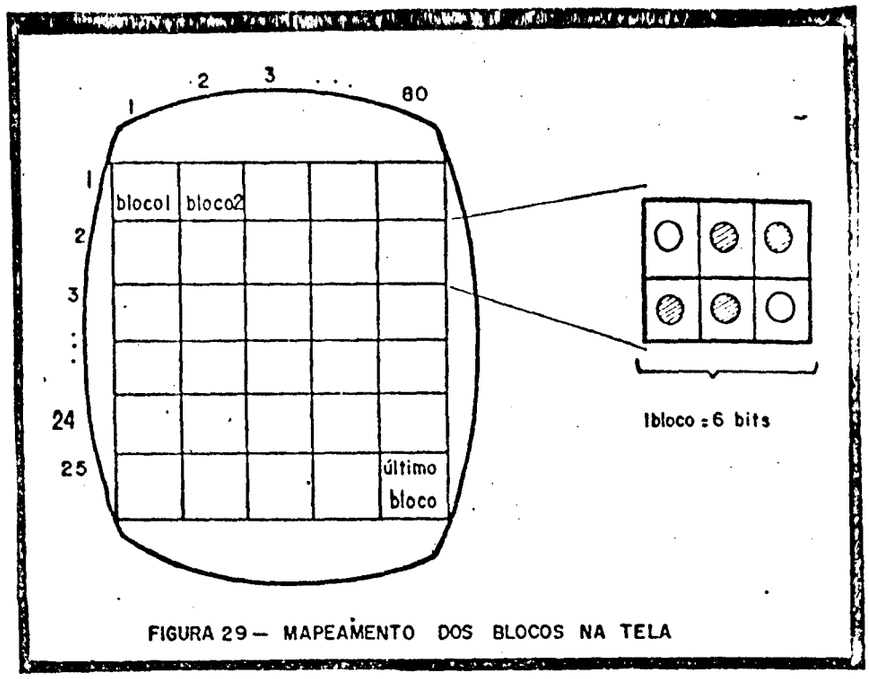


FIGURA 29 — MAPEAMENTO DOS BLOCOS NA TELA

permitisse o uso dos dois bits restantes em cada byte.

Estudando os blocos verificamos que os mesmos são divididos em dois blocos complementares, ou seja, para cada bloco existe um complementar que tem exatamente a mesma configuração de bits, porém com 0 no lugar de 1 e 1 no lugar de 0. Esta e' uma propriedade esperada, visto podermos encarar os blocos como numerais binários alinhados, como mostra a figura 30.

Ora, o CRTC e' capaz de inverter a saída de vídeo respondendo a um comando colocado entre os dados a serem exibidos (o bit 7 e' usado exatamente para distinguir esses comandos dos dados a serem exibidos). Assim, apenas metade das possíveis combinações de bits precisa ter uma representação binária biunívoca, a outra metade e' obtida mediante o comando de vídeo reverso.

EXPANSOES DO TERMINAL PROPOSTO

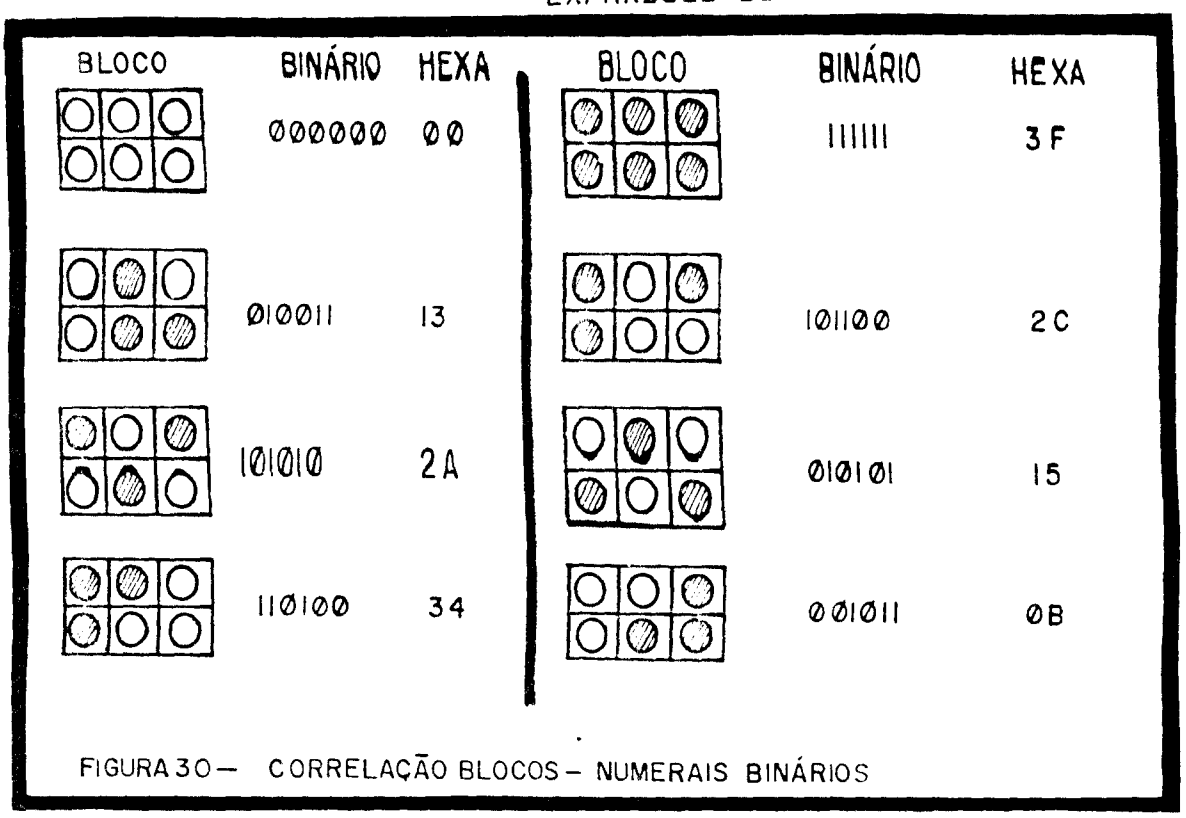


FIGURA 30— CORRELAÇÃO BLOCOS – NUMERAIS BINÁRIOS

Podemos agora pensar em blocos com a dimensão 2x4 ou 4x2, pois com apenas 7 bits podemos mapear os 128 diferentes blocos que constituem metade dos possiveis blocos com essas dimensões. A tela gráfica agora pode mostrar 320x128 ou 256x160 pontos, o que julgamos ser uma resolução razoável. O único inconveniente dessa solução e' a limitação que o CRTC impõe para o número de comandos de inversão de vídeo que uma linha pode conter. Como somente 16 desses comandos são permitidos, apenas as 16 primeiras alternâncias entre a primeira e a segunda metade do conjunto de blocos semi-gráficos serão efetuadas. Caso haja mais que essa quantidade de comandos numa linha, o programa residente do terminal se encarregara' de retirá-los da linha, substituindo os blocos complementares pelos blocos disponiveis que mais se aproximem daqueles desejados.

EXPANSOES DO TERMINAL PROPOSTO

A conseqüência deste arranjo e' que no final de cada linha da tela gráfica podem ocorrer erros de alguns bits. No caso de exibição de imagens esses erros não são significativos, pois representam menos de 1% da quantidade de informação presente na tela. Em aplicações como desenhos de figuras geométricas (comuns em CAD/CAM) esses erros são mais significativos, razão pela qual aconselhamos nesses casos o emprego do terminal gráfico completo citado anteriormente.

IV.2.3 Saida Para Video Composto -

Muitos monitores não possuem entradas separadas para sinal de video, retraço horizontal e retraço vertical. A grande maioria, com efeito, apresenta apenas uma entrada para um sinal composto por esses três eventos. A figura 30 mostra a alteração proposta por [58] para mixar as três saidas num sinal composto.

IV.2.4 Coprocessador -

Dado o pequeno custo das UCP's atualmente uma extensão de real valor seria a incorporação de um coprocessador para as tarefas mais elaboradas. A figura 32 mostra uma possivel ligação entre dois processadores, um dos quais cuidando da recepção dos caracteres pela linha serial e interpretação da linguagem de controle. Nas vezes em que o terminal e' colocado em modo local o microprocessador A recebe os dados, passa para o

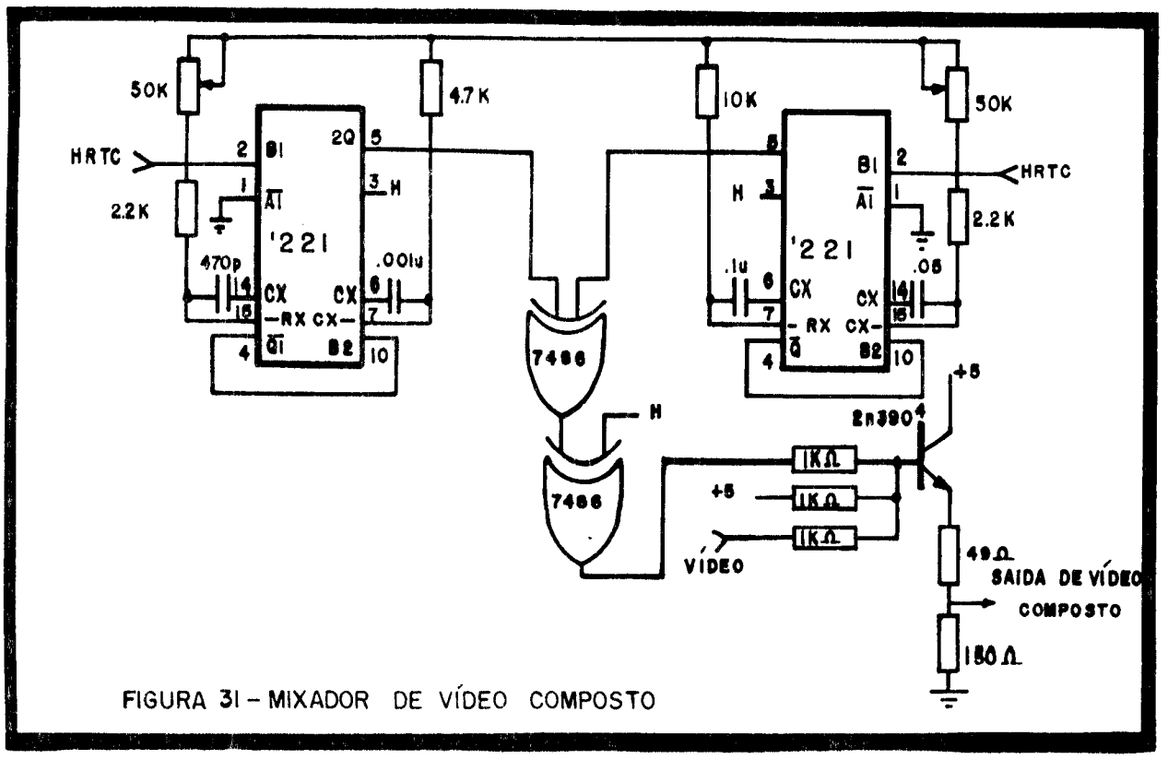


FIGURA 31 - MIXADOR DE VÍDEO COMPOSTO

microprocessador B que os interpreta e os devolve ao microprocessador A; certamnte um processo pouco prático. Entretanto, deve-se notar qe os terminais de vídeo foram projetados para operar em linha e so' ocasionalmente em modo local. Além disso, a taxa de recepção de caracteres nunca e' crítica no modo local devido `a baixa velocidade dos digitadores, que não compromete essa troca de informações entre os processadores. Ja' com o terminal operando em modo linha a velocidade de recepção pode ser extremamente alta (ate' 2.400 caracteres por segundo), exigindo uma manipulação eficiente dos caracteres.

Mesmo que a arquitetura adotada e a simulação de ADM por programação ocupem apenas 60% do tempo do processador, cumpre notar que 40% da capacidade de processamento de um Z80 não e' certamente muita coisa.

EXPANSOES DO TERMINAL PROPOSTO

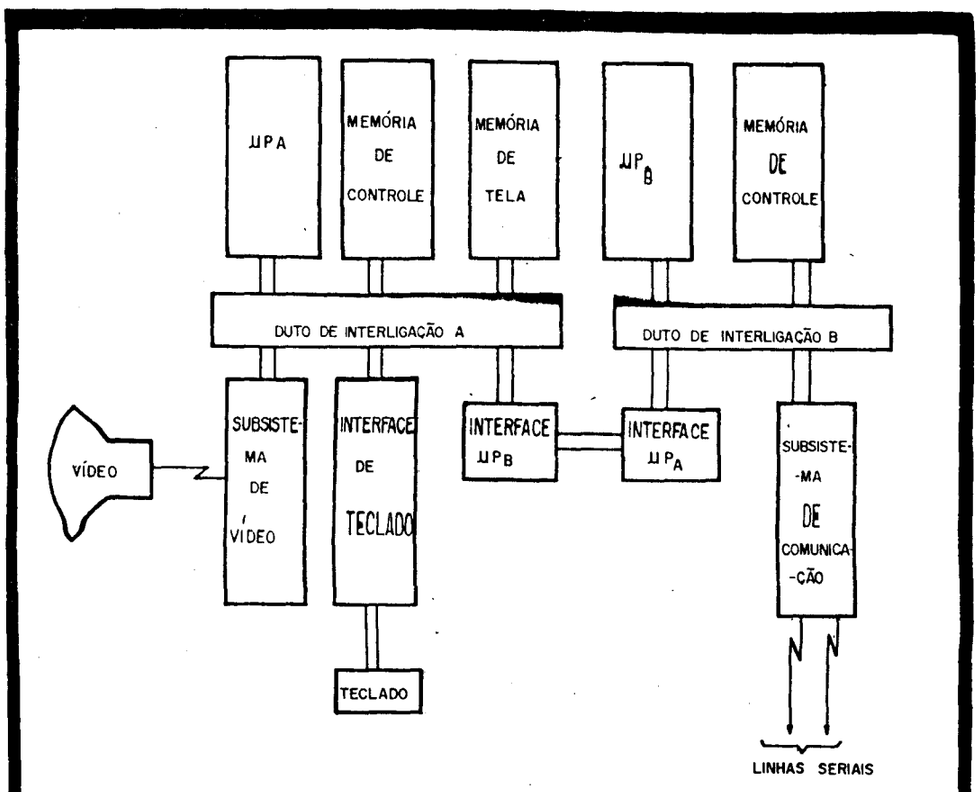


figura 32 - EXPANSÃO PARA COPROCESSADOR

Muitas das aplicações esperadas para um terminal, como a edição e interpretação local certamente esgotariam essa capacidade. Mesmo a interpretação de uma linguagem de controle muito elaborada seria suficiente para "afogar" o microprocessador. Este trabalho tornou mais que evidente a necessidade de um co-processador, e estamos fortemente inclinados a optar por um microprocessador de 16 bits como segundo processador quando continuarmos o desenvolvimennto desse dispositivo.[15]

---

15 - O microprocessador mais indicado para essa aplicação seria o 8088 ou o 80188 por apresentarem duto externo de 8 bits. Com esses integrados não seriam necessárias modificações no subsistema de memória do terminal, tornando seu emprego uma opção bastante desejável.

IV.2.5 Conexão Serial Para O Teclado -

Visando o menor custo possivel projetamos o subsistema de teclado com toda a inteligência concentrada no sistema do terminal. O teclado propriamente dito seria apenas um arranjo conveniente de chaves de dois polos sem nenhuma codificação ou capacidade adicional.

A realidade de mercado, porém, desconhece a lógica: um teclado de tecnologia capacitiva incorporando um microprocessador, memória RAM não volátil e um porto serial custa menos que um teclado de "reed switch" com um número equivalente de teclas. Assim sendo alteramos o projeto inicial para suportar esse tipo de entrada.

A modificação consiste na troca da USART adotada (Intel 8251) que pssui apenas um porto serial pela Z80-SIO, circuito integrado da Zilog que incorpora dois desses portos. Foram necessárias alterações na programação de controle, visto que os "drivers" para um e outro CI são radicalmente diferentes. Também a conexão das USART's ao sistema foi afetada ja' que o gerador de freqüência de comunicação (8253) somente possui dois canais livres. Anteriormente os mesmos eram conectados `as entradas RxC e TxC da 8251, capacitando-a a enviar e receber dados em velocidades diferentes. Como agora ha' dois portos seriaidades as entradas RxCA e TxCA do Z80-SIO foram conectadas a um canal do contador programável e as entradas RxCB e TxCB a outro. Desta

forma o terminal não pode mater taxas de recepção e transmissão diferenciadas, afastando-o um pouco mais do padrão VT-52 ou VT-100.

Uma solução fácil para o problema seria conectar as entradas TxCB e RxCB da SIO a um multivibrador de baixo custo (tipo 555 ou equivalente), ja' que a velocidade de comunicação com o teclado não necessita em absoluto ser programável. Dessa forma, os dois canais do contador programável voltam a ficar disponiveis para comunicação.

IV.2.6 Utilização De Componentes Da Ultima Geração -

Um dos aspectos mais interessantes desse trabalho e' o fato de que nenhum integrado de última geração foi empregado. A extrema simplicidade e o reduzido número de componentes do dispositivo são devidos apenas a uma arquitetura mais elaborada e uma melhor integração entre os circuitos e a programação. Como veremos adiante, se nos permitirmos o emprego de componentes mais modernos o grau de simplicidade e a capacidade do terminal serão ainda maiores. No atual instante, porém, desejamos frisar o fato de que a simples mudança de filosofia de projeto permitiu a construção de um periférico significativamente melhor sem qualquer incremento tecnológico ou de complexidade.

EXPANSOES DO TERMINAL PROPOSTO

Ja' existem circuitos integrados capazes de reduzir significativamente o número de componentes necessários para a implantação desse projeto ou aumentar grandemente suas capacidades.

CAPITULO V

PROGRAMAÇAO BASICA

V.0 PROGRAMAÇAO BASICA

O pequeno número de componentes do terminal somente foi possivel devido a complexidade das funções executadas por programação. Além do controle básico do terminal (leitura e escrita de caracteres pelo porto serial, varredura e decodificação do teclado e atualização e manipulação da memória de tela), uma série de outras tarefas não triviais tais como parte da simulação do acesso direto a memória e seqüencialização de caracteres recebidos e' efetuada por "software". Sem dúvida alguma o que mais diferencia um terminal convencional do dispositivo proposto são caracteristicas quase totalmente providas pela programação residente. O programa de controle e' bastante intrincado.

PROGRAMAÇAO BASICA

Algumas técnicas bastante apuradas foram empregadas, além de decisões baseadas em premissas diversificadas e alguns compromissos entre complexidade, rapidez e eficiência.

V.1 OS PROBLEMAS DOS PROGRAMAS DE CONTROLE

Escrever aplicativos deste tipo e' problemàtico. Tradicionalmente os implementadores desse tipo de rotinas não se preocupam com formalismo, desenvolvimento ordenado dos procedimentos, estruturação ou engenharia de programação, fato facilmente constatado analisando-se as publicações ditàticas de fabricantes conceituados [57], [58]. Mesmo sendo destinadas ao ensino e divulgação de novas técnicas, esses artigos apresentam quase que somente programação desestruturada e cheia de visiveis correções. Os programas residentes nas ROM's de muitos dos sistemas comercialmente disponiveis não são muito diferentes. Ate' mesmo produtos de última geração incorporam rotinas "remendadas" [59].

Esta prática resulta em que muito freqüentemente determinado procedimento necessita tantas alterações no processo de depuração que ao se chegar a uma versão funcional, dificilmente consegue-se reconhecer o programa original. Decisões tomadas no inicio vão paulatinamente sendo abandonadas durante a depuração, linhas de raciocinio vão sendo interrompidas, convenções de programação (como uso de registradores e posições de

memória, nomenclatura de variáveis e procedimentos, etc.) são gradativamente quebradas e pior que tudo, codificação mal costurada "apenas para testar uma idéia" acaba tornando-se parte definitiva do programa. Isto resulta em procedimentos de difícil alteração, mal documentados, desestruturados e cheios de armadilhas. Quase sempre condições de fronteira causam erros esporádicos cuja correção exigirá novos remendos. Os programadores calejados na sofrida arte de manutenção de programas residentes podem testificar a quantidade de instruções presentes "por razões históricas" na maioria das rotinas de controle.

Um dos mais importantes alvos deste trabalho foi a tentativa de se efetuar inteiramente um projeto evitando esses problemas, infelizmente tão comuns. Para tanto optamos por um desenvolvimento ordenado, implementando apenas poucas funções de cada vez, testando-as e apenas então incorporando-as ao programa controlador. A filosofia seguida foi a "bottom-up"; começamos pela análise e implementação das rotinas mais elementares e as usamos após estarem depuradas e confiáveis como blocos para a construção de outras mais complexas. Este e' o método mais comumente empregado quando ha' desenvolvimento integrado de "hardware" e "software" como no caso, pois começando-se pelas rotinas mais simples e' possivel testar os circuitos concomitantemente à programação. Inversamente, quando se desenvolve

PROGRAMAÇAO BASICA

"software" para um projeto com circuito já implementado e' mais facil ater-se `as especificações originais empregando-se a técnica "top-down", definindo inicialmente o sistema em termos genéricos e detalhando-o através de refinamentos sucessivos.

Foi nosso alvo também manter ordenada e coerente a documentação ao longo dos três anos de duração do trabalho, bem como escrever código bem estruturado, livre de erros de fronteira e de condiçoẽs especiais, expansivel e limpo.

V.2 DESCRIçÃO DAS ROTINAS

Passaremos a descrever em seguida as rotinas componentes do sistema básico do terminal, por ordem de elaboração.

V.2.1 Rotina De Inicialização -

A primeira rotina a ser escrita teve dois propósitos: Testar a funcionalidade do sistema e inicializar os dispositivos periféricos.

A execução correta desta rotina permite considerável grau de confiança no bom funcionamento dos diversos subsistemas do terminal, pois e' necessário que:

1. As tensões de +5V, +12V e o terra estejam satisfatórias e chegando a considerável parte dos componentes do protótipo.

2. O subsistema de inicialização ("reset") esteja funcionando a contento.

3. O subsistema de relógio esteja gerando as freqüências esperadas.

4. A decodificação de endereços esteja funcionando.

5. A UCP seja capaz de ler dados armazenados na EPROM.

6. A UCP tenha acesso ao CRTC corretamente.

7. A UCP tenha acesso correto ao gerador de velocidade de comunicação.

8. A UCP tenha acesso correto ao porto serial.

9. O gerador de velocidade de comunicação esteja recebendo o sinal correto de relógio e dividindo-o corretamente, conforme as instruções da UCP.

10. O porto serial esteja recebendo todos os sinais necessários ao envio de dados.

Desde o inicio do desenvolvimento sempre que enfrentamos problemas com o "hardware" a primeira ação era invariavelmente tentar executar este programa para verificar se o circuito "continuava vivo". Esta mesma rotina sem o ciclo de envio de caracteres e' empregada como rotina de inicialização dos componentes periféricos.

Essa e todas as outras rotinas foram escritas em linguagem de máquina do microprocessador Z80. Todavia antes da codificação uma descrição da mesma em linguagem tipo Pascal foi feita para documentação e análise. Nas ocasiões necessárias violamos a sintaxe da linguagem original, introduzindo comandos que descrevessem a ação desejada, embora tenhamos procurado nos ater ao Pascal

padrão sempre que possivel. Optamos por este caminho por crermos que o Pascal e' um denominador comum entre os provàveis interressados num trabalho deste tipo.

No nosso pseudo-Pascal a rotina de envio de caracteres fica[16] :

```
PROCEDURE Teste_U;
BEGIN
  Inibir_Interrupcoes_CRTC;
  Inicializar_Contadores(4800);
  Inicializar_PortoSerial;
  REPEAT
    Transmitir("U");
  UNTIL FALSE
END {Teste_U};
```

Detalhando as ações que compõem a Teste_U temos:

```
PROGRAM Teste_U(INPUT,OUTPUT);
CONST
  ocupado = 00000001b;

VAR
  a : registro;

BEGIN
  { Programa os CI's periféricos }
  OUT(dad_8275,cmd_para);
  OUT(c0_8253,prg_8253);
  OUT(c0_8253,baud_l)
  OUT(c0_8253,baud_h);
  OUT(dad_8251,cmd_8251);
  OUT(dad_8251,mod_8251);

  { Ciclo de envio de um caracter }
  REPEAT
    OUT(dad_8251,"u");
    a := IN(est_8251);
    WHILE ((a AND ocupado) <> 0) DO
      A := IN(est_8251)
  UNTIL FALSE
```

---

16 - Enviamos a letra U porque seu código ASCII e' 55H ou seja, 01010101 em binário, tratando-se de uma onda quadrada fácil de gatilhar no osciloscópio.

END {Teste_U}.

No apêndice D encontra-se o corpo da rotina Teste_U como um programa diagnóstico, codificado em linguagem montadora Z80 e executável isoladamente ("stand alone").

V.2.2 Rotina De Simulação De ADM -

A rotina Teste_U e' um procedimento genérico, pois pode ser empregada na maioria dos microcomputadores. A rotina seguinte entretanto e' totalmente especifica para o terminal: o atendimento `a interrupção gerada pelo circuito de simulação do acesso direto `a memória.

Como esta e' uma das contribuições originais desta dissertação e e' extremamente dependente do particular modo de implementação do circuito simulador, foi bastante estudada antes da implementação. Várias tentativas foram feitas antes de se chegar a uma solução satisfatória; descreveremos agora as principais decisões juntamente com as razões que nos levaram a tomá-las.

V.2.2.1 Tratamento Das Interrupções -

A primeira questão envolve qual o tipo de interrupção atendida. O controlador de video adotado gera dois tipos de pedidos de atenção da UCP: um sinal solicitando dados (o pedido de ADM, DRQ/) e um informando que chegou ao final da tela (IRQ/). O primeiro e' gerado

sempre que o feixe está prestes a iniciar uma série de varreduras referentes a uma nova linha de caracteres, e espera-se que em resposta a esse evento algum subsistema transfira para o controlador a quantidade de caracteres necesária `a exibição de uma nova linha. O IRQ/ e' gerado quando o feixe atinge a última linha de varredura da tela, sendo empregado para disparar os procedimentos caracteristicos dessa situação, tais como reinicialização do subsistema de ADM (ja' que quando ocorrer o próximo DRQ/ não sera' mostrada a linha de caracteres seguinte mas sim a primeira linha da tela), procedimento de deslocamento vertical da tela ("scroll"), re-posicionamento do cursor (que podera' voltar ao canto superior direito ou permanecer na última linha, dependendo da aplicação), etc.

Como o microprocessador empregado (Z80) possui duas entradas de interrupção a primeira idéia foi usar uma para cada um dos sinais gerados pelo CRTC. Esta opção mostrou-se entretanto inadequada devido ao fato do CRTC gerar interrupções antes de ser programado. Sendo a inicialização do dispositivo feita por programação (o 8275 não tem uma entrada de "reset") não ha' modo de impedir a emissão de alguns pedidos de interrupção antes que o microprocessador central emita um comando de desabilitação. A UCP mantém inibido o reconhecimento de interrupções até o final da execução da primeira instrução após um "reset", logo, se a mesma for um DI

("Disable Interrupt"), as interrupções ficarão mascaradas. Infelizmente apenas uma das entradas de interrupção do Z80 e' mascarável, a outra (NMI) e' sempre atendida. Assim, ocasionalmente ao ligarmos o sistema o processador atendia a um pedido de interrupção espúria gerada pelo CRTC não inicializado. O atendimento dessa rotina implicava numa transferência de dados para o controlador que ao serem interpretados como comandos desgovernavam-no completamente. Na prática, a cada 10 ou 12 vezes que o terminal fosse ligado uma vez o mesmo não funcionaria, apresentando uma tela completamente confusa ou simplesmente permanecendo mudo.

Várias alternativas foram estudadas para o problema, todas apresentando algum inconveniente. Inicialmente projetamos o circuito da figura 33 que efetuava o mascaramento da linha NMI através de comandos do microprocessador, solução considerada inadequada devido ao aumento do número de CI's do projeto. Perseguimos o alvo de número minimo de componentes com bastante tenacidade, hesitando bastante antes de acrescentar uma única porta.

Mascaramento da linha NMI

Pelo mesmo motivo descartamos o emprego de um controlador dedicado tipo Z80-CTC ou 8259 para as interrupções (figuras 34 e 35), pois embora se tratasse da inclusão de apenas mais um componente, seu custo excedia os 4 que teriamos que colocar na proposta

BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

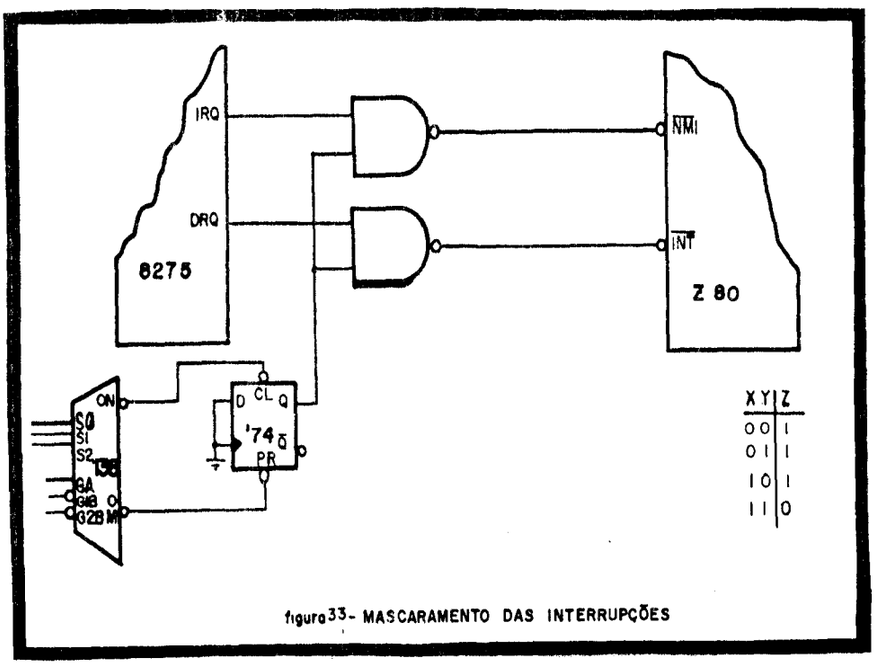


figura33- MASCARAMENTO DAS INTERRUPÇÕES

anterior. A principal razão pela qual desejávamos um terminal com poucos componentes era manter baixo seu custo.

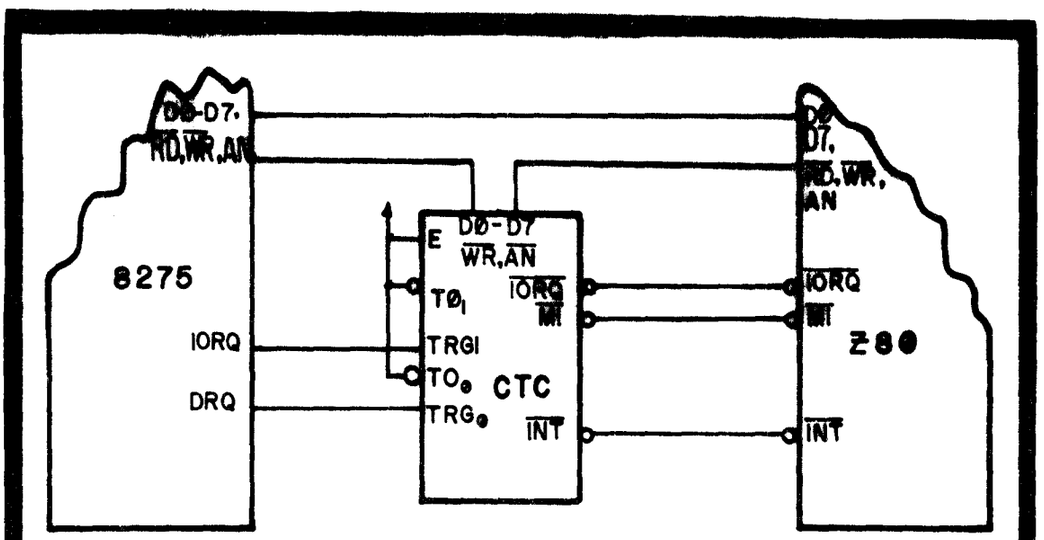


FIGURA 34— COLOCANDO UM CTC

Uma opção que exigia menos CI's era a junção das duas interrupções num so' sinal, conectado `a entrada mascarável. Em atendimento a esta interrupção a UCP faria uma varredura nas linhas IRQ e DRQ do CRTC e

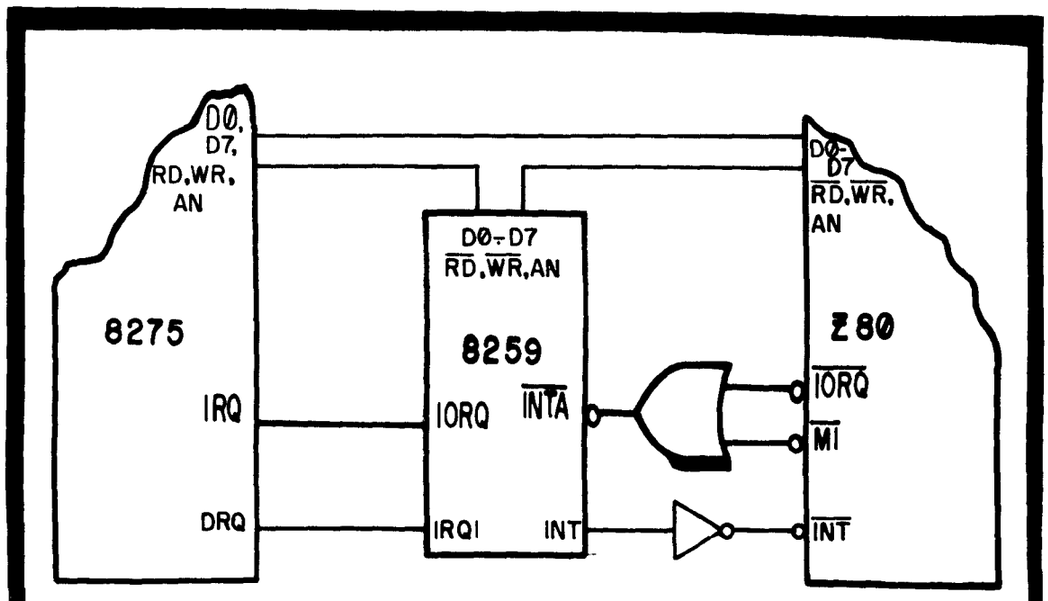


FIGURA 35 – COLOCANDO UM 8259

verificaria qual delas causara a interrupção (figura 36). Infelizmente o 8275 não mantém essas saidas (especialmente o DRQ) ativas durante o tempo necessário para sua leitura pelo microprocessador por instruções IN. Caso colocássemos registradores como na figura 37, novamente incrementariamos de 4 a quantidade de componentes.

Combinando-se as interrupções

Juntado-se as interrupções e registrado-as

Ao invés de registrar as saidas IRQ e DRQ do 8275 e' possivel determinar pelo exame dos registradores de estado do CRTC qual das duas interrupções foi acionada. Embora quase tenhamos optado por essa alternativa, esse método apresenta um inconveniente sério: cada byte de instrução adicionado `a rotina de tratamento de interrupção aumenta consideravelmente a sobrecarga ("overhead") da UCP. E acrescentamos muito mais que um

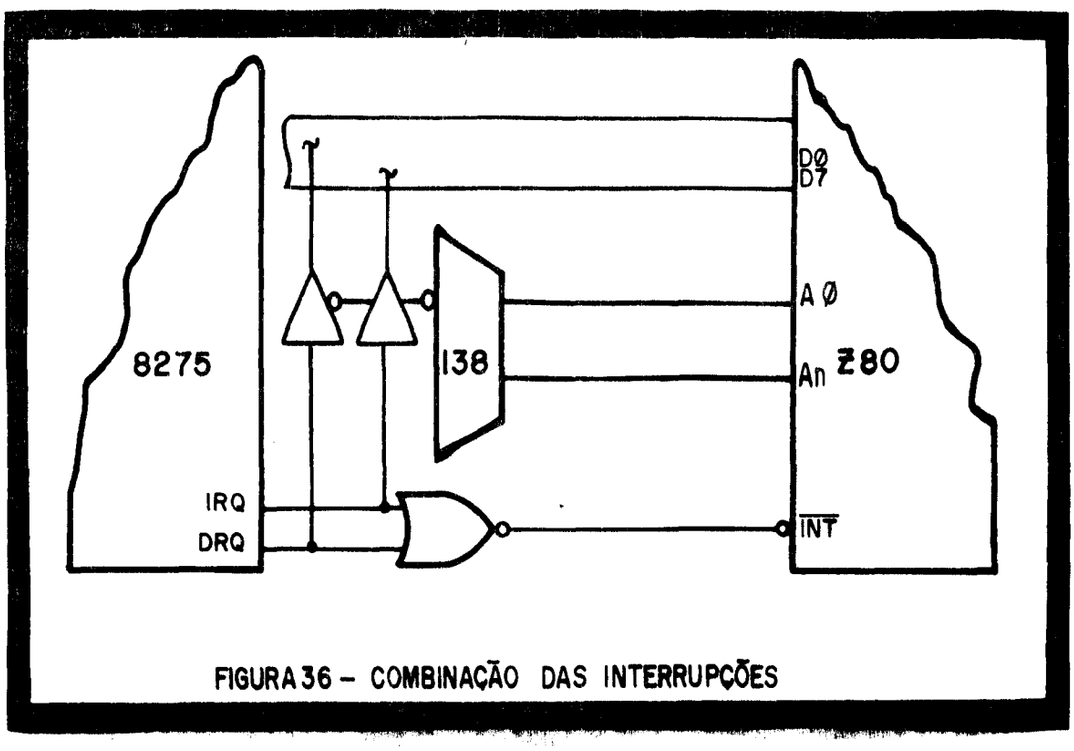


FIGURA 36 – COMBINAÇÃO DAS INTERRUPÇÕES

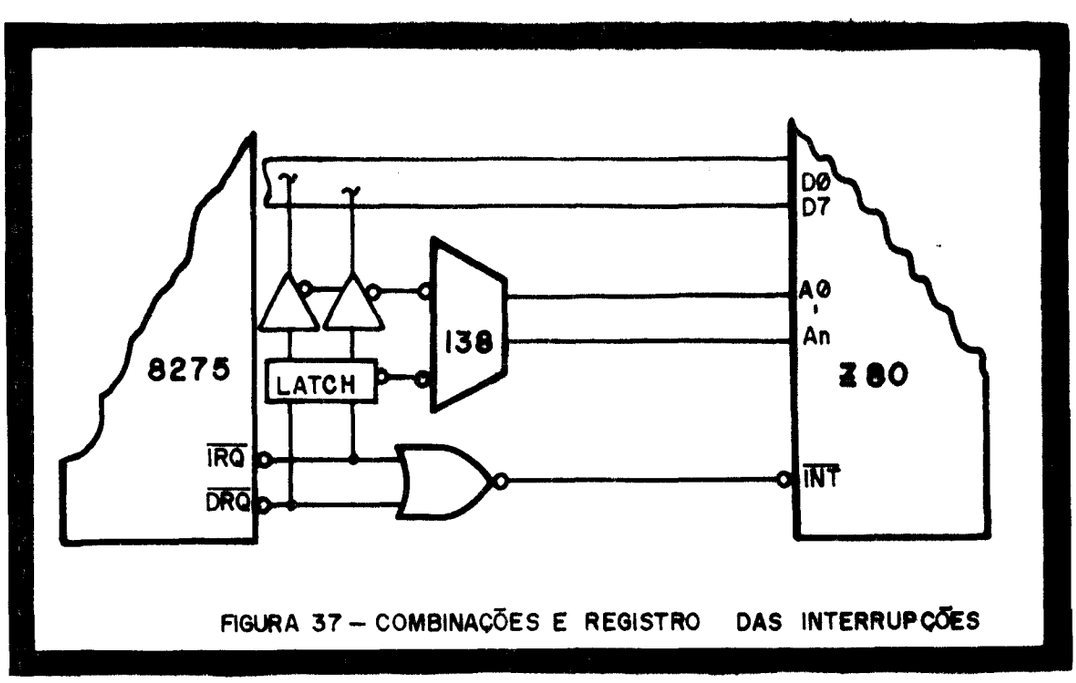


FIGURA 37 – COMBINAÇÕES E REGISTRO DAS INTERRUPÇÕES

byte, pois e' necessário desabilitar as interrupções, ler registradores de estado do CRTC, testar a ocorrência de determinados bits nos dados lidos e invocar a rotina de

tratamento apropriada ao evento, num total de mais de 12 bytes perfazendo um atraso de aproximadamente 20 us. Pode não parecer um tempo excessivamente grande, considerando-se porém que a tela e' re-escrita a cada 16.600 us e que em cada uma delas ha' 25 pedidos de ADM (um para cada linha), o atraso numa tela seria:

20 x 25 = 500 uS

Em cada periodo de 16,6 ms (uma tela) cerca de 7,5 ms são gastos no processo de transferência de dados da memória de video para o controlador. Restam portanto aproximadamente 9,1 ms para trabalho útil. O atraso devido `a inclusão de apenas seis ou sete instruções no corpo da rotina provoca um decréscimo de 5% no tempo deixado disponivel para outras aplicações. E' justificável portanto a relutância em aumentar o número de instruções da rotina de atendimento de interrupção.

Procurando uma solução mais aceitável verificamos que deduzindo-se a condição de final de tela sinalizada pela interrupção IRQ e' possivel dispensar este sinal proveniente do CRTC. Como este sinal ocorre a cada 24 linhas transferidas um simples contador faz todo o trabalho necessário. O código para manutenção desse contador, mostrado abaixo, ocupa apenas 4 bytes, demorando 6,6 us.

```
                .
                .
                .          (corpo da rotina DRQ)
                .
                .
        LXI     H,Cont_Lin
```

```
DCR     M
RZ
        .
        .
        .          (corpo da rotina IRQ)
        .
        .
RET
```

Parece dificil conseguir algum tipo de tratamento de interrupção mais eficiente que este, pois além de não implicar em nenhum acréscimo de circuitos (ja' que agora ha' apenas a interrupção DRQ ligada `a entrada mascarada do microprocessador), a redução no tempo de processamento útil e' de somente 1,8%.

V.2.2.2 Simulação Da Memória De Tela -

Como neste estágio desejamos apenas testar o mecanismo de simulação de ADM através de interrupção, a estrutura de dados da memória de vídeo sera' totalmente ignorada. Definimos na região de dados do programa uma seqüência de 80 caracteres "E" e os varremos continuamente para todos os pedidos de transferência de ADM. Assim, a tela ficara' com "E"'s em todas as posiçoẽs de todas as linhas.

V.2.2.3 Estrutura Da Rotina De Atendimento `a Interrupção -

O algoritmo da rotina de atendimento `a interrupção para simulação de ADM fica:

```
PROCEDURE Interrup(p_exibe:INTEGER;linha:t_linha);
VAR
  sp : registro16;
  sp_salvo : INTEGER;

BEGIN
  salva_ponteiro_pilha(sp,sp_salvo);
  troca_ponteiro_pilha(sp,p_exibe);
  sinaliza_inicio_adm(sp);
  transfere80(sp,linha);
  restaura_ponteiro_pilha(sp,sp_salvo)
END {Interrup};
```

A variável p_exibe (ponteiro de exibição) e' colocada como parâmetro de entrada por ser inicializada externamente por outra rotina, responsável pela inicialização de toda a estrutura de dados da memória de vídeo. Após a primeira interrupção P_Exibe e' atualizado apenas pelo procedimento Interrup. Como esta parte do código ainda não e' relevante, a inicializacão de P_Exibe fica solta, fora da rotina. Note-se que pelas mesmas razões não ha' ainda a detecção da condição da interrupção IRQ/, pois como não ha' estrutura de dados da memória de vídeo não faz sentido falar em última linha da tela. Raciocínio análogo e' válido para o parâmetro Linha.

Explicitando as ações delineadas no algoritmo anterior, temos:

```
PROCEDURE Interrup(p_exibe:INTEGER;linha:t_linha);
VAR
  sp, bc : registro16;
  sp_salvo : INTEGER;

BEGIN
```

```
  sp_salvo := sp;
  sp := p_exibe;
  {SP aponta p/ o mesmo endereco, mas com A15=1}
  sp := sp + 8000h;
  POP(bc);
  POP(bc);
    .
    .
    .
  POP(bc);
  POP(bc);
  sp := sp_salvo;
END {Interrup};
```

O programa que testa o bom funcionamento da rotina esta' listado em linguagem montadora no apêndice E. Abaixo mostramos sua estrutura empregando o Pascal livre:

```
PROGRAM Teste_E(INPUT,OUTPUT);
TYPE
  t_linha : ARRAY[1..80] OF CHAR;

VAR
  p_exibe : INTEGER;
  linha : t_linha;

PROCEDURE Interrup(p_exibe:INTEGER;linha:t_linha);
BEGIN
  .
  .
  .
END {Interrup};

PROCEDURE Inicializar;
BEGIN
  .
  .
  .
END {Inicializar};

BEGIN
  Inicializar(p_exibe,linha);
  {A intruçãO seguinte nãO e' codificada, sendo
   automaticamente executada pelos circuitos da UCP
   em resposta a uma interrupçao.}
  IF (InterrupçãO)
    THEN Interrup(p_exibe,linha)
END.
```

A rotina Interrup e', como sugere o nome, acionada por uma interrupção solicitada pelo CRTC programado para exibir uma tela de dados com 25 linhas de 80 caracteres.

V.2.3 Programação Do Subsistema De Teclado -

O subsistema de teclado possui, em sua versão mais simples, um grau de automação bastante limitado, constituindo-se apenas de portos paralelos divididos em um grupo de linhas de varredura e outro de linhas de retorno.

Nas linhas de varredura são escritos sucessivamente os padrões 11111110, 11111101, 11111011, 11110111, 11101111, 11011111, 10111111 e 01111111 (padrão de zero circulante), sendo lido nas linhas de retorno se houve alguma tecla pressionada. A detecção de uma tecla e' feita sabendo-se que as linhas de retorno estão todas ligadas a Vcc através de um resistor, ou seja, quando não ha' teclas pressionadas o padrão presente nessas linhas e' sempre 11111111. Quando uma determinada tecla e' pressionada, no instante em que o padrão de zero circulante apresenta um 0 na linha de varredura associada `aquela tecla uma das linhas de retorno sera' diferente de 1. A partir da posição do bit em 0 nas linhas de retorno e do padrão de varredura pode-se descobrir qual a tecla pressionada (figura 38).

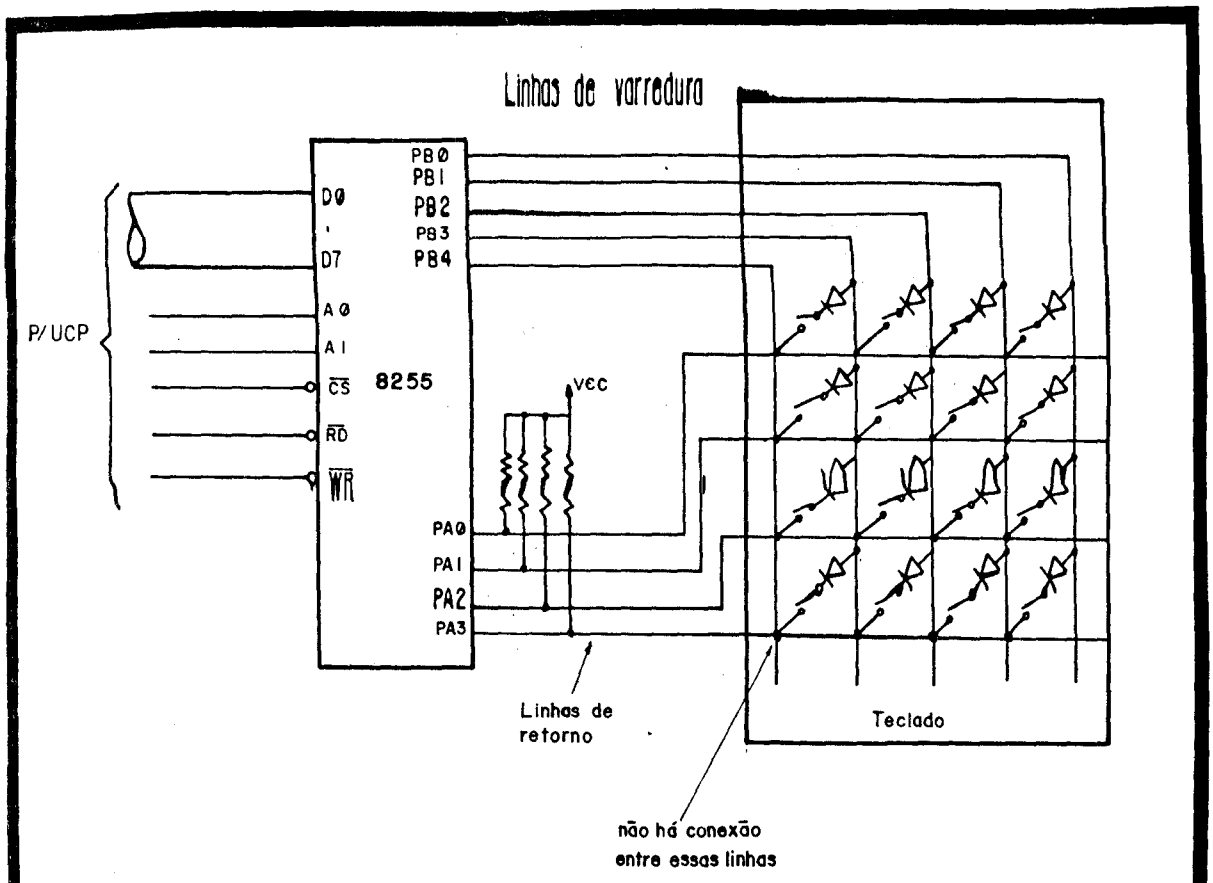


FIGURA 38 — SUBSISTEMA DE VARREDURA

A supressão dos pulsos de ruido caracteristicos do chaveamento eletro-mecânico e' feita exigindo que uma determinada leitura seja feita "n" vêzes antes de ser validada, onde "n" e' dependente das caracteristicas das chaves. Uma expansão bastante interessante e' a incorporação de um microprocessador dedicado para o processamento do teclado, tal como discutido no item IV.2.5. Por enquanto esse processamento e' feito pela UCP central na forma de um módulo executado alternadamente com os demais procedimentos componentes do programa controlador do terminal, podendo ser transformado num processo independente escalado de tempos em tempos quando for implementado o núcleo de tempo real tal como descrito em um dos parágrafos seguintes desta obra. O procedimento de tratamento do teclado tem a

```
forma:
FUNCTION Le_Tecl(in1_varr,in2_varr:INTEGER):BYTE;
VAR
  b, c, d : REGISTER;
  achou : BOOLEAN;

BEGIN
  b := in1_varr;
  c :=  0FFh;
  OUT (p2_varr, c);
  achou := FALSE;

  { Faz a varredura de todas as linhas }
  WHILE ((NOT carry) AND (NOT achou)) DO
  BEGIN
    { envia varredura e le resposta }
    OUT (p1_varr, b);
    d := IN(p_Leit);
    { mascara bits de MAIUSC, CNTRL e TRAVA }
    IF ((NOT d) AND 1Fh <> 0)
      THEN achou := TRUE;
      ELSE { atualiza e envia nova varredura }
           b := RRC(b)
  END;

  { Passa a varrer linhas do porto C }
  IF (NOT carry)
    THEN BEGIN
           { Desliga porto B }
           b := 0FFh;
           OUT (p1_varr, b);
           c := in2_varr;
           IF (in2_varr <> 0FFh)
             THEN WHILE ((NOT carry) AND (NOT achou)) DO
                    BEGIN
                      { Comeca a ler colunas do porto C }
                      OUT (p2_varr, c);
                      d :=IN(p_leit);
                      IF ((NOT d) AND 1Fh) <> 0)
                        THEN achou := TRUE
                        ELSE c := RRC(c)
                    END
             ELSE { nao achou:  retorna com erro }
                  d := 0;
         END;
  Le_Tecl := d
END {Le_Tecl};
```

V.2.4 Estruturação Da Memória De Tela -

A estrutura de dados da memória de tela tem significativo impacto na performance do terminal, pois algumas de suas ações mais básicas e corriqueiras como deslocamento de tela para cima e para baixo podem exigir continua movimentação de dados na memória caso sua estruturação não seja apropriada. Descreveremos a seguir as principais estruturas estudadas no decorrer deste trabalho.

V.2.4.1 Posicionamento Fixo -

A primeira tentativa foi simplesmente colocar as linhas em posições fixas da memória. Esta estrutura tem a caracteristica de facilitar muito o posicionamento aleatório do cursor, pois cada linha esta' sempre no mesmo endereço, como pode ser visto na figura 39.

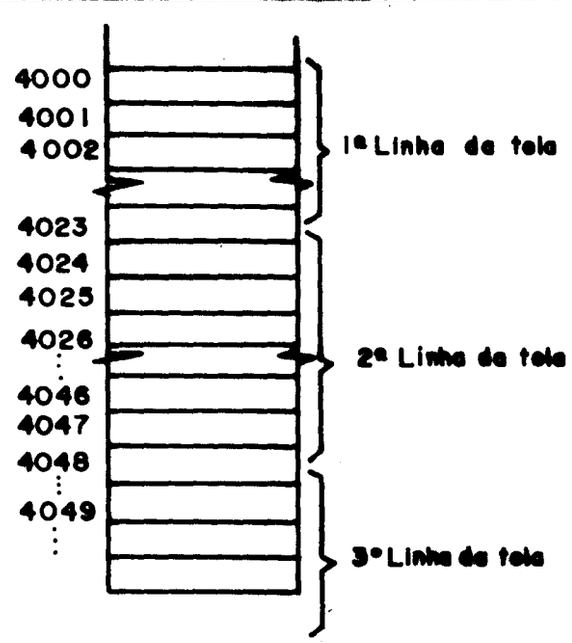


FIGURA 39 - POSICIONAMENTO FIXO

Entretanto, para se fazer o deslocamento da tela ("scroll") e' necessário deslocar todas as linhas uma posição para cima ou para baixo, numa transferência de 2k bytes. Esse esforço computacional e' demasiado grande para o microprocessador empregado, especialmente se levarmos em conta a freqüência com que essas operações são invocadas. Assim, embora com méritos de simplicidade e facilidade de endereçamento, esta estrutura foi abandonada.

V.2.4.2 Linhas Relocáveis -

Para solucionar a dificuldade imposta pelo deslocamento de tela ("scroll") implementamos um esquema especialmente projetado para facilita'-lo. Esta estrutura e' conhecida como "buffer circular", podendo se representada como na figura 40.

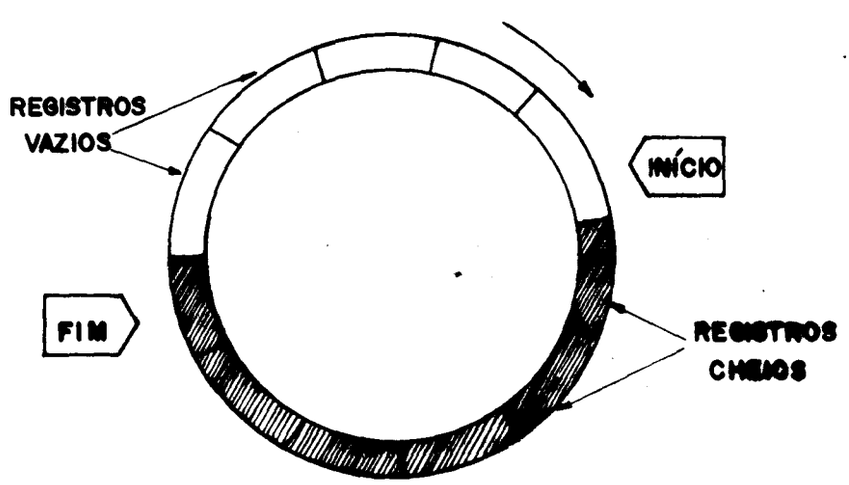


FIGURA 40— "BUFFER" Circular

PROGRAMAÇAO BASICA

A primeira linha e' apontada pelo ponteiro Inicio sendo as 23 seguintes retiradas e exibidas a partir dela. Basta portanto que o ponteiro de linha seja alterado para se efetuar o deslocamento de toda a tela, numa operação 4.000 vêzes mais eficiente que a da estrutura anterior (uma escrita versus 2.000 leituras e 2.000 escritas). Os dois problemas dessa estrutura são a implementação do "buffer" circular propriamente dito e o posicionamento direto do cursor.

A implementação do "buffer" circular não e' direta devido ao fato do espaço de memória ser endereçado linearmente. O próximo registro pode, desta forma, não ser contiguo a seu antecessor como mostrado na figura 41. A operação de apontar para o registo seguinte envolve então um teste para verificação de adjacência, implicando numa relativa perda de eficiência (recordem-se as conclusões do item V.2.2.1 nas quais um pequeno incremento de tempo neste trecho de programa provoca significativos decréscimos de eficiência em todo o sistema).

O posicionamento direto do cursor fica bastante dificultado pois não ha' nenhuma correlação entre o endereço fisico das linhas e sua numeração lógica. A solução de calcular a cada deslocamento de tela a posição fisica do cursor (que também e'alterada nesta operação) a partir do ponteiro PrimLin, embora implementável sem maiores dificuldades, e' bastante ineficiente,

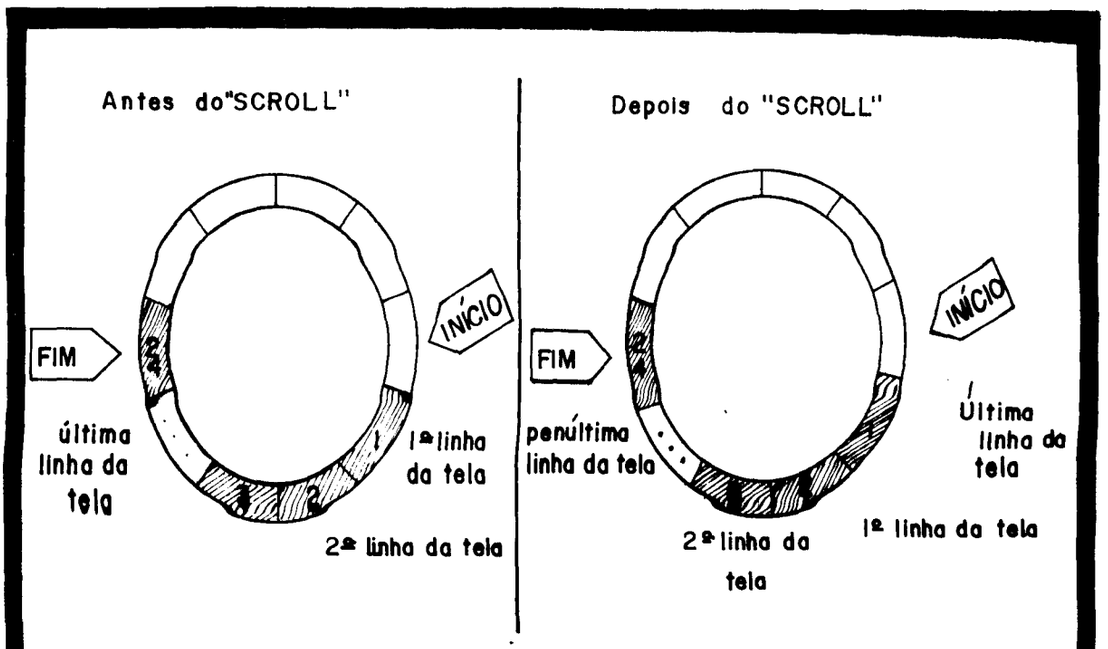


FIGURA 41 — NÃO CONTINGÜIDADE

especialmente se o cursor se mantiver com freqüência nas últimas linhas da tela, como normalmente ocorre.

Procuramos solucionar esta dificuldade tirando partido da enorme eficiência da operação de deslocamento de tela desta estrutura, mantendo um ponteiro para a posição física do cursor além dos contadores para linha e coluna lógicas. Este ponteiro e' atualizado a cada deslocamento de tela e como esta e' uma operação bastante ràpida, duas ou quatro escritas não afetarão muito profundamente sua eficiência. Embora não resolva o problema do posicionamento do cursor em posições aleatórias da tela, estes ponteiros auxiliam muito o càlculo do posicionamento do cursor nos deslocamentos da mesma.

PROGRAMAÇÃO BASICA

Três problemas ficam pendentes nessa estrutura: O posicionamento direto do cursor, ja' discutido, outro decorrente do particular CI controlador de video empregado e um terceiro por causa de uma das caracteristicas desejadas do terminal.

O 8275 possui a capacidade de definir alguns campos com caracteristicas especiais, tais como sublinhado, piscante, video reverso, outros geradores de caracter, etc. As marcas especificando o inicio e o fim desses campos são bytes de caracteres não exibiveis inseridos na memória de tela em posições apropriadas. Devido a este fato os registros da lista circular passaram a ter tamanho variável ao invés dos 80 bytes fixos que compõem a linha. Para podermos empregar essas capacidades seria necessário alterar a estrutura de dados para suporte de registros de tamanho variável, o que embora diminuisse a eficiência do programa, não se configurava particularmente dificil; ou então empregar uma outra estrutura mais adequada `as carateristicas da aplicação (facilidade de deslocamento da tela, e linhas de tamanho variável).

A dificuldade devida `a caracteristica do terminal e' de solução mais complexa. Desejava-se capacitar o dispositivo em construção a fazer deslocamentos não apenas da tela inteira mas também de porções da mesma, podendo deslocar tanto para cima como para baixo algumas linhas sem alterar o restante da tela. Para tanto, ou

transferiamos o conteúdo de algumas linhas para outras, (com baixissima eficiência), ou implementariamos a lista circular como uma lista encadeada, facilitando a relocação e registros pela alteração dos ponteiros de ligação entre os mesmos (figura 42).

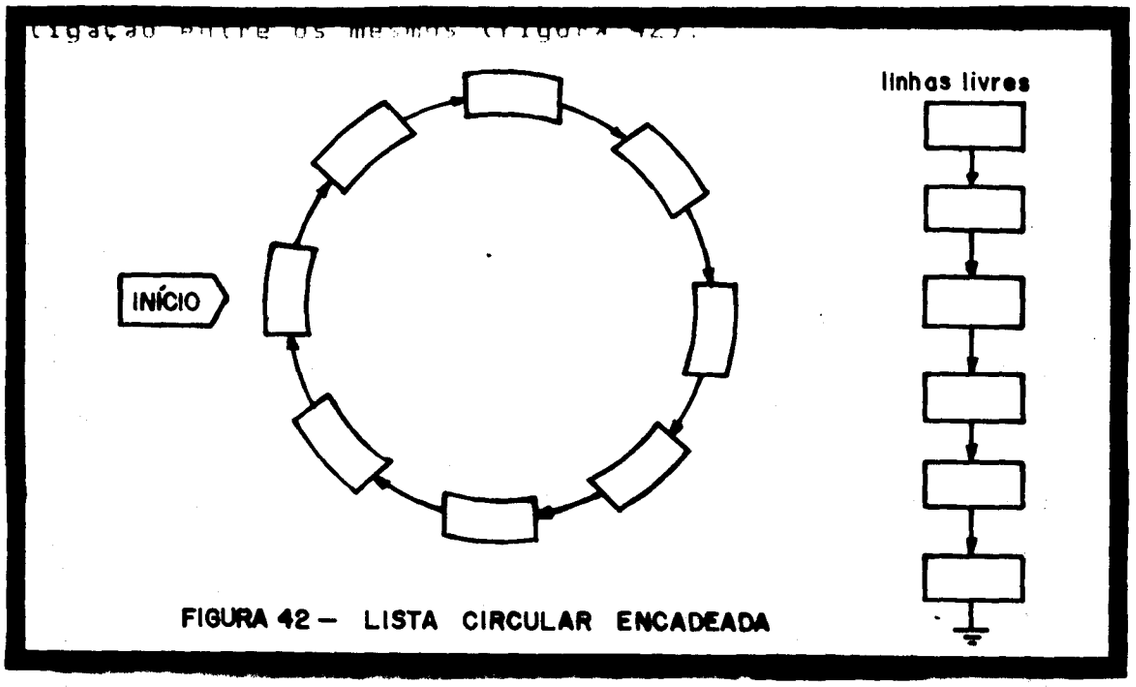


FIGURA 42 — LISTA CIRCULAR ENCADEADA

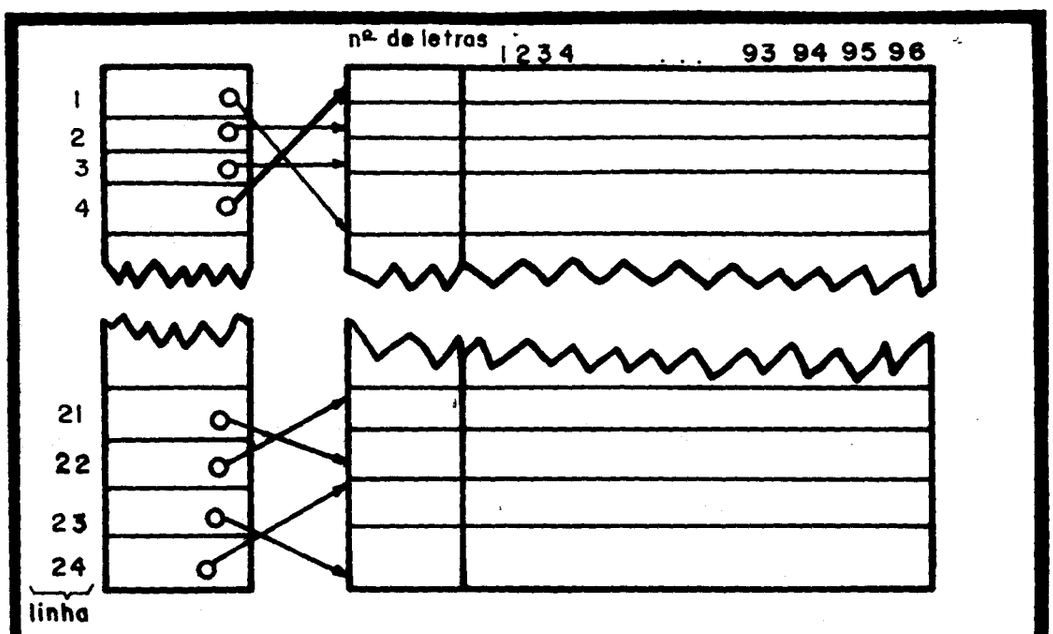
Neste ponto começamos a pensar seriamente numa estrutura alternativa pois a lista circular com registros de tamanho variável e encadeada ja' demandava quase tanto esforço computacional quanto a estrutura de dados de endereço fixo.

V.2.4.3 Registro Composto -

Elaborou-se então uma estrutura que facilitasse a manipulação dos dados da tela com as caracteristicas estudadas anteriormente, `a qual demos o nome de registro composto.

PROGRAMAÇAO BASICA

Este registro e' composto por 24 linhas de dois campos cada, o primeiro indicando a quantidade de bytes da linha e o segundo apontando para a posição num segundo arranjo de 24 linhas por 96 colunas onde estão armazenados os dados correspondentes `a linha de video cuja numeração lógica segue a numeração fisica das linhas do registro.



FIGURA— 43 — REGISTRO COMPOSTO

Embora pareça uma estrutura pouco flexivel, verifica-se que a manipulação de no máximo 24 ponteiros permite qualquer tipo de deslocamento de tela, total ou parcial. A separação da área de dados dos ponteiros permitiu a fixação de seu comprimento, que e' de 96 bytes devido `ao fato do 8275 aceitar no máximo 16 caracteres de controle por linha de video (80 caracteres exibiveis + 16 caracteres de controle = 96 bytes). E' procedente a

critica de que esta estrutura não tem um aproveitamento ótimo de espaço, pois no final de cada linha pode haver bytes desocupados que não podem ser empregados para outras finalidades. Entretanto, a facilidade e eficiência na manipulação de registros de tamanho variável compensam amplamente esses inconvenientes.

V.2.4.3.1 Inicialização -

Definida a estrutura de dados a ser empregada, projetou-se uma rotina que a testasse. Este programa inicializa as linhas com um valor constante, os contadores de linha e coluna para o canto superior esquerdo e os ponteiros de linha para cada uma das linhas previamente inicializadas. Esse algoritmo esta' descrito abaixo, e a listagem do programa linguagem montadora do Z80 esta' no apêndice F.

```
PROGRAM Teste_Tela(INPUT,OUTPUT);
CONST
  maxlin = 24;
  maxcol = 96;

TYPE
  t_pontlin : ARRAY[1..maxlin] OF INTEGER;
  t_tela : ARRAY[1..maxlin,1..maxcol] OF CHAR;

VAR
  tela : t_tela;
  pontlin : t_pontlin;
  lincor, colcor, linmostrar, pontins : INTEGER;

PROCEDURE LigaPont(VAR pontlin:t_pontlin);
VAR
  hl : REGISTER;
  i : INTEGER;
BEGIN
  hl := 1;
  FOR i:=maxlin DOWNTO 1 DO
  BEGIN
```

```
    pontlin[i] := hl;
    hl := hl + maxcol;
  END;
END {ligapont};

PROCEDURE IniTela(VAR pontlin:t_pontlin; VAR
          tela:t_tela, VAR lincor, colcor,
          linmostrar, pontins : INTEGER);
VAR
  hl : REGISTER;
  ix : REGISTER;
  b, c : REGISTER;

BEGIN
  LigaPont(pontlin);
  ix := ^pontlin;
  b := maxlin;
  {repete para todas as linhas}
  REPEAT
    hl := ^ix + 2;   {pula 2 primeiros bytes
(contadores)}
    c := maxcol;
    {repete para todas as colunas}
    REPEAT
      ^hl := " ";
      hl := hl + 1
      c := c - 1;
    UNTIL (c=0);
    ix := ix + 2;
    b := b - 1;
  UNTIL (b=0);

  {inicializa variáveis de estado}
  lincor := 0;
  colcor := 0;
  linmostrar := 0;

  ix := ^pontlin;
  pontins := ^ix + 2;

END {IniTela};

PROCEDURE Interrup;
VAR
  pilha_salva : INTEGER;

BEGIN
  EXCHANGE_REG;
  pilha_salva := sp;
  hl := pontlin[linmostrar*2];
  bc := tela[^hl];
  sp := bc + 8000h + 1;
  c := c'2;
  IF ((c MOD 2) = 1)
    THEN c := c + 1;
```

```
  c := c + 1;
  FOR c := c DOWNTO 0 DO
    POP(bc);
  linmostrar := linmostrar + 1;
  IF (linmostrar = 24)
    THEN linmostrar := 0;
  sp := pilha_salva;
  EXCHANGE_REG
END {Interup};

BEGIN
  IniTela(pontlin,tela,lincor,colcor,
          linmostrar,pontins);
END {TesteTela}.
```

Para simplificar à notação doravante todas as variáveis referentes `a estrutura de tela serão agrupadas no seguinte registro composto:

```
TYPE
  t_estrtela = RECORD OF
                 tela : ARRAY[1..maxlin,1..maxcol] OF
CHAR;
                 pontlin : ARRAY[1..maxlin] OF INTEGER;
                 lincor : INTEGER;
                 colcor : INTEGER;
                 linmostrar : INTEGER;
                 pontins : INTEGER
               END;
```

V.2.5 Conexão Com O Teclado -

Após a estruturação dos dados escrevemos o algoritmo de controle do teclado. Este programa e' relativamente complexo por englobar a varredura, supressão dos ruidos dos contatos e codificação das teclas com consulta a tabelas. Segundo a filosofia de trabalho adotada decompusemos a rotina em algumas tarefas básicas, como discutido abaixo.

V.2.5.1 Rotina De Gerenciamento Do Teclado -

Esta rotina faz todo o gerenciamento do teclado. A cada chamada duas saidas são fornecidas: um indicador avisando se uma tecla foi detectada ou não (reg. A) e o código ASCII do caracter detectado, caso haja um (reg. C). Invocando-se periodicamente esse procedimento o teclado estara' sempre atendido.

Mostramos a seguir o algoritmo da rotina teclado:

```
PROCEDURE Teclado(VAR detectou:BOOLEAN;
                  VAR codigo:REGISTER);
BEGIN
  IF (NOT Esta_Na_Hora)
    THEN Avisa(detectou)
    ELSE
      BEGIN
        Le_Tecl(coordenadas);
        IF (NOT Ha'_Tecla)
          THEN AcertaParametros
          ELSE
            IF (NOT Ja'_Tratou)
             THEN
               IF (NOT Ja'_Havia_Detectado)
                 THEN
                   BEGIN
                     Salva_Coordenadas;
                     Avisa_Que_Ja'_Detectou
                   END
                 ELSE
                   IF (coord_salvas=coordenadas)
                     THEN
                       BEGIN
                         Recebe(coordenadas, codigo));
                         Avisa_Que_Ja'_Tratou
                       END
                     ELSE
                       BEGIN
                         Salva_Coordenadas;
                         Inicializa_Contador_Repetição
                       END
             ELSE
               BEGIN
                 Decrementa_Contador_Repetição;
                 IF (contadorrepetição = 0)
                   THEN
                     BEGIN
```

```
                    Recebe(coordenadas, codigo);
                    Inicializa1_Contador_Repetição
                  END
            END
      END
END {teclado};
```

A filosofia básica do algoritmo acima e' a seguinte: O programa principal chama a rotina teclado a cada ciclo, porém as N primeiras chamadas são ignoradas, respondendo a rotina com um indicador de "não ha' dados". Isto e' feito porque a digitação pelo operador e' bastante lenta, não exigindo atenção constante por parte do sistema, e também como parte do mecanismo de supressão de ruidos dos contatos mecânicos dos interruptores das teclas. Após o número especificado de chamadas a rotina efetivamente inicia seu trabalho e consulta o teclado.

Caso haja uma tecla pressionada, verifica-se então se na leitura anterior ja' havia sido detectada uma tecla pressionada. Em caso afirmativo, as coordenadas das duas teclas são comparadas, e se forem as mesmas, a tecla sera' reconhecida e traduzida para o caracter ASCII equivalente, respondendo a rotina com um sinal de "dados detectados" e marcando esta tecla como ja' tratada. Enquanto o periodo da taxa de repetição não se esgotar a tecla não sera' reconhecida, mesmo estando pressionada, protegendo desta forma o mecanismo de depuração de ruidos. Expirado o prazo determinado pelo contador da taxa de repetição, a marca de tecla tratada e' retirada, podendo o processo ser novamente disparado na próxima

chamada da rotina se a tecla ainda continuar pressionada. Note-se que o processo completo de reconhecimento de uma tecla exige várias chamadas da rotina "teclado".

Na expansão do algoritmo mostrada abaixo as seguintes variáveis e constantes foram definidas:

1. constantes:
   1. delta_tecl - número de vêzes que a rotina deve voltar vazia.
   2. rep1 - tempo a esperar ate' reconhecer a primeira repetïçãõ.
   3. rep2 - tempo a esperar para as repetições subseqõentes.

2. variáveis:
   1. cnt_tecl - contador das vêzes que a rotina foi chamada.
   2. det - marca de tecla detectada.
   3. trat - marca de tecla ja' tratada, força a rotina a ignorá-la.
   4. cont_rep - contador da taxa de repetição de tecla.
   5. a - retorna OFF quando não foram detectados dados no teclado
   6. b, c, d - coordenadas da tecla detectada no teclado

A rotina teclado fica então:

```
PROCEDURE Teclado(VAR a:BOOLEAN; VAR c:REGISTER;
                  VAR cnt_tecl,cnd_rep,det,trat:BYTE);
VAR
  det,trat: BOOLEAN;
  cnt_rep,bsalvo, csalvo, dsalvo : BYTE;
  b,c,d : REGISTER;

BEGIN
  cnt_tecl := cnt_tecl - 1;
  IF (cnt_tecl <> 0)
```

```
THEN
  a := FALSE;
ELSE
  BEGIN
    cnt_tecl := delta_tecl;
    IF (NOT le_tecl(b,c,d))
      THEN
        BEGIN
          trat := FALSE;
          det := FALSE;
          cont_rep := rep1
        END
      ELSE
        IF (NOT trat)
          THEN
            IF (NOT det)
              THEN
                BEGIN
                  salva_bcd(b,c,d,bsalvo,
                                csalvo,dsalvo);
                  det := TRUE
                END
              ELSE
                IF (bcd_iguais(b,c,d,bsalvo,
                                csalvo,dsalvo))
                  THEN
                    BEGIN
                      trat := TRUE;
                      c := recebe(b,c,d,maiusc,
                                  cntrl,local,trava)
                    END
                  ELSE
                    BEGIN

salva_bcd(b,c,d,bsalvoçsalvo,dsalvo);
                      cont_rep := rep1;
                    END
          ELSE
            BEGIN
              cont_rep := cont_rep -1;
              IF (cont_rep <> 0)
                THEN
                  a := FALSE
                ELSE
                  BEGIN
                    c := recebe(b,c,d,maiusc,
                                cntrl,local,trava);
                    cont_rep := rep2
                  END
            END
  END
END {teclado};
```

PROGRAMAÇAO BASICA

As ações cuja inclusão neste algoritmo obscureceriam por demais o código foram transformadas em rotinas, segundo a filosofia do refinamento sucessivo ("top-down"). Três delas são funções, duas do tipo booleano e uma do tipo BYTE. As booleanas são a rotina "bcd_iguais(b,c,d)" que verifica se os registradores b, c e d são iguais aos salvos pela rotina "salva_bcd(b,c,d)" e a rotina "le_tecl(b,c,d)" que retorna verdadeira se houve uma tecla detectada (colocando as coordenadas da tecla nos registradores bc e d) ou falsa se não houve nenhuma tecla pressionada (caso em que os parâmetros b,c e d não tem significado).

A rotina mais complexa desse algoritmo e' a função "recebe(b,c,d)", do tipo BYTE, que a partir das coordenadas de uma tecla lida define um caracter válido através de pesquisa nas tabelas que definem o teclado e o retorna no registrador c. A dificuldade deste procedimento decorre do fato de que nem todas as teclas detectadas geram um código ASCII. Algumas alteram a função de outras teclas (como SHIFT e CTRL) e algumas podem criar uma seqüência de caracteres de tamanho variável (como as teclas de função). Outras ainda mudam o modo de operação do terminal, colocando-o por exemplo no modo LINHA ou no modo LOCAL, o que tem um impacto bastante grande em algumas ações da própria rotina como por exemplo as cadeias geradas pelas teclas de função que deve ser redirecionadas para a tela ou para a conexão

serial de acordo com o modo LINHA ou LOCAL do terminal.

Na nossa linguagem de definição de algoritmos esses procedimentos ficam:

V.2.5.1.1 Procedimento Salva_bcd -

```
PROCEDURE salva_bcd(b,c,d:REGISTER;
                    VAR bsalvo,csalvo,dsalvo:BYTE);
BEGIN
  bsalvo := b;
  csalvo := c;
  dsalvo := d
END {salva_bcd};


FUNCTION bcd_iguais(b, c,d:REGISTER;
                    bsalvo,csalvo,dsalvo:BYTE):BOOLEAN;
BEGIN
  bcdiguais := (b = bsalvo) AND (c = csalvo) AND
               (d=dsalvo)
END {bcd_iguais};


FUNCTION le_teclado(VAR b,c,d:REGISTER):BOOLEAN;
BEGIN
  d := IVar;
  le_teclado := TRUE;
  REPEAT
    OUT(d,pa);
    c := (NOT IN(pc)) AND masc_c;
    b := (NOT IN(pb)) AND masc_b;
    d := RRC(d)
   UNTIL Carry OR (c <> 0) OR (b <> 0);
   IF (NOT Carry)
     THEN
       le_teclado := FALSE
END {le_teclado};


FUNCTION recebe(b,c,d:REGISTER; VAR maiusc,cntrl,
                local,trava:BOOLEAN):BYTE;
VAR
  contador : BYTE;
  achou : BOOLEAN;
  lin_det,col_det : BYTE;

BEGIN
```

```
  { calcula o número do bit em bc que esta' em 0 }
  col_det := 0;
  contador := 8;
  achou := FALSE;
  REPEAT
    col_det := col_det + 1;
    b := RRC(b);
    IF (NOT Carry)
      THEN
        BEGIN
          col_det := col_det - 1;
          achou := TRUE
        END;
    contador := contador - 1
  UNTIL (contador = 0) OR achou
  REPEAT
    col_det := col_det + 1;
    c := RRC(c)
  UNTIL NOT Carry

  { calcula numero do bit em d que esta' em 0}
  lin_det := 0;
  contador := 8;
  REPEAT
    lin_det := lin_det + 1;
    d := RRC(d);
    contador := contador - 1
  UNTIL (NOT Carry) OR (contador = 0)
  lin_det := lin_det - 1;

  maiusc := FALSE;
  cntrl := FALSE;
  IF ((BIT(6,c) = 0)
    THEN
      maiusc := TRUE;
  IF ((BIT(7,c) = 0)
    THEN
      cntrl := TRUE;
  a := traduz(tab_tecl,lin_tecl,lin_detçol_det,maiusc,
       cntrl,trava);
  recebe := tecl_espec(a,tecl_alt,local,trava);
END {recebe};
```

Novamente a mesma metodologia. Deixamos para definição posterior os procedimentos que por sua complexidade confundiriam o fluxo de execução da função "recebe". Nesta codificação definimos as seguintes constantes e variáveis:

1. constantes:
   1. ivar (0111 1111) - máscara de varredura da primeira linha (bit 7).
   2. masc_b, masc_c - máscara para apagar linhas não varridas.
   3. pa, pb, pc - endereço dos portos a, b e c de varredura do teclado.
   4. tab_tecl - endereço inicial das tabelas de tradução de teclas.
   5. lin_tecl - número de linhas da matriz teclado.

2. variáveis:
   1. bsalvo, csalvo, dsalvo - variáveis de armazenamento temporário das coordenadas para comparação com a próxima leitura.
   2. lin_det, col_det - número da linha e da coluna da tecla pressionada.
   3. maiusc, cntrl - indicadores booleanos de detecção dessas teclas.
   4. contador, achou - variáveis de controle do ciclo de definição de lin_det e col_det.

As rotinas que ficaram para definição posterior são as funções "traduz" e "tecl_espec". "Traduz" tem diversos parâmetros, alguns constantes (tab_tecl, lin_tecl) e alguns dependentes da particular tecla pressionada (lin_det, col_det, maiusc, cntrl). Os parâmetros constantes definem qual a tabela de tradução a ser empregada, e os variáveis dizem qual foi a tecla pressionada. Esta função retorna um código para a tecla pressionada.

Se o código retornado indicar uma tecla especial (que gera mais de um caracter) a rotina "tecl_espec" enviara' os caracteres apropriados ao canal correspondente ao modo do terminal. Se indicar apenas um caracter ASCII normal, a função "tecl_espec" não tera' nenhuma ação e o código sera' passado para "recebe", que o enviara' para a rotina "teclado" que a invocou. "Tecl_espec" também verifica se foi detectada a tecla de LOCAL/LINHA ou a tecla de TRAVA_MAIÚSCULAS e atualiza as variáveis correspondentes.

A codificação dos procedimentos "traduz" e "tecl_espec" fica então:

```
FUNCTION traduz(tab_tecl1, lin_tecl1:INTEGER;
                lin_det1,col_det1,maiusc1,cntrl1,
                trava1:BYTE):BYTE;
VAR
  ponteiro : INTEGER;

BEGIN
  WHILE (col_det1 <> 0) DO
  BEGIN
    trab_tecl1 := tab_tecl1 + lin_tecl1;
    col_det1 := col_det1 - 1
  END;
  tab_tecl1 := tab_tecl1 + lin_det1;
  IF (cntrl1 <> 0)
    THEN
      ponteiro := tam_tab*3
    ELSE
      IF (maiusc1 <> 0)
        THEN
          ponteiro := tam_tab*2
        ELSE
          IF (trava1 <> 0)
            THEN
              ponteiro := tam_tab;
  ponteiro := tab_tecl1 + ponteiro;
  traduz := ^ponteiro
END {traduz};

FUNCTION tecl_espec(a:REGISTER; tecl_alt:INTEGER;
                    VAR trava,local:BYTE):BYTE;
```

```
VAR
  ponteiro, cont : INTEGER;

BEGIN
  tecl_espec := a;
  IF (a >= 80h)
    THEN
      IF (a = trv)
        THEN
          BEGIN
            trava := NOT trava;
            tecl_espec := 0
          END
        ELSE
          IF (a = slc)
            THEN
              BEGIN
                local := NOT local;
                tecl_espec := 0
              END
            ELSE
              BEGIN
                ponteiro := ((a - 80h) * 4) + tecl_alt;
                FOR cont := ponteiro TO ponteiro+4 DO
                manda(^cont);
                tecl_espec := 0
              END
END {tecl_espec};
```

V.2.6 Controlador Simples -

O teste de integração das rotinas discutidas anteriormente foi feito no programa controlador simples, que implementa as seguintes funções:

1. Deteção de uma tecla pressionada no teclado e codificação do caracter correspondente.

2. Inserção de um caracter na tela.

3. Rotina atendimento `a interrupção de simulação de acesso direto `a memória.

PROGRAMAÇAO BASICA

Trata-se portanto de um programa que transforma o dispositivo num terminal permanentemente no modo local. Sua implementação foi um passo decisivo na vida do projeto, pois como em todos os desenvolvimentos feitos de forma modular, muitas das caracteristicas funcionais somente podem ser completamente testadas pela integração das diversas sub-partes, com os módulos consumidores processando não dados de entrada simulados, mas sim os verdadeiramente criados pelos procedimentos geradores. A efetiva troca de mensagens e a validação da comunicação inter-módulos somente pode ser assegurada no estágio de integração[17] .

O instante da integração desempenha, desta forma, papel extremamente importante em todos os desenvolvimentos efetuados com essa filosofia, tal como ocorreu com esta dissertação, ocupando um lugar de destaque. O corpo principal do controlador simples e':

```
PROGRAM ContSimp(INPUT,OUTPUT);
CONST
  maxlin = 24;
  maxcol = 96;
TYPE
  t_tela = RECORD OF
              ARRAY[1..maxlin,1..maxcol] OF CHAR;
           END;
VAR
   estrtela : t_tela;
   pontexib : INTEGER;

PROCEDURE TrataInt(pontexib1:INTEGER;estrtela1:t_tela);
BEGIN
  IF (ocorreu_interrupção)
    THEN Interrup(pontexib1,estrtela1)
END {TrataInt};
```

```
PROCEDURE TrataTecl(estrtela2:t_tela);
VAR
  detec : BOOLEAN;     {Indica det. tecla pressionada}
  codtecl : BYTE;      {Código da tecla detectada}
BEGIN
  REPEAT
    Teclado(detec,codtecl);
    IF (detec)
      THEN InsTela(codtecl,estrtela2);
    WAIT(5);
  UNTIL FALSE
END {TrataTecl};


{Programa Principal}
BEGIN
  PrgPerif;
  IniVar(pontexib);
  IniTela(estrtela);
  LigaVideo;
  COBEGIN
    TrataTecl(estrtela);
    TrataInt(estrtela);
  COEND
END.
```

Note-se que não ha' um núcleo de tempo real para o gerenciamento dos processos. Os dois processos concorrentes TrataTecl e TrataInt têm sua execução intercalada no tempo através de uma estrutura de "foreground/background" que se mostra bastante eficaz para este particular caso, visto que o procedimento TrataInt tem como finalidade básica reportar o estado da estrutura de tela periodicamente, transferindo-a para o subsistema de visualização, sendo escalado pela

---

17 - A menos que se use mecanismos formais de prova de correção de programas. Dado o enfoque extremamente prático adotado, o formalismo necessário para a prova certamente afastaria uma parcela considerável da audiência esperada para esse trabalho.

interrupção discutida anteriormente.

O controle do acesso `as regiões criticas pode ser eficientemente implementado através da inibição de interrupções, visto que o processo de fundo ("backgrond") somente e' disparado por uma interrupção e somente devolve o controle ao processo de frente ("foreground") depois de ler os dados compartilhados. Como o processo de frente não tem mecanismos para se interpor `a execução do processo de fundo e esse processo de fundo somente pode ser escalado quando uma interrupção e' aceita, pode-se garantir a efetiva exclusão mútua dos processos em suas regiões criticas apenas pelo mecanismo de inibição de interrupções. Novamente o cuidado tomado no projeto do subsistema controlador de interrupções mostra suas vantagens, pois se fosse mantido o esquema mais simples, com um sinal de interrupção e uma rotina de tratamento para cada ação básica do subsistema de exibição, dificilmente seria possivel um efetivo gerenciamento da exclusão mútua dos diversos processos.

Mesmo tendo a estrutura de programa de frente/fundo mostrado o seu valor, visto que a implementação do programa controlador funciona a contento, um núcleo de tempo real pode vir a ser uma opção extremamente interessante no caso de se reconstruir o terminal ao redor de um microprocessador mais poderoso.

PROGRAMAÇAO BASICA

Como a atual programação praticamente esgota os recursos computacionais do limitado e antigo microprocessador Z80 empregado, o mecanismo de frente/fundo torna-se obrigatório por razões de eficiência visto que quase nenhum esforço computacional ("overhead") e' necessário para a troca de contextos (apenas instruções de salvamento de registros) com a exclusão mútua implementada com uma única instrução de máquina (e das mais rápidas, apenas 7 ciclos) e um método de escalamento de processos extremamente eficiente (o esforço desprendido pela UCP para a escalação e' o mesmo do reconhecimento de uma interrupção). Não e' necessária a manutenção de um relógio de tempo real nem algoritmos complexos com estratégias de alocação que, se implementados, certamente não permitiriam que o processo TrataInt fosse escalonado para execução com a freqüência necessária ao bom funcionamento do subsistema de exibição.

O microprocessador Z80 foi usado dentro da filosofia de se empregar os componentes mais baratos para manter o projeto como um terminal de baixo custo. Entretanto, com a diferença entre essa UCP e os dispositivos de 16 bits mais baratos em torno de 5 dólares no mercado americano (Byte, Vol 11 N.1, Janeiro 86, pp. 456) torna-se difícil justificar a adoção de uma UCP claramente insuficiente para a magnitude das tarefas envolvidas, como e' o caso do Z80. Cremos que com a troca de UCP não precisaremos

nos ater `a estrutura de frente/fundo com suas limitantes caracteristicas. Haveria capacidade para implementar um núcleo de tempo real e separar os diversos componentes da programação em processos isolados, que seriam então escalados de acordo com a ocorrência de determinados eventos. Embora possa haver uma pequena perda de eficiência devida `a grande freqüência com a qual o processo TrataInt e' invocado, esse ataque teria a vantagem de permitir uma efetiva separação de funções em módulos de baixo acoplamento, bem dentro da filosofia de todo o trabalho. Desta forma o reconhecimento das seqüências de controle, o tratamento do teclado, a recepção de dados pela conexão serial e demais tarefas componentes do programa controlador interagiriam através do núcleo de tempo real, permitindo que futuras expansões fossem agregadas como novos processos. Em especial as capacidades anteriormente verificadas como desejáveis num terminal como editor de textos local, pre'-processador de comandos e compilador cruzado seriam facilmente incorporáveis nessa estrutura. Também a carga programas ("down-load") pelo hospedeiro fica bastante simplificada, pois o mesmo pode ser considerado um procedimento a mais executando concorrentemente no terminal. A conveniência da adoção de um sistema operacional em tempo real foi uma das conclusões mais fortes dessa etapa do projeto.

PROGRAMAÇAO BASICA

No apêndice G encontra-se a listagem do principal programa controlador.

BIBLIOTECA DO INSTITUTO DE FISICA E QUIMICA DE SÃO CARLOS
FISICA

CAPITULO VI

CONCLUSOES

VI.0 CONCLUSOES

A proposta inicial desta dissertação de discutir o papel atual dos terminais de vídeo foi bastante modificada no decorrer do trabalho. Pesquisando a bibliografia disponível e estudando as arquiteturas de diversos periféricos percebeu-se que na maioria dos casos esses dispositivos apresentam sérias falhas arquitetônicas, de implementação e de programação.

Verificamos que o advento dos microprocessadores alterou profundamente a forma de construção desses aparelhos, fato nem sempre captado pelos projetistas. De modo geral os terminais apresentam uma arquitetura bastante similar `a de um microcomputador convencional, com toda a lógica de vídeo feita por circuitos discretos.

CONCLUSOES

Trata-se de uma disposição bastante inadequada para essas aplicações, fato facilmente comprovável pela construção de um terminal com as mesmas características dos comercialmente disponíveis com componentes da mesma tecnologia, mas com menos de 70% dos integrados destes. Esse terminal ainda apresenta capacidades inexploradas, como a possibilidade de implementação do sistema operacional local de tempo real, implementação do dígrafo reconhecedor de linguagens de comando modificável e carregável pelo hospedeiro, teclado alterável por programação e capacidade gráfica limitada. Cremos que se abandonarmos as restrições de custo mínimo e uso da mesma tecnologia dos projetos comerciais comparados (guias adotados para enfatizar os ganhos obtidos com uma arquitetura mais apropriada) e empregarmos componentes um pouco mais modernos como os novos controladores gráficos, microprocessadores de 16 bits, memórias mais densas, FPLA's e PAL's, podemos sem grandes dificuldades projetar e construir um periférico mais poderoso que a maioria dos atuais. Esses terminais ligados ao computador central numa estrutura de "front-end/back-end" através de uma rede local tornariam os terminais de vídeo tal como os conhecemos totalmente anacrônicos.

Normalmente o questionamento a essa posição e' feito em bases de custo. Devido a isto projetamos o terminal de ultra baixo custo descrito nos capítulos anteriores. Esse dispositivo encontra-se em pleno uso em vários

grupos e laboratórios do Departamento de Fisica e Ciência dos Materiais desta Universidade, ja' tendo superado a fase de protótipo. Esta' operacional inclusive a máscara de circuito impresso do terminal, possibilitando sua efetiva duplicação pelos diversos grupos interessados. O terminal mais poderoso citado acima, continuação natural deste trabalho (de quem inclusive aproveitaria grande parte dos conceitos e decisões programação e de arquitetura), poderia sem dúvida ser construido a um custo equivalente ao dos periféricos "burros" ou "inteligentes" atualmente adotados nos centros de processamento.

Entretanto, para a efetiva implantação desta nova filosofia algumas considerações sobre redes de processamento precisarão ser feitas. Um novo campo se descortina a partir deste trabalho pois e' impossivel prosseguir nesta direção sem uma sólida investigação na teoria de redes locais de computadores. Além de prover uma base para pesquisa nessa área, talvez o maior mérito desta dissertação tenha sido a capacitação gerada no L.I.E. para prosseguir nesses estudos. O autor sente-se extremamente recompensado pelos frutos obtidos, pela experiência adquirida e pelas discussões com o orientador, professores, técnicos e funcionários envolvidos na construção deste terminal e especialmente pelo fato de estar usando para redigir este texto um dos dispositivos construidos durante o trabalho. Também na

arquitetura de sistemas baseados em microprocessadores foram feitos vários avanços, com várias técnicas originais empregadas no projeto. Na parte de programação este trabalho deu ensejo a uma metodologia de desenvolvimento empregada em vários outros sistemas em desenvolvimento no laboratório. E' também um exemplo de um sistema complexo projetado com uma preocupação constante de documentação, facilidade de manutenção do código de controle e baixo nivel de erros.

E, sobretudo, este projeto possui o crivo da praticidade, estando disponivel e em plena utilização nas mais variadas aplicações, condição essencial sem a qual trabalhos deste tipo ficam fadados ao esquecimento.

APENDICE A

SEQUENCIAS DE CONTROLE DO VT52

1. ESC A (27 65) - Subir o cursor uma linha ou ignorar se no topo da tela.

2. ESC B (27 66) - Descer o cursor uma linha ou ignorar se no topo da tela.

3. ESC C (27 67) - Avançar o cursor uma coluna ou ignorar se no topo da tela.

4. ESC D (27 68) - Retroceder o cursor uma coluna ou ignorar se no topo da tela.

5. ESC H (27 72) - Trazer o cursor para primeira coluna, primeira linha.

6. ESC I (27 73) - Subir o cursor uma linha, descer toda a tela uma linha se no topo da tela.

7. ESC J (27 74) - Apagar da posição do cursor (inclusive) ate' o final da tela.

8. ESC K (27 75) - Apagar da posição do cursor (inclusive) ate' o final da linha.

9. ESC Y (27 89) - Posicionar o diretamente o cursor (o número da linha e da coluna seguem-se ao ESC Y).

10. ESC Z (27 90) - Identificar o terminal (VT52 responde com ESC/Z).

11. ESC < (27 60) - Faz com que as teclas do teclado numérico transmitam números quando pressionadas.

12. ESC = (27 61) - Faz com que as teclas do teclado numérico transmitam sequências de controle quando pressionadas.

13. ESC [(27 91) - Entra no modo tela a tela (a tela so/ e/ deslocada após ser pressionada uma tecla especial)

14. ESC ? (27 92) - Sai do modo tela a tela (a tela e' deslocada sempre que ocorrer um LF e o cursor estiver na última linha).

APENDICE B

COMANDOS DA LINGUAGEM REGIS

A linguagem gráfica ReGis e' um exemplo de linguagem de comando de terminais voltada para execução local, sem o auxilio da UCP central. Trata-se de uma aplicação tipica para a arquitetura de terminal proposta nesta dissertação.

Seus comandos são todos simbolos ASCII que representam primitivas nas quais podem ser decompostas quaisquer figuras. Serão descritos a seguir:

1. P (position)- Posiciona o cursor gráfico em qualquer local da tela sem escrever nenhum dado.

2. V (vector)- Desenha um vetor (linhas retas) entre as posições da tela definidas no comando.

3. C (curve)- Desenha circulos e/ou arcos usando as posicões especificadas no comando.

4. T (text)- Controla a exibição de cadeias gráficas de texto e permite a especificação do texto a exibir.

5. W (write)- Especifica controles de escrita como sombras e padrões regulares.

6. S (screen)- Especifica controles de tela como tamanho, cor, apagamento, etc.

7. F (fill)- Preenche curvas fechadas com padrões pré-estabelecidos.

8. @ (macrograph)- Define uma cadeia de comando como uma macro-operação. Usadas para armazenar e recuperar comandos ReGis que definem figuras complexas e repetidas.

9. L (load)- Controla a definição e o carregamento de caracteres alternativos que podem ser exibidos pelo comando T.

10. R (report)- Envia informações tais como posição do cursor, códigos de erro, tamanho da tela, cor ativa, etc.

11. ; (resync)- Este caracter funciona como re-sincronizador para as cadeias de comando do Regis.

Comandos típicos do Regis são:

1. S(M0(L0)1(L33)2(L67)3(L100) - Define valores para os mapas de saida para um monitor monocromático como preto para o primeiro plano, cinza escuro para o segundo plano, cinza claro para o terceiro plano e branco para o quarto plano.

2. S(A[0,0][799,479]) - Define as coordenadas da tela como 0,0 para o canto superior esquerdo e 799,479 para a extremidade inferior direita.

3. W(P(M2)) - Fator de escala de 2 escolhido.

4. W(I(H180L50S100)) - Seleciona cor de frente da tela como a cor mais próxima ao amarelo.

5. P[-400,50] - Posiciona o cursor nas coordenadas solicitadas.

APENDICE C

COMANDOS DA LINGUAGEM ANSI X.3.64 (VT100)

1. CPR (Cursor Position Report) - ‹ESC›[‹Pn›;‹Pn›R
2. CUB (Cursor Backward) - ‹ESC›[‹Pn›D
3. CUD (Cursor Down) - ‹ESC›[‹Pn›B
4. CUF (Cursor Forward) - ‹ESC›[‹Pn›C
5. CUU (Cursor Up) - ‹ESC›[‹Pn›A
6. CUP (Cursor Position) - ‹ESC›[‹Pn›;‹Pn›H
7. DECALN (Screen Alignment Display) - ‹ESC›#8
8. DECARM (Auto Repeat Mode) - ‹ESC›#
9. DECANM (ANSI/VT52 Mode) - ‹ESC›#
10. DECAWM (Autowrap Mode) - ‹ESC›#
11. DECCKM (Cursor Keys Mode) - ‹ESC›#
12. DECCOLM (Column Mode) - ‹ESC›#
13. DECDHL (Double Height Mode) - ‹ESC›#3, ‹ESC›#4
14. DECDWL (Double Width Mode) - ‹ESC›#6
15. DECID (Identify Terminal) - ‹ESC›Z

16. DECINLM (Interlace Mode) - ‹ESC›

17. DECKPAM (Keypad Alternate Mode) - ‹ESC›=

18. DECKPNM (Keypad Numeric Mode) - ‹ESC››

19. DECLL (Load Leds) - ‹ESC›[‹Ps›q

20. DECCOM (Origin Mode) - ‹ESC›

21. DECRC (Restore Cursor) - ‹ESC›8

22. DECREPTPARM (Report Terminal Parameters) - ‹ESC›[‹Pn›... x

23. DECREQTPARM (Request Terminal Parameters) - ‹ESC›[‹Pn›x

24. DECSC (Save Cursor) - ‹ESC›7

25. DECSCLM (Scrolling Mode) - ‹ESC›

26. DECSCNM (Screen Mode) - ‹ESC›

27. DECSTBM (Set Top and Bottom Margins) - ‹ESC›[‹Pn›;‹Pm›r

28. DECTST (Invoke Confidence Test) - ‹ESC›[2;‹Ps›y

29. DSR (Device Status Report) - ‹ESC›[,Ps›R

30. ED (Erase in Display) - ‹ESC›[‹Ps›J

31. EL (Erase in Line) - ‹ESC›[‹Ps›K

32. HTS (Horizontal Tabulation Set) - ‹ESC›H

33. HVP (Horizontal Vertical Position) - ‹ESC›[‹Pn›;‹Pm›f

34. DA (Device Atributes) - ‹ESC›[‹Pn›c

35. DECSWL (Single With Line) - ‹ESC›#5

36. IND (Index) - ‹ESC›D

37. LNM (Line Feed/New Line Mode)

38. NEL (Next Line) - ‹ESC›E

39. RI (Reverse Index) - ‹ESC›M

40. RIS (Reset to Initial State) - ‹ESC›c

41. RM (Reset Mode) - ‹ESC›[‹Ps›... l

42. SCS (Select Character Set)

43. SGR (Select Graphic Rendition) - ‹ESC›[‹Ps›...
m

44. SM (Set Mode) - ‹ESC›[‹Ps›... h

45. TBC (Tabulation Clear) - ‹ESC›[‹Ps›g

APENDICE D

ROTINA TESTE_U

```
                        ;
                        ;                    TESTE DA 8251 E 8253
                        ;
                        ;      Programa o temporizador (8253) e o porto serial (8251) para a
                        ; transmissao continua de um caracter a 4800 bps.
                        ;
                        ;#######################################################################
                        ;
                        ;
                        ; V1.0  Original                        Alvaro            02-MAI-1983
                        ; V1.1  Execucao sob M80                Alvaro            08-FEV-1984
                        ; V1.2  Execucao na placa UCP64D        Caetano/Alvaro    02-MAR-1984
                        ; V1.3  Execucao na placa TVI100, in-   Alvaro/Gonzalo    15-MAR-1985
                        ;       clui re-inicializacao da 51 e
                        ;       passa para mnemonicos do Z80.
                        ;
                        ;                                                [33,34]TESTEU13.MAC
                        ;#######################################################################
                        ;                    USP-IFQSC-DFCM-LIE, CP 369, 13560-S. Carlos, SP
                        ;


                                ORG 0000H               ; Espaco de EPROM

                        ;
                        ; Endereco dos perifericos
                        ;
0010                    C0_8253         EQU     10H             ; 8253: Contador 0
0011                    C1_8253         EQU     C0_8253+1       ; 8253: Contador 1
0012                    C2_8253         EQU     C0_8253+2       ; 8253: Contador 2
0013                    M_8253          EQU     C0_8253+3       ; 8253: Modo

0020                    D_8251          EQU     20H             ; 8251: Dados
0021                    E_8251          EQU     D_8251+1        ; 8251: Estado

                        ;
                        ; Constantes do programa
                        ;
0055                    CARAC           EQU     55H     ; Caracter a ser enviado pela 8251
0005                    T_BAUD          EQU     0005    ; Valor Inicial do contador para 4800 bp
0005                    R_BAUD          EQU     0005    ; Valor Inicial do contador para 4800 bp
00EF                    MODO_8251       EQU     0EFH    ; Palavra de modo da 8251:
                                                        ;       - Baud Rate div 64
```

```
                                                        ;       - Palavras de 8 bits
                                                        ;       - Sem paridade
                                                        ;       - 2 bits de parada

0037                    CMD_8251        EQU     37H     ; Palavra de comando da 8251
                                                        ;       - Habilita transmissao
                                                        ;       - Habilita recepcao
                                                        ;       - Habilita pinos DTR' e RTS'
                                                        ;       - "Reseta" em erro

0076                    CONT1_53        EQU     76H     ; Palavra de controle do contador 1
                                                        ;       - Seleciona contador 1
                                                        ;       - Carrega B-S seguido por B+S
                                                        ;       - Modo 3 (gerador de onda quadrad
                                                        ;       - Contador em 16 bits

00B6                    CONT2_53        EQU     0B6H    ; Palavra de controle do contador 2
                                                        ;       - Seleciona contador 2
                                                        ;       - Carrega B-S seguido por B+S
                                                        ;       - Modo 3 (gerador de onda quadrad
                                                        ;       - Contador em 16 bits

                        .Z80

                        ;
                        ; Programacao da 8253
                        ;
0000'                   INIC:
0000'   3E 76                   LD      A,CONT1_53      ; Programa contador 1
0002'   D3 13                   OUT     (M_8253),A
0004'   21 0005                 LD      HL,T_BAUD       ; HL <-- valor de contagem para gerar TxD
0007'   7D                      LD      A,L             ; Valor inicial do contador
0008'   D3 11                   OUT     (C1_8253),A
000A'   7C                      LD      A,H
000B'   D3 11                   OUT     (C1_8253),A
000D'   3E B6                   LD      A,CONT2_53      ; Programa o contador 2
000F'   D3 13                   OUT     (M_8253),A
0011'   21 0005                 LD      HL,R_BAUD       ; HL <-- valor de contagem para gerar TxD
0014'   7D                      LD      A,L             ; Valor inicial do contador
0015'   D3 12                   OUT     (C2_8253),A
0017'   7C                      LD      A,H
0018'   D3 12                   OUT     (C2_8253),A
                        ;
                        ; Programacao da 8251
                        ;
001A'   3E 00                   LD      A,00            ; Envia"SYNC's"
```

```
001C'   D3 21                   OUT     (E_8251),A
001E'   D3 21                   OUT     (E_8251),A
0020'   D3 21                   OUT     (E_8251),A
0022'   3E 40                   LD      A,40H           ; Faz "reset" na 51
0024'   D3 21                   OUT     (E_8251),A
0026'   3E EF                   LD      A,MODO_8251     ; Programa modo
0028'   D3 21                   OUT     (E_8251),A
002A'   3E 37                   LD      A,CMD_8251      ; Envia comando
002C'   D3 21                   OUT     (E_8251),A

                        ;
                        ; Loop de envio de um caracter
                        ;
002E'   3E 55           LOOP:   LD      A,CARAC         ; Caracter a ser escrito
0030'   D3 20                   OUT     (D_8251),A      ; Envia caracter

0032'   DB 21           ESP:    IN      A,(E_8251)      ; Captura estado do porto serial
0034'   E6 01                   AND     1H              ; Verifica se TxRDY
0036'   CA 0032'                JP      Z,ESP           ; Espera TxRDY = 1
0039'   C3 002E'                JP      LOOP            ; Envia outra vez o caracter


                                END     INIC
```

Macros:

Symbols:

C0_825	0010	C1_825	0011	C2_825	0012	CARAC	0055
CMD_82	0037	CONT1_	0076	CONT2_	00B6	D_8251	0020
ESP	0032'	E_8251	0021	INIC	0000'	LOOP	002E'
MODO_8	00EF	M_8253	0013	R_BAUD	0005	T_BAUD	0005

No Fatal error(s)

APÊNDICE E

ROTINA TESTE_E

```
                        ;     LIE - Laboratorio de  Instrumentacao Eletronica - USP
                        ;
                        ;                    TESTE DE ESCRITA NA TELA
                        ;
                        ;   Inicializa uma area de RAM com 'E' e escreve na tela com ADM
                        ; simulado por interrupcao armazenada.
                        ;
                        ;################################################################
                        ;
                        ; V1.0  Original                     Gonzalo/Alvaro   27-NOV-1984
                        ; V1.1  Alteracao do endereco de tela Gonzalo/Alvaro   04-FEV-1985
                        ; V1.2  Inclui programacao da 8253    Gonzalo/Alvaro   05-FEV-1985
                        ; V1.3  Inclui programacao dp cursor  Gonzalo/Alvaro   07-FEV-1985
                        ; V1.4  Apenas interrupcao para DRQ   Gonzalo/Alvaro   08-FEV-1985
                        ;
                        ;################################################################
                        ;                  LIE-DFCM-IFQSC-USP,  CP 369, 13560-S Carlos SP


                        ;#################################################################
                        ;#                                                               #
                        ;#                   Constantes do programa                      #
                        ;#                                                               #
                        ;#################################################################

0000                    CMD_PARA        EQU     00      ; Comando para parar a operacao do 8275
00C0                    CMD_NINT        EQU     0C0H    ; Comando para desabilitar interrupcao
00E0                    CMD_ZERA        EQU     0E0H    ; Comando para zerar contadores do 8275
0080                    CMD_CURS        EQU     80H     ; Comando para carregar o cursor
0023                    CMD_LIGA        EQU     23H     ; Comando para comecar a operacao do 8275
                                                        ; 4:2 - 0 ciclos entre pedidos de ADM
                                                        ; 1:0 - 8 ciclos de ADM por pedido

004F                    PAR_1           EQU     4FH     ; Parametro 1 do CMD_PARA
                                                        ; 7 - tela normal
                                                        ; 6:0 - 80 caracteres por linha

0058                    PAR_2           EQU     58H     ; Parametro 2 do CMD_PARA
                                                        ; 7:6 - 2 linhas por retraco vertical
                                                        ; 5:0 - 25 linhas por tela
```

```
0089                    PAR_3       EQU     89H     ; Parametro 3 do CMD_PARA
                                                    ; 7:4 - Subscrito colocado na 9 varredura
                                                    ; 3:0 - 10 varreduras por caracter

00DD                    PAR_4       EQU     0DDH    ; Parametro 4 do CMD_PARA
                                                    ; 7  -  Incrementa de 1 contagem de retra
                                                    ; 6  -  Modo nao transparente
                                                    ; 5:4 - Cursor sublinhado piscante
                                                    ; 3:0 - 32 retracos por HRTC

0032                    CNTR0_53    EQU     32H     ; Palavra de controle da 8253
                                                    ; 7:6 - Contador 0
                                                    ; 5:4 - Le B-S e depois B+S
                                                    ; 3:1 - Modo 1: Monoestavel
                                                    ; 0  -  Conta em binario
00B6                    CNTR2_53    EQU     0B6H    ; Palavra de controle da 8253
                                                    ; 7:6 - Contador 2
                                                    ; 5:4 - Le B-S e depois B+S
                                                    ; 3:1 - Modo 3: Gerador de onda quadrada
                                                    ; 0  -  Conta em binario

                        ;
                        ; Enderecos de perifericos
                        ;
0070                    P_8275  EQU     70H         ; Porto de parametros da 8275
0071                    C_8275  EQU     P_8275+1    ; Porto de comando da 8275

5000                    C0_8253 EQU     5000H       ; 8253: Contador 0
5001                    C1_8253 EQU     C0_8253+1   ; 8253: Contador 1
5002                    C2_8253 EQU     C0_8253+2   ; 8253: Contador 2
5003                    P_8253  EQU     C0_8253+3   ; 8253: Porto de programacao


                        ;#################################################################
                        ;#                                                               #
                        ;#                    Programa Principal                         #
                        ;#                                                               #
                        ;#################################################################

                        .Z80
                                ORG     0
0000'                   INIC:
0000'   F3                      DI
0001'   C3 0070'                JP      PPRINC
```

```
                        ;#################################################################
                        ;#                                                               #
                        ;#          Vetor acessado quando ocorre uma interrupcao         #
                        ;#                                                               #
                        ;#################################################################

                                ORG     38H
0038'   F3                      DI
0039'   D9                      EXX                     ; Salva estado da maquina
003A'   08                      EX      AF,AF'

003B'   21 0000                 LD      HL,0            ; Salva ponteiro de pilha
003E'   39                      ADD     HL,SP

003F'   31 A500                 LD      SP,TELA+8000H   ; Ponteiro de pilha aponta p/ inicio da

                                REPT    40              ; Envia para 8275
                                  POP   BC
                                ENDM
0042'   C1              +         POP   BC
0043'   C1              +         POP   BC
0044'   C1              +         POP   BC
0045'   C1              +         POP   BC
0046'   C1              +         POP   BC
0047'   C1              +         POP   BC
0048'   C1              +         POP   BC
0049'   C1              +         POP   BC
004A'   C1              +         POP   BC
004B'   C1              +         POP   BC
004C'   C1              +         POP   BC
004D'   C1              +         POP   BC
004E'   C1              +         POP   BC
004F'   C1              +         POP   BC
0050'   C1              +         POP   BC
0051'   C1              +         POP   BC
0052'   C1              +         POP   BC
0053'   C1              +         POP   BC
0054'   C1              +         POP   BC
0055'   C1              +         POP   BC
0056'   C1              +         POP   BC
0057'   C1              +         POP   BC
0058'   C1              +         POP   BC
```

```
0059'  C1          +          POP   BC
005A'  C1          +          POP   BC
005B'  C1          +          POP   BC
005C'  C1          +          POP   BC
005D'  C1          +          POP   BC
005E'  C1          +          POP   BC
005F'  C1          +          POP   BC
0060'  C1          +          POP   BC
0061'  C1          +          POP   BC
0062'  C1          +          POP   BC
0063'  C1          +          POP   BC
0064'  C1          +          POP   BC
0065'  C1          +          POP   BC
0066'  C1          +          POP   BC
0067'  C1          +          POP   BC
0068'  C1          +          POP   BC
0069'  C1          +          POP   BC

006A'  F9                     LD    SP,HL          ; Restaura ponteiro de pilha

006B'  D9                     EXX                  ; Restaura estado anterior
006C'  08                     EX    AF,AF'
006D'  FB                     EI
006E'  ED 4D                  RETI


                        ;#################################################################
                        ;#                                                               #
                        ;#                    Programa Principal                         #
                        ;#                                                               #
                        ;#################################################################

0070'                   PPRINC:
0070'  31 212C'               LD    SP,I_PILHA
0073'  ED 56                  IM    1              ; Define modo de interrupcao
                        ;
                        ; Programa o contador (8253)
                        ;
0075'  3E 32                  LD    A,CNTRO_53     ; Palavra de controle
0077'  32 5003                LD    (P_8253),A
007A'  32 5000                LD    (CO_8253),A    ; Inicializa contador 0
007D'  3E 00                  LD    A,00
007F'  32 5000                LD    (CO_8253),A


                        ;
```

```
                        ; Programa o controlador de video (8275)
                        ;
0082'  3E 00                    LD      A,CMD_PARA      ; Carrega reg. com. com ordem de para
0084'  D3 71                    OUT     (C_8275),A
0086'  3E 4F                    LD      A,PAR_1         ; Carrega parametros no 8275
0088'  D3 70                    OUT     (P_8275),A
008A'  3E 58                    LD      A,PAR_2
008C'  D3 70                    OUT     (P_8275),A
008E'  3E 89                    LD      A,PAR_3
0090'  D3 70                    OUT     (P_8275),A
0092'  3E DD                    LD      A,PAR_4
0094'  D3 70                    OUT     (P_8275),A

0096'  3E E0                    LD      A,CMD_ZERA      ; Zera contadores do 8275
0098'  D3 71                    OUT     (C_8275),A

009A'  3E 23                    LD      A,CMD_LIGA      ; Dispara o 8275
009C'  D3 71                    OUT     (C_8275),A

009E'  3E C0                    LD      A,CMD_NINT      ; Nao habilita interrupcoes do 8275
00A0'  D3 71                    OUT     (C_8275),A

00A2'  3E 80                    LD      A,CMD_CURS      ; Define posicao inicial do cursor
00A4'  D3 71                    OUT     (C_8275),A
00A6'  3E 00                    LD      A,0
00A8'  D3 70                    OUT     (P_8275),A
00AA'  D3 70                    OUT     (P_8275),A

                        ;
                        ; Inicializa tela
                        ;
00AC'  21 2500                  LD      HL,TELA
00AF'  3E 45                    LD      A,'E'
00B1'  0E 50                    LD      C,80
00B3'                   INI_TELA:
00B3'  77                       LD      (HL),A
00B4'  23                       INC     HL
00B5'  0D                       DEC     C
00B6'  C2 00B3'                 JP      NZ,INI_TELA

00B9'  FB                       EI

                        ;
                        ; Espera interrupcao
                        ;
00BA'  C3 00BA'         CA:     JP      CA              ; Fica por aqui ate' o final dos tempos
```

```
                        ;#################################################################
                        ;#                                                               #
                        ;#                        Area de dados                          #
                        ;#                                                               #
                        ;#################################################################

                                ORG     2000H           ; Inicio da RAM

2000'                   Espaco_de_pilha:DS      300
212C'                   I_pilha         EQU     $
2500                    TELA            EQU     2500H
                                        .
                                END     INIC
```

Macros:

Symbols:

C0_825	5000	C1_825	5001	C2_825	5002	CA	00BA'
CMD_CU	0080	CMD_LI	0023	CMD_NI	00C0	CMD_PA	0000
CMD_ZE	00E0	CNTR0_	0032	CNTR2_	00B6	C_8275	0071
ESPACO	2000'	INIC	0000'	INI_TE	00B3'	I_PILH	212C'
PAR_1	004F	PAR_2	0058	PAR_3	0089	PAR_4	00DD
PPRINC	0070'	P_8253	5003	P_8275	0070	TELA	2500

No Fatal error(s)

APENDICE F

TESTE DE INTEGRAÇAO TECLADO-VIDEO

```
                    ;
                    ;          Laboratorio de Instrumentacao Eletronica - USP
                    ;
                    ;#######################################################################
                    ;
                    ;                         TESTE DE TECLADO
                    ;
                    ;      Varre o teclado pelo porto paralelo e escreve na tela a letra
                    ; correspondente.
                    ;                                                    [33]TSTECL24.MAC
                    ;#######################################################################
                    ;
                    ;
                    ; V1.0  Escreve no porto serial        Alvaro/Gonzalo     22-MAI-1985
                    ; V2.0  Escreve na tela do terminal    Alvaro/Gonzalo     23-MAI-1985
                    ; V2.1  Matriz generica LINxCOL        Alvaro/Gonzalo     24-MAI-1985
                    ; V2.2  Rotina para gerar Lin_Det      Alvaro/Gonzalo     24-MAI-1985
                    ;       e Col_Det
                    ; V2.3  Tabela de traducao ASCII       Alvaro/Gonzalo     31-MAI-1985
                    ; V2.4  Trata teclas controle          Alvaro/Gonzalo      2-JUN-1985
                    ;
                    ;#######################################################################
                    ;                      USP-IFQSC-DFCM-LIE, CP 369, 13560-S. Carlos, SP
                    ;

                          .Z80
                          ORG     0


                    ;#######################################################################
                    ;#                                                                     #
                    ;#                    Constantes do Programa                           #
                    ;#                                                                     #
                    ;#######################################################################

                    ;
                    ; Bloco de Parametros do Terminal
                    ;
00C8                DELTA_TECL      EQU     200     ; Intervalo entre varreduras
```

```
0005                    Lin_Tecl        EQU     5       ; Numero de linhas da matriz teclado
000A                    Col_Tecl        EQU     10      ; Numero de colunas da matriz teclado
0005                    T_BAUD          EQU     0005    ; Valor Inicial do contador para 4800 bps
0005                    R_BAUD          EQU     0005    ; Valor Inicial do contador para 4800 bps
0019                    NUM_LIN         EQU     25      ; Numero de linhas do terminal
0062                    TAM_LIN         EQU     98      ; Tamanho de cada linha na memoria
0014                    LIMITE          EQU     20      ; Numero de linhas armazenadas em TELA
                                                        ; As restantes sao guardadas em TELA_1
                        ;
                        ; Constantes
                        ;
000A                    LF              EQU     0AH     ; ASCII para avancar linha
000D                    CR              EQU     0DH     ; ASCII para retorno de carro
000C                    FF              EQU     0CH     ; ASCII para Novo Formulario
0008                    BS              EQU     08H     ; ASCII para retornar um caracter
001B                    ESC             EQU     1BH     ; ASCII para Escape
0000                    NULO            EQU     0

00EF                    MODO_8251       EQU     0EFH    ; Palavra de modo da 8251:
                                                        ;       - Baud Rate div 64
                                                        ;       - Palavras de 8 bits
                                                        ;       - Sem paridade
                                                        ;       - 2 bits de parada

0037                    CMD_8251        EQU     37H     ; Palavra de comando da 8251
                                                        ;       - Habilita transmissao
                                                        ;       - Habilita recepcao
                                                        ;       - Habilita pinos DTR' e RTS'
                                                        ;       - "Reseta" em erro

0032                    CONT0_53        EQU     32H     ; Palavra de controle da 8253
                                                        ; 7:6 - Seleciona o contador 0
                                                        ; 5:4 - Carrega B-S seguido por B+S
                                                        ; 3:1 - Modo 1: Monoestavel
                                                        ; 0   - Conta em 16 bits


0076                    CONT1_53        EQU     76H     ; Palavra de controle do contador 1
                                                        ;       - Seleciona contador 1
                                                        ;       - Carrega B-S seguido por B+S
                                                        ;       - Modo 3 (gerador de onda quadrad
                                                        ;       - Contador em 16 bits

00B6                    CONT2_53        EQU     0B6H    ; Palavra de controle do contador 2
                                                        ;       - Seleciona contador 2
                                                        ;       - Carrega B-S seguido por B+S
```

```
                                                        ;       - Modo 3 (gerador de onda qua
                                                        ;       - Contador em 16 bits

0090                    PRG_55          EQU     90H     ; Modo de operacao da 8255:
                                                        ;       - Modo 0 (E/S generica)
                                                        ;       - Porta A: entrada
                                                        ;       - Portas B e C: saida

0000                    CMD_PARA        EQU     00      ; Comando para parar a operacao do 82
00C0                    CMD_NINT        EQU     0C0H    ; Comando para desabilitar interrupca
00E0                    CMD_ZERA        EQU     0E0H    ; Comando para zerar contadores do 82
0023                    CMD_LIGA        EQU     23H     ; Comando para comecar a operacao do
                                                        ; 4:2 - 0 ciclos entre pedidos de ADM
                                                        ; 1:0 - 8 ciclos de ADM por pedido

004F                    PAR_1           EQU     4FH     ; Parametro 1 do CMD_PARA
                                                        ; 7 - tela normal
                                                        ; 6:0 - 80 caracteres por linha

0058                    PAR_2           EQU     58H     ; Parametro 2 do CMD_PARA
                                                        ; 7:6 - 2 linhas por retraco vertical
                                                        ; 5:0 - 25 linhas por tela

0089                    PAR_3           EQU     89H     ; Parametro 3 do CMD_PARA
                                                        ; 7:4 - Subscrito colocado na 9 varre
                                                        ; 3:0 - 10 varreduras por caracter

00DD                    PAR_4           EQU     0DDH    ; Parametro 4 do CMD_PARA
cos                                                     ; 7  -  Incrementa de 1 contagem de re
                                                        ; 6  -  Modo nao transparente
                                                        ; 5:4 - Cursor sublinhado piscante
                                                        ; 3:0 - 32 retracos por HRTC


                        ;#################################################################
                        ;#                                                               #
                        ;#                    Endereco dos perifericos                   #
                        ;#                                                               #
                        ;#################################################################

0000                    P_8275          EQU     00H             ; Porto de parametros da 8275
0001                    C_8275          EQU     P_8275+1        ; Porto de comando da 8275

0010                    C0_8253         EQU     10H             ; 8253: Contador 0
```

```
0011                    C1_8253     EQU     C0_8253+1       ; 8253: Contador 1
0012                    C2_8253     EQU     C0_8253+2       ; 8253: Contador 2
0013                    M_8253      EQU     C0_8253+3       ; 8253: Modo

0020                    D_8251      EQU     20H             ; 8251: Dados
0021                    E_8251      EQU     D_8251+1        ; 8251: Estado


0030                    P_Leit      EQU     30H
0031                    P1_Varr     EQU     P_Leit+1
0032                    P2_Varr     EQU     P_Leit+2
0033                    M_8255      EQU     P_Leit+3
                                    .
                                    PAGE
```

```
                        ;#################################################################
                        ;#                                                               #
                        ;#                    Programa Principal                         #
                        ;#                                                               #
                        ;#################################################################

0000'                   PPRINC:
0000'   F3                      DI
0001'   31 312C'                LD      SP,INI_PILHA
0004'   ED 56                   IM      1

0006'   CD 00A9'                CALL    PRG_PERIF               ; Incializa perifericos
0009'   CD 00F2'                CALL    INI_VAR                 ; Inicializa variaveis
000C'   CD 0148'                CALL    INI_TELA                ; Inicializa a tela
000F'   CD 0197'                CALL    LIGA_VIDEO              ; Comeca a mostrar a tela

0012'                   PPR_1:
0012'   CD 01A1'                CALL    TECLADO                 ; Trata teclado
0015'   3E 05                   LD      A,5
0017'                   PPR_2:                                  ; Limpa ruido das teclas
0017'   3D                      DEC     A
0018'   C2 0017'                JP      NZ,PPR_2
001B'   C3 0012'                JP      PPR_1

                                PAGE
```

```
                        ;#################################################################
                        ;#                                                               #
                        ;#          Vetor acessado quando ocorre uma interrupcao         #
                        ;#                                                               #
                        ;#################################################################

                                ORG     38H

0038'   F3                      DI
0039'   D9                      EXX                     ; Salva estado da maquina
003A'   08                      EX      AF,AF'
003B'   21 0000                 LD      HL,0            ; Salva ponteiro de pilha
003E'   39                      ADD     HL,SP
003F'   22 3137'                LD      (PILHA_SALVA),HL
                        ;
                        ; Pega contador de bytes/linha, aponta SP para primeiro byte da linha
                        ;
0042'   21 3139'                LD      HL,PONT_LIN
0045'   3A 3136'                LD      A,(LIN_MOSTRAR) ; HL <-- PONT_LIN[LIN_MOSTRAR*2]
0048'   87                      ADD     A,A
0049'   5F                      LD      E,A
004A'   16 00                   LD      D,0
004C'   19                      ADD     HL,DE

004D'   5E                      LD      E,(HL)          ; DE <-- endereco do contador de bytes/li
004E'   23                      INC     HL              ; (tambem endereca inicio dessa linha)
004F'   56                      LD      D,(HL)

0050'   EB                      EX      DE,HL           ; BC <-- TELA[ DE ] (contador bytes/linha
0051'   4E                      LD      C,(HL)
0052'   23                      INC     HL
0053'   46                      LD      B,(HL)
```

```
0054'  11 8001                    LD      DE,8000H+1      ; SP := primeiro byte da linha + 8000H
0057'  19                         ADD     HL,DE
0058'  F9                         LD      SP,HL
                            ;
                            ; Transferencia dos bytes para o controlador de video
                            ;
0059'  21 006D'                   LD      HL,POPS         ; Calcula nro bytes a mais que 80 nessa
005C'  CB 29                      SRA     C               ; C := C/2
005E'  30 01                      JR      NC,TR_1         ; Se (CY = 1) entao E := E+1
0060'  0C                         INC     C
0061'                       TR_1:
0061'  B7                         OR      A               ; Limpa indicador CY
0062'  ED 42                      SBC     HL,BC
0064'  E9                         JP      (HL)

                                  REPT    8               ; Transferencia dos bytes a mais nessa
                                   POP    BC
                                  ENDM
0065'  C1              +           POP    BC
0066'  C1              +           POP    BC
0067'  C1              +           POP    BC
0068'  C1              +           POP    BC
0069'  C1              +           POP    BC
006A'  C1              +           POP    BC
006B'  C1              +           POP    BC
006C'  C1              +           POP    BC
006D'                       POPS:
                                  REPT    40              ; Transferencia de uma linha normal
                                   POP    BC
                                  ENDM
006D'  C1              +           POP    BC
006E'  C1              +           POP    BC
006F'  C1              +           POP    BC
0070'  C1              +           POP    BC
0071'  C1              +           POP    BC
0072'  C1              +           POP    BC
0073'  C1              +           POP    BC
0074'  C1              +           POP    BC
0075'  C1              +           POP    BC
0076'  C1              +           POP    BC
0077'  C1              +           POP    BC
0078'  C1              +           POP    BC
0079'  C1              +           POP    BC
007A'  C1              +           POP    BC
007B'  C1              +           POP    BC
007C'  C1              +           POP    BC
```

```
007D'   C1          +          POP     BC
007E'   C1          +          POP     BC
007F'   C1          +          POP     BC
0080'   C1          +          POP     BC
0081'   C1          +          POP     BC
0082'   C1          +          POP     BC
0083'   C1          +          POP     BC
0084'   C1          +          POP     BC
0085'   C1          +          POP     BC
0086'   C1          +          POP     BC
0087'   C1          +          POP     BC
0088'   C1          +          POP     BC
0089'   C1          +          POP     BC
008A'   C1          +          POP     BC
008B'   C1          +          POP     BC
008C'   C1          +          POP     BC
008D'   C1          +          POP     BC
008E'   C1          +          POP     BC
008F'   C1          +          POP     BC
0090'   C1          +          POP     BC
0091'   C1          +          POP     BC
0092'   C1          +          POP     BC
0093'   C1          +          POP     BC
0094'   C1          +          POP     BC
                        ;
                        ; Atualiza ponteiro de linha a mostrar
                        ;
0095'   21 3136'                LD      HL,LIN_MOSTRAR  ; LIN_MOSTRAR := LIN_MOSTRAR + 1
0098'   34                      INC     (HL)

0099'   7E                      LD      A,(HL)          ; Se (LIN_MOSTRAR = 25) entao LIN_MOSTRAR
009A'   D6 19                   SUB     25
009C'   20 02                   JR      NZ,TR_2
009E'   36 00                   LD      (HL),0
                        ;
                        ; Restaura estado da maquina e retorna
                        ;
00A0'                   TR_2:
00A0'   2A 3137'                LD      HL,(PILHA_SALVA)
00A3'   F9                      LD      SP,HL
00A4'   D9                      EXX
00A5'   08                      EX      AF,AF'
00A6'   FB                      EI
00A7'   ED 4D                   RETI

                                PAGE
```

```
                        ;#######################################################################
                        ;#
                        ;#                        Rotinas Nivel I
                        ;#
                        ;#######################################################################


                        ; Nome:         PRG_PERIF
                        ;
                        ; Funcao:       Inicializa e programa os perif'ericos (51, 53 e 75)
                        ; Entrada:      Nenhuma
                        ; Saida:        Nenhuma
                        ; Emprega:      Na saida, perif'ericos programados
                        ; Registradores destruidos:     A,H,L
                        ; Referencias externas:         Nenhuma
                        ;
00A9'                   PRG_PERIF:
                        ;
                        ; Programa o controlador de video (8275)
                        ;
00A9'   3E 00                   LD      A,CMD_PARA      ; Carrega reg. com. com ordem de parada
00AB'   D3 01                   OUT     (C_8275),A
00AD'   3E 4F                   LD      A,PAR_1         ; Carrega parametros no 8275
00AF'   D3 00                   OUT     (P_8275),A
00B1'   3E 58                   LD      A,PAR_2
00B3'   D3 00                   OUT     (P_8275),A
00B5'   3E 89                   LD      A,PAR_3
00B7'   D3 00                   OUT     (P_8275),A
00B9'   3E DD                   LD      A,PAR_4
00BB'   D3 00                   OUT     (P_8275),A

00BD'   3E E0                   LD      A,CMD_ZERA      ; Zera contadores do 8275
```


BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

```
00BF'   D3 01                   OUT     (C_8275),A
                        ;
                        ; Programacao da 8253
                        ;
00C1'   3E 32                   LD      A,CONT0_53      ; Programa contador 0
00C3'   D3 13                   OUT     (M_8253),A
00C5'   D3 10                   OUT     (C0_8253),A
00C7'   3E 00                   LD      A,00
00C9'   D3 10                   OUT     (C0_8253),A

00CB'   3E 76                   LD      A,CONT1_53      ; Programa contador 1
00CD'   D3 13                   OUT     (M_8253),A
00CF'   21 0005                 LD      HL,T_BAUD       ; HL <-- valor de contagem para gerar TxD
00D2'   7D                      LD      A,L             ; Valor inicial do contador
00D3'   D3 11                   OUT     (C1_8253),A
00D5'   7C                      LD      A,H
00D6'   D3 11                   OUT     (C1_8253),A

00D8'   3E B6                   LD      A,CONT2_53      ; Programa o contador 2
00DA'   D3 13                   OUT     (M_8253),A
00DC'   21 0005                 LD      HL,R_BAUD       ; HL <-- valor de contagem para gerar TxD
00DF'   7D                      LD      A,L             ; Valor inicial do contador
00E0'   D3 12                   OUT     (C2_8253),A
00E2'   7C                      LD      A,H
00E3'   D3 12                   OUT     (C2_8253),A
                        ;
                        ; Programacao da 8251
                        ;
00E5'   3E EF                   LD      A,MODO_8251     ; Programa modo
00E7'   D3 21                   OUT     (E_8251),A
00E9'   3E 37                   LD      A,CMD_8251      ; Envia comando
00EB'   D3 21                   OUT     (E_8251),A
                        ;
                        ; Programacao da 8255
                        ;
00ED'   3E 90                   LD      A,PRG_55
00EF'   D3 33                   OUT     (M_8255),A

00F1'   C9                      RET


                        ; Nome:     INIcializa_VARiaveis.
```

```
                        ;
                        ; Funcao:   Inicializa as variaveis utilizadas pela rotina TECLADO.
                        ; Entradas: nenhuma
                        ; Saida:    Variaveis Maiusc, Cntrl, Trava, Trat, Det e Cnt_Tecl inicia
                        ; Destroi:    A
                        ; Referencia: ninguem.
                        ;
00F2'                   INI_VAR:
00F2'   3E 00                   LD      A,0
00F4'   32 312D'                LD      (Trat),A
00F7'   32 312E'                LD      (Det),A
00FA'   3E C8                   LD      A,Delta_Tecl
00FC'   32 312C'                LD      (Cnt_Tecl),A

00FF'   3E 01                   LD      A,1
0101'   32 335C'                LD      (Maiusc),A
0104'   32 335B'                LD      (Trava),A
0107'   32 335D'                LD      (Cntrl),A

010A'   3E 05                   LD      A,Lin_Tecl
010C'   32 3357'                LD      (Lin),A
010F'   3E 0A                   LD      A,Col_Tecl
0111'   32 3358'                LD      (Col),A

0114'   3A 3358'                LD      A,(Col)
0117'   FE 09                   CP      9
0119'   47                      LD      B,A
011A'   3E FF                   LD      A,0FFh
011C'   D2 0131'                JP      NC,IN_2

011F'   3F                      CCF                     ; CY <-- 0
0120'                   IN_1:
0120'   CB 17                   RL      A               ; FOR B := Col DOWNTO 1 DO
0122'   05                      DEC     B               ;   CY <--- A7 <-..- A0 <--- CY;
0123'   C2 0120'                JP      NZ,IN_1

0126'   32 3355'                LD      (In1_Varr),A    ; Resposta: In1_Varr calculado
0129'   3E FF                   LD      A,0FFh          ; In2_Varr e' inativo (menos que 8 colun
012B'   32 3356'                LD      (In2_Varr),A
012E'   C3 0147'                JP      IN_4


0131'                   IN_2:
0131'   78                      LD      A,B             ; B := Col-8
0132'   D6 08                   SUB     8
0134'   47                      LD      B,A
```

```
0135'  37                          SCF
0136'  3F                          CCF
0137'  3E FF                       LD      A,0FFH          ; Prepara acumulador para o calculo
0139'                      IN_3:
0139'  CB 17                       RL      A               ; FOR B := Col-8 DOWNTO 1 DO
013B'  05                          DEC     B               ;   CY <--- A7 <-..- A0 <--- CY;
013C'  C2 0139'                    JP      NZ,IN_3

013F'  32 3356'                    LD      (In2_Varr),A    ; Resposta: In2_Varr calculado
0142'  3E 7F                       LD      A,NOT 80h       ; In1_Varr e' colocado no limite
0144'  32 3355'                    LD      (In1_Varr),A
0147'                      IN_4:

0147'  C9                          RET
```

```
                           ;
                           ; Nome:       INIcializa_TELA
                           ;
                           ; Funcao:     Inicializa a estrutura de dados da tela
                           ; Entradas:   Nenhuma
                           ; Saida:      Inicializados:  TELA com brancos, PONT_LIN com ponteiro
                           ;             para as linhas, primeiro byte de cada linha com 0 apontan
                           ;             do para o primeiro byte livre da primeira linha.
                           ; Destroi:    A,B,C,D,E,H,L,IX
                           ; Referencia: Ninguem
                           ;
0148'                      INI_TELA:
0148'  DD 21 3139'                 LD      IX,PONT_LIN
014C'  01 0062                     LD      BC,TAM_LIN

014F'  3E 14                       LD      A,LIMITE                ; Inicializa primeiras linhas
0151'  21 4000'                    LD      HL,TELA
0154'  CD 02BE'                    CALL    LIGA_PONT

0157'  3E 05                       LD      A,NUM_LIN-LIMITE        ; Inicializa linhas finais
```

```
0159'  21 316B'                LD      HL,TELA_1
015C'  CD 02BE'                CALL    LIGA_PONT
                        ;
                        ; Inicializa linhas com brancos
                        ;
015F'  DD 21 3139'             LD      IX,PONT_LIN
0163'  06 19                   LD      B,NUM_LIN
0165'                   INI_T2:
0165'  DD 6E 00                LD      L,(IX)          ; Pega ponteiro do inicio da lin
0168'  DD 66 01                LD      H,(IX+1)
016B'  23                      INC     HL              ; Pula 2 primeiros bytes (contad
016C'  23                      INC     HL
016D'  0E 60                   LD      C,96            ; Contador de bytes/linha
016F'                   INI_T3:
016F'  36 2B                   LD      (HL),'+'        ; Inicializa posicao de tela
0171'  23                      INC     HL
0172'  0D                      DEC     C
linha  0173'  20 FA            JR      NZ,INI_T3       ; Repete para todas as colunas d

0175'  DD 23                   INC     IX              ; IX <-- pont. da proxima linha
0177'  DD 23                   INC     IX
ela    0179'  05               DEC     B               ; Repete para todas as linhas da
       017A'  20 E9            JR      NZ,INI_T2

rente  017C'  3E 00            LD      A,0             ; Inicializa ponteiro de linha c
       017E'  32 3134'         LD      (LIN_COR),A
       0181'  32 3135'         LD      (COL_COR),A
pcao   0184'  32 3136'         LD      (LIN_MOSTRAR),A ; Inicializa variavel para inter

sicao  0187'  DD 21 3139'      LD      IX,PONT_LIN     ; Inicializa P_INS com primeira
ra     018B'  DD 6E 00         LD      L,(IX)          ; vaga na primeira coluna da pri
       018E'  DD 66 01         LD      H,(IX+1)        ; linha.
       0191'  23               INC     HL
       0192'  23               INC     HL
       0193'  22 3132'         LD      (P_INS),HL

       0196'  C9               RET


                        ; Nome:         LIGA_VIDEO
                        ;
                        ; Funcao:       Permitir que o controlador de v'ideo comece a interrom-
```

```
                        ;               per o programa para receber dados.
                        ; Entrada:     Nenhuma
                        ; Saida:       Nenhuma
                        ; Estraga:     A
                        ; Referencia:  Ninguem
                        ;
0197'                   LIGA_VIDEO:
0197'   3E 23                   LD      A,CMD_LIGA      ; Dispara o 8275
0199'   D3 01                   OUT     (C_8275),A

019B'   3E C0                   LD      A,CMD_NINT      ; Nao habilita interrupcoes do 8275
019D'   D3 01                   OUT     (C_8275),A
019F'   FB                      EI                      ; Habilita interrupcoes na UCP
01A0'   C9                      RET


                        ; Nome:     TECLADO
                        ;
                        ; Funcao:   Realiza o processamento de varredura do teclado.
                        ; Entradas: Portos A, B e C. Variaveis globais Trat e Det
                        ; Saida:    nenhuma no momento.
                        ; Destroi:    A, B, C, D, H, L.
                        ; Referencia: LE_TECL, IG_REG_SALVO e RECEBE.
                        ;

01A1'                   TECLADO:
01A1'   21 312C'                LD      HL,Cnt_Tecl             ; cnt_tecl := cnt_tecl - 1
01A4'   35                      DEC     (HL)                    ; IF (cnt_tecl = 0)
01A5'   7E                      LD      A,(HL)                  ;  THEN BEGIN
01A6'   B7                      OR      A
01A7'   C0                      RET     NZ

01A8'   CD 01FE'                CALL    LE_TECL                 ;     IF (nao_ha_tecla)
01AB'   B7                      OR      A                       ;      THEN BEGIN
01AC'   C2 01B9'                JP      NZ,TECL_1

01AF'   AF                      XOR     A                       ;             Trat := 0
01B0'   32 312D'                LD      (Trat),A                ;             Det := 0
01B3'   32 312E'                LD      (Det),A                 ;           END
01B6'   C3 01F8'                JP      TECL_2
```

```
                01B9'                   TECL_1:                         ;       ELSE BEGIN
                01B9'   3A 312D'                LD      A,(Trat)        ;               IF (Trat = 0)
                01BC'   B7                      OR      A               ;               THEN BEGIN
                01BD'   C2 01F8'                JP      NZ,TECL_2

                01C0'   3A 312E'                LD      A,(Det)         ;                       IF (Det = (
                01C3'   B7                      OR      A               ;                       THEN BEGIN
                01C4'   C2 01D8'                JP      NZ,TECL_4

                01C7'   78                      LD      A,B             ;                               sal
= 1             01C8'   32 312F'                LD      (BSalvo),A      ;                               Det
                01CB'   79                      LD      A,C             ;                               END
                01CC'   32 3130'                LD      (CSalvo),A
                01CF'   7A                      LD      A,D
                01D0'   32 3131'                LD      (DSalvo),A
                01D3'   3E 01                   LD      A,1
                01D5'   32 312E'                LD      (Det),A
                01D8'   C3 01F8'                JP      TECL_2

                01DB'                   TECL_4:
                01DB'   CD 023B'                CALL    IG_REG_SALVO    ;                       ELSE BEGIN
guais)          01DE'   C2 01EC'                JP      NZ,TECL_5       ;                               IF

EGIN            01E1'   CD 024A'                CALL    RECEBE          ;                               TH
  recebe        01E4'   3E 01                   LD      A,1             ;
  TRAT := 1     01E6'   32 312D'                LD      (Trat),A        ;
END             01E9'   C3 01F8'                JP      TECL_2          ;

EGIN            01EC'                   TECL_5:                         ;                               EL
  salva(A,B,C)01EC'     78                      LD      A,B             ;
END             01ED'   32 312F'                LD      (BSalvo),A      ;
                01F0'   79                      LD      A,C
                01F1'   32 3130'                LD      (CSalvo),A
                01F4'   7A                      LD      A,D
                01F5'   32 3131'                LD      (DSalvo),A

                01F8'                   TECL_2:                         ;       nct_tecl := delta_tecl
                01F8'   3E C8                   LD      A,DELTA_TECL    ;   END;
                01FA'   32 312C'                LD      (Cnt_Tecl),A
                01FD'   C9                      RET

                                                PAGE
```

```
                        ;#######################################################################
                        ;#
                        ;#                          Rotinas Nivel II
                        ;#
                        ;#######################################################################

                        ; Nome:  LE_TECLado
                        ;
                        ; Funcao:     Varre a matriz de teclado e detecta uma tecla pressionada.
                        ; Entradas:   Nenhuma
                        ; Saida:      A: 0 se nao detectou tecla, <> 0 se detectou
                        ;             B, C: Colunas da tecla detectada
                        ;             D: Linha da tecla  detectada
                        ; Emprega:    Portos B e C para varredura, porto A para recepcao
                        ; Destroi:    A, B, C, D
                        ; Referencia: Ninguem
                        ;
01FE'                   LE_TECL:
01FE'   3A 3355'                LD      A,(IN1_Varr)            ; Endereco inicial para varredura
0201'   47                      LD      B,A

0202'   3E FF                   LD      A,0FFh                  ; Coloca porta 2 inativa (tudo 1)
0204'   4F                      LD      C,A
0205'   D3 32                   OUT     (P2_Varr),A
0207'                   LE_T1:
0207'   78                      LD      A,B                     ; envia valor de varredura
0208'   D3 31                   OUT     (P1_Varr),A

020A'   DB 30                   IN      A,(P_Leit)              ;   le respota do teclado
020C'   57                      LD      D,A
020D'   2F                      CPL
020E'   E6 1F                   AND     1Fh                     ; mascara bits MAIUSC, CNTRL, TRA
0210'   C2 0239'                JP      NZ,LE_ACH               ; se encontrou resposta retorna


0213'   CB 08                   RRC     B                       ; atualiza valores de varredura
0215'   DA 0207'                JP      C,LE_T1                 ; envia novos valores
```

```
0218'  3E FF                LD      A,0FFH          ; Se ja varreu todas as linhas da
021A'  47                   LD      B,A
021B'  D3 31                OUT     (P1_VARR),A     ; destiva as colunas da porta 1

021D'  3A 3356'             LD      A,(IN2_Varr)
0220'  4F                   LD      C,A
0221'  FE FF                CP      0FFh            ; Se (In2_Varr <> 0FFh) entao
0223'  CA 0237'             JP      Z,LE_T3

0226'                 LE_T2:                        ; Comeca a ler colunas do porto 2
0226'  79                   LD      A,C
0227'  D3 32                OUT     (P2_Varr),A     ;

0229'  DB 30                IN      A,(P_Leit)      ; le resposta
022B'  57                   LD      D,A
022C'  2F                   CPL
022D'  E6 1F                AND     1Fh
022F'  C2 0239'             JP      NZ,LE_ACH       ; se achou retorna com o valor

0232'  CB 09                RRC     C
0234'  DA 0226'             JP      C,LE_T2         ; verifca a proxima linha

0237'                 LE_T3:
0237'  16 00                LD      D,0             ; se nao achou retorna com indica
                                                    ; de erro
0239'                 LE_ACH:
0239'  7A                   LD      A,D
023A'  C9                   RET


                      ;
                      ; Nome:  IGuais_REGistradores_SALVO
                      ;
                      ; Funcao:     Verifica se os registradoes B, C e D sao iguais as variavei
                      ;             BSalvo, CSalvo e DSalvo.
                      ; Entradas:   B, C e D com os valores a ser comparados
                      ; Saida:      flag ZERO setado caso sejam iguais
                      ; Emprega:    Nada
                      ; Destroi:    A
                      ; Referencia: ninguem.
                      ;
023B'                 IG_REG_SALVO:
023B'  3A 312F'             LD      A,(BSalvo)
```

```
023E'   B8                      CP      B
023F'   C0                      RET     NZ

0240'   3A 3130'                LD      A,(CSalvo)
0243'   B9                      CP      C
0244'   C0                      RET     NZ

0245'   3A 3131'                LD      A,(DSalvo)
0248'   BA                      CP      D
0249'   C9                      RET
```

```
                        ; Nome:     RECEBE
                        ;
                        ; Funcao:   Faz o processamento necessario para enviar uma letra `a tela
                        ;
                        ; Entradas: B e C: coluna da matriz teclado
                        ;           D: linha da matriz teclado
                        ; Saida:    Nenhuma
                        ; Emprega:  Tela para a saida dos caracteres detectados
                        ; Destroi:  A, B, C, D, E
                        ; Referencia: TRADUZ,INS_TELA
                        ;
024A'                   RECEBE:
024A'   1E 08                   LD      E,8             ; Numero maximo de rotacoes
024C'   3E 00                   LD      A,0             ; Valor inicial do contador
024E'                   REC_0:
024E'   3C                      INC     A               ; Incrementa contador
024F'   CB 08                   RRC     B               ; Roda o reg. B, bit B0 --> CY
0251'   D2 025E'                JP      NC,REC_1        ; Se ja' detectou 0 vai imprimir contador
0254'   1D                      DEC     E
0255'   C2 024E'                JP      NZ,REC_0        ; Repete 8 vezes no maximo
                        ;
                        ; Nao detectou zero no reg. B. Tenta no reg. C
```

```
                        ;
0258'                   REC_2:
0258'   3C                      INC     A               ; Incrementa contador
0259'   CB 09                   RRC     C               ; Roda o reg. C, bit C0 --> CY
025B'   DA 0258'                JP      C,REC_2

025E'                   REC_1:
025E'   3D                      DEC     A               ; Ajusta contador (estava adiantado de 1)
025F'   32 335A'                LD      (Col_Det),A     ; Salva coordenada da tecla detectada
                        ;
                        ; Encontra posicao do bit em 0 no registrador D
                        ;
0262'   1E 08                   LD      E,8             ; Numero maximo de repeticoes
0264'   3E 00                   LD      A,0             ; Valor inicial do contador
0266'   6A                      LD      L,D             ; Salva em L bit em 0 nas linhas
0267'                   REC_4:
0267'   3C                      INC     A               ; Incrementa contador
0268'   CB 0A                   RRC     D               ; Roda o reg. D, bit D0 --> CY
026A'   D2 0281'                JP      NC,REC_5        ; Se ja' detectou 0 vai imprimir contador
026D'   1D                      DEC     E
026E'   C2 0267'                JP      NZ,REC_4        ; Repete 8 vezes no maximo
                        ;
                        ; Nao achou nenhum bit em 0 no registrador D
                        ;
0271'   3E 3F                   LD      A,'?'           ; Envia mensagem de erro e para
0273'   CD 0306'                CALL    INS_TELA
0276'   3E 31                   LD      A,'1'
0278'   CD 0306'                CALL    INS_TELA
027B'   3E 3F                   LD      A,'?'
027D'   CD 0306'                CALL    INS_TELA
0280'   76                      HALT
                        ;
                        ; Define variavel do bit em 0 no registrador D
                        ;
0281'                   REC_5:
0281'   3D                      DEC     A               ; Ajusta contador (estava adiantado de 1)
0282'   32 3359'                LD      (Lin_Det),A     ; Salva coordenada linha da tecla detecta

                        ;
                        ; Verifica se estava apertado teclas de MAIUSC, TRAVA ou CNTRL
                        ;

0285'   3E 01                   LD      A,1
0287'   32 335C'                LD      (Maiusc),A
028A'   32 335B'                LD      (Trava),A
028D'   32 335D'                LD      (Cntrl),A
```

```
0290'   3E 00                       LD      A,0

0292'   CB 6D                       BIT     5,L             ; Se (Bit_maiusc = 0)
0294'   C2 029A'                    JP      NZ,REC_6
0297'   32 335C'                    LD      (Maiusc),A

029A'                       REC_6:
029A'   CB 75                       BIT     6,L             ; Se (Bit_cntrl = 0)
029C'   C2 02A2'                    JP      NZ,REC_7
029F'   32 335D'                    LD      (Cntrl),A

02A2'                       REC_7:
02A2'   CB 7D                       BIT     7,L             ; Se (Bit_trava = 0)
02A4'   C2 02AA'                    JP      NZ,REC_8
02A7'   32 335B'                    LD      (Trava),A

02AA'                       REC_8:
02AA'   21 0360'                    LD      HL,Tab_Tecl     ; Prepara parametros para TRADUZ
02AD'   11 0005                     LD      DE,Lin_Tecl
02B0'   3A 3359'                    LD      A,(Lin_Det)
02B3'   4F                          LD      C,A
02B4'   3A 335A'                    LD      A,(Col_Det)

02B7'   CD 02D3'                    CALL    TRADUZ          ; Traduz p/ ASCII a tecla detectada

02BA'   CD 0306'                    CALL    INS_TELA        ; Coloca na tela

02BD'   C9                          RET

                                    PAGE
```

```
                        ;#################################################################
                        ;
                        ;         R O T I N A S   N I V E I S   III   E   IV
                        ;
                        ;#################################################################


                        ; Nome:        LIGA_PONTeiros
                        ;
                        ; Funcao:      Faz a ligacao entre a  tabela de ponteiros (armazenada
                        ;               em PONT_LIN) e as linhas de dados do terminal (arnaze-
                        ;               nadas em TELA e TELA_1).
                        ; Entradas:     A  - Numero de ligacoes a fazer
                        ;               BC - Numero de bytes/linha
                        ;               HL - Endereco da ma:triz de linhas
                        ;               IX - Endereco da tabela de ponteiros
                        ; Saida:        Ponteiros a partir de IX
                        ; Registradores destruidos:     A,HL,IX
                        ; Referencias externas:         Nenhuma
                        ;
02BE'                   LIGA_PONT:
02BE'   DD 75 00                LD      (IX),L                  ; VAR i:INTEGER;
02C1'   DD 74 01                LD      (IX+1),H                ;
02C4'   36 00                   LD      (HL),0                  ; FOR i := A DOWNTO 1 DO
02C6'   23                      INC     HL                      ; BEGIN
02C7'   36 00                   LD      (HL),0                  ;   pont_lin[i] := HL;
02C9'   2B                      DEC     HL              ; Patchao !!!
02CA'   09                      ADD     HL,BC                   ;    [HL] := 0;
02CB'   DD 23                   INC     IX                      ;    HL := HL + compr_linha
02CD'   DD 23                   INC     IX                      ; END;
02CF'   3D                      DEC     A
02D0'   20 EC                   JR      NZ,LIGA_PONT

02D2'   C9                      RET
```

```
                        ; Nome:       TRADUZ
                        ;
                        ; Funcao:     Gera um codigo ASCII correspondente `a tecla detetada
                        ; Entradas:   HL: Inicio da tabela de traducao
                        ;             DE: Numero de linhas da matriz de teclado
                        ;             A: Numero da coluna da tecla detetada
                        ;             C: Numero da linha da tecla detetada
                        ;             Variaveis Cntrl:  0 se a tecla de controle foi detetada
                        ;                       Maiusc: 0 se a tecla de maiusculas foi detetada
                        ;                       Trava:  0 se a tecla de trava foi detetada
                        ;             Constante Tam_Tab: Tamanho da tabela de minusculas
                        ; Saida:      Reg. A: Codigo ASCII do caracter
                        ; Destroi:    A,B
                        ; Referencia: Ninguem
                        ;
02D3'                   TRADUZ:
02D3'   B7                      OR      A                       ; WHILE (Col_Det <> 0) DO
02D4'   CA 02DC'                JP      Z,TRAD_2

02D7'                   TRAD_1:                                 ;   BEGIN
02D7'   19                      ADD     HL,DE                   ;     Trad_Tecl := Trad_Tecl+Lin_
02D8'   3D                      DEC     A                       ;     Col_Det := Col_Det - 1
02D9'   C2 02D7'                JP      NZ,TRAD_1               ;   END;

02DC'                   TRAD_2:
02DC'   06 00                   LD      B,0                     ; Trad_Tecl := Trad_Tecl + Lin_De
02DE'   09                      ADD     HL,BC

02DF'                   TRAD_3:
02DF'   3A 335D'                LD      A,(Cntrl)               ; IF (Cntrl <> 0)
02E2'   B7                      OR      A
02E3'   C2 02EC'                JP      NZ,TRAD_4

02E6'   11 0096                 LD      DE,Tam_Tab*3            ;  THEN DE := Tam_Tab*3
02E9'   C3 0303'                JP      TRAD_6

02EC'                   TRAD_4:
02EC'   3A 335C'                LD      A,(Maiusc)              ;  ELSE IF (Maiusc <> 0)
02EF'   B7                      OR      A
02F0'   C2 02F9'                JP      NZ,TRAD_5

02F3'   11 0064                 LD      DE,Tam_Tab*2            ;        THEN DE := Tam_Tab*2
02F6'   C3 0303'                JP      TRAD_6

02F9'                   TRAD_5:
```

```
02F9'   3A 335B'                LD      A,(Trava)               ;       ELSE IF (Trava <> 0)
02FC'   B7                      OR      A
02FD'   C2 0304'                JP      NZ,TRAD_7

0300'   11 0032                 LD      DE,Tam_Tab              ;               THEN DE := Tam_

0303'                   TRAD_6:
0303'   19                      ADD     HL,DE                   ; Calcula valor do ponteiro
0304'                   TRAD_7:
0304'   7E                      LD      A,(HL)                  ; Retorno: Pega caraceter na tat

0305'   C9                      RET
```

```
                        ; Nome:         INS_TELA
                        ;
                        ; Funcao:       Inserir dados lidos na estrutura de dados da tela
                        ; Entrada:      Dado no reg. A.
                        ; Saida:        Memoria de tela, P_INS, COL_COR
                        ; Registradores destruidos:      A,B
                        ; Referencias externas:          Nenhuma
                        ;
0306'                   INS_TELA:
0306'   2A 3132'                LD      HL,(P_INS)      ; Pega endereco a inserir
0309'   FE 0A                   CP      LF
030B'   CA 031B'                JP      Z,NOVA_LINHA
030E'   77                      LD      (HL),A          ; insere caracter
030F'   23                      INC     HL
0310'   22 3132'                LD      (P_INS),HL      ; atualiza ponteiro de insersao

0313'   21 3135'                LD      HL,COL_COR      ; Verifica se acabou esta linha
0316'   34                      INC     (HL)
0317'   7E                      LD      A,(HL)

0318'   FE 50                   CP      80              ; Se nro bytes na linha = 80, passa p/ p:
031A'   C0                      RET     NZ              ; Se nao passou, pronto.
```

```
                        ;
                        ; Caso tenha mudado de linha
                        ;
031B'                   NOVA_LINHA:
031B'   21 3134'                LD      HL,LIN_COR
031E'   34                      INC     (HL)            ; Incrementa contador de linhas
031F'   7E                      LD      A,(HL)
0320'   FE 19                   CP      25              ; Se passou da ultima linha, roda tela
0322'   CC 0343'                CALL    Z,RODA_CIMA
                        ;
                        ; Atualiza P_INS
                        ;
0325'   3A 3134'                LD      A,(LIN_COR)     ; pega linha corrente certa
0328'   87                      ADD     A,A             ; indexa para conseguir ponteiro
0329'   4F                      LD      C,A
032A'   06 00                   LD      B,0
032C'   DD 21 3139'             LD      IX,PONT_LIN
0330'   DD 09                   ADD     IX,BC

0332'   DD 6E 00                LD      L,(IX)          ; HL <-- endereco do 1. byte da linha
0335'   DD 66 01                LD      H,(IX+1)
0338'   23                      INC     HL              ; Pula 2 bytes do contador
0339'   23                      INC     HL
033A'   22 3132'                LD      (P_INS),HL

033D'   21 3135'                LD      HL,COL_COR      ; Zera contador de caracteres/linha
0340'   36 00                   LD      (HL),0

0342'   C9                      RET


0343'                   RODA_CIMA:
0343'   3E 00                   LD      A,0
0345'   32 3134'                LD      (LIN_COR),A     ; Reinicializa ponteiro para primeira lin

0348'   DD 21 3139'             LD      IX,PONT_LIN     ; atualiza P_INS para primeira posicao da
034C'   DD 6E 00                LD      L,(IX)
034F'   DD 66 01                LD      H,(IX+1)
0352'   23                      INC     HL
0353'   23                      INC     HL
0354'   22 3132'                LD      (P_INS),HL

0357'   C9                      RET
```

```
                        ; Nome:         TX_OK
                        ;
                        ; Funcao:     Espera porto serial terminar de enviar caracter
                        ; Entradas:   Nenhuma
                        ; Saida:      Segura o processamento ate' porto estar OK
                        ; Emprega:   Estado do porto serial
                        ; Destroi:    A
                        ; Referencia: Ninguem
                        ;
0358'                   TX_OK:
0358'   DB 21                   IN      A,(E_8251)      ; Captura estado do porto serial
035A'   E6 01                   AND     1               ; Verifica se TxRDY
035C'   CA 0358'                JP      Z,TX_OK         ; Espera TxRDY = 1
035F'   C9                      RET


                        ;#################################################################
                        ;#                                                               #
                        ;#                      Tabelas Fixas                            #
                        ;#                                                               #
                        ;#################################################################


                        ;
                        ; Tabela de minusculas
                        ;
0360'                   Tab_Tecl:
0360'   2D 70 00 2F             DB      '-',    'p',    NULO,   '/',    CR
0364'   0D
0365'   3A 6F 08 2E             DB      ':',    'o',    BS,     '.',    NULO
0369'   00
036A'   30 69 3B 2C             DB      '0',    'i',    ';',    ',',    NULO
036E'   00
036F'   39 75 6C 6D             DB      '9',    'u',    'l',    'm',    NULO
0373'   00
0374'   38 79 6B 6E             DB      '8',    'y',    'k',    'n',    ' '
```

```
0378'   20
0379'   37 74 6A 62                DB      '7',    't',    'j',    'b',    'a'
037D'   61
037E'   36 72 68 76                DB      '6',    'r',    'h',    'v',    ESC
0382'   1B
0383'   35 65 67 63                DB      '5',    'e',    'g',    'c',    '1'
0387'   31
0388'   34 77 66 78                DB      '4',    'w',    'f',    'x',    '2'
038C'   32
038D'   33 71 64 7A                DB      '3',    'q',    'd',    'z',    's'
0391'   73

0032                       Tam_Tab EQU     $-Tab_Tecl

                           ;
                           ; Tabela de trava
                           ;
0392'   2D 50 00 2F                DB      '-',    'P',    NULO,   '/',    CR
0396'   0D
0397'   3A 4F 08 2E                DB      ':',    'O',     BS,    '.',    NULO
039B'   00
039C'   30 49 3B 2C                DB      '0',    'I',    ';',    ',',    NULO
03A0'   00
03A1'   39 55 4C 4D                DB      '9',    'U',    'L',    'M',    NULO
03A5'   00
03A6'   38 59 4B 4E                DB      '8',    'Y',    'K',    'N',    ' '
03AA'   20
03AB'   37 54 4A 42                DB      '7',    'T',    'J',    'B',    'A'
03AF'   41
03B0'   36 52 48 56                DB      '6',    'R',    'H',    'V',    ESC
03B4'   1B
03B5'   35 45 47 43                DB      '5',    'E',    'G',    'C',    '1'
03B9'   31
03BA'   34 57 46 58                DB      '4',    'W',    'F',    'X',    '2'
03BE'   32
03BF'   33 51 44 5A                DB      '3',    'Q',    'D',    'Z',    'S'
03C3'   53

                           ;
                           ; Tabela de maiusculas
                           ;
03C4'   3D 50 00 3F                DB      '=',    'P',    NULO,   '?',    CR
03C8'   0D
03C9'   2A 4F 08 3E                DB      '*',    'O',     BS,    '>',    NULO
03CD'   00
```

```
03CE'   30 49 2B 3C                DB      '0',    'I',    '+',    '<',    NULO
03D2'   00
03D3'   29 55 4C 4D                DB      ')',    'U',    'L',    'M',    NULO
03D7'   00
03D8'   28 59 4B 4E                DB      '(',    'Y',    'K',    'N',    ' '
03DC'   20
03DD'   27 54 4A 42                DB      27h,    'T',    'J',    'B',    'A'
03E1'   41
03E2'   26 52 48 56                DB      '&',    'R',    'H',    'V',    ESC
03E6'   1B
03E7'   25 45 47 43                DB      '%',    'E',    'G',    'C',    '!'
03EB'   21
03EC'   24 57 46 58                DB      '$',    'W',    'F',    'X',    '"'
03F0'   22
03F1'   23 51 44 5A                DB      '#',    'Q',    'D',    'Z',    'S'
03F5'   53

                           ;
                           ; Tabela de controle
                           ;
03F6'   00 10 00 00                DB      NULO, 10H, NULO, NULO, NULO
03FA'   00
03FB'   00 0F 00 00                DB      NULO, 0FH, NULO, NULO, NULO
03FF'   00
0400'   00 09 00 00                DB      NULO, 09H, NULO, NULO, NULO
0404'   00
0405'   00 15 0C 0D                DB      NULO, 15H,  FF,  CR, NULO
0409'   00
040A'   00 19 1B 0E                DB      NULO, 19H, 1BH, 0EH, NULO
040E'   00
040F'   00 14 0A 02                DB      NULO, 14H,  LF, 02H, 01H
0413'   01
0414'   00 05 07 03                DB      NULO, 05H, 07H, 03H, NULO
0418'   00
0419'   00 12 08 16                DB      NULO, 12H, 08H, 16H, NULO
041D'   00
041E'   00 17 06 18                DB      NULO, 17H, 06H, 18H, NULO
0422'   00
0423'   00 11 04 1A                DB      NULO, 11H, 04H, 1AH, NULO
0427'   00
```

```
                    ;#################################################################
                    ;#                                                               #
                    ;#                    Area de Dados                              #
                    ;#                                                               #
                    ;#################################################################


                                ORG     3000H   ; Primeira RAM

3000'               Pilha:      DS      300
312C'               INI_PILHA   EQU     $
312C'               Cnt_Tecl:   DS      1       ; Contador de interrupcoes
312D'               Trat:       DS      1       ; Indicador de byte tratado.0..
312E'               Det:        DS      1       ; Indicador de byte detetado
312F'               BSalvo:     DS      1       ; Salvadores de registradores
3130'               CSalvo:     DS      1
3131'               DSalvo:     DS      1
3132'               P_INS:      DS      2       ; Ponteiro de insercao na memoria
3134'               LIN_COR:    DS      1       ; Linha corrente
3135'               COL_COR:    DS      1       ; Coluna corrente
3136'               LIN_MOSTRAR: DS     1       ; Linha a ser mostrada
3137'               PILHA_SALVA: DS     2       ; Endereco da pilha
3139'               PONT_LIN:   DS      NUM_LIN*2
316B'               TELA_1:     DS      TAM_LIN*(NUM_LIN-LIMITE)
3355'               IN1_Varr:   DS      1       ; Valor Inicial da varredura das linhas
3356'               IN2_Varr:   DS      1       ; Valor Inicial da varredura das colunas
3357'               LIN:        DS      1
3358'               COL:        DS      1
3359'               Lin_Det:    DS      1       ; Armazena a linha da tecla detectada
335A'               Col_Det:    DS      1       ; Armazena a coluna da tecla detectada
335B'               Trava:      DS      1       ; Armazena 0 se detetou a tecla de trava
335C'               Maiusc:     DS      1       ; Armazena 0 se detetou a tecla de maiusc
335D'               Cntrl:      DS      1       ; Armazena 0 se detetou a tecla de contro

                                ORG     4000H   ; Segunda RAM

4000'               TELA:       DS      TAM_LIN*LIMITE
47A8'               F           EQU     $       ; So' para aparecer na listagem
```

```
                    END     PPRINC
```

Macros:

Symbols:

BS	0008	BSALVO	312F'	C0_825	0010	C1_825	0011
C2_825	0012	CMD_82	0037	CMD_LI	0023	CMD_NI	00C0
CMD_PA	0000	CMD_ZE	00E0	CNTRL	335D'	CNT_TE	312C'
COL	3358'	COL_CO	3135'	COL_DE	335A'	COL_TE	000A
CONT0_	0032	CONT1_	0076	CONT2_	00B6	CR	000D
CSALVO	3130'	C_8275	0001	DELTA_	00C8	DET	312E'
DSALVO	3131'	D_8251	0020	ESC	001B	E_8251	0021
F	47A8'	FF	000C	IG_REG	023B'	IN1_VA	3355'
IN2_VA	3356'	INI_PI	312C'	INI_T2	0165'	INI_T3	016F'
INI_TE	0148'	INI_VA	00F2'	INS_TE	0306'	IN_1	0120'
IN_2	0131'	IN_3	0139'	IN_4	0147'	LE_ACH	0239'
LE_T1	0207'	LE_T2	0226'	LE_T3	0237'	LE_TEC	01FE'
LF	000A	LIGA_P	02BE'	LIGA_V	0197'	LIMITE	0014
LIN	3357'	LIN_CO	3134'	LIN_DE	3359'	LIN_MO	3136'
LIN_TE	0005	MAIUSC	335C'	MODO_8	00EF	M_8253	0013
M_8255	0033	NOVA_L	031B'	NULO	0000	NUM_LI	0019
P1_VAR	0031	P2_VAR	0032	PAR_1	004F	PAR_2	0058
PAR_3	0089	PAR_4	00DD	PILHA	3000'	PILHA_	3137'
PONT_L	3139'	POPS	006D'	PPRINC	0000'	PPR_1	0012'
PPR_2	0017'	PRG_55	0090	PRG_PE	00A9'	P_8275	0000
P_INS	3132'	P_LEIT	0030	RECEBE	024A'	REC_0	024E'
REC_1	025E'	REC_2	0258'	REC_4	0267'	REC_5	0281'
REC_6	029A'	REC_7	02A2'	REC_8	02AA'	RODA_C	0343'
R_BAUD	0005	TAB_TE	0360'	TAM_LI	0062	TAM_TA	0032
TECLAD	01A1'	TECL_1	01B9'	TECL_2	01F8'	TECL_4	01D8'
TECL_5	01EC'	TELA	4000'	TELA_1	3168'	TRADUZ	02D3'
TRAD_1	02D7'	TRAD_2	02DC'	TRAD_3	02DF'	TRAD_4	02EC'
TRAD_5	02F9'	TRAD_6	0303'	TRAD_7	0304'	TRAT	312D'
TRAVA	335B'	TR_1	0061'	TR_2	00A0'	TX_OK	0358'
T_BAUD	0005						

No Fatal error(s)

APÊNDICE G

PROGRAMA DE CONTROLE DO TERMINAL

```
                        ;          Laboratorio de Instrumentacao Eletronica - USP
                        ;
                        ;#######################################################################
                        ;
                        ;                        Terminal Reduzido
                        ;
                        ;      Funcoes basicas do terminal implementadas "na marra", sem sis-
                        ; tema operacional residente.
                        ;                                                    [33]TERMRD1.MAC
                        ;#######################################################################
                        ;
                        ;
                        ; V1.0  Original                      Alvaro/Gonzalo      5-AGO-1985
                        ; V2.0 temporariamente
                        ;
                        ;#######################################################################
                        ;                      USP-IFQSC-DFCM-LIE, CP 369, 13560-S. Carlos, SP
                        ;

                                .Z80
                                ORG     0


                        ;#######################################################################
                        ;#                                                                     #
                        ;#                    Constantes do Programa                           #
                        ;#                                                                     #
                        ;#######################################################################

                        ;
                        ; Bloco de Parametros do Terminal
                        ;
0046                    DELTA_TECL      EQU     70      ; Intervalo entre varreduras
003C                    REP1            EQU     60
000A                    REP2            EQU     10
0008                    Lin_Tecl        EQU     8       ; Numero de linhas da matriz teclado
000D                    Col_Tecl        EQU     13      ; Numero de colunas da matriz teclado
0005                    T_BAUD          EQU     5       ; Valor Inicial do contador para 4800 bps
0005                    R_BAUD          EQU     5       ; Valor Inicial do contador para 4800 bps
```

```
0018                    Num_Lin         EQU     24      ; Numero de linhas do terminal
0050                    Num_Col         EQU     80      ; Numero de colunas do terminal
0062                    Tam_Lin         EQU     98      ; Tamanho de cada linha na memoria
0014                    LIMITE          EQU     20      ; Numero de linhas armazenadas em TELA
                                                        ; As restantes sao guardadas em TELA_1
                        ;
                        ; Constantes
                        ;
000A                    LF              EQU     0AH     ; ASCII para avancar linha
000D                    CR              EQU     0DH     ; ASCII para retorno de carro
000C                    FF              EQU     0CH     ; ASCII para Novo Formulario
0008                    BS              EQU     08H     ; ASCII para retornar um caracter
007F                    DEL             EQU     7FH     ; ASCII para "delete"
001B                    ESC             EQU     1BH     ; ASCII para Escape
0009                    TAB             EQU     09H     ; ASCII para Tabulacao
0000                    NIL             EQU     0
00F0                    TRV             EQU     0F0H    ; Codigo especial para TRAVA (teclado)
00F1                    SLC             EQU     0F1H    ; Codigo especial para LOCAL/LINHA (tecla

                                                        ;

0032                    CONT0_53        EQU     32H     ; Palavra de controle da 8253
                                                        ; 7:6 - Seleciona o contador 0
                                                        ; 5:4 - Carrega B-S seguido por B+S
                                                        ; 3:1 - Modo 1: Monoestavel
                                                        ; 0   - Conta em 16 bits


0076                    CONT1_53        EQU     76H     ; Palavra de controle do contador 1
                                                        ;       - Seleciona contador 1
                                                        ;       - Carrega B-S seguido por B+S
                                                        ;       - Modo 3 (gerador de onda quadrad
                                                        ;       - Contador em 16 bits

00B6                    CONT2_53        EQU     0B6H    ; Palavra de controle do contador 2
                                                        ;       - Seleciona contador 2
                                                        ;       - Carrega B-S seguido por B+S
                                                        ;       - Modo 3 (gerador de onda quadrad
                                                        ;       - Contador em 16 bits

008B                    PRG_55          EQU     8BH     ; Modo de operacao da 8255:
                                                        ;       - Modo 0 (E/S generica)
                                                        ;       - Portas A e C: entrada
                                                        ;       - Porta B: saida

0000                    CMD_PARA        EQU     00      ; Comando para parar a operacao do 8275
```

```
00C0                    CMD_NINT        EQU     0C0H    ; Comando para desabilitar interrupcao
00E0                    CMD_ZERA        EQU     0E0H    ; Comando para zerar contadores do 8275
0080                    Cmd_Curs        EQU     80h     ; Comando para carregar a posicao do cu
0023                    CMD_LIGA        EQU     23H     ; Comando para comecar a operacao do 82
                                                        ; 4:2 - 0 ciclos entre pedidos de ADM
                                                        ; 1:0 - 8 ciclos de ADM por pedido

004F                    PAR_1           EQU     4FH     ; Parametro 1 do CMD_PARA
                                                        ; 7 - tela normal
                                                        ; 6:0 - 80 caracteres por linha

0057                    PAR_2           EQU     57H     ; Parametro 2 do CMD_PARA
                                                        ; 7:6 - 2 linhas por retraco vertical
                                                        ; 5:0 - 25 linhas por tela

0089                    PAR_3           EQU     89H     ; Parametro 3 do CMD_PARA
                                                        ; 7:4 - Subscrito colocado na 9 varredu
                                                        ; 3:0 - 10 varreduras por caracter

00DD                    PAR_4           EQU     0DDH    ; Parametro 4 do CMD_PARA
                                                        ; 7  -  Incrementa de 1 contagem de ret
                                                        ; 6  -  Modo nao transparente
                                                        ; 5:4 - Cursor sublinhado piscante
                                                        ; 3:0 - 32 retracos por HRTC

007F                    MascB           EQU     01111111B  ; 7 colunas de varredura no porto B
003F                    MascC           EQU     00111111B  ; 6 colunas de varredura no porto C

                                .8080

0001                    T1      SET     1

                                REPT    Lin_Tecl-1

                        T2      SET     2*T1
                        T1      SET     T2

                                ENDM
0002                +   T2      SET     2*T1
0002                +   T1      SET     T2
0004                +   T2      SET     2*T1
0004                +   T1      SET     T2
0008                +   T2      SET     2*T1
0008                +   T1      SET     T2
0010                +   T2      SET     2*T1
```

```
0010                +     T1      SET     T2
0020                +     T2      SET     2*T1
0020                +     T1      SET     T2
0040                +     T2      SET     2*T1
0040                +     T1      SET     T2
0080                +     T2      SET     2*T1
0080                +     T1      SET     T2
                                  .Z80

007F                      I_Var   EQU     T1-1


                          ;#################################################################
                          ;#                                                               #
                          ;#                    Endereco dos perifericos                   #
                          ;#                                                               #
                          ;#################################################################

0000                      P_8275          EQU     00H             ; Porto de parametros da 8275
0001                      C_8275          EQU     P_8275+1        ; Porto de comando da 8275

0010                      C0_8253         EQU     10H             ; 8253: Contador 0
0011                      C1_8253         EQU     C0_8253+1       ; 8253: Contador 1
0012                      C2_8253         EQU     C0_8253+2       ; 8253: Contador 2
0013                      M_8253          EQU     C0_8253+3       ; 8253: Modo


0020                      D_SIO           EQU     020H
0021                      E_SIO           EQU     D_SIO+1

0022                      D_AUX           EQU     D_SIO+2
0023                      E_AUX           EQU     D_SIO+3


0030                      PA              EQU     30H
0031                      PB              EQU     PA+1
0032                      PC              EQU     PA+2
0033                      M_8255          EQU     PA+3

                                          PAGE
```

```
                        ;#######################################################################
                        ;#
                        ;#                          Programa Principal
                        ;#
                        ;#######################################################################

0000'                   PPRINC:
0000'   F3                      DI
0001'   31 312C'                LD      SP,INI_PILHA
0004'   ED 56                   IM      1

0006'   CD 00E9'                CALL    PRG_PERIF               ; Incializa perifericos
0009'   CD 0121'                CALL    INI_VAR                 ; Inicializa variaveis
000C'   CD 0142'                CALL    INI_TELA                ; Inicializa a tela
000F'   CD 01A4'                CALL    LIGA_VIDEO              ; Comeca a mostrar a tela

0012'   C3 00BC'                JP      PPR_1                   ; Vai para inicio do programa pri

                                PAGE
```

```
                        ;#################################################################
                        ;#                                                               #
                        ;#          Vetor acessado quando ocorre uma interrupcao         #
                        ;#                                                               #
                        ;#################################################################

                                ORG     38H

0038'   F3                      DI
0039'   D9                      EXX                     ; Salva estado da maquina
003A'   08                      EX      AF,AF'
003B'   21 0000                 LD      HL,0            ; Salva ponteiro de pilha
003E'   39                      ADD     HL,SP
003F'   22 3136'                LD      (PILHA_SALVA),HL
                        ;
                        ; Pega contador de bytes/linha, aponta SP para primeiro byte da linha
                        ;
0042'   21 32CE'                LD      HL,PONT_LIN
0045'   3A 32CD'                LD      A,(LIN_MOSTRAR) ; HL <-- PONT_LIN[LIN_MOSTRAR*2]
0048'   87                      ADD     A,A
0049'   5F                      LD      E,A
004A'   16 00                   LD      D,0
004C'   19                      ADD     HL,DE

004D'   5E                      LD      E,(HL)          ; DE <-- endereco do contador de bytes/li
004E'   23                      INC     HL              ; (tambem endereca inicio dessa linha)
004F'   56                      LD      D,(HL)

0050'   EB                      EX      DE,HL           ; BC <-- TELA[ DE ] (contador bytes/linha
0051'   4E                      LD      C,(HL)
0052'   06 00                   LD      B,0

0054'   11 8002                 LD      DE,8000H+2      ; SP := primeiro byte da linha + 8000H
```

```
0057'  19                          ADD     HL,DE
0058'  F9                          LD      SP,HL
                            ;
                            ; Transferencia dos bytes para o controlador de video
                            ;
0059'  21 006F'                    LD      HL,POPS         ; Calcula nro bytes a mais que 80 nessa l
005C'  B7                          OR      A
005D'  CB 29                       SRA     C               ; C := C/2
005F'  D2 0063'                    JP      NC,TR_1         ; Se (CY = 1) entao E := E+1
0062'  0C                          INC     C
0063'                       TR_1:
0063'  B7                          OR      A               ; Limpa indicador CY
0064'  ED 42                       SBC     HL,BC
0066'  E9                          JP      (HL)

                                   REPT    8               ; Transferencia dos bytes a mais nessa li
                                    POP    BC
                                   ENDM
0067'  C1                +          POP    BC
0068'  C1                +          POP    BC
0069'  C1                +          POP    BC
006A'  C1                +          POP    BC
006B'  C1                +          POP    BC
006C'  C1                +          POP    BC
006D'  C1                +          POP    BC
006E'  C1                +          POP    BC
006F'                       POPS:
                                   REPT    40              ; Transferencia de uma linha normal
                                    POP    BC
                                   ENDM
006F'  C1                +          POP    BC
0070'  C1                +          POP    BC
0071'  C1                +          POP    BC
0072'  C1                +          POP    BC
0073'  C1                +          POP    BC
0074'  C1                +          POP    BC
0075'  C1                +          POP    BC
0076'  C1                +          POP    BC
0077'  C1                +          POP    BC
0078'  C1                +          POP    BC
0079'  C1                +          POP    BC
007A'  C1                +          POP    BC
007B'  C1                +          POP    BC
007C'  C1                +          POP    BC
007D'  C1                +          POP    BC
007E'  C1                +          POP    BC
```

```
007F'  C1          +          POP   BC
0080'  C1          +          POP   BC
0081'  C1          +          POP   BC
0082'  C1          +          POP   BC
0083'  C1          +          POP   BC
0084'  C1          +          POP   BC
0085'  C1          +          POP   BC
0086'  C1          +          POP   BC
0087'  C1          +          POP   BC
0088'  C1          +          POP   BC
0089'  C1          +          POP   BC
008A'  C1          +          POP   BC
008B'  C1          +          POP   BC
008C'  C1          +          POP   BC
008D'  C1          +          POP   BC
008E'  C1          +          POP   BC
008F'  C1          +          POP   BC
0090'  C1          +          POP   BC
0091'  C1          +          POP   BC
0092'  C1          +          POP   BC
0093'  C1          +          POP   BC
0094'  C1          +          POP   BC
0095'  C1          +          POP   BC
0096'  C1          +          POP   BC
                        ;
                        ; Atualiza ponteiro de linha a mostrar
                        ;
0097'  21 32CD'                LD    HL,LIN_MOSTRAR  ; LIN_MOSTRAR := LIN_MOSTRAR + 1
009A'  34                      INC   (HL)

009B'  7E                      LD    A,(HL)          ; Se (LIN_MOSTRAR = 24) entao LIN_MOSTRAR
009C'  FE 18                   CP    Num_Lin
009E'  C2 00A3'                JP    NZ,TR_2
00A1'  36 00                   LD    (HL),0
                        ;
                        ; Restaura estado da maquina e retorna
                        ;
00A3'                   TR_2:
00A3'  2A 3136'                LD    HL,(PILHA_SALVA)
00A6'  F9                      LD    SP,HL
00A7'  D9                      EXX
00A8'  08                      EX    AF,AF'
00A9'  FB                      EI
00AA'  ED 4D                   RETI

                               PAGE
```

```
                        ;
                        ; Continuacao do programa principal do terminal
                        ;


                        ;
                        ; Bloco de parametros para a programacao da SIO
                        ;
00AC'                   BLOCO_SIO:
                        ;
                        ; programacao do canal A
                        ;
00AC'   04                      DB      4               ; prepara para escrever registrador 4
00AD'   CC                      DB      11001100B       ; sem paridade, 2 bits de parada, modo
                                                        ; assincrono, clock x64.

00AE'   05                      DB      5               ; prepara para escrever registrador 5
00AF'   EA                      DB      11101010B       ; & bits por caracter na transmissao

00B0'   03                      DB      3               ; prepara para escrever registrador 3
00B1'   E1                      DB      11100001B       ; & bits por caracter na recepcao

00B2'   01                      DB      1               ; prepara para escrever registrador 1
00B3'   00                      DB      00000000B       ; interrupacoes desabilitadas
                        ;
                        ; Programacao do canal B
                        ;
00B4'   04                      DB      4               ; prepara para escrever registrador 4
00B5'   CC                      DB      11001100B       ; sem paridade, 2 bits de parada, modo
                                                        ; assincrono, clock x64.

00B6'   05                      DB      5               ; prepara para escrever registrador 5
00B7'   EA                      DB      11101010B       ; & bits por caracter na transmissao

00B8'   03                      DB      3               ; prepara para escrever registrador 3
00B9'   E1                      DB      11100001B       ; & bits por caracter na recepcao

00BA'   01                      DB      1               ; prepara para escrever registrador 1
00BB'   00                      DB      00000000B       ; interrupacoes desabilitadas
```

```
                        ;
                        ; Inicio da parte principal do programa
                        ;
                        ;
00BC'                   PPR_1:
00BC'   CD 01C9'                CALL    TECLADO                 ; Trata teclado
00BF'   FE FF                   CP      0FFH
00C1'   CA 00D4'                JP      Z,PPR_2
00C4'   3A 32FE'                LD      A,(Local)
00C7'   B7                      OR      A
00C8'   CA 00D1'                JP      Z,PPR_3
00CB'   CD 023E'                CALL    POE_NA_TELA
00CE'   C3 00D4'                JP      PPR_2
00D1'                   PPR_3:
00D1'   CD 0251'                CALL    ENVIA
00D4'                   PPR_2:
00D4'   CD 025C'                CALL    LINHA
00D7'   FE FF                   CP      0FFH
00D9'   CA 00BC'                JP      Z,PPR_1
00DC'   3A 32FE'                LD      A,(LOCAL)
00DF'   B7                      OR      A
00E0'   C2 00BC'                JP      NZ,PPR_1
00E3'   CD 023E'                CALL    POE_NA_TELA
00E6'   C3 00BC'                JP      PPR_1


                        ;#######################################################################
                        ;#
                        ;#                      Rotinas Nivel I
                        ;#
                        ;#######################################################################


                        ; Nome:          PRG_PERIF
                        ;
                        ; Funcao:        Inicializa e programa os perif'ericos (51, 53 e 75)
                        ; Entrada:       Nenhuma
```

```
                        ; Saida:        Nenhuma
                        ; Emprega:      Na saida, perif'ericos programados
                        ; Registradores destruidos:     A,H,L
                        ; Referencias externas:         Nenhuma
                        ;
00E9'                   PRG_PERIF:

                        ;
                        ; Programacao da 8253
                        ;
00E9'   3E 32                   LD      A,CONT0_53      ; Programa contador 0
00EB'   D3 13                   OUT     (M_8253),A
00ED'   D3 10                   OUT     (C0_8253),A
00EF'   3E 00                   LD      A,00
00F1'   D3 10                   OUT     (C0_8253),A

00F3'   3E 76                   LD      A,CONT1_53      ; Programa contador 1
00F5'   D3 13                   OUT     (M_8253),A
00F7'   21 0005                 LD      HL,T_BAUD       ; HL <-- valor de contagem para gerar TxC
00FA'   7D                      LD      A,L             ; Valor inicial do contador
00FB'   D3 11                   OUT     (C1_8253),A
00FD'   7C                      LD      A,H
00FE'   D3 11                   OUT     (C1_8253),A

0100'   3E B6                   LD      A,CONT2_53      ; Programa o contador 2
0102'   D3 13                   OUT     (M_8253),A
0104'   21 0005                 LD      HL,R_BAUD       ; HL <-- valor de contagem para gerar TxC
0107'   7D                      LD      A,L             ; Valor inicial do contador
0108'   D3 12                   OUT     (C2_8253),A
010A'   7C                      LD      A,H
010B'   D3 12                   OUT     (C2_8253),A

                        ;  Programacao da SIO

010D'   0E 21                   LD      C,E_SIO         ; Prepara para programar o canal A
010F'   21 00AC'                LD      HL,BLOCO_SIO    ; bloco de parametros
0112'   06 08                   LD      B,8             ; contador de bytes a transferir
0114'   ED B3                   OTIR                    ; transferencia dos bytes para a SIO

0116'   0E 21                   LD      C,E_SIO         ; Prepara para programar o canal A
0118'   06 08                   LD      B,8             ; contador de bytes a transferir
011A'   ED B3                   OTIR                    ; transferencia dos bytes para a SIO

                        ;
                        ; Programacao da 8255
                        ;
```

```
011C'   3E 8B                   LD      A,PRG_55
011E'   D3 33                   OUT     (M_8255),A

0120'   C9                      RET


                        ; Nome:     INIcializa_VARiaveis.
                        ;
                        ; Funcao:   Inicializa as variaveis utilizadas pela rotina TECLADO.
                        ; Entradas: nenhuma
                        ; Saida:    Variaveis Maiusc, Cntrl, Trava, Trat, Det e Cnt_Tecl iniciali
                        ; Destroi:    A
                        ; Referencia: ninguem.
                        ;
0121'                   INI_VAR:
0121'   3E 00                   LD      A,0
0123'   32 312D'                LD      (Trat),A
0126'   32 312E'                LD      (Det),A
0129'   3E 46                   LD      A,Delta_Tecl
012B'   32 312C'                LD      (Cnt_Tecl),A

012E'   3E FF                   LD      A,0FFH
0130'   32 32C3'                LD      (Maiusc),A
0133'   32 32C4'                LD      (Cntrl),A

0136'   3E 00                   LD      A,0
0138'   32 32C2'                LD      (Trava),A

013B'   21 0331'                LD      HL,NORMAL
013E'   22 32FF'                LD      (TECALT),HL

0141'   C9                      RET


                        ;
                        ; Nome:       INIcializa_TELA
                        ;
```

```
                        ; Funcao:     Inicializa a estrutura de dados da tela
                        ; Entradas:   Nenhuma
                        ; Saida:      Inicializados:  TELA com  brancos,  PONT_LIN com  ponteiro
                        ;             para as linhas,  primeiro byte de cada  linha com 0 apontan
                        ;             do  para  o primeiro  byte livre da primeira linha.
                        ; Destroi:    A,B,C,D,E,H,L,IX
                        ; Referencia: Ninguem
                        ;
0142'                   INI_TELA:
0142'   3E 00                   LD      A,0
0144'   32 3134'                LD      (LIN_COR),A
0147'   32 3135'                LD      (COL_COR),A
014A'   DD 21 32CE'             LD      IX,PONT_LIN
014E'   01 0062                 LD      BC,TAM_LIN

0151'   3E 14                   LD      A,LIMITE                ; Inicializa primeiras linhas
0153'   21 4000'                LD      HL,TELA
0156'   CD 0499'                CALL    LIGA_PONT

0159'   3E 04                   LD      A,NUM_LIN-LIMITE        ; Inicializa linhas finais
015B'   21 3138'                LD      HL,TELA_1
015E'   CD 0499'                CALL    LIGA_PONT
                        ;
                        ; Inicializa linhas com brancos
                        ;
0161'   DD 21 32CE'             LD      IX,PONT_LIN
0165'   06 18                   LD      B,NUM_LIN
0167'                   INI_T2:
0167'   DD 6E 00                LD      L,(IX)                  ; Pega ponteiro do inicio da linh
016A'   DD 66 01                LD      H,(IX+1)
016D'   36 00                   LD      (HL),0
016F'   23                      INC     HL
0170'   36 00                   LD      (HL),0
0172'   23                      INC     HL
0173'   36 F1                   LD      (HL),0F1H

0175'   0E 5F                   LD      C,95
0177'                   INI_T1:
0177'   36 20                   LD      (HL),' '
0179'   23                      INC     HL
017A'   0D                      DEC     C
017B'   C2 0177'                JP      NZ,INI_T1

017E'   DD 23                   INC     IX                      ; IX <-- pont. da proxima linha
0180'   DD 23                   INC     IX
0182'   05                      DEC     B                       ; Repete para todas as linhas da
```


BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

```
0183'  C2 0167'          JP      NZ,INI_T2

0186'  AF                XOR     A                    ; Inicializa ponteiro de linha co
0187'  32 32CB'          LD      (INI_RODA),A
018A'  32 32CD'          LD      (LIN_MOSTRAR),A      ; Inicializa variavel para interr

018D'  32 32C5'          LD      (IndEsc),A

0190'  32 32C6'          LD      (PAR_CURS),A         ; Inicializa indicador de espera
                                                      ; parametro de posicionamento de

0193'  3E 17             LD      A,23
0195'  32 32CC'          LD      (FIM_RODA),A

0198'  3E 00             LD      A,0
019A'  32 32FE'          LD      (LOCAL),A

019D'  CD 07E6'          CALL    PosicHL              ; Posiciona HL na linha 0 coluna
01A0'  22 3132'          LD      (P_INS),HL

01A3'  C9                RET


                         ; Nome:        LIGA_VIDEO
                         ;
                         ; Funcao:      Permitir que o controlador de v'ideo comece a interrom-
                         ;               per o programa para receber dados.
                         ; Entrada:     Nenhuma
                         ; Saida:       Nenhuma
                         ; Estraga:     A
                         ; Referencia:  Ninguem
                         ;
01A4'                    LIGA_VIDEO:
                         ;
                         ; Programa o controlador de video (8275)
                         ;

01A4'  3E 00             LD      A,CMD_PARA      ; Carrega reg. com. com ordem de parada
01A6'  D3 01             OUT     (C_8275),A
```

```
01A8'   3E 4F                   LD      A,PAR_1         ; Carrega parametros no 8275
01AA'   D3 00                   OUT     (P_8275),A
01AC'   3E 57                   LD      A,PAR_2
01AE'   D3 00                   OUT     (P_8275),A
01B0'   3E 89                   LD      A,PAR_3
01B2'   D3 00                   OUT     (P_8275),A
01B4'   3E DD                   LD      A,PAR_4
01B6'   D3 00                   OUT     (P_8275),A

01B8'   CD 07CB'                CALL    CARR_CURS

01BB'   3E E0                   LD      A,CMD_ZERA      ; Zera contadores do 8275
01BD'   D3 01                   OUT     (C_8275),A


01BF'   3E 23                   LD      A,CMD_LIGA      ; Dispara o 8275
01C1'   D3 01                   OUT     (C_8275),A

01C3'   3E C0                   LD      A,CMD_NINT      ; Nao habilita interrupcoes do 8275
01C5'   D3 01                   OUT     (C_8275),A

01C7'   FB                      EI                      ; Habilita interrupcoes na UCP
01C8'   C9                      RET


                        ; Nome:     TECLADO
                        ;
                        ; Funcao:   Realiza o processamento de varredura do teclado.
                        ; Entradas: Portos A, B e C. Variaveis globais Trat e Det
                        ; Saida:    nenhuma no momento.
                        ; Destroi:    A, B, C, D, H, L.
                        ; Referencia: LE_TECL, IG_REG_SALVO e RECEBE.
                        ;

01C9'                   TECLADO:
01C9'   21 312C'                LD      HL,CNT_TECL
01CC'   35                      DEC     (HL)
01CD'   7E                      LD      A,(HL)
01CE'   B7                      OR      A
```

```
01CF'   CA 01D5'                JP      Z,TECL_0
01D2'   3E FF                   LD      A,0FFH
01D4'   C9                      RET

01D5'                   TECL_0:
01D5'   3E 46                   LD      A,DELTA_TECL
01D7'   77                      LD      (HL),A
01D8'   CD 0273'                CALL    LE_TECL
01DB'   B7                      OR      A
01DC'   C2 01EE'                JP      NZ,TECL_1
01DF'   AF                      XOR     A
01E0'   32 312D'                LD      (TRAT),A
01E3'   32 312E'                LD      (DET),A

                        ;
                        ; Para auto-repeticao {
                        ;
01E6'   3E 3C                   LD      A,REP1
01E8'   32 3301'                LD      (CONTREP),A
                        ; }

01EB'   C3 021E'                JP      TECL_4
01EE'                   TECL_1:
01EE'   3A 312D'                LD      A,(TRAT)
01F1'   B7                      OR      A
01F2'   C2 0221'                JP      NZ,TECL_5
01F5'   3A 312E'                LD      A,(DET)
01F8'   B7                      OR      A
01F9'   C2 0207'                JP      NZ,TECL_2
01FC'   CD 0231'                CALL    SALVABCD
01FF'   3E 01                   LD      A,1
0201'   32 312E'                LD      (DET),A
0204'   C3 021E'                JP      TECL_4
0207'                   TECL_2:
0207'   CD 028F'                CALL    IG_REG_SALVO
020A'   C2 0216'                JP      NZ,TECL_3
020D'   3E 01                   LD      A,1
020F'   32 312D'                LD      (TRAT),A
0212'   CD 029E'                CALL    RECEBE
0215'   C9                      RET

0216'                   TECL_3:
0216'   CD 0231'                CALL    SALVABCD
                        ;
                        ; Para auto-repeticao {
                        ;
```

```
0219'   3E 3C                   LD      A,REP1
021B'   32 3301'                LD      (CONTREP),A
                        ; }

021E'                   TECL_4:
021E'   3E FF                   LD      A,0FFH
0220'   C9                      RET
                        ;
                        ; Para auto-repeticao {
                        ;
0221'                   TECL_5:
0221'   21 3301'                LD      HL,CONTREP
0224'   35                      DEC     (HL)
0225'   C2 021E'                JP      NZ,TECL_4

0228'   CD 029E'                CALL    RECEBE

022B'   3E 0A                   LD      A,REP2
022D'   32 3301'                LD      (CONTREP),A
0230'   C9                      RET
                        ; }


0231'                   SALVABCD:
0231'   78                      LD      A,B
0232'   32 312F'                LD      (BSALVO),A
0235'   79                      LD      A,C
0236'   32 3130'                LD      (CSALVO),A
0239'   7A                      LD      A,D
023A'   32 3131'                LD      (DSALVO),A
023D'   C9                      RET

                                PAGE
```

```
                                ;
                                ; Rotina POE_NA_TELA
                                ;
023E'                           POE_NA_TELA:
023E'   79                              LD      A,C
023F'   CD 04E2'                        CALL    INTERP
0242'   B7                              OR      A
0243'   C4 0780'                        CALL    NZ,INS_TELA
0246'   C9                              RET

                                ;
                                ; ROTINA PARA MANDAR O CARACTER RECEBIDO PELO CANAL CORRETO:
                                ;     Tela se em modo LOCAL
                                ;     Porto serial se em modo linha
                                ;
0247'                           MANDA:
0247'   3A 32FE'                        LD      A,(LOCAL)
024A'   B7                              OR      A
024B'   CA 0251'                        JP      Z,ENVIA
024E'   C3 023E'                        JP      POE_NA_TELA
                                ;
                                ;
                                ;  ROTINA ENVIA
                                ;
0251'                           ENVIA:
0251'   79                              LD      A,C
0252'   FE 00                           CP      NIL
0254'   C8                              RET     Z

0255'   CD 07DA'                        CALL    TX_OK
0258'   79                              LD      A,C
0259'   D3 20                           OUT     (D_SIO),A
025B'   C9                              RET

                                ;
                                ;  ROTINA LINHA
                                ;
025C'                           LINHA:

025C'   3E 00                           LD      A,0
025E'   D3 21                           OUT     (E_SIO),A
```

```
0260'  DB 21                    IN      A,(E_SIO)
0262'  E6 01                    AND     01H
0264'  CA 026D'                 JP      Z,LIN_1
0267'  DB 20                    IN      A,(D_SIO)
0269'  E6 7F                    AND     7FH
026B'  4F                       LD      C,A
026C'  C9                       RET
026D'                   LIN_1:
026D'  3E FF                    LD      A,0FFH
026F'  C9                       RET


0270'                   SEM_DADO:
0270'  3E FF                    LD      A,0FFH
0272'  C9                       RET

                        ;#######################################################################
                        ;#
                        ;#                      Rotinas Nivel II
                        ;#
                        ;#######################################################################

                        ; Nome:  LE_TECLado
                        ;
                        ; Funcao:     Varre a matriz de teclado e detecta uma tecla pressionada.
                        ; Entradas:   Nenhuma
                        ; Saida:      A: 0 se nao detectou tecla, <> 0 se detectou
                        ;             B, C: Colunas da tecla detectada
                        ;             D: Linha da tecla  detectada
                        ; Emprega:    Portos B e C para varredura, porto A para recepcao
                        ; Destroi:    A, B, C, D
                        ; Referencia: Ninguem
                        ;
0273'                   LE_TECL:
0273'  16 7F                    LD      D,I_Var

0275'                   Varre:
0275'  7A                       LD      A,D
0276'  D3 30                    OUT     (PA),A
0278'  DB 32                    IN      A,(PC)
027A'  4F                       LD      C,A
027B'  DB 31                    IN      A,(PB)
027D'  47                       LD      B,A

027E'  2F                       CPL
027F'  E6 7F                    AND     MascB
```

```
0281'   C0                      RET     NZ
0282'   79                      LD      A,C
0283'   2F                      CPL
0284'   E6 3F                   AND     MascC
0286'   C0                      RET     NZ
0287'   CB 0A                   RRC     D
0289'   DA 0275'                JP      C,Varre

028C'   3E 00                   LD      A,0
028E'   C9                      RET


                        ;
                        ; Nome:  IGuais_REGistradores_SALVO
                        ;
                        ; Funcao:     Verifica se os registradoes B, C e D sao iguais as variavei
                        ;             BSalvo, CSalvo e DSalvo.
                        ; Entradas:   B, C e D com os valores a ser comparados
                        ; Saida:      flag ZERO setado caso sejam iguais
                        ; Emprega:    Nada
                        ; Destroi:    A
                        ; Referencia: ninguem.
                        ;
028F'                   IG_REG_SALVO:
028F'   3A 312F'                LD      A,(BSalvo)
0292'   B8                      CP      B
0293'   C0                      RET     NZ

0294'   3A 3130'                LD      A,(CSalvo)
0297'   B9                      CP      C
0298'   C0                      RET     NZ

0299'   3A 3131'                LD      A,(DSalvo)
029C'   BA                      CP      D
029D'   C9                      RET


                        ; Nome:     RECEBE
                        ;
                        ; Funcao:   Faz o processamento necessario para enviar uma letra `a tela
                        ;
```

```
                        ; Entradas: B e C: coluna da matriz teclado
                        ;           D: linha da matriz teclado
                        ; Saida:    Nenhuma
                        ; Emprega:  Tela para a saida dos caracteres detectados
                        ; Destroi:  A, B, C, D, E
                        ; Referencia: TRADUZ,INTERP
                        ;
029E'                   RECEBE:
029E'   69                      LD      L,C             ; Salva indicadores de SHIFT e CONTROL
029F'   1E 08                   LD      E,8             ; Numero maximo de rotacoes
02A1'   3E 00                   LD      A,0             ; Valor inicial do contador
02A3'                   REC_0:
02A3'   3C                      INC     A               ; Incrementa contador
02A4'   CB 08                   RRC     B               ; Roda o reg. B, bit B0 --> CY
02A6'   D2 02B3'                JP      NC,REC_1        ; Se ja' detectou 0 vai imprimir contado
02A9'   1D                      DEC     E
02AA'   C2 02A3'                JP      NZ,REC_0        ; Repete 8 vezes no maximo
                        ;
                        ; Nao detectou zero no reg. B. Tenta no reg. C
                        ;
02AD'                   REC_2:
02AD'   3C                      INC     A               ; Incrementa contador
02AE'   CB 09                   RRC     C               ; Roda o reg. C, bit C0 --> CY
02B0'   DA 02AD'                JP      C,REC_2

02B3'                   REC_1:
02B3'   3D                      DEC     A               ; Ajusta contador (estava adiantado de 1
02B4'   32 32C1'                LD      (Col_Det),A     ; Salva coordenada da tecla detectada
                        ;
                        ; Encontra posicao do bit em 0 no registrador D
                        ;
02B7'   1E 08                   LD      E,8             ; Numero maximo de repeticoes
02B9'   3E 00                   LD      A,0             ; Valor inicial do contador
02BB'                   REC_4:
02BB'   3C                      INC     A               ; Incrementa contador
02BC'   CB 0A                   RRC     D               ; Roda o reg. D, bit D0 --> CY
02BE'   D2 02C5'                JP      NC,REC_5        ; Se ja' detectou 0 vai imprimir contador
02C1'   1D                      DEC     E
02C2'   C2 02BB'                JP      NZ,REC_4        ; Repete 8 vezes no maximo
                        ;
                        ; Define variavel do bit em 0 no registrador D
                        ;
02C5'                   REC_5:
02C5'   3D                      DEC     A               ; Ajusta contador que estava adiantado de
02C6'   32 32C0'                LD      (Lin_Det),A     ; Salva coordenada linha da tecla detecta
```

```
                        ;
                        ; Verifica se estava apertado teclas de MAIUSC ou CNTRL
                        ;

02C9'   3E 01                   LD      A,1
02CB'   32 32C3'                LD      (Maiusc),A
02CE'   32 32C4'                LD      (Cntrl),A

02D1'   3E 00                   LD      A,0

02D3'   CB 75                   BIT     6,L             ; Se (Bit_maiusc = 0)
02D5'   C2 02DB'                JP      NZ,REC_6
02D8'   32 32C3'                LD      (Maiusc),A

02DB'                   REC_6:
02DB'   CB 7D                   BIT     7,L             ; Se (Bit_cntrl = 0)
02DD'   C2 02E3'                JP      NZ,REC_7
02E0'   32 32C4'                LD      (Cntrl),A

02E3'                   REC_7:
02E3'   21 081D'                LD      HL,Tab_Tecl     ; Prepara parametros para TRADUZ
02E6'   11 0008                 LD      DE,Lin_Tecl
02E9'   3A 32C0'                LD      A,(Lin_Det)
02EC'   4F                      LD      C,A
02ED'   3A 32C1'                LD      A,(Col_Det)

02F0'   CD 04AF'                CALL    TRADUZ          ; Traduz p/ ASCII a tecla detectada
02F3'   CD 02F8'                CALL    TEC_ESP
02F6'   4F                      LD      C,A
02F7'   C9                      RET
                        ;
                        ; Esta rotina serve para o tratamento de teclas especiais
                        ; Por enquanto so trata trava.
                        ;
02F8'                   TEC_ESP:
02F8'   FE 80                   CP      80H
02FA'   D8                      RET     C

02FB'   FE F0                   CP      TRV
02FD'   CA 031D'                JP      Z,TES_1
0300'   FE F1                   CP      SLC
0302'   CA 0327'                JP      Z,TES_2

0305'   D6 80                   SUB     80H
0307'   CB 27                   SLA     A
0309'   CB 27                   SLA     A
```

```
030B'   4F                          LD      C,A
030C'   06 00                       LD      B,0
030E'   2A 32FF'                    LD      HL,(TECALT)
0311'   09                          ADD     HL,BC
0312'   06 04                       LD      B,4
0314'                       ENV:
0314'   4E                          LD      C,(HL)
0315'   CD 0247'                    CALL    MANDA
0318'   23                          INC     HL
0319'   10 F9                       DJNZ    ENV

031B'   AF                          XOR     A
031C'   C9                          RET

031D'                       TES_1:
031D'   3A 32C2'                    LD      A,(Trava)
0320'   2F                          CPL
0321'   32 32C2'                    LD      (Trava),A

0324'   3E 00                       LD      A,0
0326'   C9                          RET

0327'                       TES_2:
0327'   3A 32FE'                    LD      A,(LOCAL)
032A'   2F                          CPL
032B'   32 32FE'                    LD      (LOCAL),A

032E'   3E 00                       LD      A,0
0330'   C9                          RET


0331'                       NORMAL:
0331'   1B 50 00 00                 ESC,'P',  0,  0
0335'   1B 51 00 00                 ESC,'Q',  0,  0
0339'   1B 52 00 00                 ESC,'R',  0,  0
033D'   1B 53 00 00                 ESC,'S',  0,  0
0341'   37 00 00 00                 '7',  0,  0,  0
0345'   38 00 00 00                 '8',  0,  0,  0
0349'   39 00 00 00                 '9',  0,  0,  0
034D'   2D 00 00 00                 '-',  0,  0,  0
0351'   34 00 00 00                 '4',  0,  0,  0
0355'   35 00 00 00                 '5',  0,  0,  0
0359'   36 00 00 00                 '6',  0,  0,  0
035D'   2C 00 00 00                 ',',  0,  0,  0
0361'   31 00 00 00                 '1',  0,  0,  0
0365'   32 00 00 00                 '2',  0,  0,  0
```

```
0369'   33 00 00 00                '3',  0,  0,  0
036D'   0D 00 00 00                 CR,  0,  0,  0
0371'   2E 00 00 00                '.',  0,  0,  0
0375'   30 00 00 00                '0',  0,  0,  0
0379'   00 00 00 00                  0,  0,  0,  0
037D'   00 00 00 00                  0,  0,  0,  0
0381'   00 00 00 00                  0,  0,  0,  0
0385'   00 00 00 00                  0,  0,  0,  0
0389'   00 00 00 00                  0,  0,  0,  0
038D'   00 00 00 00                  0,  0,  0,  0
0391'   00 00 00 00                  0,  0,  0,  0
0395'   00 00 00 00                  0,  0,  0,  0
0399'   1B 41 00 00              ESC,'A',  0,  0
039D'   1B 44 00 00              ESC,'D',  0,  0
03A1'   1B 42 00 00              ESC,'B',  0,  0
03A5'   1B 43 00 00              ESC,'C',  0,  0

03A9'                       ALTERNATE:
03A9'   1B 50 00 00              ESC,'P',  0,  0
03AD'   1B 51 00 00              ESC,'Q',  0,  0
03B1'   1B 52 00 00              ESC,'R',  0,  0
03B5'   1B 53 00 00              ESC,'S',  0,  0
03B9'   1B 3F 77 00              ESC,'?','w',  0
03BD'   1B 3F 78 00              ESC,'?','x',  0
03C1'   1B 3F 79 00              ESC,'?','y',  0
03C5'   1B 3F 6D 00              ESC,'?','m',  0
03C9'   1B 3F 74 00              ESC,'?','t',  0
03CD'   1B 3F 75 00              ESC,'?','u',  0
03D1'   1B 3F 76 00              ESC,'?','v',  0
03D5'   1B 3F 6C 00              ESC,'?','l',  0
03D9'   1B 3F 71 00              ESC,'?','q',  0
03DD'   1B 3F 72 00              ESC,'?','r',  0
03E1'   1B 3F 73 00              ESC,'?','s',  0
03E5'   1B 3F 4D 00              ESC,'?','M',  0
03E9'   1B 3F 6E 00              ESC,'?','n',  0
03ED'   1B 3F 70 00              ESC,'?','p',  0
03F1'   00 00 00 00                  0,  0,  0,  0
03F5'   00 00 00 00                  0,  0,  0,  0
03F9'   00 00 00 00                  0,  0,  0,  0
03FD'   00 00 00 00                  0,  0,  0,  0
0401'   00 00 00 00                  0,  0,  0,  0
0405'   00 00 00 00                  0,  0,  0,  0
0409'   00 00 00 00                  0,  0,  0,  0
040D'   00 00 00 00                  0,  0,  0,  0
0411'   1B 41 00 00              ESC,'A',  0,  0
0415'   1B 44 00 00              ESC,'D',  0,  0
```

```
0419'   1B 42 00 00                 ESC,'B',  0,  0
041D'   1B 43 00 00                 ESC,'C',  0,  0

0421'                       WS:
0421'   00 00 00 00                   0,  0,  0,  0
0425'   00 00 00 00                   0,  0,  0,  0
0429'   00 00 00 00                   0,  0,  0,  0
042D'   00 00 00 00                   0,  0,  0,  0
0431'   37 00 00 00                 '7',  0,  0,  0
0435'   38 00 00 00                 '8',  0,  0,  0
0439'   39 00 00 00                 '9',  0,  0,  0
043D'   2D 00 00 00                 '-',  0,  0,  0
0441'   34 00 00 00                 '4',  0,  0,  0
0445'   35 00 00 00                 '5',  0,  0,  0
0449'   36 00 00 00                 '6',  0,  0,  0
044D'   2C 00 00 00                 ',',  0,  0,  0
0451'   31 00 00 00                 '1',  0,  0,  0
0455'   32 00 00 00                 '2',  0,  0,  0
0459'   33 00 00 00                 '3',  0,  0,  0
045D'   0D 00 00 00                  CR,  0,  0,  0
0461'   2E 00 00 00                 '.',  0,  0,  0
0465'   30 00 00 00                 '0',  0,  0,  0
0469'   11 46 00 00                 11H,'F',  0,  0
046D'   0B 42 00 00                 0BH,'B',  0,  0
0471'   16 00 00 00                 16H,  0,  0,  0
0475'   0B 59 00 00                 0BH,'Y',  0,  0
0479'   0A 00 00 00                 0AH,  0,  0,  0
047D'   0B 4B 00 00                 0BH,'K',  0,  0
0481'   03 00 00 00                 03H,  0,  0,  0
0485'   12 00 00 00                 12H,  0,  0,  0
0489'   05 00 00 00                 05H,  0,  0,  0
048D'   13 00 00 00                 13H,  0,  0,  0
0491'   18 00 00 00                 18H,  0,  0,  0
0495'   04 00 00 00                 04H,  0,  0,  0

                                    PAGE
```

```
                        ;#################################################################
                        ;
                        ;          R O T I N A S   N I V E I S   III  E  IV
                        ;
                        ;#################################################################


                        ; Nome:        LIGA_PONTeiros
                        ;
                        ; Funcao:      Faz a ligacao entre a  tabela de ponteiros (armazenada
                        ;               em PONT_LIN) e as linhas de dados do terminal (armaze-
                        ;               nadas em TELA e TELA_1).
                        ; Entradas:    A  - Numero de ligacoes a fazer
                        ;               BC - Numero de bytes/linha
                        ;               HL - Endereco da matriz de linhas
                        ;               IX - Endereco da tabela de ponteiros
                        ; Saida:       Ponteiros a partir de IX
                        ; Registradores destruidos:     A,HL,IX
                        ; Referencias externas:         Nenhuma
                        ;
0499'                   LIGA_PONT:
0499'   DD 75 00                LD      (IX),L                  ; VAR i:INTEGER;
049C'   DD 74 01                LD      (IX+1),H                ;
049F'   36 00                   LD      (HL),0                  ; FOR i := A DOWNTO 1 DO
04A1'   23                      INC     HL                      ; BEGIN
04A2'   36 00                   LD      (HL),0                  ;   pont_lin[i] := HL;
04A4'   2B                      DEC     HL
04A5'   09                      ADD     HL,BC                   ;    [HL] := 0;
04A6'   DD 23                   INC     IX                      ;    HL := HL + compr_linha
04A8'   DD 23                   INC     IX                      ; END;
04AA'   3D                      DEC     A
04AB'   C2 0499'                JP      NZ,LIGA_PONT

04AE'   C9                      RET
```

```
                        ; Nome:       TRADUZ
                        ;
                        ; Funcao:     Gera um codigo ASCII correspondente 'a tecla detetada
                        ; Entradas:   HL: Inicio da tabela de traducao
                        ;             DE: Numero de linhas da matriz de teclado
                        ;             A: Numero da coluna da tecla detetada
                        ;             C: Numero da linha da tecla detetada
                        ;             Variaveis Cntrl:  0 se a tecla de controle foi detetada
                        ;                       Maiusc: 0 se a tecla de maiusculas foi detetada
                        ;                       Trava:  0 se a tecla de trava foi detetada
                        ;             Constante Tam_Tab: Tamanho da tabela de minusculas
                        ; Saida:      Reg. A: Codigo ASCII do caracter
                        ; Destroi:    A,B
                        ; Referencia: Ninguem
                        ;
04AF'                   TRADUZ:
04AF'   B7                      OR      A                       ; WHILE (Col_Det <> 0) DO
04B0'   CA 04B8'                JP      Z,TRAD_2

04B3'                   TRAD_1:                                 ;   BEGIN
04B3'   19                      ADD     HL,DE                   ;     Trad_Tecl := Trad_Tecl+Lin_
04B4'   3D                      DEC     A                       ;     Col_Det := Col_Det - 1
04B5'   C2 04B3'                JP      NZ,TRAD_1               ;   END;

04B8'                   TRAD_2:
04B8'   06 00                   LD      B,0                     ; Trad_Tecl := Trad_Tecl + Lin_De
04BA'   09                      ADD     HL,BC

04BB'                   TRAD_3:
04BB'   3A 32C4'                LD      A,(Cntrl)               ; IF (Cntrl <> 0)
04BE'   B7                      OR      A
04BF'   C2 04C8'                JP      NZ,TRAD_4

04C2'   11 0150                 LD      DE,Tam_Tab*3            ;  THEN DE := Tam_Tab*3
04C5'   C3 04DF'                JP      TRAD_6

04C8'                   TRAD_4:
04C8'   3A 32C3'                LD      A,(Maiusc)              ;  ELSE IF (Maiusc <> 0)
04CB'   B7                      OR      A
04CC'   C2 04D5'                JP      NZ,TRAD_5

04CF'   11 00E0                 LD      DE,Tam_Tab*2            ;        THEN DE := Tam_Tab*2
04D2'   C3 04DF'                JP      TRAD_6

04D5'                   TRAD_5:
```

```
04D5'   3A 32C2'                LD      A,(Trava)               ;       ELSE IF (Trava <> 0)
04D8'   B7                      OR      A
04D9'   C2 04E0'                JP      NZ,TRAD_7

04DC'   11 0070                 LD      DE,Tam_Tab              ;               THEN DE := Tam_Ta

04DF'                   TRAD_6:
04DF'   19                      ADD     HL,DE                   ; Calcula valor do ponteiro
04E0'                   TRAD_7:
04E0'   7E                      LD      A,(HL)                  ; Retorno: Pega caraceter na tabe

04E1'   C9                      RET


                        ; Nome:     INTERP
                        ;
                        ; Funcao:   Interpreta o caracter recebido, insere se for letra, trata se
                        ;           de controle.
                        ;
                        ; Entradas: A: Codigo ASCII do caracter recebido.
                        ; Saida:    A: ASCII do caracter recebido - deve ser inserido na tela
                        ;              0 - nao deve ser inserido na tela.
                        ; Destroi:  A, B
                        ; Referencia: Ninguem
                        ;

04E2'                   INTERP:
04E2'   F5                      PUSH    AF
04E3'   3A 32C6'                LD      A,(PAR_CURS)
04E6'   FE FF                   CP      0FFH
04E8'   CA 0694'                JP      Z,LE_PAR
04EB'   3A 32C5'                LD      A,(INDESC)
04EE'   B7                      OR      A
04EF'   C2 055C'                JP      NZ,DETESC
04F2'   F1                      POP     AF

04F3'   01 0007                 LD      BC,7
04F6'   21 0509'                LD      HL,CAR_ESP
04F9'   ED B1                   CPIR
04FB'   C2 051E'                JP      NZ,NAO_ESP
04FE'   21 0510'                LD      HL,END_ESP
0501'   CB 21                   SLA     C
0503'   09                      ADD     HL,BC
0504'   5E                      LD      E,(HL)
```

```
0505'   23                          INC     HL
0506'   56                          LD      D,(HL)
0507'   EB                          EX      DE,HL
0508'   E9                          JP      (HL)

0509'                       CAR_ESP:
0509'   0D                          DB      CR
050A'   0A                          DB      LF
050B'   08                          DB      BS
050C'   09                          DB      TAB
050D'   7F                          DB      DEL
050E'   1B                          DB      ESC
050F'   0C                          DB      FF

0510'                       END_ESP:
0510'   0546'                       DW      FORMFEED
0512'   053F'                       DW      ESCAPE
0514'   0521'                       DW      IGNORA
0516'   0523'                       DW      TABULAR
0518'   05FC'                       DW      ESQUERDA
051A'   06EC'                       DW      DESCLIN
051C'   06DC'                       DW      INILIN

                            ;
                            ; Verifica se o caracter pode ser escrito
                            ;
051E'                       NAO_ESP:
051E'   FE 20                       CP      20H             ; Ve se e imprimivel
0520'   D0                          RET     NC              ; Volta avisando que pode ser escrito.

0521'                       Ignora:
0521'   AF                          XOR     A               ; Volta avisando que nao pode ser escrito
0522'   C9                          RET

0523'                       Tabular:
0523'   3A 3135'                    LD      A,(Col_Cor)
0526'   C6 08                       ADD     A,8
0528'   E6 F8                       AND     11111000B
052A'   FE 50                       CP      80
052C'   DA 0531'                    JP      C,Tab_1
052F'   3E 4F                       LD      A,79
0531'                       Tab_1:
0531'   32 3135'                    LD      (Col_Cor),A
0534'   CD 07CB'                    CALL    Carr_Curs
0537'   CD 07E6'                    CALL    PosicHL
053A'   22 3132'                    LD      (P_Ins),HL
```

```
053D'   AF                          XOR     A
053E'   C9                          RET

                            ;
                            ; Avisa a detecao de um ESC
                            ;
053F'                       Escape:

053F'   3E FF                       LD      A,0FFh
0541'   32 32C5'                    LD      (IndEsc),A

0544'   AF                          XOR     A               ; Volta avisando que nao pode ser escrito
0545'   C9                          RET
                            ;
                            ; Faz <FF> limpar a tela
                            ;
0546'                       FORMFEED:
0546'   3E 00                       LD      A,0
0548'   32 3134'                    LD      (LIN_COR),A
054B'   32 3135'                    LD      (COL_COR),A
054E'   CD 07CB'                    CALL    CARR_CURS
0551'   CD 07E6'                    CALL    POSICHL
0554'   22 3132'                    LD      (P_INS),HL
0557'   CD 064F'                    CALL    LMPRESTO
055A'   AF                          XOR     A
055B'   C9                          RET

                            ;
                            ; Decodifica uma sequencia de caracteres de controle
                            ;
055C'                       DetEsc:
055C'   3E 00                       LD      A,0             ; Apaga indicador de detecao do ESC
055E'   32 32C5'                    LD      (IndEsc),A

0561'   F1                          POP     AF              ; Restaura valor lido

0562'   01 000D                     LD      BC,13
0565'   21 0578'                    LD      HL,ESC_SEQ
0568'   ED B1                       CPIR
056A'   C2 059F'                    JP      NZ,NAO_SEQ
056D'   CB 21                       SLA     C
056F'   21 0585'                    LD      HL,ESC_END
0572'   09                          ADD     HL,BC
0573'   5E                          LD      E,(HL)
0574'   23                          INC     HL
0575'   56                          LD      D,(HL)
```

```
0576'  EB                          EX      DE,HL'
0577'  E9                          JP      (HL)

0578'                       ESC_SEQ:
0578'  3D                          DB      '='
0579'  3E                          DB      '>'
057A'  41                          DB      'A'
057B'  42                          DB      'B'
057C'  43                          DB      'C'
057D'  44                          DB      'D'
057E'  48                          DB      'H'
057F'  49                          DB      'I'
0580'  4A                          DB      'J'
0581'  4B                          DB      'K'
0582'  57                          DB      'W'
0583'  59                          DB      'Y'
0584'  5A                          DB      'Z'
0585'                       ESC_END:
0585'  077E'                       DW      ID_TERM
0587'  068A'                       DW      POSIC
0589'  05B1'                       DW      ENTWS
058B'  0663'                       DW      LMPLINHA
058D'  064F'                       DW      LMPRESTO
058F'  0624'                       DW      LININV
0591'  0611'                       DW      ORIGEM
0593'  05FC'                       DW      ESQUERDA
0595'  05E5'                       DW      DIREITA
0597'  05CE'                       DW      DESCE
0599'  05B9'                       DW      SOBE
059B'  05A9'                       DW      SAIALTER
059D'  05A1'                       DW      ENTALTER

059F'                       NAO_SEQ:
059F'  AF                          XOR     A               ; Caso nao seja nenhum desses, ignora
05A0'  C9                          RET


05A1'                       ENTALTER:
05A1'  21 03A9'                    LD      HL,ALTERNATE
05A4'  22 32FF'                    LD      (TECALT),HL
05A7'  AF                          XOR     A
05A8'  C9                          RET
05A9'                       SAIALTER:
05A9'  21 0331'                    LD      HL,NORMAL
05AC'  22 32FF'                    LD      (TECALT),HL
```

```
05AF'   AF                          XOR     A
05B0'   C9                          RET
05B1'                       ENTWS:
05B1'   21 0421'                    LD      HL,WS
05B4'   22 32FF'                    LD      (TECALT),HL
05B7'   AF                          XOR     A
05B8'   C9                          RET

                            ;
                            ; Sobe o cursor para a linha de cima, se possivel
                            ;
05B9'                       Sobe:
05B9'   3A 3134'                    LD      A,(Lin_Cor)     ; Se na primeira linha, ignora comando.
05BC'   FE 00                       CP      0
05BE'   C8                          RET     Z

05BF'   3D                          DEC     A               ; Senao, passa `a linha superior
05C0'   32 3134'                    LD      (Lin_Cor),A
05C3'   CD 07CB'                    CALL    Carr_Curs       ; Carrega no 8275 nova posicao do cursor.
05C6'   CD 07E6'                    CALL    PosicHL         ; Calcula nova posicao do P_Ins.
05C9'   22 3132'                    LD      (P_Ins),HL
05CC'   AF                          XOR     A               ; Este caracter nao vai para a tela
05CD'   C9                          RET


                            ;
                            ; Desce o cursor para  a linha de baixo, se possivel
                            ;
05CE'                       Desce:
05CE'   3A 3134'                    LD      A,(Lin_Cor)     ; Se na 'ultima linha, ignora comando.
05D1'   FE 17                       CP      Num_Lin-1
05D3'   CA 05E3'                    JP      Z,Desc_1

05D6'   3C                          INC     A               ; Senao, passa `a linha inferior
05D7'   32 3134'                    LD      (Lin_Cor),A
05DA'   CD 07CB'                    CALL    Carr_Curs       ; Carrega no 8275 nova posicao do cursor.
05DD'   CD 07E6'                    CALL    PosicHL         ; Calcula nova posicao do P_Ins.
05E0'   22 3132'                    LD      (P_Ins),HL
05E3'                       Desc_1:
05E3'   AF                          XOR     A
05E4'   C9                          RET
```

```
                        ;
                        ; Avanca o cursor uma posicao `a direita, se possivel
                        ;
05E5'                   Direita:
05E5'   3A 3135'                LD      A,(Col_Cor)     ; Se na 'ultima linha, ignora comando.
05E8'   FE 4F                   CP      Num_Col-1
05EA'   CA 05FA'                JP      Z,Dir_1

05ED'   3C                      INC     A               ; Senao, passa ao caracter `a direita
05EE'   32 3135'                LD      (Col_Cor),A
05F1'   CD 07CB'                CALL    Carr_Curs       ; Carrega no 8275 nova posicao do cursor.
05F4'   CD 07E6'                CALL    PosicHL         ; Calcula nova posicao do P_Ins.
05F7'   22 3132'                LD      (P_Ins),HL
05FA'                   Dir_1:
05FA'   AF                      XOR     A
05FB'   C9                      RET


                        ;
                        ; Avanca o cursor uma posicao `a esquerda, se possivel
                        ;
05FC'                   Esquerda:
05FC'   3A 3135'                LD      A,(Col_Cor)     ; Se na primeira linha, ignora comando.
05FF'   FE 00                   CP      0
0601'   C8                      RET     Z

0602'   3D                      DEC     A               ; Senao, passa ao caracter `a esquerda
0603'   32 3135'                LD      (Col_Cor),A
0606'   CD 07CB'                CALL    Carr_Curs       ; Carrega no 8275 nova posicao do cursor.
0609'   CD 07E6'                CALL    PosicHL
060C'   22 3132'                LD      (P_Ins),HL
060F'   AF                      XOR     A               ; Nao deve ser inserido
0610'   C9                      RET


                        ;
                        ; Traz o cursor para a posicao 1,1
                        ;
0611'                   Origem:
0611'   3E 00                   LD      A,0
0613'   32 3134'                LD      (Lin_Cor),A
0616'   32 3135'                LD      (Col_Cor),A
0619'   CD 07CB'                CALL    Carr_Curs
```

```
061C'  CD 07E6'          CALL   PosicHL
061F'  22 3132'          LD     (P_Ins),HL
0622'  AF                XOR    A
0623'  C9                RET


                    ;
                    ; Faz um avanco de tela invertido
                    ;
0624'               LinInv:
0624'  3A 3134'          LD     A,(Lin_Cor)
0627'  FE 00             CP     0
0629'  CA 063B'          JP     Z,Roda_Baixo
062C'  3D                DEC    A
062D'  32 3134'          LD     (Lin_Cor),A
0630'  CD 07CB'          CALL   Carr_Curs
0633'  CD 07E6'          CALL   PosicHL
0636'  22 3132'          LD     (P_Ins),HL
0639'  AF                XOR    A
063A'  C9                RET

063B'               Roda_Baixo:
063B'  3A 32CC'          LD     A,(Fim_Roda)
063E'  CD 076E'          CALL   Apaga
0641'  11 FFFE           LD     DE,0FFFEH      ; Parametro para rotina de scroll que ind
0644'  CD 0717'          CALL   Roda           ; direcao. Roda para baixo
0647'  CD 07E6'          CALL   PosicHL        ; Posiciona cursor na nova linha
064A'  22 3132'          LD     (P_Ins),HL
064D'  AF                XOR    A
064E'  C9                RET


                    ;
                    ; Limpa da posicao do cursor (inclusive) ate' o fim da tela.
                    ;
064F'               LmpResto:
064F'  CD 0663'          CALL   LMPLINHA
0652'  3A 3134'          LD     A,(LIN_COR)
0655'               LMPR1:
0655'  3C                INC    A
0656'  FE 18             CP     NUM_LIN
0658'  CA 0661'          JP     Z,LMPR2
065B'  CD 076E'          CALL   APAGA
065E'  C3 0655'          JP     LMPR1
```

```
0661'                       LMPR2:
0661'   AF                          XOR     A
0662'   C9                          RET


                            ;
                            ; Limpa da posicao do cursor (inclusive) ate' o fim da linha.
                            ;
0663'                       LmpLinha:
0663'   3A 3134'                    LD      A,(LIN_COR)
0666'   CD 0809'                    CALL    POSIC_LIN
0669'   EB                          EX      DE,HL
066A'   2A 3132'                    LD      HL,(P_INS)
066D'   36 F1                       LD      (HL),0F1H
066F'   23                          INC     HL
0670'   36 F1                       LD      (HL),0F1H
0672'   2B                          DEC     HL
0673'   B7                          OR      A
0674'   ED 52                       SBC     HL,DE
0676'   3A 3135'                    LD      A,(COL_COR)
0679'   4F                          LD      C,A
067A'   06 00                       LD      B,0
067C'   ED 42                       SBC     HL,BC
067E'   EB                          EX      DE,HL
067F'   2B                          DEC     HL
0680'   2B                          DEC     HL
0681'   73                          LD      (HL),E
0682'   3A 3135'                    LD      A,(COL_COR)
0685'   83                          ADD     A,E
0686'   23                          INC     HL
0687'   77                          LD      (HL),A

0688'   AF                          XOR     A
0689'   C9                          RET


                            ;
                            ; Posiciona o cursor em qualquer lugar da tela
                            ;
068A'                       Posic:
068A'   3E FF                       LD      A,0FFh
068C'   32 32C6'                    LD      (Par_Curs),A
068F'   AF                          XOR     A
0690'   32 32C7'                    LD      (Num_Par),A
0693'   C9                          RET
```

```
0694'                        Le_Par:
0694'   3A 32C7'                     LD      A,(Num_Par)
0697'   FE 00                        CP      0
0699'   C2 06A6'                     JP      NZ,Pos_1
069C'   F1                           POP     AF
069D'   32 32C8'                     LD      (Guarda_Par),A
06A0'   21 32C7'                     LD      HL,Num_Par
06A3'   34                           INC     (HL)
06A4'   AF                           XOR     A
06A5'   C9                           RET
06A6'                        Pos_1:
06A6'   F1                           POP     AF
06A7'   D6 20                        SUB     20H
06A9'   FE 50                        CP      NUM_COL
06AB'   D4 06CC'                     CALL    NC,ACERCOL
06AE'   32 3135'                     LD      (Col_Cor),A
06B1'   3A 32C8'                     LD      A,(Guarda_Par)
06B4'   D6 20                        SUB     20H
06B6'   FE 18                        CP      NUM_LIN
06B8'   D4 06D4'                     CALL    NC,ACERLIN
06BB'   32 3134'                     LD      (Lin_Cor),A
06BE'   CD 07E6'                     CALL    PosicHL
06C1'   22 3132'                     LD      (P_Ins),HL
06C4'   CD 07CB'                     CALL    Carr_Curs
06C7'   AF                           XOR     A
06C8'   32 32C6'                     LD      (Par_Curs),A
06CB'   C9                           RET

06CC'                        ACERCOL:
06CC'   D6 50                        SUB     NUM_COL
06CE'   FE 50                        CP      NUM_COL
06D0'   D2 06CC'                     JP      NC,ACERCOL
06D3'   C9                           RET

06D4'                        ACERLIN:
06D4'   D6 18                        SUB     NUM_LIN
06D6'   FE 18                        CP      NUM_LIN
06D8'   D2 06D4'                     JP      NC,ACERLIN
06DB'   C9                           RET

                             ;
                             ; Volta `a posicao 1 dessa linha (retorno de carro) .
                             ;
06DC'                        IniLin:
06DC'   3E 00                        LD      A,0
```

```
06DE'   32 3135'                LD      (Col_Cor),A
06E1'   CD 07CB'                CALL    Carr_Curs
06E4'   CD 07E6'                CALL    PosicHL
06E7'   22 3132'                LD      (P_Ins),HL
06EA'   AF                      XOR     A               ; Nao deve ser inserido
06EB'   C9                      RET


                        ;
                        ; Desce para a linha de baixo, nessa mesma posicao. Se na ultima linha,
                        ; faz a tela toda subir (avanco de formulario).
                        ;
06EC'                   DescLin:
06EC'   3A 3134'                LD      A,(Lin_Cor)
06EF'   3C                      INC     A
06F0'   FE 18                   CP      Num_Lin
06F2'   CA 0703'                JP      Z,Roda_Cima
06F5'   32 3134'                LD      (Lin_Cor),A
06F8'   CD 07E6'                CALL    PosicHL
06FB'   22 3132'                LD      (P_Ins),HL
06FE'   CD 07CB'                CALL    Carr_Curs
0701'   AF                      XOR     A
0702'   C9                      RET

0703'                   Roda_Cima:
0703'   3A 32CB'                LD      A,(Ini_Roda)
0706'   CD 076E'                CALL    Apaga
0709'   11 0002                 LD      DE,2
070C'   CD 0717'                CALL    Roda
070F'   CD 07E6'                CALL    PosicHL
0712'   22 3132'                LD      (P_Ins),HL
0715'   AF                      XOR     A
0716'   C9                      RET


0717'                   Roda:
0717'   3A 32CB'                LD      A,(Ini_Roda)
071A'   87                      ADD     A,A
071B'   47                      LD      B,A
071C'   3A 32CC'                LD      A,(Fim_Roda)
071F'   87                      ADD     A,A
0720'   4F                      LD      C,A
0721'   7B                      LD      A,E
```

```
0722'  B7                          OR      A
0723'  F2 0729'                    JP      P,Nao_Troca
0726'  78                          LD      A,B
0727'  41                          LD      B,C
0728'  4F                          LD      C,A
0729'                      Nao_Troca:
0729'  DD 21 32CE'                 LD      IX,Pont_Lin
072D'  FD 21 32CE'                 LD      IY,Pont_LIn
0731'  EB                          EX      DE,HL
0732'  58                          LD      E,B
0733'  16 00                       LD      D,0
0735'  DD 19                       ADD     IX,DE
0737'  FD 19                       ADD     IY,DE
0739'  EB                          EX      DE,HL
073A'  FD 19                       ADD     IY,DE
073C'  DD 6E 00                    LD      L,(IX)
073F'  DD 66 01                    LD      H,(IX+1)
0742'  22 32C9'                    LD      (Pont_Salvo),HL
0745'  78                          LD      A,B
0746'                      RD_1:
0746'  B9                          CP      C
0747'  CA 0762'                    JP      Z,RD_2
074A'  F3                          DI
074B'  FD 7E 00                    LD      A,(IY)
074E'  DD 77 00                    LD      (IX),A
0751'  FD 7E 01                    LD      A,(IY+1)
0754'  DD 77 01                    LD      (IX+1),A
0757'  FB                          EI
0758'  DD 19                       ADD     IX,DE
075A'  FD 19                       ADD     IY,DE
075C'  78                          LD      A,B
075D'  83                          ADD     A,E
075E'  47                          LD      B,A
075F'  C3 0746'                    JP      RD_1
0762'                      RD_2:
0762'  2A 32C9'                    LD      HL,(Pont_Salvo)
0765'  F3                          DI
0766'  DD 75 00                    LD      (IX),L
0769'  DD 74 01                    LD      (IX+1),H
076C'  FB                          EI

076D'  C9                          RET


076E'                      Apaga:
076E'  CD 0809'                    CALL    POSIC_LIN
```

```
0771'   36 F1                   LD      (HL),0F1H
0773'   23                      INC     HL
0774'   36 F1                   LD      (HL),0F1H
0776'   2B                      DEC     HL
0777'   2B                      DEC     HL
0778'   36 00                   LD      (HL),0
077A'   2B                      DEC     HL
077B'   36 00                   LD      (HL),0

077D'   C9                      RET


077E'                   Id_Term:
077E'   AF                      XOR     A
077F'   C9                      RET


                        ; Nome:         INS_TELA
                        ;
                        ; Funcao:       Inserir dados lidos na estrutura de dados da tela
                        ;
                        ; Entrada:      Dado no reg. A.
                        ; Saida:        Memoria de tela, P_INS, COL_COR
                        ; Registradores destruidos:     A,B
                        ; Referencias externas:         Nenhuma
                        ;
0780'                   INS_TELA:
0780'   F5                      PUSH    AF
0781'   2A 3132'                LD      HL,(P_Ins)
0784'   3E F1                   LD      A,0F1h
0786'   BE                      CP      (HL)
0787'   C2 078E'                JP      NZ,INS1
078A'   23                      INC     HL
078B'   36 F1                   LD      (HL),0F1h
078D'   2B                      DEC     HL
078E'                   INS1:
078E'   F1                      POP     AF
078F'   77                      LD      (HL),A
0790'   23                      INC     HL
0791'   22 3132'                LD      (P_Ins),HL
0794'   21 3135'                LD      HL,Col_Cor
0797'   34                      INC     (HL)
0798'   7E                      LD      A,(HL)
0799'   FE 50                   CP      80
079B'   CC 07A2'                CALL    Z,NovaLin
079E'   CD 07CB'                CALL    CARR_CURS
```

```
07A1'   C9                          RET

07A2'                       NovaLin:
07A2'   3A 3134'                    LD      A,(LIN_COR)
07A5'   3C                          INC     A
07A6'   32 3134'                    LD      (LIN_COR),A
07A9'   FE 18                       CP      NUM_LIN
07AB'   CC 07BA'                    CALL    Z,SCROLL
07AE'   3E 00                       LD      A,0
07B0'   32 3135'                    LD      (COL_COR),A
07B3'   CD 07E6'                    CALL    POSICHL
07B6'   22 3132'                    LD      (P_INS),HL
07B9'   C9                          RET

07BA'                       SCROLL:
07BA'   3A 32CB'                    LD      A,(Ini_Roda)
07BD'   CD 076E'                    CALL    Apaga
07C0'   11 0002                     LD      DE,2
07C3'   CD 0717'                    CALL    RODA
07C6'   21 3134'                    LD      HL,Lin_Cor
07C9'   35                          DEC     (HL)
07CA'   C9                          RET


                            ; Nome:       Carr_Curs
                            ;
                            ; Funcao:     Carrega no 8275 a posicao do cursor.
                            ; Entradas:   Nenhuma
                            ; Saida:      Segura o processamento ate' porto estar OK
                            ; Emprega:   Estado do porto serial
                            ; Destroi:    A
                            ; Referencia: Ninguem
                            ;
                            ;
07CB'                       Carr_Curs:
07CB'   3E 80                       LD      A,Cmd_Curs      ; Envia comando de carregamento
07CD'   D3 01                       OUT     (C_8275),A
07CF'   3A 3135'                    LD      A,(Col_Cor)     ; Envia posicao X
07D2'   D3 00                       OUT     (P_8275),A
07D4'   3A 3134'                    LD      A,(Lin_Cor)     ; Envia posicao Y
07D7'   D3 00                       OUT     (P_8275),A
07D9'   C9                          RET
```

```
                        ; Nome:       TX_OK
                        ;
                        ; Funcao:     Espera porto serial terminar de enviar caracter
                        ; Entradas:   Nenhuma
                        ; Saida:      Segura o processamento ate' porto estar OK
                        ; Emprega:    Estado do porto serial
                        ; Destroi:    A
                        ; Referencia: Ninguem
                        ;
07DA'                   TX_OK:
07DA'   3E 00                   LD      A,0
07DC'   D3 21                   OUT     (E_SIO),A
07DE'   DB 21                   IN      A,(E_SIO)
07E0'   E6 04                   AND     04H
07E2'   CA 07DA'                JP      Z,TX_OK
07E5'   C9                      RET


                        ; Nome:       PosicHL
                        ;
                        ; Funcao:     Calcula a posicao do inicio da linha corrente na mem'oria.
                        ; Entradas:   Lin_Cor - Numero da linha corrente.
                        ;             Pont_Lin - Inicio da tabela de ponteiros de linhas.
                        ; Saida:      HL - Endereco do primeiro byte de dados dessa linha.
                        ; Destroi:    A, BC, IX, HL
                        ; Referencia: Ninguem
                        ;
07E6'                   PosicHL:
07E6'   3A 3134'                LD      A,(LIN_COR)
07E9'   CD 0809'                CALL    POSIC_LIN
07EC'   3A 3135'                LD      A,(COL_COR)
07EF'   4F                      LD      C,A
07F0'   FE 00                   CP      0
07F2'   C8                      RET     Z
07F3'   3E F1                   LD      A,0F1H
07F5'                   POS1:
07F5'   BE                      CP      (HL)
07F6'   28 05                   JR      Z,POS2
07F8'   23                      INC     HL
```


BIBLIOTECA DO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS - USP
FÍSICA

```
07F9'   0D                          DEC     C
07FA'   20 F9                       JR      NZ,POS1

07FC'   C9                          RET
07FD'                       POS2:
07FD'   0C                          INC     C
07FE'                       POS4:
07FE'   0D                          DEC     C
07FF'   28 05                       JR      Z,POS3
0801'   36 20                       LD      (HL),' '
0803'   23                          INC     HL
0804'   18 F8                       JR      POS4
0806'                       POS3:
0806'   36 F1                       LD      (HL),0F1H

0808'   C9                          RET



0809'                       POSIC_LIN:
0809'   4F                          LD      C,A
080A'   CB 21                       SLA     C
080C'   06 00                       LD      B,0
080E'   DD 21 32CE'                 LD      IX,PONT_LIN
0812'   DD 09                       ADD     IX,BC
0814'   DD 6E 00                    LD      L,(IX)
0817'   DD 66 01                    LD      H,(IX+1)
081A'   23                          INC     HL
081B'   23                          INC     HL
081C'   C9                          RET


                            ;#####################################################################
                            ;#                                                                   #
                            ;#                          Tabelas Fixas                            #
                            ;#                                                                   #
                            ;#####################################################################


                            ;
                            ; Tabela de minusculas
                            ;
081D'                       Tab_Tecl:
081D'   32 09 31 3C                 DB      '2',    TAB,    '1',    '<',    '8',    'y',    'h',    '
0821'   38 79 68 6E
```

```
0825'  F0 00 20 00          DB    TRV,  NIL,  ' ',  NIL,  '9',  'u',  'j',
0829'  39 75 6A 6D
082D'  37 74 67 62          DB    '7',  't',  'g',  'b',  '0',  'i',  'k',
0831'  30 69 6B 2C
0835'  36 72 66 76          DB    '6',  'r',  'f',  'v',  '-',  'o',  'l',
0839'  2D 6F 6C 2E
083D'  35 65 64 63          DB    '5',  'e',  'd',  'c',  '=',  'p',  ';',
0841'  3D 70 3B 5C
0845'  34 77 73 78          DB    '4',  'w',  's',  'x',  60H,  '[',  27H,
0849'  60 5B 27 00
084D'  33 71 61 7A          DB    '3',  'q',  'a',  'z',  DEL,  ']',  '/',
0851'  7F 5D 2F 0D
0855'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,
0859'  00 00 00 00
085D'  92 93 94 80          DB    92H,  93H,  94H,  80H,  81H,  82H,  83H,
0861'  81 82 83 00
0865'  95 98 99 84          DB    95H,  98H,  99H,  84H,  85H,  86H,  87H,
0869'  85 86 87 00
086D'  91 9A 90 88          DB    91H,  9AH,  90H,  88H,  89H,  8AH,  8BH,
0871'  89 8A 8B 00
0875'  9B 9C 9D 8C          DB    9BH,  9CH,  9DH,  8CH,  8DH,  8EH,  8FH,
0879'  8D 8E 8F 00
087D'  00 00 00 97          DB    NIL,  NIL,  NIL,  97H,  96H,  NIL,  NIL,
0881'  96 00 00 00
0885'  1B F1 00 00          DB    ESC,  SLC,  NIL,  NIL,  NIL,  NIL,  NIL,
0889'  00 00 00 00

0070                  Tam_Tab EQU   $-Tab_Tecl

                      ;
                      ; Tabela de TRAVA
                      ;
088D'  32 09 31 3C          DB    '2',  TAB,  '1',  '<',  '8',  'Y',  'H',
0891'  38 59 48 4E
0895'  F0 00 20 00          DB    TRV,  NIL,  ' ',  NIL,  '9',  'U',  'J',
0899'  39 55 4A 4D
089D'  37 54 47 42          DB    '7',  'T',  'G',  'B',  '0',  'I',  'K',
08A1'  30 49 4B 2C
08A5'  36 52 46 56          DB    '6',  'R',  'F',  'V',  '-',  'O',  'L',
08A9'  2D 4F 4C 2E
08AD'  35 45 44 43          DB    '5',  'E',  'D',  'C',  '=',  'P',  ';',
08B1'  3D 50 3B 5C
08B5'  34 57 53 58          DB    '4',  'W',  'S',  'X',  60H,  '[',  27H,
08B9'  60 5B 27 00
08BD'  33 51 41 5A          DB    '3',  'Q',  'A',  'Z',  DEL,  ']',  '/',
08C1'  7F 5D 2F 0D
```

```
08C5'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
08C9'  00 00 00 00
08CD'  92 93 94 80          DB    92H,  93H,  94H,  80H,  81H,  82H,  83H,  N
08D1'  81 82 83 00
08D5'  95 98 99 84          DB    95H,  98H,  99H,  84H,  85H,  86H,  87H,  N
08D9'  85 86 87 00
08DD'  91 9A 90 88          DB    91H,  9AH,  90H,  88H,  89H,  8AH,  8BH,  N
08E1'  89 8A 8B 00
08E5'  9B 9C 9D 8C          DB    9BH,  9CH,  9DH,  8CH,  8DH,  8EH,  8FH,  N
08E9'  8D 8E 8F 00
08ED'  00 00 00 97          DB    NIL,  NIL,  NIL,  97H,  96H,  NIL,  NIL,  N
08F1'  96 00 00 00
08F5'  1B F1 00 00          DB    ESC,  SLC,  NIL,  NIL,  NIL,  NIL,  NIL,  N
08F9'  00 00 00 00
                      ;
                      ; Tabela de MAIUSCULAS
                      ;
08FD'  40 09 21 3E          DB    '@',  TAB,  '!',  '>',  '*',  'Y',  'H',  '
0901'  2A 59 48 4E
0905'  F0 00 20 00          DB    TRV,  NIL,  ' ',  NIL,  '(',  'U',  'J',  '
0909'  28 55 4A 4D
090D'  26 54 47 42          DB    '&',  'T',  'G',  'B',  ')',  'I',  'K',  '
0911'  29 49 4B 2C
0915'  5E 52 46 56          DB    '^',  'R',  'F',  'V',  '_',  'O',  'L',  '
0919'  5F 4F 4C 2E
091D'  25 45 44 43          DB    '%',  'E',  'D',  'C',  '+',  'P',  ':',  '
0921'  2B 50 3A 7C
0925'  24 57 53 58          DB    '$',  'W',  'S',  'X',  '~',  '{',  '"',  N
0929'  7E 7B 22 00
092D'  23 51 41 5A          DB    '#',  'Q',  'A',  'Z',   BS,  '}',  '?',  L
0931'  08 7D 3F 0A
0935'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
0939'  00 00 00 00
093D'  92 93 94 80          DB    92H,  93H,  94H,  80H,  81H,  82H,  83H,  N
0941'  81 82 83 00
0945'  95 98 99 84          DB    95H,  98H,  99H,  84H,  85H,  86H,  87H,  N
0949'  85 86 87 00
094D'  91 9A 90 88          DB    91H,  9AH,  90H,  88H,  89H,  8AH,  8BH,  N
0951'  89 8A 8B 00
0955'  9B 9C 9D 8C          DB    9BH,  9CH,  9DH,  8CH,  8DH,  8EH,  8FH,  N
0959'  8D 8E 8F 00
095D'  00 00 00 97          DB    NIL,  NIL,  NIL,  97H,  96H,  NIL,  NIL,  N
0961'  96 00 00 00
0965'  1B F1 00 00          DB    ESC,  SLC,  NIL,  NIL,  NIL,  NIL,  NIL,  N
0969'  00 00 00 00
                      ;
```

```
                        ; Tabela de CONTROLE
                        ;
0969'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  19H,  08H,  0
0971'  00 19 08 0E
0975'  F0 00 00 00          DB    TRV,  NIL,  NIL,  NIL,  NIL,  15H,  0AH,  0
0979'  00 15 0A 0D
097D'  00 14 07 02          DB    NIL,  14H,  07H,  02H,  NIL,  09H,  0BH,  N
0981'  00 09 0B 00
0985'  00 12 06 16          DB    NIL,  12H,  06H,  16H,  NIL,  0FH,  0CH,  N
0989'  00 0F 0C 00
098D'  00 05 04 03          DB    NIL,  05H,  04H,  03H,  NIL,  10H,  NIL,  N
0991'  00 10 00 00
0995'  00 17 13 18          DB    NIL,  17H,  13H,  18H,  NIL,  NIL,  NIL,  N
0999'  00 00 00 00
099D'  00 11 01 1A          DB    NIL,  11H,  01H,  1AH,  NIL,  NIL,  NIL,  N
09A1'  00 00 00 00
09A5'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09A9'  00 00 00 00
09AD'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09B1'  00 00 00 00
09B5'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09B9'  00 00 00 00
09BD'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09C1'  00 00 00 00
09C5'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09C9'  00 00 00 00
09CD'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09D1'  00 00 00 00
09D5'  00 00 00 00          DB    NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  NIL,  N
09D9'  00 00 00 00


                        ;#################################################
                        ;#                                               #
                        ;#                Area de Dados                  #
                        ;#                                               #
                        ;#################################################

                                ORG     3000H   ; Primeira RAM
```

```
3000'                   Pilha:          DS      300
312C'                   INI_PILHA       EQU     $
312C'                   Cnt_Tecl:       DS      1       ; Contador de interrupcoes
312D'                   Trat:           DS      1       ; Indicador de byte tratado.0..
312E'                   Det:            DS      1       ; Indicador de byte detetado
312F'                   BSalvo:         DS      1       ; Salvadores de registradores
3130'                   CSalvo:         DS      1
3131'                   DSalvo:         DS      1
3132'                   P_INS:          DS      2       ; Ponteiro de insercao na memoria
3134'                   LIN_COR:        DS      1       ; Linha corrente
3135'                   COL_COR:        DS      1       ; Coluna corrente
3136'                   PILHA_SALVA:    DS      2       ; Endereco da pilha
3138'                   TELA_1:         DS      TAM_LIN*(NUM_LIN-LIMITE)
32C0'                   Lin_Det:        DS      1       ; Armazena a linha da tecla detectada
32C1'                   Col_Det:        DS      1       ; Armazena a coluna da tecla detectada
32C2'                   Trava:          DS      1       ; Armazena 0 se detetou a tecla de trava
32C3'                   Maiusc:         DS      1       ; Armazena 0 se detetou a tecla de maiusc
32C4'                   Cntrl:          DS      1       ; Armazena 0 se detetou a tecla de contro
32C5'                   IndEsc:         DS      1       ; FF se a letra detectada anteriormemte f
32C6'                   Par_Curs:       DS      1       ; Flag que indica que a proxima entrada
                                                        ; e' parametro para posicionamento de cur
32C7'                   Num_Par:        DS      1       ; Numero de parametros
32C8'                   Guarda_Par:     DS      1       ; Armazena parametro inicial do posiciona
32C9'                   Pont_Salvo:     DS      2       ; Salva ponteiro de insercao

32CB'                   Ini_Roda:       DS      1       ; Linha de inicio da regiao de Scroll
32CC'                   Fim_Roda:       DS      1       ; Linha de fim da regiao de Scroll
32CD'                   LIN_MOSTRAR:    DS      1       ; Linha a ser mostrada
32CE'                   PONT_LIN:       DS      NUM_LIN*2
32FE'                   LOCAL:          DS      1
32FF'                   TECALT:         DS      2
3301'                   CONTREP:        DS      1

                                        ORG     4000H   ; Segunda RAM

4000'                   TELA:           DS      TAM_LIN*LIMITE
47A8'                   F               EQU     $       ; So' para aparecer na listagem

                                END     PPRINC
```

Macros:

Symbols:

ACERCO	06CC'	ACERLI	06D4'	ALTERN	03A9'	APAGA	076E'
BLOCO_	00AC'	BS	0008	BSALVO	312F'	C0_825	0010
C1_825	0011	C2_825	0012	CARR_C	07CB'	CAR_ES	0509'
CMD_CU	0080	CMD_LI	0023	CMD_NI	00C0	CMD_PA	0000
CMD_ZE	00E0	CNTRL	32C4'	CNT_TE	312C'	COL_CO	3135'
COL_DE	32C1'	COL_TE	000D	CONT0_	0032	CONT1_	0076
CONT2_	00B6	CONTRE	3301'	CR	000D	CSALVO	3130'
C_8275	0001	DEL	007F	DELTA_	0046	DESCE	05CE'
DESCLI	06EC'	DESC_1	05E3'	DET	312E'	DETESC	055C'
DIREIT	05E5'	DIR_1	05FA'	DSALVO	3131'	D_AUX	0022
D_SIO	0020	END_ES	0510'	ENTALT	05A1'	ENTWS	0581'
ENV	0314'	ENVIA	0251'	ESC	001B	ESCAPE	053F'
ESC_EN	0585'	ESC_SE	0578'	ESQUER	05FC'	E_AUX	0023
E_SIO	0021	F	47A8'	FF	000C	FIM_RO	32CC'
FORMFE	0546'	GUARDA	32C8'	ID_TER	077E'	IGNORA	0521'
IG_REG	028F'	INDESC	32C5'	INILIN	06DC'	INI_PI	312C'
INI_RO	32CB'	INI_T1	0177'	INI_T2	0167'	INI_TE	0142'
INI_VA	0121'	INS1	078E'	INS_TE	0780'	INTERP	04E2'
I_VAR	007F	LE_PAR	0694'	LE_TEC	0273'	LF	000A
LIGA_P	0499'	LIGA_V	01A4'	LIMITE	0014	LINHA	025C'
LININV	0624'	LIN_1	026D'	LIN_CO	3134'	LIN_DE	32C0'
LIN_MO	32CD'	LIN_TE	0008	LMPLIN	0663'	LMPR1	0655'
LMPR2	0661'	LMPRES	064F'	LOCAL	32FE'	MAIUSC	32C3'
MANDA	0247'	MASCB	007F	MASCC	003F	M_8253	0013
M_8255	0033	NAO_ES	051E'	NAO_SE	059F'	NAO_TR	0729'
NIL	0000	NORMAL	0331'	NOVALI	07A2'	NUM_CO	0050
NUM_LI	0018	NUM_PA	32C7'	ORIGEM	0611'	PA	0030
PAR_1	004F	PAR_2	0057	PAR_3	0089	PAR_4	00DD
PAR_CU	32C6'	PB	0031	PC	0032	PILHA	3000'
PILHA_	3136'	POE_NA	023E'	PONT_L	32CE'	PONT_S	32C9'
POPS	006F'	POS1	07F5'	POS2	07FD'	POS3	0806'
POS4	07FE'	POSIC	068A'	POSICH	07E6'	POSIC_	0809'
POS_1	06A6'	PPRINC	0000'	PPR_1	00BC'	PPR_2	00D4'
PPR_3	00D1'	PRG_55	008B	PRG_PE	00E9'	P_8275	0000
P_INS	3132'	RD_1	0746'	RD_2	0762'	RECEBE	029E'
REC_0	02A3'	REC_1	02B3'	REC_2	02AD'	REC_4	02BB'
REC_5	02C5'	REC_6	02DB'	REC_7	02E3'	REP1	003C
REP2	000A	RODA	0717'	RODA_B	063B'	RODA_C	0703'
R_BAUD	0005	SAIALT	05A9'	SALVAB	0231'	SCROLL	07BA'
SEM_DA	0270'	SLC	00F1	SOBE	05B9'	T1	0080
T2	0080	TAB	0009	TABULA	0523'	TAB_1	0531'
TAB_TE	081D'	TAM_LI	0062	TAM_TA	0070	TECALT	32FF'
TECLAD	01C9'	TECL_0	01D5'	TECL_1	01EE'	TECL_2	0207'
TECL_3	0216'	TECL_4	021E'	TECL_5	0221'	TEC_ES	02F8'
TELA	4000'	TELA_1	3138'	TES_1	031D'	TES_2	0327'
TRADUZ	04AF'	TRAD_1	04B3'	TRAD_2	04B8'	TRAD_3	04BB'

TRAD_4	04C8'	TRAD_5	04D5'	TRAD_6	04DF'	TRAD_7	04E0'
TRAT	312D'	TRAVA	32C2'	TRV	00F0	TR_1	0063'
TR_2	00A3'	TX_OK	070A'	T_BAUD	0005	VARRE	0275'
WS	0421'						

No Fatal error(s)

APÊNDICE H

ESQUEMA ELETRICO DO TERMINAL

APÊNDICE I

SINAIS DE CONTROLE DO CRTC 8275

Os sinais de controle do CRTC adotado são:

1. LC0-LC3 [4,3,2,1]

   Saida do registrador contador de linhas usada para endereçar o gerador de caracter. Indica qual o retraço ativo na matriz de caracteres.

2. DRQ [5]

   Saida indicadora de que o 8275 deseja acesso `a memória de tela para mais dados.

3. DACK/ [6]

   Entrada do mestre da memória indicando o reconhecimento do pedido DRQ.

4. HRTC [7]

   Saida ativa durante o periodo programado de retraço horizontal. VSP permanece alto e LTEN fica baixo durante esse tempo.

5. VRTC [8]

   Saida ativa durante o periodo programado de retraço vertical. VSP permanece alto e LTEN fica baixo durante esse tempo.

SINAIS DE CONTROLE DO CRTC 8275

6. RD/ [9]

Entrada do pulso de ativação de leitura.

7. WR/ [10]

Entrada do pulso de ativação de escrita.

8. LPEN [11]

Entrada do pulso de ativação de caneta de luz.

9. DB0-DB7 [12-19]

Linhas de dados bi-direcionais de três estados.

10. VSS [20]

Terra.

11. A0 [21]

Entrada de seleção do porto de comando/parâmetros.

12. CS/ [22]

Entrada de seleção do integrado. Os sinais de RD/ e WR/ são mascarados pelo CS/.

13. CC0-CC6 [23-29]

Saida dos códigos de caracter usada para endereçar o gerador de caracteres. Durante todas as varreduras que compoem um dado caracter, esses códigos ficam constantes letra a letra.

14. CCLK [30]

Entrada do relógio de caracter.

15. IRQ [31]

Solicitação de interrupção.

16. HLGT [32]

Saida sinalizadora de intensificação do sinal de vídeo, programável por códigos de atributo.

17. GPA0,GPA1 [33,34]

Códigos de atributos gerais, alteráveis por programação.

18. VSP [35]

Saida do sinal de supressão de vídeo, gerado durante os retraços, nas margens superiores e inferiores da tela e quando o pedido de ADM não e' atendida. Também e' usada para implementar os caracteres piscantes.

19. RVV [36]

Saida do sinal ativável por programação que indica a reversão do sinal de vídeo.

20. LTEN [37]

Saida do sinal que força a ocorrência de um ponto na tela. Usado para implementar o cursor e o sublinhado.

21. LA0,LA1 [38,39]

Saidas usadas para implementar algumas primitivas gráficas (extremamente limitadas). Esses sinais precisam ser decodificados externamente.

22. VCC [40]

Alimentação (+5v).

BIBLIOGRAFIA

1. SIEWIOREK, D., BELL, C. D., NEWELL, A., "Computer Structures: Principles and Examples", 1- edição, McGraw-Hill, Toquio, Japão, 1982.

2. HUSKEY, H. D., GRANINO, A. K., "Computer Handbook", 1- edição, McGraw-Hill, Nova Iorque, Estados Unidos, 1962.

3. BOGNI, C. J., MARRONE, L. A., "Computadoras: Introducción a las arquitecturas paralelo", 1- edição, Ed. Unicamp. Campinas, 1986.

4. WAKERLY, J. F., "Microcomputer Architecture and Programming", 2- edição, John Wiley & Sons, Toronto, Canada', 1981.

5. GUIMARAES, C. C., Principios de Sistemas Operacionais", 1- edição, Campus-Mec/Seplan, Rio de Janeiro, 1986.

6. STONE, H. S. et al, "Introduction to Computer Architecture", SRA - Science Research Associates, INC, 1- edição, Chicago, Estados Unidos, 1962.

7. REDDY, D. R., "Some Numerical Problems in Artificial Intelligence: Implications for Complexity and Machine Architecture", Complexity of Sequential and Parallel Numerical Algorithms, Academic Press, pp 131-148.

8. THURBER, K. J., "Parallel Processor Architectures", Computer Design, jan/fev 1979, pp 89-97 e 103-114.

9. GEPPERT, D. V.,"Basic Electron Tubes", McGraw-Hill Publishing Co, Nova Iorque, Estados Unidos, 1951.

10. SPANGENBERG, K. R., "Vacuum Tubes", McGraw Hill Publishing Co, Palo Alto, Estados Unidos, 1948.

11. DEVERE, C., "Cathode Ray Tubes", Academic Press, Beaverton, Estados Unidos, 1967.

12. DEVERE, C., "Storage Cathode-Ray Tubes and Circuits", Academic Press, Beaverton, 1967.

13. HAWKINS, P.,"The Vector Option", Systems International, abril, 1981.

14. McMANIGAL, D. e STEVENSON, D., "Architecture of the IBM 3277 Graphics Attachment", IBM Systems Journal, Vol. 19 Num. 3, Palo Alto, Estados Unidos, 1980, pp 331-334.

15. MEAD, S. D., TAYLOR, W. R., MINTZ, K. A., POTTER, C. M., "A Color Presentation Graphics Workstation", Hewlett-Packard Journal, Vol. 34, Num. 9, setembro 1983.

16. KEITH, J. C., SCHULTE, L. T., e VOGEN, A. K., "System 45 Hardware Design", Hewlett-Packard Journal, Vol 28, Num 8, abril, 1978.

17. HEWLETT-PACKARD, Especificações técnicas dos terminais 2382A, 2621B, 2622A, 2623A, 2627A, 2624B, 2626W, 2626A, 2645A, 2647F, 2648A, sistema 2700, sistema 100, sistema 200/10, 200/20, 200/26, 200/36 e sistema 9845, in HP Products Guide, Estados Unidos, 1975, pp 582-642.

18. TEKTRONIX, Especificações técnicas dos terminais 4051, 4006-1, 4010-1,4012, 4013, 4014-1, 4015-1, 4023, 611, 613, 602, 603, 604, 605 in Tektronix Products, Beaverton, 1976.

19. PARKER, Richard, "Input/Output Technology", Computer Design Vol 21 numero 12, Dez. 82, Penwell Publishing Co, Littleton, MA, pp.157-180.

20. PLESHKO, P., TANG, V.,"Inherent Pixel Memory in AC Plasma Displays Increases Screen Data", Computer Technology Review, pp 63-71, Verao de 1983

21. WEDDING, D. K., ERNSTHAUSEN, R. E, SOPER, T. J, e EWING, D. J., "Mammoth Displays that measure in meters", Computer Design, julho 1983, pp 177-184.

22. _____,"AC Plasma Panel Flat Screen Displays", Electronic Product Design, janeiro 1983, pp 39-43.

23. BOARDMAN, C. M. e MICHEL J. P., "Future Flat Displays - The liquid Crystal Answer", Electronic Engineering, fevereiro 1982, pp 38-53.

24. TSANTES, J., "Electroluminescent Display Devices Shine as LED, LCD, CRT Replacements", Electronic Design News (EDN), Vol. 25 Num 2, 20 janeiro 1980, pp 35-49.

25. SLOTTOW, H. G., "Plasma Displays" in IEEE Transactions on Electronic Devices, Vol. 23 Num. 7, 1976, pp 57-66.

26. TANNAS, L., "Flat Panel Displays: A Critique" in IEEE Spectrum, Vol. 15 Num. 7, 1978, pp 45-49.

27. WOODSFORD, P. A., "The HRD-1 Laser Display System" in SIGGRAPH 74 Conference Proceedings, Vol. 10 Num. 2, 1976, pp 68-73.

28. RUGGIERO, C. A.,"Projeto e Construção de um Terminal de Video Gráfico", Dissertação de Mestrado apresentada ao Instituto de Física e Química de São Carlos, 1983.

29. FOLEY, J. D. e Van DAM, A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley, Nova Iorque, Estados Unidos, 1979.

30. WAITE, M., "Computer Graphics Primer", Howard W. Sons & Co. Inc., Indianápolis, Estados Unidos, 1979.

31. HUNT, R. W. G., "The Reproduction of Color", John Wiley & Sons, Nova Iorque, Estados Unidos, 1975.

32. NEMCSICS, A., "The Coloroid Color System", Color Research adn Aplications, Vol. 5 Num. 2, 1980, pp 113-120.

33. WASSERMANN, G., "Color Vison: An Historical Introduction", John Wiley and Sons, Nova Iorque, Estados Unidos, 1978.

34. MURCH, T. e HUBER, J.,"Colour - the logical step", New Electronics, 15 junho 1982, pp 30-32.

35. OSTWALD, .W. - "Colour Science", Winsor & Winsor, Londres, Inglaterra, 1931.

36. GOMES, L. - "A Picture of the Future: Drawing on Computers", Intel Solutions, novembro/dezembro 1983, Santa Clara, Califórnia, Estados Unidos, pp 2-5.

37. _____,"Full Collor Graphics Added to Workstation", Computer Design, Vol. 222 num. 1, janeiro 1983, pag. 168

38. _____, "Raster Graphics Survey" in Systems International, janeiro, 1981, pp. 21-24.

39. MYERS, Ware - "Interactive Computer Graphics: Flying High - Part I", in Computer, Vol 12 numero 7, julho 1979, pp. 8-17.

40. MYERS, Ware - "Interactive Computer Graphics: Flying High - Part II" in Computer, Vol 12 numero 8, julho 1979, pp.52-67.

41. Digital Equipment Corporation, "VT-240 Programmer Reference Manual", publicação EK-VT240-RM-002, 2- edição, 1984.

42. Digital Equipment Corporation, "VT-240 Owner's Manual", publicação EK-VT240-UG-001, 2- edição, 1984.

43. BRINCH HANSEN, P. - "Operating Systems Principles", Prentice Hall, Englewoods, NJ, Estados Unidos, 1970.

44. THOMAS, R., YATES, J. - "A User Guide to the UNIX system", 1- edição, McGraw-Hill/Osborne, Berkeley, Estados Unidos, 1982.

45. MAGALHAES, ,"Software para Tempo Real", anais da I Escola Brasileira de Computação, Editora da Unicamp, Campinas, 1986.

46. Intel Corporation, "Intellec Hardware Reference Manual", 3- edição, 1984.

47. _____,"O sistema SOB para o CP500",Editele, São Paulo, 1984.

48. Digital Equipment Corporation, "VAX-VMS Digital Control Language Reference Manual", Maynard, Estados Unidos, 1986.

49. CHRISTIAN, K.,"Sistema Operacional Unix", 1- edição, Campus, Rio de Janeiro, 1985.

50. International Business Machines, "OS Job Control Language Reference Manual", Estados Unidos, 1986.

51. BELL, G., MUDGE, C., McNamara, J., "Computer Engineering", DEC Press, 1- edição, Maynard, Massachussets, Estados Unidos, 1978.

52. Digital Equipment Corporation, "Local Area Network Concepts", 1- edição, DEC, Estados Unidos, 1983.

53. Digital Equipment Corporation, "Terminals & Printers Handbook", edição 1983-84, Maynard, Massachussets, Estados Unidos.

54. Commodore Inc, "Commodore 64 Hardware Reference Manual", Estados Unidos, 1985.

55. Digital Equipment Corporation, "VAX Architechture Handbook", Maynard, Massachussets, 1982.

56. Motorola Semiconductors, "Microprocessors Data Manual", Suiça, 1982.

57. Motorola Semiconductors, "16 Bit Microprocessors Data Manual", Suiça, 1983.

58. Intel Corporation, "Peripheral Design Handbook", Santa Clara, Estados Unidos, 1981.

59. Scopus, Manual de Referência Técnica do Nexus 1600, São Paulo, 1986.

60. CCE Eletrônica, "Magic Window Reference Handbook", 1- edição, São Paulo, 1984.

61. KANE, G.,"CRT Controller Handbook", 1- edição, Osborne/McGraw-Hill, Berkeley, CA, Estados Unidos, 1980.

62. Phillips Holland,"Digital Imaging", publicação interna, 1984.